Gildas Avoine
Orhun Kara (Eds.)

LNCS 8162

Lightweight Cryptography for Security and Privacy

Second International Workshop, LightSec 2013
Gebze, Turkey, May 2013
Revised Selected Papers

Lecture Notes in Computer Science 8162

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Gildas Avoine Orhun Kara (Eds.)

Lightweight Cryptography for Security and Privacy

Second International Workshop, LightSec 2013
Gebze, Turkey, May 6-7, 2013
Revised Selected Papers

Springer

Volume Editors

Gildas Avoine
Université catholique de Louvain
1348 Louvain-la-Neuve, Belgium
E-mail: gildas.avoine@uclouvain.be

Orhun Kara
TÜBİTAK BİLGEM UEKAE
National Research Institute of Electronics and Cryptology
41470 Gebze, Kocaeli, Turkey
E-mail: orhun.kara@tubitak.gov.tr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-40391-0 e-ISBN 978-3-642-40392-7
DOI 10.1007/978-3-642-40392-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013945416

CR Subject Classification (1998): E.3, K.6.5, C.2, D.4.6, F.2, K.4.1

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

LightSec 2013 was the Second International Workshop on Lightweight Cryptography for Security and Privacy, which was held in TÜSSİDE Gebze/Kocaeli Turkey during May 6–7, 2013. The workshop was sponsored by TÜBİTAK BİLGEM UEKAE (The Scientific and Technological Research Council of Turkey, National Research Institute of Electronics and Cryptology).

The Program Committee (PC) consisted of 34 members representing 18 countries. There were 30 papers from 15 countries submitted to the workshop and three of them were withdrawn without review. The 27 remaining submissions were reviewed by the PC members themselves or assigned to external reviewers. Each submission was double-blind reviewed by at least three PC members and the submissions by PC members were assigned to five reviewers. The vast majority of the papers were reviewed by four reviewers. The PC members selected 10 papers out of the 27 submissions for presentation.

The program also included three excellent invited talks by experts in the field. The first talk was given by Axel Poschmann about "Implementation Aspects of Lightweight Cryptography." Gregor Leander presented the second talk on "Design Strategies for Lightweight Block Ciphers: PRINCE and Beyond." Jacques Stern gave the third invited presentation on the topic of "Randomness and Lack of Randomness in Cryptography."

We would like to thank all the people who contributed to making the workshop successful. First, we greatly appreciated the valuable work of the authors and we thank them for submitting their manuscripts to LightSec 2013. We are also grateful to the PC members and the external reviewers whose admirable effort in reviewing the submissions definitely enhanced the scientific quality of the program. Thanks also to the invited speakers, Axel Poschmann, Gregor Leander, and Jacques Stern, for their willingness to participate in LightSec 2013. Last but not least, we would like to thank Hüseyin Demirci and the organizing team for their management of the organization.

May 2013

Gildas Avoine
Orhun Kara

Organization

Organizing Committee

General Chair

Hüseyin Demirci	TÜBİTAK BİLGEM UEKAE, Turkey

Adnan Baysal
Muhammed Ali Bingöl
Serkan Çelik
Süleyman Kardaş
Mehmet Sarıyüce
Güven Yücetürk

Program Committee

Program Chairs

Gildas Avoine	Université catholique de Louvain, Belgium
Orhun Kara	TÜBİTAK BİLGEM UEKAE, Turkey

Onur Acıiçmez	Samsung, USA
Jean-Philippe Aumasson	NAGRA, Switzerland
Paulo Barreto	University of São Paulo, Brazil
Lejla Batina	Radboud University Nijmegen, The Netherlands
Guido Bertoni	STMicroelectronics, Italy
Mike Burmester	Florida State University, USA
Roberto Di Pietro	Università di Roma Tre, Italy
Orr Dunkelman	University of Haifa, Israel
Kris Gaj	George Mason University, USA
Helena Handschuh	Cryptography Research Inc., USA
Julio Hernandez-Castro	University of Kent, UK
Marc Joye	Technicolor, France
Pascal Junod	HEIG-VD, Switzerland
Mehmet Sabır Kiraz	TÜBİTAK BİLGEM UEKAE, Turkey
Çetin Kaya Koç	UCSB, USA
Xuejia Lai	Shanghai Jiao Tong University, China
Albert Levi	Sabanci University, Turkey
Thomas Brochmann Pedersen	TÜBİTAK BİLGEM UEKAE, Turkey
Josef Pieprzyk	Macquarie University, Australia

David Pointcheval	CNRS/ENS/INRIA, France
Axel Poschmann	Nanyang Technological University, Singapore
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Arash Reyhani-Masoleh	University of Western Ontario, Canada
Vincent Rijmen	Katholieke Universiteit Leuven, Belgium
Matt Robshaw	Orange Labs, France
Francisco Rodríguez-Henríquez	CINVESTAV-IPN, Mexico
Erkay Savaş	Sabanci University, Turkey
Mike Scott	CertiVox labs, Ireland
Ali Aydın Selçuk	Bilkent University, Turkey
François-Xavier Standaert	Université catholique de Louvain, Belgium
Serge Vaudenay	EPFL, Switzerland
Amr Youssef	Concordia University, Canada

Additional Reviewers

Aydın Aysu	Kazuhiko Minematsu
Laurent Bussard	Mehran Mozaffari-Kermani
Chien-Ning Chen	Salim Sarımurat
Barış Ege	Yannick Seurin
Jens Hermans	Dave Singelee
Jialin Huang	Marc Stöttinger
Süleyman Kardaş	Kazım Yumbul

Sponsoring Institution

TÜBİTAK BİLGEM UEKAE (The Scientific and Technological Research Council of Turkey, National Research Institute of Electronics and Cryptology)

Table of Contents

A Lightweight ATmega-Based Application-Specific Instruction-Set Processor for Elliptic Curve Cryptography

Erich Wenger

Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, 8010 Graz, Austria
erich.wenger@iaik.tugraz.at

Abstract. It is inevitable that future Radio-Frequency Identification (RFID) technology must support complex protocols and public-key cryptography. In this paper, we present an Application-Specific Instruction-Set Processor (ASIP) based on a clone of the ATmega128 microprocessor. A leakage-resilient, constant-runtime, and assembly-optimized software implementation of an elliptic curve point multiplication, which outperforms related work, requires 9,230–34,928 kCycles or 681–2,576 ms for standard conform elliptic curves (`secp160r1`, `secp192r1`, `secp224r1`, and `secp256r1`). Because this is too slow for most applications, the microprocessor has been equipped with a multiply-accumulate and a bit-serial instruction-set extension. Therefore, the runtime has been reduced to practically usable 96–248 ms, while keeping the power below 1.1 mW, and the area consumption between 19–27 kGE.

Keywords: ATmega, Elliptic Curve Cryptography, Instruction Set Extension, Application Specific Instruction-set Processor, Constant Runtime.

1 Introduction

The future Internet of things will consist of embedded smart cards, wireless sensor networks, and Radio-Frequency Identification (RFID) tags. Those devices must be capable to communicate with other entities over an air interface and must provide *privacy* and *security* capabilities. At Asiacrypt 2007, Serge Vaudenay [20] showed that "...an RFID scheme that achieves narrow-strong privacy ... essentially needs public-key cryptography techniques."

Among the three most popular public-key cryptographic systems (RSA, ElGamal, and ECC), Elliptic Curve Cryptography (ECC) is the least resource demanding and most suitable for embedded systems. In the past ECC has been well studied and standardized by SECG [2] and NIST [17]. One could also investigate non-standardized curves (e.g., by Gallant, Lambert, and Vanstone [7] or Bernstein *et al.* [1]), but for open-loop systems one should stick to the given

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 1–15, 2013.

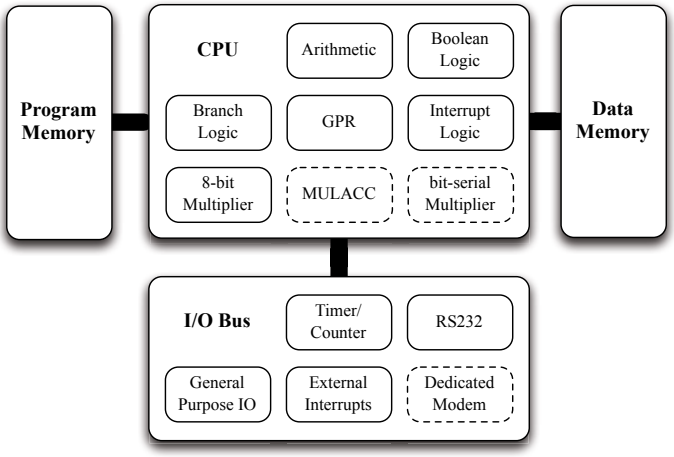


Fig. 1. Schematic diagram of the used processing architecture

standards. In this work we focus on the four prime-field based Weierstrass curves `secp160-256r1` as those are already used for mainstream applications such as TLS, IPSec, and SSH.

While public-key systems are very resource demanding, RFID tags must consume little power, be cheap (have a small chip area) and support real-time applications (respond within a given time). The traditional approach, which will pretty soon exceed its realms of possibility, is to equip the state machine of an RFID tag with a dedicated hardware block doing public-key cryptography. A more sophisticated solution is to base the design on a microprocessor, in particular on an Application-Specific Instruction-Set Processor (ASIP). An ASIP unites the advantages of programmable microprocessors (flexibility, extendability) with the advantages of dedicated hardware blocks (high-performance, low-power). Therefore, dedicated hardware units can be avoided and the overall hardware footprint decreases. In this paper, we transform a commercially available microprocessor into an ASIP targeting RFID and public-key cryptography.

For this paper, we base our design on the popular 8-bit Atmel ATmega128 AVR processor. This processor comes with an extensive instruction set and a dedicated hardware multiplier (important for prime-field arithmetic). The ATmega128 is used for a magnitude of applications and is supported by many toolchains (e.g., `avr-gcc`, IAR, Crossworks). In Wenger *et al.* [21], we presented 'Just Another AVR' (JAAVR, see Figure 1), a feature-complete clone of the popular ATmega128 which is written in VHDL and only requires 6,140 GE, making it perfectly suitable for area-sensitive embedded designs. In [21], we equipped an earlier version of JAAVR with a dedicated RFID modem, and evaluated ECC, AES, and Grøstl. As expected, those results show that ECC is the dominating component.

Our Contribution. In this paper we present an JAAVR-based ASIP optimized for ECC. First, we present new assembly optimized ATmega-compatible

runtime results in which we outperform related software implementations (including our own in [21]). Second, we optimize the cycles-per-instruction (CPI) of all load, store, and multiply instructions of JAAVR in order to achieve speedups of 25–27 %. Third, we are the first to actually build an ATmega128-compatible processor with multiply-accumulate instruction-set extensions as ASIC. Previous work was either simulated or only performed on FPGAs. Fourth, we are also the first to build a tightly-coupled bit-serial multiplier as instruction-set extension of JAAVR for prime-field based ECC. Utilizing all those techniques, we present a 19 kGE small design suitable for RFID and other real-time applications.

This paper is structured as follows: Section 2 elaborates some basic design decisions. Section 3 discusses efficient software implementation techniques for ECC. Sections 4-6 deal with the improvement of the CPI of JAAVR, the utilization of a multiply-accumulate instruction, and the integration of a bit-serial multiplier, respectively. Section 7 discusses the results in connection with related work. Section 8 concludes the work. The most important results are gathered in Table 2. They are discussed throughout the paper.

2 Basic Reasoning

For RFID applications, the runtime of an algorithm is important in two respects. First, it must be sufficiently fast to support real-time applications. Second, by having a fast implementation, one can reduce the clock frequency and therefore reduce the power consumption. For a passively powered RFID tag, the power consumption is of upmost importance. For a typical ISO-14443-compatible [13] tag, we assume the following requirements. The clock should be an integer fraction of 13.56 MHz, the maximum power consumption below 2 mW, and the maximum runtime for an ECC point multiplication is 100-500 ms. It should be noted that all our hardware designs easily exceed this minimum clock frequency of 13.56 MHz.

Tools. For all of our implementations, we performed hardware synthesis (Cadence RTL Compiler v08.10), power simulations (Cadence First Encounter v08.10), and cycle-accurate post-synthesis and post-layout hardware simulations (using NCSim v08.20). As process technology the UMC 130 nm low-leakage CMOS technology with Faraday design libraries in combination with area-efficient single-port register-based RAM macros and Via-1 ROM macros is used. Previous experiments showed that synthesizing the program memory as standard logic cells results in smaller (post synthesis) but less routable designs (post place-and-route). A decreased cell density increases the size of the synthesized program memory to a point where the available Via-1 ROM macro is effectively smaller.

Practical Security. When implementing cryptography, the designer must consider practical attack scenarios such as timing, side-channel and fault attacks. Regarding timing attacks, all assembly-optimized implementations perform the point multiplication in constant runtime. Further, all implementations provide a basic resistance against power-analysis attacks. The Montgomery ladder formula by Hutter *et al.* [11] performs key-independent double-and-add operations.

Table 1. `secp160r1` point multiplication results using different multi-precision integer multiplication methods: operand-scanning (OS), product-scanning (PS), hybrid, and operand-caching (OC)

Impl.	Point-Multiplication [kCycles]	Integer-Multiplication [Cycles]	Program-Memory Size [Bytes]	Integer Mul. [%]	Chip Area [GE]
OS in C	37,168	9,807	4,188	3	17,738
OS	17,607	5,505	12,110	62	23,540
PS looped	17,226	5,367	4,636	4	17,638
PS	13,546	4,035	9,860	54	21,701[a]
Hybrid	10,609	2,972	9,050	49	21,701[a]
OC	9,230	2,473	8,218	46	21,701[a]

[a] Identical, because only certain discrete ROM macros are available.

With its requirement of 16 field multiplications and 17 field additions per key bit, it is reasonably fast. The finite-field multiplication is used for multiplications as well as for squarings. At the end of the Montgomery ladder, a y-coordinate-recovery and a constant-runtime inversion based on exponentiation (Fermat's little theorem) are performed. For side channel and fault security we also perform projective point randomization [4] before (against side-channel attacks) and point verification before and after (against fault attacks) the point multiplication. Because we did not perform practical power analysis attacks or fault simulations, we do not claim to be side channel or fault secure, but we use algorithms that improve resistance against those attacks. Thus all our results are practically relevant.

3 The Baseline: Efficient Software Implementation

By choosing an 8-bit processor we start with a rather small but "arithmetically speaking" slow processor. Our first not constant-time, plain-C implementation showed excruciatingly-slow runtimes of 37–131 million cycles, 2.7–9.6 seconds (@ 13.56 MHz). Thus optimizing the existing code in assembly is mandatory. For all following comparisons we consider our C implementations as baseline. In hardware it requires 16.9–19.5 kGE and 561–656 μW.

The first (and most laborious) optimization we have performed is the replacement of all field operations with constant-runtime assembly functions. This not only improves the runtime but also makes all timing attacks infeasible. The field addition and subtraction operations have been unrolled and perform the reduction without branches. For the field multiplications, we have taken advantage of the standardized Mersenne-like primes to get branch-free code using only addition and shift operations.

The most time-consuming algorithm is the multi-precision integer multiplication. Hutter and Wenger [12] did a thorough comparison between the Schoolbook's operand-scanning (OS), Comba's [3] product-scanning (PS), Gura *et*

al.'s [9] hybrid and their own operand-caching (OC) multiplication methods. We implemented unrolled and looped versions of those algorithms in assembly. Table 1 shows that by doing so the runtimes of integer and point multiplications for `secp160r1` were improved by factors of 3.97 and 4.03, respectively. Our fastest implementation, based on the operand-caching method, achieved a runtime of 9,230 kCycles for a point multiplication. For comparison: Gura *et al.* [9], Szczechowiak *et al.* [19], Wenger *et al.* [21], and Liu *et al.* [16] achieved runtimes of 6,480 kCycles, 9,376 kCycles, 13,027 kCycles and 16,939 kCycles, respectively. However, most of those implementations do not consider side-channel attacks. For instance, Gura *et al.* used a Jacobian-based NAF point-multiplication formula. For reference, we applied the same technique as Gura *et al.* and improved their fastest implementation by 50 kCycles to 6,430 kCycles.

Apart from the expected runtime differences (OS > PS > Hybrid > OC), unrolling the integer multiplication has a huge impact on the size of the program code. Up to 62 % of the entries in the program memory are due to the unrolled integer multiplication. Compared to the C implementation, the chip size of the program memory increased by up to 76 %. Despite of that, assembly optimization and 'unrolling' improved the area-time-product by a factor of up to 3.3, thus establishing themselves as one of the most important optimization techniques.

The focus of this section was to perform software optimizations both applicable to the ATmega128 and JAAVR. In the next sections, we present hardware optimization techniques that improve both the execution time as well as the total hardware footprint.

4 Improving the CPU

Already during the design of JAAVR, we realized several avenues for optimization potentials. It was necessary to artificially introduce `NOP` operations in order to achieve identical cycles-per-instruction (CPI) counts compared to the original ATmega128. The most significant difference is that the ATmega128 uses a two-stage pipeline and JAAVR does not. So all we needed to improve the performance of *store* and *multiply* operations was to deactivate the `NOP` operations.

Unlike our previous paper [21], we also optimized memory *load* operations. For the cycle-accurate (CA) design, two cycles are needed to load data from the synchronous data memory. During the first cycle the address is applied to memory and during the second cycle the obtained data word is stored to a general purpose registers (GPR).

In order to reduce the latency of all *load* operations, we decided to introduce a pipelining structure. While the first cycle of the operation stays identical, the second cycle is performed as part of the subsequent operation. As Figure 2 shows, multiplexers before and after each general purpose register were added. MUX2 is used to update the next value stored within the GPR. MUX1 overrides the current contents within the GPR. Thus the ALU is working on an updated set of GPR. The impact of the multiplexers on the critical path is hardly noticeable.

By switching JAAVR from the CA to the FAST mode, the following instructions improve: `MUL*`, `ST`, `STD`, `PUSH`, `LD`, `LDD`, `POP`, `IJMP`, `RJMP`, `CBI`, `SBI` ($2 \rightarrow 1$),

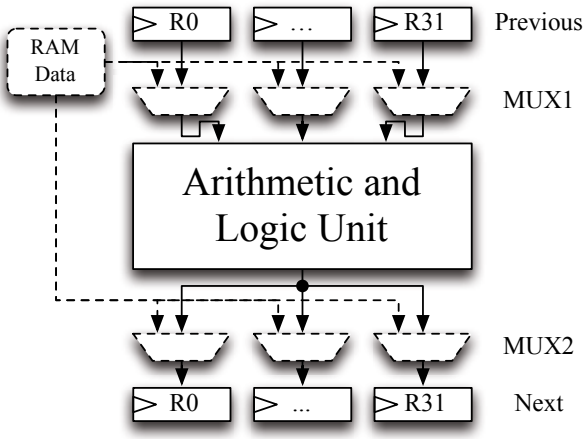


Fig. 2. The multiplexers were added to reduce the necessary cycles per load instruction

`RCALL`, `ICALL`, `LPM`, `ELPM` ($3 \rightarrow 2$), `CALL`, `RET`, and `RETI` ($4 \rightarrow 3$). This increased the size of JAAVR form 6,140 GE to 6,791 GE (by 10 %), while the runtime of the fastest point multiplication improved by 26 %. Thus, the area-runtime product improved by a factor of 1.31.

After enabling those optimizations, JAAVR is still instruction-set compatible with the original ATmega128. So any (cryptographic) algorithm would benefit from the improved instruction-timing. The next two sections are dedicated on optimizing JAAVR for ECC using instruction-set extensions, transforming our design into an ASIP.

5 The Power of the MULACC Command

When investigating the instructions used for the unrolled product-scanning multi-precision multiplication, one can observe that there are four instructions, always used in consecutive order: `MUL`, `ADD`, `ADC`, and `ADC`. The idea behind the multiply-accumulate instruction-set extension is to combine those instructions into a single `MULACC` command, as it has been done in related work.

Already in 2004, Großschädl and Savaş [8] used five custom instructions to accelerate prime fields and binary extension fields on a MIPS32 core and gained a speedup of about six for binary extension fields. In 2005, Eberle *et al.* [5] presented multiply-accumulate instruction-set extensions for binary-extension fields on an ATmega128. They improved `sect223r1` by a factor of 13.6, but did not use ISE for prime fields as we do it in this paper.

In fact, we used the `MULACC` instruction to improve the fastest multi-precision multiplication formula: the operand-caching method. We are the first to combine the operand caching method with an instruction-set extension. Like the product-scanning method, this method uses the same sequence of instructions as mentioned above. So, by combining the operand-caching multiplication, which

reduces the number of load and store operations, and the multiply-accumulate instruction, which reduces the number of additions, we achieved a new speed record: 631 cycles for a 160-bit integer multiplication.

There are two main challenges concerning the introduction of new instructions: First, most of the 2^{16} possibilities of the 16-bit instruction words are already assigned to existing instructions. Thus, the introduction of a new instruction would mean to modify existing instructions and being no longer compatible with the original ATmega128. Second, adapting the source code of `avr-gcc`, `avr-as`, and `avr-ld` to add new instructions does not seem to be straightforward.

Our solution is to introduce a new, within the I/O memory mapped, register that can switch the processor to a special operating mode. In this special operating mode, certain existing instructions are reinterpreted. For this solution, none of the `avr-gcc` tools had to be modified.

In order to improve the performance of the operand-caching multiplication, we introduced two instructions: `MULACC` and `ST_SHIFTACC`. `MULACC` multiplies two registers Rd and Rr and adds the result to the accumulator stored in R0-R2: $(R2, R1, R0) \leftarrow (R2, R1, R0) + Rd \times Rr$. This operation can be performed 2^8 times without the risk of an overflowing accumulator. This is more than the required $e = 10$ accumulations performed within the operand-caching multiplication algorithm (e is a parameter to adjust the operand caching method, see [12]). After e `MULACC` operations, `ST_SHIFTACC` is used to store the lowest byte of the accumulator (`ST R0,Z+`) and shifts the accumulator by 8 bits to the right ($R0 \leftarrow R1$, $R1 \leftarrow R2$, $R2 \leftarrow 0$). Because of those optimizations, we freed up two registers that were used as temporary storage of the product. In order not to waste them, we increased e from 10 to 11, which further decreased the number of necessary load and store instructions.

By using those instruction-set extensions, a 160×160-bit multiplication can be performed three times faster. It takes 631 cycles compared to 1,896 cycles. A detailed decomposition of the used instructions can be found in Appendix A. Further, the ISE had hardly any impact on the size of JAAVR. Only 257 GE or 3.8 % of additional logic had to be added. At the same time, the size of the program memory decreased: from 11,807 GE to 8,202 GE (-31 %). Adding all those improvements together, the area-time product improved by a factor of 2.4.

A point multiplication in `secp160r1` takes 3,268 kCycles. A profiling analysis showed that 83.2 % of the total runtime are spent on the field multiplications. The optimized reduction algorithm for the `secp160r1` prime $2^{160} - 2^{31} - 1$ utilizes 26.1 % of the total runtime or a third of the field multiplication. Thus, any further optimizations must not only consider the integer multiplication, but the field multiplication as a whole. This is done in the following section.

6 Using a Dedicated Digit-Serial Multiplier

When investigating related work on ECC, one can either find ECC designs based on an word-level multiplier (cf. [10,22]), as we used in the previous sections,

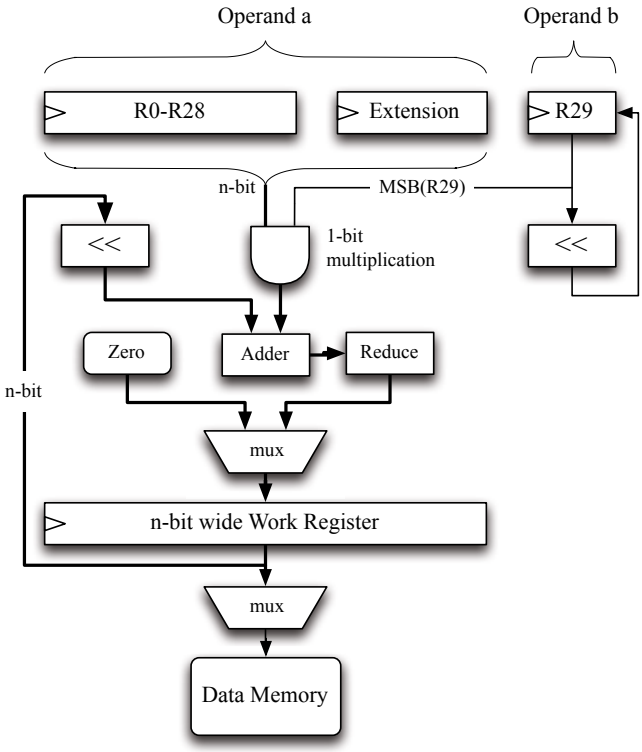


Fig. 3. The bit-serial multiplier is merged with the CPU

or designs based on a digit-serial multiplier (cf. [6]). A digit-serial multiplier simultaneously operates on all digits of the multiplicand a, but only a single digit b_i of the multiplicand b.

In this section we want to introduce the concept of a 'tightly-coupled' bit-serial multiplier, which merges a bit-serial multiplier with the CPU. By reusing existing registers, we were able to keep the impact on the total chip area to a minimum and avoid unnecessary data transfers.

A block diagram of our bit-serial multiplier is depicted in Figure 3. Algorithm 1 shows the pseudo-code to control it. An MSB-first multiplier is used which accesses all b_i starting with the most significant bit. The Z-register (`R30`, `R31`) is used to address the memory, and `R29` is used to store the byte containing b_i. During each cycle, `R29` is shifted to the left using the `LSL` (Logic Shift Left) instruction of JAAVR. At the same time the `Work` register is updated in the following manner: $\texttt{Work} \leftarrow (a \times b_i + \texttt{Work} \ll |b_i|) \pmod p$. In each cycle the most significant bit of `R29` (b_i) is multiplied with a, the product is added to a shifted version of the n-bit[1] `Work` register, and the `Work` register is updated with the reduced sum. After the last computation cycle, the product $a \times b \pmod p$

[1] n relates to the number of bits needed to represent any value in $\mathbb{F}_p$.

Algorithm 1. Pseudocode for the bit-serial multiplication

```
 1: PUSH all call-saved registers.
 2: LD operand a to R0-R28 and Extension.
 3: Switch to ISE mode.                          (memory mapped config register)
 4: Clear Work register.
 5: for i from ⌈n/8⌉ − 1 to 0 do
 6:     LD R29, -Z                               (load b_i, pre-decrement pointer register Z)
 7:     8 times: LSL R29                         (triggers bit-serial multiplier)
 8: end for
 9: Store Work register.
10: Switch back to normal mode.
11: POP all call-saved registers.
```

is stored in the `Work` register. A modified store instruction (`ST`) is used to write the `Work` register to memory, one byte at a time.

To reuse existing registers, a is stored in register `R0` to `R28` and the `Extension` register. For `secp256r1`, three IO-memory-mapped 8-bit `Extension` registers are necessary. So for `secp160-224r1` it was only necessary to add the `Work` register and the combinatoric logic.

A field multiplication for `secp160r1` takes 271 cycles. $9 \times 20 = 180$ cycles are used by the digit-serial multiplication, $2 \times 20 = 40$ cycles are necessary for loading operand a and storing the result, and 51 cycles are necessary to comply with the C-calling convention (`PUSH`, `POP`, `CALL`, `RET`) and to switch between the normal ATmega128 compatible operation mode and the instruction-set-extension mode.

Compared to the original software implementations in C, a speedup between 30 (`secp160r1`) and 44 (`secp256r1`) was achieved. The bit-serial approach is also 2.3–3.7 times faster than the `MULACC` instruction-set extension. The size of the program memory decreased significantly by 25 %–41 %. However, the size of the CPU core increased by 61 %–107 %. 4,551 GE are required for the bit-serial multiplier for `secp160r1` and 7,792 GE have to be added for `secp256r1`. The question now is whether adding the bit-serial multiplier improves or worsens the area-runtime product. In fact, it improves by a factor of 2.1–3.1, which makes the tightly-coupled digit-serial multiplier (by far) the fastest, even though not the smallest hardware implementation presented in this paper.

7 Results

The most important results of our implementations are summarized in Table 2 and have already been discussed in the previous sections. It contains figures that are characteristic for software and hardware implementations. Every row labeled with cycle accuracy (CA) is applicable for an ATmega128 as well as JAAVR. Using a TCL-based simulation script, we measured the data-memory requirements (including stack) of all implementations. The bit-serial designs needs the least data memory, because the memory for a temporary $2n$-bit product was

Table 2. Summary of all experiments. SARP stands for 'scaled area-runtime product'.

Impl.	Runtime	Program Memory	Data Memory	Area Requirement JAAVR	ROM	RAM	Total	Power @13.56 MHz	Energy	Runtime @13.56 MHz	SARP
	[kCycles]	[Bytes]	[Bytes]	[kGE]	[kGE]	[kGE]	[kGE]	[μW]	[μJ]	[ms]	
`secp160r1`											
CA in C	37,168	4,188	418	6,140	7,744	3,855	17,738	561	1,539	2,741	22.5
CA	9,230	8,218	402	6,140	11,807	3,754	21,701	662	450	681	6.8
FAST	6,764	8,218	402	6,791	11,807	3,754	22,352	824	411	499	5.2
MULACC	3,268	5,688	402	7,048	8,202	3,754	19,004	850	205	241	2.1
bit-serial	1,298	4,286	350	11,341	7,744	3,452	22,537	1,013	97	96	1.0
`secp192r1`, NIST P-192											
CA in C	55,365	3,916	483	6,140	6,505	4,233	16,877	640	2,615	4,083	21.6
CA	15,093	10,070	462	6,140	11,807	4,107	22,054	677	753	1,113	7.7
FAST	11,101	10,070	462	6,791	11,807	4,107	22,705	832	681	819	5.8
MULACC	5,022	6,396	462	7,048	10,040	4,107	21,195	864	320	370	2.5
bit-serial	1,813	4,490	406	12,302	7,744	3,779	23,825	1,084	145	134	1.0
`secp224r1`, NIST P-224											
CA in C	86,058	3,926	569	6,140	6,363	4,712	17,215	663	4,208	6,346	23.9
CA	23,213	12,374	526	6,140	15,484	4,485	26,109	689	1,179	1,712	9.8
FAST	17,114	12,374	526	6,791	15,484	4,485	26,760	843	1,063	1,262	7.4
MULACC	7,537	7,404	526	7,048	10,040	4,485	21,573	848	472	556	2.6
bit-serial	2,469	4,808	466	13,237	7,744	4,132	25,113	1,032	188	182	1.0
`secp256r1`, NIST P-256											
CA in C	130,695	5,604	645	6,140	8,202	5,165	19,506	656	6,320	9,638	27.8
CA	34,928	15,888	590	6,140	17,029	4,838	28,006	663	1,707	2,576	10.7
FAST	26,290	15,888	590	6,791	17,029	4,838	28,657	811	1,572	1,939	8.2
MULACC	11,900	9,372	590	7,048	11,807	4,838	23,693	836	733	878	3.1
bit-serial	3,367	5,532	522	14,583	8,202	4,460	27,244	1,031	256	248	1.0

saved. The RAM and ROM macros are chosen according to the data and program memory requirements. Because those macros are only available in certain sizes, not every difference measured in Bytes is reflected in the actual area of the ROM macro (in gate equivalents).

7.1 Reached Goals

All targeted goals ($< 2\,\text{mW}$, $< 500\,\text{ms}$ @ $13.56\,\text{MHz}$) have been met. An exception are the larger 224-bit and 256-bit elliptic curves which render the `MULACC` based approach as too slow. The runtimes of 100–200 ms show that the clock frequency can be decreased by factors of 2–4, which in turn would decrease the power consumptions by a factor of 2–4.

7.2 Related Work

For a fair comparison with related work, it is important to not only consider plain numbers (chip area, runtime, power), but also the provided features. We distinguish whether a design is microprocessor-based (MCU), comes with a C-compiler, considers side-channel attacks, or performs binary- or prime-field based ECC. One must also consider the used manufacturing technology, but this would go beyond the scope of this paper.

A fair comparison with dedicated hardware designs is tough. While they are optimized to the limit, they lack the rich set of features our ASIP provides. The ASIP is easily extendable and provides a compiler toolchain. Also our

Table 3. Comparison with related work

Reference	Runtime [kCycles]	Area [GE]	Characteristics
ISE - `secp160r1`			
Gura [9]	4,720	-	ATmega-based
OC + `MULACC`	3,268	19,004	ATmega-based
Dedicated Hardware - `secp160r1`			
bit-serial	1,298	22,537	ATmega-based
OC + `MULACC`	3,268	19,004	ATmega-based
Fürbass [6]	362	19,000	ECDSA-like
Dedicated Hardware - `secp192r1`			
Satoh [18]	4,165	29,655	ECC
bit-serial	1,813	23,825	ATmega-based
Fürbass [6]	502	23,600	ECDSA-like
OC + `MULACC`	5,022	21,195	ATmega-based
Hutter [10]	859	19,115	ECDSA, MCU
Wenger [22]	1,377	11,686	ECDSA, MCU

microprocessor-based approach has not yet reached its limits (c.f. Appendix C), but applying those ideas would make our design incompatible with a standard ATmega128. Table 3 summarizes the comparison with related work.

Fürbass *et al.* [6], Hutter *et al.* [10], Satoh *et al.* [18], and Wenger *et al.* [22] worked on dedicated prime-field based elliptic curve hardware designs. They require 12–30 kGE of hardware and 362–4,165 kCycles of runtime. Although most of their solutions are faster, it is important to notice that our solutions provide sufficiently fast response times. Hutter *et al.* and Wenger *et al.* implemented the full ECDSA signature algorithm and Fürbass *et al.* implemented ECDSA without a hash algorithm. The designs by Hutter *et al.* and Wenger *et al.* is micro controller based, but does not provide a C-compiler. The designs by Fürbass *et al.* and Satoh *et al.* are not micro controller based, so it probably is easier to adapt our designs for real-world scenarios.

Most comparable to this paper are the works of Gura *et al.* [9], Kumar and Paar [15], and Koschuch *et al.* [14]. Gura *et al.* added simulated ISE to an AVR processor, but achieved slower runtimes results. Kumar and Paar used the ATSTK94 FPSLIC demonstration board to extend an AVR processor with a bit-serial multiplier extension for binary extension fields. They however have not applied their methodology to prime fields and do not provide results for an ASIC. Koschuch *et al.* synthesized an 8051-compatible microprocessor and equipped it with a hardware accelerator for binary extension fields. In total, they needed 29 kGE and 1.2 MCycles. Even though we use prime fields, our results are smaller and approximately of similar speed.

8 Conclusion

After our thorough analysis of instruction-set extensions for ECC, the following conclusions can be drawn: First, if the area footprint is most important

(e.g., for RFID) our MULACC-based ASIP is the best choice. The chip area is on par with related work and reasonable response times of less than 370 ms are achievable. Second, for applications such as wireless sensor networks or embedded smart cards, the tightly-coupled bit-serial ASIP approach is most suitable. It provides the best energy efficiency and the best area-time product. Third, the design space for ECC implementations is huge: the ratios between the best and the worst implementation across all tested elliptic curves in the categories of area-runtime product, runtime, and energy are 87:1, 101:1, and 65:1, respectively. Our results show that the figures vary by up to two orders of magnitude across the hardware/software design space, which gives a designer a multitude of options to fine-tune a design for a given set of requirements.

Acknowledgements. The author wants to thank Thomas Plos for the support during the creation of this paper. This work has been supported by the Austrian Science Fund (FWF) under grant number TRP 251-N23 (Realizing a Secure Internet of Things - ReSIT).

References

1. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards Curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008)
2. Certicom Research. Standards for Efficient Cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0 (2000)
3. Comba, P.: Exponentiation cryptosystems on the IBM PC. IBM Systems Journal 29(4), 526–538 (1990)
4. Coron, J.-S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
5. Eberle, H., Wander, A., Gura, N., Chang-Shantz, S., Gupta, V.: Architectural Extensions for Elliptic Curve Cryptography over GF(2^m) on 8-bit Microprocessors. In: International Conference on Application-specific Systems, Architectures and Processors, pp. 343–349. IEEE Computer Society (July 2005)
6. Fürbass, F., Wolkerstorfer, J.: ECC Processor with Low Die Size for RFID Applications. In: Proceedings of 2007 IEEE International Symposium on Circuits and Systems. IEEE (May 2007)
7. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001)
8. Großschädl, J., Savaş, E.: Instruction Set Extensions for Fast Arithmetic in Finite Fields GF(p) and GF(2^m). In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 133–147. Springer, Heidelberg (2004)
9. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004)
10. Hutter, M., Feldhofer, M., Plos, T.: An ECDSA Processor for RFID Authentication. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 189–202. Springer, Heidelberg (2010)

11. Hutter, M., Joye, M., Sierra, Y.: Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 170–187. Springer, Heidelberg (2011)
12. Hutter, M., Wenger, E.: Fast Multi-precision Multiplication for Public-Key Cryptography on Embedded Microprocessors. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 459–474. Springer, Heidelberg (2011)
13. International Organization for Standardization (ISO). ISO/IEC 14443-3: Identification Cards - Contactless Integrated Circuit(s) Cards - Proximity Cards - Part3: Initialization and Anticollision (2001)
14. Koschuch, M., Lechner, J., Weitzer, A., Großschädl, J., Szekely, A., Tillich, S., Wolkerstorfer, J.: Hardware/Software Co-design of Elliptic Curve Cryptography on an 8051 Microcontroller. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 430–444. Springer, Heidelberg (2006)
15. Kumar, S., Paar, C.: Reconfigurable Instruction Set Extension for Enabling ECC on an 8-Bit Processor. In: Becker, J., Platzner, M., Vernalde, S. (eds.) FPL 2004. LNCS, vol. 3203, pp. 586–595. Springer, Heidelberg (2004)
16. Liu, A., Ning, P.: TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks. In: International Conference on Information Processing in Sensor Networks, pp. 245–256 (2008)
17. National Institute of Standards and Technology (NIST). FIPS-186-3: Digital Signature Standard, DSS (2009)
18. Satoh, A., Takano, K.: A Scalable Dual-Field Elliptic Curve Cryptographic Processor. IEEE Transactions on Computers 52, 449–460 (2003)
19. Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.: NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 305–320. Springer, Heidelberg (2008)
20. Vaudenay, S.: On Privacy Models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)
21. Wenger, E., Baier, T., Feichtner, J.: JAAVR: Introducing the Next Generation of Security-Enabled RFID Tags. In: DSD, pp. 640–647 (2012)
22. Wenger, E., Feldhofer, M., Felber, N.: Low-Resource Hardware Design of an Elliptic Curve Processor for Contactless Devices. In: Chung, Y., Yung, M. (eds.) WISA 2010. LNCS, vol. 6513, pp. 92–106. Springer, Heidelberg (2011)

A Decomposition of Instructions

Table 4 shows the decomposition of used instructions for performing a 160×160-bit multiplication with and without instruction-set extension. Using the ISE, the necessary additions were nearly eliminated.

Table 4. Decomposition of the number of cycles per type of instruction necessary for a 160×160-bit multiplication

Instruction	CA	FAST	ISE
`MUL`	800	400	0
`MULACC`	0	0	400
`LD`	160	80	76
`ST`	120	60	0
`ST_SHIFTACC`	0	0	58
`ADC`	820	820	18
`ADD`	420	420	18
`CLR`	63	63	4
`PUSH`	36	18	18
`POP`	36	18	18
Others	18	17	21
Total	**2,473**	**1,896**	**631**

B Runtimes of Finite-Field Operations

The runtimes of all finite-field operations for `secp160r1` are presented in Table 5. Especially the comparable slow finite-field multiplication greatly profited from the performed optimizations. Because the inversion is based on an exponentiation, the speedup of the inversion is a direct reflection of the speedup of the multiplication. Using the bit-serial multiplier, the ratio between additions and multiplications is only 1.5. This needs to be taken in concern when a method or formula for the point multiplication is chosen.

Table 5. Runtimes of finite-field operations for `secp160r1`

Operation	CA	FAST	ISE	bit-serial
Addition	291	176	176	176
Subtraction	291	176	176	176
Multiplication	3,024	2,249	984	271
Inversion	519,217	386,368	170,053	48,130

C The Limits

In this paper we concentrated on delivering sufficiently fast and power-aware ASIPs for future RFID technology. We sticked to modifying the processing core and only optimized the finite-field operations in assembly language. However, if you want to make our design into an actual product, further optimizations need to be considered.

Constants consume space within the program and data memory. At startup they are loaded from the program memory and stored to the data memory. If one would add a memory-mapped table within the data memory bus, one could significantly reduce the size of the necessary RAM macro. The RAM macro could be shrunken by at least seven times 160-bit = 140 bytes.

Memory Management is currently performed by the compiler by reserving memory on the stack. If the whole source code would be written in assembly, unnecessary and redundant data memory entries could be avoided.

Processor Features that are not needed for ECC could be removed. E.g. the I/O bus is mapped within the data memory, or `MOVW` instructions are not needed for the finite-field operations. Removing those feature would decrease the size of JAAVR by 420 GE.

Processor Instructions that are not needed for ECC could be removed. For instance the `FMUL*` and `MULS*` multiplier instructions are not needed for an ECC point multiplication.

Program Memory is currently synthesized as Via-1 ROM macros. Using smaller ROM macros would significantly decrease the size of the program memory. Because the program memory is the largest part of the presented design, decreasing its size has a significant impact on the total chip area.

ITUbee: A Software Oriented Lightweight Block Cipher

Ferhat Karakoç[1,2], Hüseyin Demirci[1], and A. Emre Harmancı[2]

[1] Tübitak BILGEM UEKAE, 41470, Gebze, Kocaeli, Turkey
{ferhat.karakoc,huseyin.demirci}@tubitak.gov.tr
[2] Istanbul Technical University, Computer Engineering Department, 34469, Maslak, Istanbul, Turkey
harmanci@itu.edu.tr

Abstract. In this paper, we propose a software oriented lightweight block cipher, ITUbee. The cipher is especially suitable for resource constrained devices including an 8-bit microcontroller such as sensor nodes in wireless sensor networks. For a sensor node one of the most important constraints is the low energy consumption because of the limited battery power. Also, the memory on sensor nodes are restricted. We have simulated the performance of ITUbee in the AVR ATtiny45 microcontroller using the integrated development platform Atmel Studio 6. We have evaluated the memory usage and clock cycles needed for an encryption. The number of clock cycles gives a metric for energy consumption. The simulation results show that ITUbee is a competitive block cipher on 8-bit software platforms in terms of energy consumption. Also, less memory requirement of the cipher is remarkable. In addition, we have shown that the attacks which are effective on software oriented lightweight block ciphers can not reduce the 80-bit security level of ITUbee.

Keywords: Lightweight block cipher, cryptanalysis, sensor nodes, AVR ATtiny.

1 Introduction

Lightweight cryptography is needed for the security and privacy demands of the applications where resource constrained devices such as RFID tags and sensor nodes are used. Because of the increase in the usage of these devices in daily life, designing lightweight primitives is getting prominent. Block ciphers are essential primitives in cryptographic applications and therefore there have been several designed lightweight block ciphers. Some of the proposed lightweight block ciphers are DESXL [21], HIGHT [16], KASUMI [1], KATAN [8], KLEIN[14], LBlock [30], LED [15], mCrypton [22], PRESENT [6], PRINCE [7], PRINTcipher [19], SEA [28] and TEA [29]. However, most of the ciphers in this list have a hardware oriented design.

In this paper, we propose a new software oriented lightweight block cipher, ITUbee, for resource constrained devices which include a microcontroller and

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 16–27, 2013.

have a limited battery power such as sensor nodes in wireless sensor networks. To reduce the energy consumption of the cipher we have preferred not to use a key schedule. Also, we have designed the cipher based on a Feistel structure. Generally the ciphers having no key schedule are of SPN structure such as LED and PRINCE. To the best of our knowledge, the only cipher based on Fesitel Structure with a very simple key schedule is GOST [32]. However, a cipher based on Fesitel Structure with no key schedule (or a very simple key schedule) is subject to related key attacks as observed in GOST cipher [20]. To prevent this weakness we have used a new approach in the design of the round function. In every round the round key is injected between two nonlinear operations.

To evaluate the performance of ITUBEE we have simulated the energy consumption and memory usage of the cipher on the Atmel ATtiny45 8-bit microcontoller using Atmel Studio 6. The simulation results show that ITUBEE is an energy efficient block cipher and the memory requirement is very low. We have compared ITUBEE with the ciphers mentioned in a recent work [13] to present its efficiency in terms of energy consumption. Also, we have analyzed the security of the cipher against some attacks applied on software oriented lightweight block ciphers. We claim that ITUBEE offers 80-bit security.

The rest of the paper is organized as follows. In Section 2, we give the definition of ITUBEE. We explain the design rationale in Section 3. In Section 4, we present preliminary security analysis of the cipher. After giving simulation details and performance results of ITUBEE, we compare the performance results with the results of some existing ciphers in Section 5. We conclude the paper with Section 6.

2 Definition of ITUBEE

2.1 Notation

Throughout this paper we use the following notations:

$A\|\|B$	: Concatenation of two bit strings A and B.
K	: 80-bit master key.
K_L	: The left half of the master key.
K_R	: The right half of the master key.
P	: 80-bit plaintext.
P_L	: The left half of the plaintext.
P_R	: The right half of the plaintext.
C	: 80-bit ciphertext.
C_L	: The left half of the ciphertext.
C_R	: The right half of the ciphertext.
RC_i	: The round constant used in the i-th round.

2.2 Definition of the Cipher

The key length and block size of ITUBEE are 80 bits. The cipher has a Feistel structure consisting of 20 rounds and there are key whitening layers at the top and at the bottom of the cipher as illustrated in Figure 1.

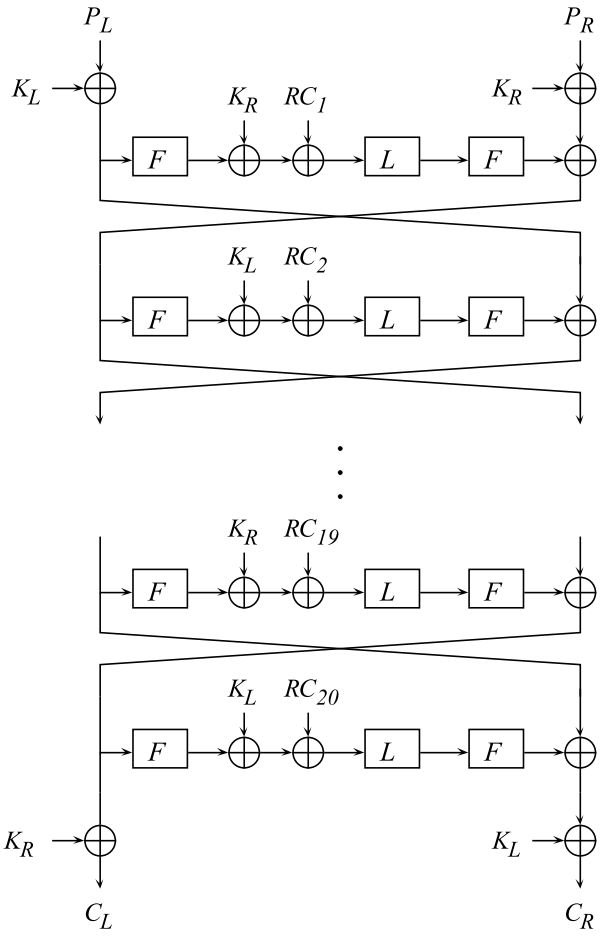


Fig. 1. ITUBEE encryption algorithm

The encryption operation proceeds as follows:

1. $X_1 \leftarrow P_L \oplus K_L$ and $X_0 \leftarrow P_R \oplus K_R$.
2. for $i = 1...20$ do
   (a) if $i \in \{1, 3, ..., 19\}$
      i. $RK \leftarrow K_R$.
   (b) else
      i. $RK \leftarrow K_L$.
   (c) $X_{i+1} \leftarrow X_{i-1} \oplus F(L(RK \oplus RC_i \oplus F(X_i)))$. Note that 16-bit round constant is added to the rightmost 16 bits.
3. $C_L \leftarrow X_{20} \oplus K_R$ and $C_R \leftarrow X_{21} \oplus K_L$.

The definitions of the functions used in the rounds are:

Table 1. Round constants used in ITUBEE

i	RC_i	i	RC_i	i	RC_i	i	RC_i
1	0x1428	6	0x0f23	11	0x0a1e	16	0x0519
2	0x1327	7	0x0e22	12	0x091d	17	0x0418
3	0x1226	8	0x0d21	13	0x081c	18	0x0317
4	0x1125	9	0x0c20	14	0x071b	19	0x0216
5	0x1024	10	0x0b1f	15	0x061a	20	0x0115

- $F(X) = S(L(S(X)))$.
- $S(a\|b\|c\|d\|e) = s[a]\|s[b]\|s[c]\|s[d]\|s[e]$ where a, b, c, d, e are 8-bit values and s is the S-box used in AES [10].
- $L(a\|b\|c\|d\|e) = (e \oplus a \oplus b)\|(a \oplus b \oplus c)\|(b \oplus c \oplus d)\|(c \oplus d \oplus e)\|(d \oplus e \oplus a)$.

The round constants used in ITUBEE are given in Table 1.

The whitening and round keys are derived from the master key directly. $(K_L\|K_R)$ and $(K_R\|K_L)$ are used as whitening keys at the top and at the bottom of the encryption algorithm respectively and for odd rounds K_R is used as round keys while for even rounds K_L is used.

The decryption process is the same as the encryption process except that in the decryption process the key is $(K_R\|K_L)$ and the round constants are used in reversed order.

2.3 Security Goal

ITUBEE offers 80-bit security. Also, the cipher provides the same security level against attacks in related key models as well as in single key attack models. In addition, the cipher has no weak keys.

3 Design Rationale

Power consumption is a major consideration in the design of ITUBEE because of the limited battery power of sensor nodes. Also, we have paid attention on the memory requirement.

We have preferred a Feistel structure which enables us to use the same program code for encryption and decryption processes in a microcontroller which leads to less memory requirement. Also, the elimination of the key schedule reduces the energy consumption and memory requirement. The usage of a Fesitel structure with no key schedule providing security against related key attacks makes ITUBEE different from previously proposed ciphers.

S-boxes satisfy good confusion with less number of operations in microcontrollers. Thus, we have used 8-bit S-boxes to confuse 8 bits with just a table look-up which reduces the power consumption dramatically. In the linear layer we have used a cellular automaton just consisting of 15 XOR operations. To make the proofs for the security of the cipher against differential, linear and

related key differential attacks simple we have constructed a 40-bit substitution box, F function, in a light way which is $S \circ L \circ S$.

The cipher has a Feistel structure and also has no key schedule. These two properties of the cipher makes the cipher weaker for related key differential attacks. One example of such an attack was applied on full GOST [20]. To save the security of the cipher against for such an attack we have injected the round keys between two nonlinear operations (two F functions). To the best of our knowledge this rationale is a not followed before. Also, we have used a linear layer between two F functions to avoid using two consecutive S-boxes.

Some attacks such as reflection [17], slide [4], and slidex [12] use similarities between round functions. To break the similarities between round functions we have used round constants. However, we have reduced the size of the constants to save from the number of operations and memory requirement as in [19]. We have derived that the number of bits of a round constant should be at least 16. If we had used 8-bit round constants then there would be some patterns in the ciphertexts as described in Proposition 1 and 2.

Proposition 1. *F function preserves the pattern for the inputs (a, b, b, a, c) where a, b, c are 8-bit values.*

Proof. The first S-box layer does not change the pattern. After the linear layer the output will be $((s[c] \oplus s[a] \oplus s[b]) \| (s[a] \oplus s[b] \oplus s[b]) \| (s[b] \oplus s[b] \oplus s[a]) \| (s[b] \oplus s[a] \oplus s[c]) \| (s[a] \oplus s[c] \oplus s[a])) = ((s[c] \oplus s[a] \oplus s[b]) \| s[a] \| s[a] \| s([b] \oplus s[a] \oplus s[c]) \| s[c])$ which is the same pattern. Thus, this pattern is not changed by F function.

Proposition 2. *Assume that 1-byte round constants are used in* ITUBEE *and the round constants are added on the rightmost byte of 40-bit value. When the input and the key are of the form $(a \| b \| b \| a \| c \| d \| e \| e \| d \| f)$ then the ciphertext will be in the same form where a, b, c, d, e, f are 1-bytes.*

Proof. P_L, P_R, K_L, K_R have the same pattern as in Proposition 1. Thus, the key additions in the algorithm does not change the pattern. Also, F, L and S-box layers preserve this pattern as presented in Proposition 1. In addition, the round constant addition does not change the pattern because the constant changes only the rightmost byte which does not affect the pattern. As a result, C_L and C_R will have the same pattern and this is independent of the number of rounds.

Also, we have chosen i-th round constant as $(0x15 - i) \| (0x29 - i)$ not to use any memory for the constants in the case where only encryption process is required in the microcontroller.

We have observed that the maximum number of rounds that cryptanalytic attacks can be applied is around 10. To have a high security margin we have decided the number of rounds as 20.

4 Security Analysis

4.1 Differential and Linear Cryptanalysis

The differential and linear cryptanalysis are the mostly used cryptanalysis techniques [3,25]. In these techniques, nonlinear operations in an encryption process

are treated as linear operations with a probability to model the whole cipher as a linear algorithm.

The only nonlinear part in ITUBEE is the S-box operation. Thus, counting the number of active S-boxes in differential and linear characteristics is the main work of the differential and linear cryptanalysis of the cipher. The branch number of the linear layer in F is 4, that means at least 4 S-boxes are active when one F is active. For one round, if the left half of the input is active then there will be 2 active F functions. In the Feistel Structure the left halves of the inputs of at least 2 rounds out of 3 consecutive rounds have differences if the function which produces the output to add to the right half is one-to-one. In our cipher, this function is one-to-one, so we can say that we have at least 4 active F functions that is 16 active S-boxes for 3 consecutive rounds. The S-box used in ITUBEE is the AES S-box and the best probability for one input-output difference is 2^{-6} and the best linear bias for an input-output mask is 2^{-4}. Thus, for consecutive 3 rounds the best probability of a differential characteristic and the best linear bias of a linear characteristic will be $(2^{-6})^{16} = 2^{-96}$ and $2^{15} \times (2^{-4})^{16} = 2^{-49}$ respectively which are not usable in differential and linear attacks. Therefore, it seems that differential and linear attacks can not be applied on 20-round ITUBEE.

However, to give a proof against the differential and linear attacks it is not sufficient to count the number of active S-boxes in the characteristics because of the differential and the linear hull effects.

To analyze the differential effects on our cipher we have focused on the differential probabilities for the F function. We have seen that while one active F function has at least 4 active S-boxes which leads to the probability $(2^{-6})^4 = 2^{-24}$, the F function can have differentials having a greater probability than 2^{-24}. The reason is the following summation of the probabilities of the characteristics:

$$Pr(\Delta X \xrightarrow{F} \Delta Y) = \sum_{\Delta Z} Pr(\Delta X \xrightarrow{S} \Delta Z) \times Pr(L(\Delta Z) \xrightarrow{S} \Delta Y).$$

This effect can increase the probability of a differential for the F function but the probability will be less than 2^{-17}. In the case where only 4 bytes are active totally in the input and output of the F function, there will only be one free active byte which can take any difference. This free active byte can take at most 2^7 different values. Thus, the summation of the probabilities of differential characteristics which leads to a differential for the F function will be less than $(2^{-6})^4 \times 2^7 = 2^{-17}$.

In consecutive 6 rounds there will be at least 8 active F functions and therefore the probability of a differential characteristic for 6 rounds will be less than $(2^{-17})^8 = 2^{-136}$. However, it is necessary to consider the differential effect on consecutive two F functions. To see this effect we have performed experiments on a toy version of F functions. In this version the S-box is replaced with a 3-bit S-box and the word size of 8-bit is replaced with 3-bit and the other operations are same. In this version the maximum probability of F function is less than or equal to $(2^{-2})^4 \times 2^2 = 2^{-6}$. The experiment result shows that the maximum probability of consecutive two F functions is not bigger than 2^{-6}. As a result

of this experiment, we assume that maximum differential probability for consecutive two F functions is not bigger than 2^{-17}. For 8 rounds there is at least 5 active rounds and so this leads to a probability smaller than $(2^{-17})^5 = 2^{-85}$ which is not usable in a classical differential attack. Also, note that the differentials which has a maximum probability for two consecutive F functions are key dependent so this leads to another difficulty for these type of attacks.

4.2 Meet-in-the-Middle Type Attacks

Each key bit affects all bits of the output after consecutive 3 rounds. Thus, we assume that the basic meet-in-the-middle attacks are not applicable to our cipher. Recently, there have been some extensions of the basic meet-in-the-middle attacks such as the multidimensional meet-in-the-middle attack [33], the biclique attack [5]. In the case of independent biclique type attacks, let us assume that the key whitenings does not exist. The maximum number of rounds on which a biclique can be constructed is 2. Also, in the meet-in-the-middle step of this attack the number of F functions in the recalculation step which is done for the whole key space is about 32. Thus, the complexity of such an attack is approximately $\frac{32\times 2^{80}}{40} \approx 2^{-79.678}$ so this attack can not reduce the security level of ITUBEE more than 1-bit. The multidimensional meet-in-the-middle attack is usable if the key length of the cipher is bigger than the block size. For ITUBEE the block size and key length are the same. We conclude that also this attack does not threat the security level of our cipher.

4.3 Related Key Differential Attacks

In the related key attack model, the adversary is able to collect plaintext and ciphertext pairs under related keys. In our algorithm, we divide the master key into two parts K_L and K_R and we use these parts between F functions in the rounds. It is trivial to see that when there is a difference in the key used between F functions, then at least one of the F functions will be active. In the best case for the adversary, the difference will be only one part of the key K_L or K_R. As a result, in two consecutive rounds there exists at least one active F function in the case of the related key attack. The probabilities of differentials for the F function is less than 2^{-17} as given in the Section 4.1. Thus, for 10 consecutive rounds the probabilities will be less than $(2^{-17})^5 = 2^{-85}$ which is not usable in an attack.

4.4 Impossible Differential Attacks

One of the most powerful attacks on lightweight block ciphers is the impossible differential attack [2] which have been applied on many lightweight ciphers [9,18,23,24,27,31]. We could not find any impossible differential characteristic for 6 or more rounds. We conclude that this attack technique is not applicable to our cipher.

4.5 Self-similarity Attacks

The sel-similarty attacks such as reflection [17], slide [4], and slidex [12] use similarities of round functions. For ITUBEE the round functions are very similar due to the non-existence of the key schedule. The only part in the cipher which prevent the cipher from these attacks is the round constant addition. We believe that because of the round constant these attacks can not be applied to ITUBEE.

5 Simulation Results

We have written the code of ITUBEE in assembly and simulated the energy consumption and memory usage of it on the Atmel 8-bit AVR ATtiny45 RISC-based microcontroller using Atmel Studio 6. The microcontroller has a Harvard architecture in which the instruction and data memory are separated. The instruction and data memory are a 4-kB Flash memory and 256-byte static RAM, respectively. In the implementation of our cipher we have stored the 8-bit S-box used in the cipher in the instruction memory. Also, we have used CPU registers for all internal variables and we have not used any SRAM except for the plaintext/ciphertext and master key. To compare the performance of our cipher with some other lightweight ciphers we present the memory requirement of the encryption process and the number of clock cycles needed for an encryption operation of some ciphers in Table 2. Note that the implementations in [13] were also on an ATtiny45 microcontroller. Also, we give the number of clock cycles per one byte in the table dividing the number of cycles for an encryption to the block size in bytes. In addition, we present the product of the number of clock cycles per one byte with the memory requirement of the ciphers in the table to give another metric to compare the performance results of the ciphers.

The table demonstrates that an ITUBEE encryption process is performed in less clock cycles than the other ciphers. The number of clock cycles is strongly correlated with the energy consumption [11,13]. As a result, it can be extracted from the table that ITUBEE is the encryption algorithm which has the least energy consumption in the list. 716 byte memory requirement of the cipher is also remarkable. In the case of using less memory, the energy consumption will increase but it is still remarkable.

6 Conclusion

We have proposed a software oriented lightweight block cipher named ITUBEE for applications such as wireless sensor networks consisting of low-power nodes including an 8-bit microcontroller. We have used a Feistel structure with no key schedule to reduce the energy consumption. To prevent the cipher from related key attacks we have used the round key addition between two nonlinear layers in each round. We have simulated the performance of the cipher in terms of energy consumption and memory usage on the 8-bit ATtiny45 microcontroller. We have shown that ITUBEE consumes less energy than the ciphers whose performance

Table 2. Performance results of some lightweight ciphers

Cipher	Block size [bits]	Key size [bits]	Memory [bytes]	Clock cycles per one enc.	Clock cycles per one byte	Cycle × Memory
AES [13]	128	128	1689	4557	284	479676
DESXL [13]	64	184	868	84602	10575	9179100
HIGHT [13]	64	128	434	19503	2437	1057658
IDEA [13]	64	128	1068	≈ 8250	1031	1101108
KASUMI [13]	64	128	1288	11939	1492	1921696
KATAN [13]	64	80	356	72063	9007	3206492
KLEIN [13]	64	80	1286	6095	761	978646
mCrypton [13]	64	96	1104	16457	2057	2270928
NOEKEON [13]	128	128	396	23517	1469	581724
PRESENT [13]	64	80	1018	11342	1417	1442506
SEA [13]	96	96	450	41604	3467	1560150
TEA [13]	64	128	672	7408	926	622272
LBlock [30]	64	80	not given	3955	494	-
ITUBEE [this paper] cycle optimized	80	80	716	2607	261	186876
ITUBEE [this paper] memory optimized	80	80	586	2937	294	172284

results are given in a recent work [13] while the memory usage of ITUBEE is also remarkable. In addition, we have analyzed the security of the cipher against some attacks which are effective on software oriented lightweight block ciphers and we have concluded that these attacks can not reduce the 80-bit security level of ITUBEE.

Acknowledgments. We thank to Özkan Boztaş and Cevat Manap for their helpful comments on the design of the cipher. We also thank to anonymous reviewers for their valuable comments which helped us to improve the quality of this paper.

References

1. ETSI. TS 135 202 V7.0.0: Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI specification (3GPP TS 35.202 version 7.0.0 Release 7), `http://www.etsi.org`
2. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999)
3. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991)
4. Biryukov, A., Wagner, D.: Slide attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)

5. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
6. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
7. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
8. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
9. Chen, J., Wang, M., Preneel, B.: Impossible Differential Cryptanalysis of the Lightweight Block Ciphers TEA, XTEA and HIGHT. In: Mitrokotsa, Vaudenay (eds.) [26], pp. 117–137
10. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
11. de Meulenaer, G., Gosset, F., Standaert, F.-X., Pereira, O.: On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks. In: WiMob, pp. 580–585. IEEE (2008)
12. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 336–354. Springer, Heidelberg (2012)
13. Eisenbarth, T., Gong, Z., Güneysu, T., Heyse, S., Indesteege, S., Kerckhof, S., Koeune, F., Nad, T., Plos, T., Regazzoni, F., Standaert, F.-X., van Oldeneel tot Oldenzeel, L.: Compact Implementation and Performance Evaluation of Block Ciphers in ATtiny Devices. In: Mitrokotsa, Vaudenay (eds.) [26], pp. 172–187
14. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A New Family of Lightweight Block Ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
15. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
16. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
17. Kara, O.: Reflection Cryptanalysis of Some Ciphers. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 294–307. Springer, Heidelberg (2008)
18. Karakoç, F., Demirci, H., Emre Harmanci, A.: Impossible Differential Cryptanalysis of Reduced-Round LBlock. In: Askoxylakis, I., Pöhls, H.C., Posegga, J. (eds.) WISTP 2012. LNCS, vol. 7322, pp. 179–188. Springer, Heidelberg (2012)
19. Knudsen, L., Leander, G.R., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)

20. Ko, Y., Hong, S., Lee, W., Lee, S., Kang, J.-S.: Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 299–316. Springer, Heidelberg (2004)
21. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
22. Lim, C.H., Korkishko, T.: mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
23. Liu, Y., Gu, D., Liu, Z., Li, W.: Impossible Differential Attacks on Reduced-Round LBlock. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 97–108. Springer, Heidelberg (2012)
24. Liu, Y., Gu, D., Liu, Z., Li, W.: Improved Results on Impossible Differential Cryptanalysis of Reduced-Round Camellia-192/256. Journal of Systems and Software 85(11), 2451–2458 (2012)
25. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
26. Mitrokotsa, A., Vaudenay, S. (eds.): AFRICACRYPT 2012. LNCS, vol. 7374. Springer, Heidelberg (2012)
27. Özen, O., Varıcı, K., Tezcan, C., Kocair, Ç.: Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 90–107. Springer, Heidelberg (2009)
28. Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 222–236. Springer, Heidelberg (2006)
29. Wheeler, D.J., Needham, R.M.: TEA, a Tiny Encryption Algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)
30. Wu, W., Zhang, L.: LBlock: A Lightweight Block Cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
31. Wu, W., Zhang, L., Zhang, W.: Improved Impossible Differential Cryptanalysis of Reduced-Round Camellia. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 442–456. Springer, Heidelberg (2009)
32. Zabotin, I.A., Glazkov, G.P., Isaeva, V.B.: Cryptographic Protection for Information Processing Systems. Cryptographic Transformation Algorithm. Government Standard of the USSR, GOST 28147-89 (1989)
33. Zhu, B., Gong, G.: Multidimensional meet-in-the-middle attack and its applications to katan32/48/64. Cryptology ePrint Archive, Report 2011/619 (2011), http://eprint.iacr.org/

A AES S-Box

```
s[256] = {
0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76
0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0
0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15
0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75
0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84
0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF
0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8
```

```
0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2
0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73
0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB
0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79
0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08
0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A
0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E
0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF
0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
}
```

B Test Vectors

Plaintext	Key	Ciphertext
00000000000000000000	00000000000000000000	471330577984cbecf6c8
01000000000000000000	00000000000000000080	761b8299b3f6a99f0838
6925278951fbf3b25ccc	c538bd9289822be43363	c42e0f48cd5a87d0055f

Related-Key Slide Attacks on Block Ciphers with Secret Components

Meltem Sönmez Turan

National Institute of Standards and Technology, Gaithersburg, MD
meltem.turan@nist.gov

Abstract. Lightweight cryptography aims to provide sufficient security with low area/power/energy requirements for constrained devices. In this paper, we focus on the lightweight encryption algorithm specified and approved in *NRS 009-6-7:2002* by Electricity Suppliers Liaison Committee to be used with tokens in prepayment electricity dispensing systems in South Africa. The algorithm is a 16-round SP network with 64-bit key using two 4-to-4 bit S-boxes and a 64-bit permutation. The S-boxes and the permutation are kept secret and provided only to the manufacturers of the system under license conditions. We present related-key slide attacks to recover the secret key and secret components using four scenarios; (i) known S-box and permutation with 2^{48} time complexity using $2^{16}+1$ chosen plaintexts; (ii) unknown S-box and known permutation with 2^{55} time complexity using $2^{22.71}+1$ chosen plaintexts; (iii) known S-box and unknown permutation with 2^{48} time complexity using $2^{16}+1$ chosen plaintexts and $2^{12.28}$ adaptively chosen plaintexts; and finally, (iv) unknown S-box and permutation, with 2^{48} time complexity using $2^{22.71}+1$ chosen plaintexts and $2^{31.29}$ adaptively chosen plaintexts. We also extend these attacks to recover the secret components in a chosen-key setting with practical complexities.

Keywords: Lightweight Block Ciphers, Related-Key Slide Attacks, Secret Components.

1 Introduction

Lightweight cryptography aims to provide sufficient security with low area/power/energy requirements for constrained devices; such as RFID tags, smart cards, tiny computing devices etc. Although there have been constant efforts to improve the performance of AES [1,2], the smallest AES implementation requires 2400 GEs [3], mainly due to the large key and block sizes. This makes AES implementations inappropriate for many devices with tight area constraints. Due to the limitations of AES, several lightweight block ciphers with smaller key and block sizes have been proposed recently, such as PRESENT [4], Hight [5], DESL [6], PRINTcipher [7] etc.

Security through obscurity is not a widely accepted approach to provide security. Still, in some of the schemes using small key sizes, it is common to find

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 28–42, 2013.

"obscurity" being used as a complementary mean to achieve security. One famous example having security by obscurity is the stream cipher A5/1 that is used in GSM applications. Although the algorithm was initially kept secret, it became public by reverse engineering [8]. Another example is the block cipher C2 [9], designed for digital rights management, which uses secret 8-to-8 bit S-box that is available only through licensing. Borghoff et al. [10] presented an attack on C2 that recovers the secret S-box and the key, with $2^{53.5}$ time complexity. In this paper, we focus on the lightweight block cipher specified and approved in the standard *NRS 009-6-7:2002* [11] by Electricity Suppliers Liaison Committee to be used with tokens in prepayment electricity dispensing systems in South Africa. The respective algorithm is a 16-round SP network with 64-bit key, with simple rounds similar to PRESENT [4]. The description of the main components, namely two 4-to-4 bit S-boxes and the 64-bit permutation, are not publicly available, and being provided only to the manufacturers of the system, under license conditions.

Recently, Borghoff et al. [12,13] presented a generic differential-type attack strategy to attack PRESENT-like ciphers in which the S-boxes are chosen uniformly at random at each round and the bit permutation is secret. The attack is based on *slender sets* which are defined as the set of eight input pairs that yield the same output different Hamming weight one, to recover an S-box. The attacker constructs truncated differentials for the full cipher to determine the slender sets of the S-boxes. The main assumption of the attack is that the probability of the truncated differentials is higher when the input difference to the second round has Hamming weight of one, which is not necessarily true for ciphers having a strong confusion layer. Hence, as also mentioned in [12,13], the attack does not apply to ciphers where cryptographically strong S-boxes are used.

In this paper, we apply the well-known slide attacks, proposed by Biryukov and Wagner [14], to attack the target cipher. We present related-key slide attacks to recover the secret key and secret components, in four different scenarios of adversary capabilities: (i) known S-box and permutation with 2^{48} time complexity using $2^{16}+1$ chosen plaintexts; (ii) unknown S-box and known permutation with 2^{55} time complexity using $2^{22.71}+1$ chosen plaintexts; (iii) known S-box and unknown permutation with 2^{48} time complexity using $2^{16}+1$ chosen plaintexts and $2^{12.28}$ adaptively chosen plaintexts; and finally, (iv) unknown S-box and permutation, with 2^{48} time complexity using $2^{22.71}+1$ chosen plaintexts and $2^{31.29}$ adaptively chosen plaintexts. We also extend these attacks to recover the secret components of the algorithm in a chosen-key setting with practical complexities. Moreover, we present a generic way to improve the exhaustive search around 19 percent. The attack complexities are summarized in Table 1.

The organization of the paper is as follows. In Sec. 2, we give a brief description of the encryption algorithm. In Sec. 3, we present the details of the related-key slide attack for different scenarios. In Sec. 4, we extend secret component recovery attacks using a chosen-key setting. In Sec. 5, we provide a method to improve the

Table 1. Complexities of the related-key slide attacks with different adversary capabilities

§	S-box	Perm.	Time Comp.	# CP	# Adapt. CP	# Related Keys
3.1	✓	✓	2^{48}	$2^{16}+1$	-	2
3.1	✗	✓	2^{55}	$2^{22.71}+1$	-	2
3.3	✓	✗	2^{48}	$2^{16}+1$	$2^{12.28}$	8
3.4	✗	✗	2^{48}	$2^{22.71}+1$	$2^{31.29}$	8

attack complexities. Finally, we conclude this paper and summarize our results in Sec. 6.

2 The Algorithm Specification

The encryption algorithm specified by *NRS 009-6-7:2002* [11] is a 64-bit block cipher with an SP-network structure, having 16 rounds. Let the 64-bit key K be $(k_0, k_1, \ldots, k_{63})$. The 16-bit round keys for encryption, $\mathsf{RK}_i = (r_{i,0}, r_{i,1}, \ldots, r_{i,15})$ $(0 \leq i \leq 15)$ are defined as;

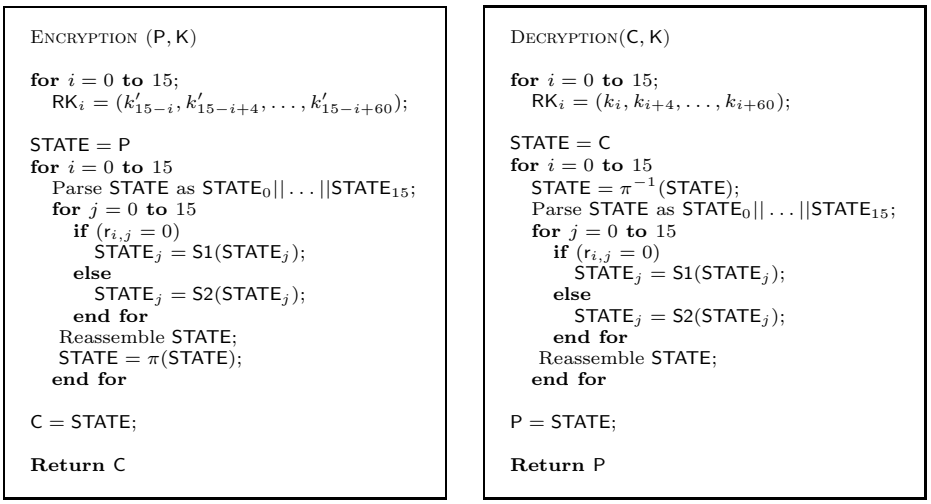
$$\mathsf{RK}_i = (k'_{15-i}, k'_{15-i+4}, k'_{15-i+8}, \ldots, k'_{15-i+60}),$$

where all the indices are calculated modulo 64 and k'_i represents the bitwise complement of k_i. Hence, in each round 16 bits of the key are used.

Each round consists of a key-dependent nonlinear substitution layer S and a linear bitwise permutation layer π. In the nonlinear layer, the state is partitioned into 16 four-bit sub-blocks and a 4-to-4 bit S-box is applied to each sub-block. The algorithm uses two S-boxes $\mathsf{S1}$ and $\mathsf{S2}$, which are selected alternatively according to the corresponding round-key bit. The permutation $\pi(x_1, \ldots, x_{64})$ is defined as $(x_{p(1)}, \ldots, x_{p(64)})$, where $p(i)$ is a permutation of $\{1, 2, \ldots, 64\}$.

In the decryption algorithm, the round keys are defined as $\mathsf{RK}_i = (k_i, k_{i+4}, k_{i+8}, \ldots, k_{i+60})$, for $0 \leq i \leq 15$ and the round operations are applied in the reverse order. The encryption round keys are the complement of the decryption round keys, i.e. the ith round key for encryption is the bitwise complement of $(15-i)$th round key for decryption. Therefore, in order to satisfy $P = D_K(E_K(P))$ for any plaintext P, $\mathsf{S1}$ should be the inverse of $\mathsf{S2}$. The pseudocodes of the encryption and decryption algorithms are provided in Fig. 1.

It should also be noted that the cipher $\mathsf{Cipher1}$ that uses the S-boxes $\mathsf{S1}, \mathsf{S2}$, and key K, denoted as $\mathsf{Cipher1}(\mathsf{S1}, \mathsf{S2}, \mathsf{K})$ is equivalent to the cipher $\mathsf{Cipher2}(\mathsf{S2}, \mathsf{S1}, \bar{\mathsf{K}})$, where $\bar{\mathsf{K}}$ is the bitwise complement of K. In other words, the encryption of any plaintext P using $\mathsf{Cipher1}$ having K and $\mathsf{Cipher2}$ having $\bar{\mathsf{K}}$ are equal. In this paper, we assume that the attack is successful, whenever the attacker can recover $\mathsf{Cipher1}$ or $\mathsf{Cipher2}$.



ENCRYPTION (P, K)

for $i = 0$ **to** 15;
  $\mathsf{RK}_i = (k'_{15-i}, k'_{15-i+4}, \ldots, k'_{15-i+60})$;

$\mathsf{STATE} = \mathsf{P}$
for $i = 0$ **to** 15
  Parse STATE as $\mathsf{STATE}_0 || \ldots || \mathsf{STATE}_{15}$;
  for $j = 0$ **to** 15
    if $(\mathsf{r}_{i,j} = 0)$
      $\mathsf{STATE}_j = \mathsf{S1}(\mathsf{STATE}_j)$;
    else
      $\mathsf{STATE}_j = \mathsf{S2}(\mathsf{STATE}_j)$;
    end for
  Reassemble STATE;
  $\mathsf{STATE} = \pi(\mathsf{STATE})$;
  end for

$\mathsf{C} = \mathsf{STATE}$;

Return C

DECRYPTION(C, K)

for $i = 0$ **to** 15;
  $\mathsf{RK}_i = (k_i, k_{i+4}, \ldots, k_{i+60})$;

$\mathsf{STATE} = \mathsf{C}$
for $i = 0$ **to** 15
  $\mathsf{STATE} = \pi^{-1}(\mathsf{STATE})$;
  Parse STATE as $\mathsf{STATE}_0 || \ldots || \mathsf{STATE}_{15}$;
  for $j = 0$ **to** 15
    if $(\mathsf{r}_{i,j} = 0)$
      $\mathsf{STATE}_j = \mathsf{S1}(\mathsf{STATE}_j)$;
    else
      $\mathsf{STATE}_j = \mathsf{S2}(\mathsf{STATE}_j)$;
    end for
  Reassemble STATE;
  end for

$\mathsf{P} = \mathsf{STATE}$;

Return P

Fig. 1. Pseudocode of the encryption and the decryption algorithm

3 Related-Key Slide Attacks

The slide attack, introduced by Biryukov and Wagner [14], is a generic attack that can be applied to block ciphers with periodic key schedule and the attack is applicable independent of the number of rounds.

We consider an attacker with black box access to two encryption devices initialized with secret keys K and K', respectively, where K' is equal to the left cyclic rotation of K by 1 bit, i.e. $\mathsf{K} = \mathsf{K}' \lll 1$. Consequently, the round keys generated by K and K', denoted as RK_i and RK'_i respectively, satisfies $\mathsf{RK}_{i+1} = \mathsf{RK}'_i$, for $i = 0, \ldots, 14$ and $\mathsf{RK}'_{15} = \mathsf{RK}_0 \lll 4$.

The plaintexts P and P' are considered to be *slid pairs*, if one round encryption of P using the round key RK_0 is P' and the following 15 rounds are identical as given in Fig. 2. Whenever the attacker has a slid pair P and P', he can search for the round key RK_0 (or $\mathsf{RK'}_{15}$) using the following partial encryptions,

$$\mathsf{P}' = \pi(\mathsf{S}(\mathsf{P}, \mathsf{RK}_0)), \text{or}$$
$$\mathsf{C}' = \pi(\mathsf{S}(\mathsf{C}, \mathsf{RK'}_{15})).$$

After recovering 16 bits of the secret key, the attacker can exhaustively search for the remaining key bits. However, before the exhaustive search, any components that remain secret should first be recovered. This recovery might need more slid pairs, which can be accomplished (provided one slid pair is already known) by the algorithm described in Appendix A.

In the following four subsections, we present related-key slide attacks for four scenarios; (*i*) known S-box and known permutation, (*ii*) unknown S-box and known permutation, (*iii*) known S-box and unknown permutation and lastly, (*iv*) unknown S-box and unknown permutation.

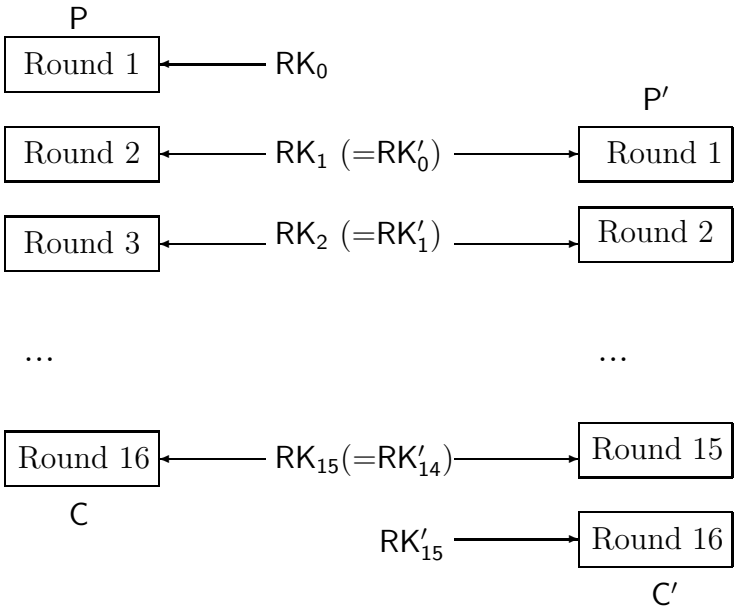


Fig. 2. Slid pairs P and P'

3.1 Known S-Box and Permutation Case

First, the attacker tries to identify slid pairs P and P', such that one round partial encryption of P using RK_0 is equal to P', i.e., $\mathsf{P}' = \pi(\mathsf{S}(\mathsf{P}, \mathsf{RK}_0))$. The attacker fixes P' to all zero input, and since π is a bitwise permutation, $\pi(0, \ldots, 0) = (0, \ldots, 0)$ is satisfied, then P also satisfies

$$\mathsf{S}(\mathsf{P}, \mathsf{RK}_0) = (0, \ldots, 0) \tag{1}$$

Then, the attacker finds $N_p = 2^{16}$ candidates $\{P_1, \ldots, P_{N_p}\}$ for P such that (1) holds. These candidates for P should be of the form $(a_0, a_1, \ldots, a_{15})$, where four-bit a_i's can take at most two distinct values, let's say a and b, such that $\mathsf{S1}(a) = 0$ and $\mathsf{S2}(b) = 0$, i.e., $a_i \in \{a, b\}$ $(0 \leq i \leq 15)$. For each candidate $P_i = (a_0, a_1, \ldots, a_{15})$, there exists a unique round key $R_i = (r_0, r_1, \ldots, r_{15})$, where

$$r_i = \begin{cases} 0, & \text{if } a_i = a \\ 1, & \text{if } a_i = b \end{cases}$$

such that (1) holds $(0 \leq i \leq 15)$. Next, the attacker obtains $C_i = E_{\mathsf{K}}(P_i)$ values and finds the correct (P_i, C_i) pair that satisfy

$$\mathsf{C}' = \pi(\mathsf{S}(C_i, \mathsf{RK}'_{15}), \tag{2}$$

where $\mathsf{RK}'_{15} = R_i \lll 4$. After obtaining the correct P_i, the attacker determines R_i, hence the 16 bits of the secret key K. The attacker exhaustively searches for

the rest of the key bits. The time complexity of the attack is $2^{48} + 2^{16} \approx 2^{48}$ with $2^{16} + 1$ chosen plaintexts, where 2^{16} plaintexts are encrypted using key K and one plaintext is encrypted using key K'.

3.2 Unknown S-Box and Known Permutation Case

In this case, the values of a and b satisfying $\mathsf{S1}(a) = 0$ and $\mathsf{S2}(b) = 0$ are unknown to the attacker, therefore, for all possible selections of a and b, the attacker considers the plaintexts of the form

$$(a_0, a_1, \ldots, a_{15}) \tag{3}$$

where $a_i \in \{a, b\}$'s $(0 \leq i \leq 15)$. Then, N_p is $2^{16}\binom{16}{2} \approx 2^{22.91}$. For a strong S-box, we can assume that there exist no fixed points (particularly, $\mathsf{S1}(0) \neq 0$ and $\mathsf{S2}(0) \neq 0$), then N_p reduces to $2^{16}\binom{15}{2} \approx 2^{22.71}$. It should be noted that the corresponding first round key for each P_i is not unique. There are two possibilities, where one of them is the bitwise complement of the other. The attacker is allowed to use any of them, and depending on his selection he either recovers Cipher1 or Cipher2, defined in Sec. 2. Note that Cipher1 and Cipher2 are equivalent.

After obtaining $\mathsf{C}' = E_{\mathsf{K}'}(\mathsf{P}')$, where P' is the all zero plaintext, the attacker obtains N_p ciphertexts, $(C_i = E_{\mathsf{K}}(P_i))$. Since the permutation is known, the attacker can determine $\pi^{-1}(\mathsf{C}')$, and try to find the correct (P_i, C_i) pair that satisfy

$$\pi^{-1}(\mathsf{C}') = \mathsf{S}(C_i, \mathsf{RK}'_{15}), \tag{4}$$

where $\mathsf{RK}'_{15} = RK_i \lll 4$. Since the S-boxes are unknown to the attacker, it is not trivial to eliminate the C_i's that do not satisfy (4). However, for each (P_i, C_i) pair, the attacker can partially construct the S-boxes and eliminate the P_i's that result in invalid S-boxes.

For the correct $P_i = (a_0, \ldots, a_{15})$ (with corresponding $C_i = (c_0, \ldots, c_{15})$ and $R_i = (r_{i,0}, \ldots, r_{i,15})$), the following should hold:

- **Property 1:** If $r_{i,j} = 1$, then $\mathsf{S1}(a_j) = 0$, otherwise $\mathsf{S2}(a_j) = 0$, for all $0 \leq j \leq 15$.
- **Property 2:** Due to the symmetry of the S-boxes, if $r_{i,j} = 1$, then $\mathsf{S2}(0) = a_j$, otherwise $\mathsf{S1}(0) = a_j$, for all $0 \leq j \leq 15$.
- **Property 3:** Let $\mathsf{C}'' = \pi^{-1}(\mathsf{C}') = (\mathsf{c}''_1, \mathsf{c}''_2, \ldots, \mathsf{c}''_{16})$. Due to the key schedule, $\mathsf{RK}_{15} = R_i \lll 4$. Then, if $r_{i,j+4} = 1$, then $\mathsf{S1}(a_j) = \mathsf{c}''_{\mathsf{j}}$, otherwise $\mathsf{S2}(a_j) = \mathsf{c}''_{\mathsf{j}}$, for all $0 \leq j \leq 15$.
- **Property 4:** Due to the symmetry of the S-boxes, if $r_{i,j+4} = 1$, then $\mathsf{S2}(\mathsf{c}''_{\mathsf{j}}) = a_j$, otherwise $\mathsf{S1}(\mathsf{c}''_{\mathsf{j}}) = a_j$, for all $0 \leq j \leq 15$.

Using this approach, we obtain 36 constraints (4 from Property 1 and 2; 32 from Property 3 and 4) for the S-boxes (some of which might be equivalent). Whenever we obtain contradicting constraints, we conclude that the candidate is incorrect. The correct P_i will never result in a contradiction, however, in

theory, with very small probability, it is possible that an incorrect P_i satisfies all constraints. We have experimentally tried 2^{25} random inputs and observed that for none of the inputs, all of the 36 constraints are satisfied, so we assume that the expected number of false alarms in $2^{22.71}$ plaintexts is zero.

After determining the value of P, the attacker can partially construct S-box, however, it is possible that the S-boxes are not uniquely determined. Based on Proposition 1 provided Appendix B, the attacker is expected to determine 11 outputs of $\mathsf{S1}$ and $\mathsf{S2}$ (out of 16) and this enables the attacker to reduce the possible S-boxes from $2^{44}(= 16!)$ to approximately $2^7(= 5!)$.

Example 1. Let

$$\begin{aligned}
\mathsf{P} &= \texttt{(10,1,1,1,10,1,1,10,10,10,1,10,10,1,1,1)},\\
\mathsf{P}' &= \texttt{(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)},\\
\mathsf{C} &= \texttt{(7,2,4,1,6,12,14,5,13,7,6,1,10,11,14,2)},\\
\pi^{-1}(\mathsf{C}') &= \texttt{(12,3,8,5,3,10,13,13,14,15,14,0,0,9,13,3)},\\
\mathsf{RK}_1 &= \texttt{(1,0,0,0,1,0,0,1,1,1,0,1,1,0,0,0)},\\
\mathsf{RK}_{15} &= \texttt{(1,0,0,1,1,1,0,1,1,0,0,0,1,0,0,0)}.
\end{aligned}$$

Due to Property 1 and 2, the following values of the S-boxes are determined.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	10	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*
S2	1	*	*	*	*	*	*	*	*	*	0	*	*	*	*	*

Due to Property 3 and 4, the following values of the S-boxes are determined.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1	10	0	3	6	8	1	14	15	*	*	12	9	7	5	13	*
S2	1	5	*	2	*	13	3	12	4	11	0	*	10	14	6	7

Using the properties, 13 (out of 16) values of S-boxes are determined. The undetermined outputs of $\mathsf{S1}$ can take three values, namely 2, 4, 11. Therefore, there are only 6 different S-boxes that can satisfy the given constraints.

The attacker obtains 16 key bits with a time complexity of $2^{22.71}$ and $2^{22.71}+1$ chosen plaintexts. Then, the attacker searches for the rest of the keys using exhaustive search, for each possible S-box. The complexity of this part is $2^{55}(= 2^{48+7})$. The attacker can verify the obtained key and the S-box using the previously obtained plaintext and ciphertext pairs. So, the total complexity of the attack is $2^{55} + 2^{22.71} \approx 2^{55}$ with $2^{22.71} + 1$ chosen plaintexts.

3.3 Known S-Box and Unknown Permutation Case

Since the S-boxes are known (similar to the case given in Sec. 3.1), P_i's can take 2^{16} values of the following form $(a_0, a_1, \ldots, a_{15})$ where $a_i \in \{a, b\}$ (for $0 \leq i \leq 15$)

such that $\mathsf{S1}(a) = 0$ and $\mathsf{S2}(b) = 0$. To identify P, the attacker obtains the corresponding C_i's and applies one more substitution layer to the C_i's using the corresponding round keys. For the correct i, $\mathsf{S}(C_i, R_i \lll 4) = \pi(\mathsf{C}')$ holds. Although the attacker does not know the permutation, based on the following observation, it is possible to eliminate the C_i's, when the weight of $\mathsf{S}(C_i, R_i \lll 4)$ is different from the weight of C'

Observation 1. *Let $X = (x_1, \ldots, x_{64})$ be a 64 bit value. The weight of X, denoted as $w(X)$ is the number of x_i's that are equal to 1. Then,*

$$w(X) = w(\pi(X)) = w(\pi^{-1}(X)),$$

since the permutation π only changes the positions of the bits.

For an incorrect i, we can assume that $\mathsf{S}(C_i, R_i \lll 4)$ and C' are independent. Since the weight of randomly chosen 64-bit inputs is binomially distributed with mean 32 and variance 16, the probability that $w(\mathsf{S}(C_i, R_i \lll 4))$ and $w(\mathsf{C}')$ are equal is

$$\begin{aligned} Pr(w(\mathsf{S}(C_i, R_i \lll 4)) = w(\mathsf{C}')) &= \sum_{i=0}^{64} Pr(w(C_i) = i) \times Pr(w(\mathsf{C}') = i) \\ &= \sum_{i=0}^{64} \binom{64}{i} 0.5^{64} \times \binom{64}{i} 0.5^{64} \\ &= 0.07. \end{aligned}$$

With probability $0.93\ (= 1 - 0.07)$, the attacker eliminates the incorrect P_i's. In out of $N_p = 2^{16}$ candidates, $0.07N_p \approx 2^{12.17}$ of them are expected to result in a false alarm. For $2^{12.17}$ P_i's, the attacker generates other slid pairs (See Appendix A) and checks the corresponding weights of the slid pairs as described above. If the weights are not equal, the attacker concludes that the corresponding P_i was a false alarm. By using one additional slid pair, the expected number of false alarms reduces from $2^{12.17}$ to $2^{8.34}$. As seen from Table 2, five slid pairs are enough to identify the correct P_i.

To identify P, hence 16 bits of the key, the required time complexity is $2^{16.11}(= 2^{16} + 1 + 2^{12.28})$ with $2^{16} + 1$ chosen plaintexts and $2^{12.28}$ adaptively chosen plaintexts. To recover the rest of the 48 bits of the key, the attacker first needs to recover the bit permutation π.

Table 2. False alarms and complexities using additional slid pairs

# Slid pairs	# False Alarms	# Chosen Plaintext	Adaptively Chosen Plaintext
1	$2^{12.17}$	$2^{16}+1$	0
2	$2^{8.34}$	$2^{16}+1$	$2^{12.17}$
3	$2^{4.51}$	$2^{16}+1$	$2^{12.27}$
4	$2^{0.69}$	$2^{16}+1$	$2^{12.28}$
5	$2^{-3.18}$	$2^{16}+1$	$2^{12.28}$

Recovering the Permutation π. For each slid pair X and Y, the attacker has an input/output pair for π, i.e.

$$Y = \pi(\mathsf{S}(X, RK_0)). \tag{5}$$

Let the attacker has N (X_i, Y_i) pairs such that $X_i = (x_1^i, \ldots, x_{64}^i)$ and $Y_i = \pi(X_i) = (x_{p(1)}^i, \ldots, x_{p(64)}^i)$. Initially, $p(j) \in \{1, \ldots, 64\}$, for $1 \leq j \leq 64$. For each (X_i, Y_i), the attacker updates the possible values of $p(j)$ by

$$p(j) \in \{l | y_j^i = x_l^i, 1 \leq l \leq 64\}.$$

After checking each input pairs, the possible values of $p(j)$ halves, on the average. Then, after a number of pairs, it is possible to recover $p(j)$ uniquely. We have implemented this approach to recover 64-bit permutations. For 1000 randomly constructed permutations, on the average, 28 input/output pairs were enough to uniquely recover the permutation.

Example 2. Let p' be a permutation of 16 values and $\pi'(x_1, \ldots, x_{16}) = (x_{p'(1)}, x_{p'(2)}, \ldots, x_{p'(16)})$. Initially, $p'(j) \in \{1, \ldots, 16\}$, for all j. Let

$$\pi'(1,1,1,1,0,0,0,0,1,0,1,0,1,0,1,0) = (0,1,0,1,1,1,1,0,0,1,1,0,0,0,0,1).$$

Then, we conclude that $p'(1)$ can only be $\{5,6,7,8,10,12,14,16\}$, since $y_{p'(1)} = x_i$, only for $i \in \{5,6,7,8,10,12,14,16\}$. Similarly, we can say that

$$p'(3), p'(8), p'(9), p'(12), p'(13), p'(14), p'(15) \in \{5,6,7,8,10,12,14,16\}$$
$$p'(2), p'(4), p'(5), p'(6), p'(7), p'(10), p'(11), p'(16) \in \{1,2,3,4,9,11,13,15\}$$

Using the second pair

$$\pi'(0,1,1,1,0,0,1,1,0,1,0,1,0,1,1,1) = (1,0,1,1,1,0,1,0,1,0,0,0,1,1,1,1),$$

following constraints hold

$$p'(1), p'(3), p'(10), p'(14), p'(15) \in \{7,8,10,12,14,16\}$$
$$p'(2), p'(6), p'(11), p'(12) \in \{1,9,11,13\}$$
$$p'(4), p'(5), p'(8), p'(16) \in \{2,3,4,15\}$$
$$p'(9), p'(13) \in \{5,6\}$$

Using the following pairs,

$$\pi'(0,0,0,1,0,1,0,1,0,1,0,0,0,1,1,0) = (1,0,1,0,0,0,1,1,0,0,0,0,0,1,1),$$
$$\pi'(0,0,1,0,1,1,1,1,1,0,0,1,0,1,0,1) = (0,0,1,0,1,0,0,1,1,0,1,1,1,1,1,0),$$
$$\pi'(1,1,1,1,1,0,0,1,1,0,0,0,0,0,0,1) = (1,0,0,1,1,0,1,1,1,1,1,1,0,0,0,0,0),$$
$$\pi'(1,0,1,1,1,1,1,1,0,1,0,1,0,0,1,1) = (1,0,1,0,1,0,1,1,1,1,0,1,1,1,0,1),$$
$$\pi'(1,1,1,1,0,0,0,1,1,0,0,1,1,1,0,1) = (0,0,1,1,1,1,1,0,1,1,1,0,1,0,1,0),$$

we uniquely determine $p'(i)$ as:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$p'(i)$	10	4	5	7	12	8	14	3	11	1	2	13	6	15	16	9

The attacker recovers the permutation, on the average using 28 slid pairs. The attacker exhaustively searches for the rest of the key bits and recovers the key and the permutation with $2^{48}(= 2^{48} + 2^{16.11})$ time complexity and $2^{16} + 1$ chosen plaintexts and $2^{12.28}$ adaptively chosen plaintexts.

3.4 Unknown S-Box and Permutation Case

In this case, the attacker needs to identify the slid pairs P and P', without the knowledge of the S-boxes and the permutation. Similar to other cases, the attacker fixes P' to the all zero plaintext and searches for P among $2^{22.71}$ many possible candidates, denoted as P_i for $0 \leq i \leq 2^{22.71}$, as given in Sec. 3.2.

Here, the attacker is assumed to have access to eight encryption devices using the keys $\mathsf{K}, \mathsf{K} \lll 16, \mathsf{K} \lll 32, \mathsf{K} \lll 48, \mathsf{K}', \mathsf{K}' \lll 16, \mathsf{K}' \lll 32$ and $\mathsf{K}' \lll 48$.

The attacker first iteratively encrypts the all zero plaintext P and obtains the $\mathsf{Y_j}$ values as given below;

$$\mathsf{P} \xrightarrow{E_{\mathsf{K}}} \mathsf{Y}_1 \xrightarrow{E_{\mathsf{K}\lll 16}} \mathsf{Y}_2 \xrightarrow{E_{\mathsf{K}\lll 32}} \mathsf{Y}_3 \xrightarrow{E_{\mathsf{K}\lll 48}} \mathsf{Y}_4 \xrightarrow{E_{\mathsf{K}}} \mathsf{Y}_5 \xrightarrow{E_{\mathsf{K}\lll 16}} \dots$$

Similarly, for each candidate P_i, the attacker generates $X_{i,j}$ using

$$P_i \xrightarrow{E_{\mathsf{K}'}} X_{i,1} \xrightarrow{E_{\mathsf{K}'\lll 16}} X_{i,2} \xrightarrow{E_{\mathsf{K}'\lll 32}} X_{i,3} \xrightarrow{E_{\mathsf{K}'\lll 48}} X_{i,4} \xrightarrow{E_{\mathsf{K}'}} X_{i,5} \xrightarrow{E_{\mathsf{K}'\lll 16}} \dots$$

For the correct P_i, i.e. $P_i = \mathsf{P}$, the attacker gets input/output values for the last round encryption, hence the following is satisfied,

$$\mathsf{Y_j} = \pi(S(\mathsf{X}_{i,j}, R_i)) \tag{6}$$

for $j = 1, 2, \dots$. To determine the correct P, the attacker eliminates the candidates, for which $\mathsf{Y_j} = \pi(S(X_{i,j}, R_i))$ cannot be satisfied for any valid S-box and permutation selection.

For each P_i, the attacker considers the $X_{i,j}$ values in the set $\mathsf{S}_{a,0}$, where

$$\mathsf{S}_{a,0} = \{X_{i,j} | X_{i,j} = (a, *, *, \dots, *), R_i = (0, *, \dots, *)\}.$$

After applying the S-box layer to the $X_{i,j}$'s in $\mathsf{S}_{a,0}$, the first four bits of the outputs will be the same. Since the permutation π only changes the bit positions, there will be four bit positions, let's say i_1, i_2, i_3 and i_4, that will be take the same value in all Y_i's that corresponds to the $X_{i,j}$'s in the set $\mathsf{S}_{a,0}$. Let Q be the number of bit positions that take the same value for these Y_i's. For the correct P_i, Q cannot be less than 4. For an incorrect P_i, the probability that $Q \geq 4$ using n input/output pairs is

$$Pr(Q \geq 4) = \binom{64}{4}(1/16)^{n-1}.$$

Using this formula, the expected number of incorrect P_i's with $Q \geq 4$ is $2^{-24.72}$ with $n = 12$. Hence, if the attacker can construct the set $S_{a,i}$ with $n = 12$ elements, he can uniquely determine P among $2^{22.71}$ candidates. It should be noted that the attacker is not limited to the first S-box input, and can apply the technique for any of the 16 S-box inputs, and the input a can take any value from $\{0, 1, \ldots, 15\}$. To guarantee that the attacker can construct such a set, with at least 12 elements, the attacker requires approximately $2^{8.58}$(= 12x16x2) slid pairs. We experimentally verified that it is possible to determine P using $2^{8.58}$ slid pairs. Next, the attacker aims to recover the secret components.

Recovering the Secret Components. After determining P, the attacker can generate up to 2^{32} $(\mathsf{X}_i, \mathsf{Y'}_i, R_i)$ values for the last round as given in Equation (6). Then, for each bit of Y_i, the attacker finds the input S-box, using the technique described above.

Example 3. Let X_i $(i = 1, 2, 3)$ be 16 bit inputs having the first four bits equal to 1. Let Y_i's be the one round encryption of X_i's with four parallel S-boxes and a permutation.

$$X_1 = (1111\ 0010\ 1101\ 0001) \quad Y_1 = (0\mathbf{0}11\ \mathbf{1}011\ 11\mathbf{0}0\ 1\mathbf{0}01)$$
$$X_2 = (1111\ 1011\ 1001\ 1101) \quad Y_2 = (0\mathbf{0}00\ \mathbf{1}111\ 01\mathbf{0}1\ 1\mathbf{0}00)$$
$$X_3 = (1111\ 0001\ 1010\ 0000) \quad Y_3 = (1\mathbf{0}11\ \mathbf{1}100\ 10\mathbf{0}1\ 0\mathbf{0}11)$$

It is seen that the bits 2, 5, 11 and 13 are equal in all Y_i's. The attacker concludes that these bits are the outputs of the first S-box. The attacker cannot determine the output of 1111 for the first S-box, however, attacker knows that it is a permutation of the bits 0100, hence can be 1000, 0100, 0010 or 0001. Although, the attacker does not know the exact value, he knows what the value will be after the permutation, i.e., the bits 2, 5, 10 and 14 will be 0,1,0,0, respectively.

Attacker repeats this experiment, for all 16 S-boxes and uniquely determines the composition of the individual S-boxes and partial permutations. Although the attacker cannot individually determine the S-boxes, the composition of S-box with the permutation is enough to construct an equivalent cipher. $2^{8.58}$ adaptively chosen plaintexts for the related keys are enough to recover the composition of one S-box input and the partial permutation. To recover all 16 output of the S-box, $2^{12.58}$ adaptively chosen plaintexts are enough, with similar time complexity. We experimentally observed that 2^{10} slid pairs are enough recover composition of S-box and permutation.

In the last step, the attacker recovers the rest of the 48 bits, by exhaustively searching. The attack requires 2^{48} time complexity and $2^{31.29} = (2^{22.71+8.58})$ adaptively chosen plaintexts using eight related keys to recover the key and the secret components.

4 Chosen-Key Component Recovery

In the previous section, we assumed an attacker having access to encryption devices using related keys. In this section, we assume that the attacker is able

to choose the key and aim to recover the secret components. For the keys that satisfy $\mathsf{K} = (\mathsf{K} \lll 1)$, namely $(0, 0, \ldots, 0)$ or $(1, 1, \ldots, 1)$, the attacks described to recover the S-boxes and the permutations can be applied using only one key. Without loss of generality, we assume that the attacker fixes K to $(0, 0, \ldots, 0)$.

- **S-Box Recovery.** In this case, possible candidates for P take the form $(a, a, \ldots, a)$, where a can be any 4-bit value. Therefore, $N_p = 2^4$. By applying the attack described in Sec. 3.2, the attacker is able to obtain the S-box with time complexity of $2^{11} (= 2^{4+7})$.
- **Permutation Recovery.** With known S-boxes, there is only one candidate for P, namely $(a, a, \ldots, a)$, where $\mathsf{S1}(a) = 0$. Then, by applying the attack described in Sec. 3.3, the attacker determines the permutation with 56 adaptively chosen plaintexts with similar time complexity. Note that the number of required adaptively chosen plaintext is 28 for each key in Sec. 3.3.
- **S-Box and Permutation Recovery.** In this case, there are 2^4 candidates for P having the form $(a, a, \ldots, a)$, where a can be any 4-bit value. Therefore, $N_p = 2^4$. Then, the attacker constructs an equivalent cipher with $2^{8.58} \times 2$ adaptively chosen plaintext with similar time complexity as given in Sec. 3.4.

5 Improved Exhaustive Key Search

In this section, we present a method to improve exhaustive key search, by taking the advantage of a weakness in the key schedule. It is assumed that the attacker knows the S-boxes ($\mathsf{S1}$ and $\mathsf{S2}$), the permutation π and a plaintext/ciphertext pair (P_1, C_1).

The attack begins with the attacker partitioning the key space into 2^{48} classes; $\mathsf{Class\ 1}, \ldots, \mathsf{Class\ } 2^{\mathsf{48}}$, with each class containing 2^{16} keys. Let $B_i = (b_1, b_2, \ldots, b_{48})$ be the binary representation of the integer i using 48 bits, for $0 \leq i \leq 2^{48}$. $\mathsf{Class\ i}$ consists of 2^{16} keys having the following format;

$$(b_1, b_2, b_3, *, b_4, b_5, b_6, *, \ldots, b_{46}, b_{47}, b_{48}, *),$$

where $*$ can be either 0 or 1. In the generic exhaustive key search, to search for a key in a given class, 2^{16} encryptions are required. However due to the key schedule of the respective algorithm, it is possible to reuse some encryptions. The attacker partially encrypts P_1 using the first three round keys, which only requires $(b_1, \ldots, b_{48})$, and stores the result as C'. Note that there is a unique C' for each class. Then, trying all 2^{16} keys in the class, encrypts C' for 13 more rounds and compares the resulting value with the given ciphertext C_1. Using this approach, the total amount of work in terms of number of encryption is $2^{16}(13/16) + 3 \approx 2^{15.70}$. Considering all classes, the complexity is reduced by 19 percent, in compared to the generic exhaustive key search.

Similarly, the method also works to improve the exhaustive search of the 48-bit key space, when 16 bits of the key are known, as required in all the attacks presented in the paper. In that case, the complexity of the attack reduces to $2^{47.70}$ from 2^{48}.

6 Conclusion

In this paper, we analyzed the security of the lightweight encryption algorithm specified and approved in *NRS 009-6-7:2002* by Electricity Suppliers Liaison Committee to be used with tokens in prepayment electricity dispensing systems in South Africa. The algorithm aims to achieve security by secrecy of its main building blocks, namely the S-boxes and the permutations.

In the paper, we presented related-key slide attacks to recover the secret key for the four respective scenarios composed of known vs. unknown S-box and known vs. unknown permutation. We showed that it is possible to recover the key even without the knowledge of the main building blocks.

The main weakness of the algorithm is due to the cyclic key schedule, and the identical round structure. The attacks can be avoided by breaking the symmetry between the round functions. Including an additional layer that uses a counter or a constant might be an easy countermeasure for the attacks. It should also be noted that increasing the number of rounds does not increase the security of the cipher against slide attacks, on the contrary, the cipher may become weaker. For example, if the number of rounds is increased to 64, finding slid pairs would require two related keys, rather than eight, when the permutation is secret.

Acknowlegments. The author would like to thank Çağdaş Çalık for presenting the work. Also, the author would like thank Özgül Küçük, Luís Brandão and anonymous reviewers for very helpful comments that increased the quality of the paper.

References

1. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEE Proceedings / Information Security 152, 13–20 (2005)
2. Hamalainen, P., Alho, T., Hannikainen, M., Hamalainen, T.D.: Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In: Proceedings of the 9th EUROMICRO Conference on Digital System Design, DSD 2006, pp. 577–583. IEEE Computer Society, Washington, DC (2006)
3. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
5. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
6. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)

7. Knudsen, L.R., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
8. Wagner, D., Briceno, M., Goldberg, I.: A Pedagogical Implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms, `http://www.scard.org/gsm/a51.html` (accessed January 23, 2013)
9. 4C Entity. C2 Block Cipher Specification, Revision 1.0, `http://www.4centity.com/`
10. Borghoff, J., Knudsen, L.R., Leander, G., Matusiewicz, K.: Cryptanalysis of C2. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 250–266. Springer, Heidelberg (2009)
11. NRS 009-6-7:2002. Rationalized User Specification, Electricity Sales Systems, Part 6: Interface standards Section 7: Standard Transfer Specification/Credit Dispensing Unit – Electricity dispenser – Token Encoding and Data Encryption and Decryption (2002)
12. Borghoff, J., Knudsen, L.R., Leander, G., Thomsen, S.S.: Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 270–289. Springer, Heidelberg (2011)
13. Borghoff, J., Knudsen, L.R., Leander, G., Thomsen, S.S.: Slender-Set Differential Cryptanalysis. J. Cryptology 26(1), 11–38 (2013)
14. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)

Appendix

A Constructing Slid Pairs

In this part, we present a method to construct slid pairs, given the slid pair P and P' using the keys K and $\mathsf{K}' = \mathsf{K} \lll 1$ (See Fig. 2). Let the attacker has access to eight encryption devices using keys $\mathsf{K} \lll 16i$ and $\mathsf{K}' \lll 16i$ for $i = 0, 1, 2, 3$. If P and P' are slid pairs, then the ciphertexts $\mathsf{C}_1 = E_{\mathsf{K}}(\mathsf{P})$ and $\mathsf{C}'_1 = E_{\mathsf{K}'}(\mathsf{P}')$ are slid pairs using the keys $\mathsf{K} \lll 16$ and $\mathsf{K}' \lll 16$, respectively. Similarly, C_2 and C'_2 are slid pairs using the keys $\mathsf{K} \lll 32$ and $\mathsf{K}' \lll 32$. In general, C_i and C'_i are slid pairs using the keys $\mathsf{K} \lll 16(i \mod 4)$ and $\mathsf{K}' \lll 16(i \mod 4)$. To generate N slid pairs with keys K and K', the attacker make $4N$ iterative encryptions and gets the values C_i and C'_i for $i = 0 \mod 4$ as given in Figure 3. Due to the birthday bound, the attacker can generate approximately 2^{32} slid pairs using the slid pair P and P'.

B Number of Possible S-Boxes

In this part, we aim to find the number of possible S-boxes that satisfy a number of given constraints.

Proposition 1. *Let $X = (x_1, \ldots, x_n)$ be a random sample of n, with replacement, from a population of $D = \{1, 2, \ldots, N\}$. For $i \in D$, let Y_i denote the number of times i occurs in the sample,*

$$Y_i = \#\{j \in \{1, 2, \ldots, N\} : x_j = i\}.$$

$$\mathsf{P} \xrightarrow{E_{\mathsf{K}}} \mathsf{C}_1 \xrightarrow{E_{\mathsf{K}\lll 16}} \mathsf{C}_2 \xrightarrow{E_{\mathsf{K}\lll 32}} \mathsf{C}_3 \xrightarrow{E_{\mathsf{K}\lll 48}} \mathsf{C}_4 \xrightarrow{E_{\mathsf{K}}} \mathsf{C}_5 \xrightarrow{E_{\mathsf{K}\lll 16}} \ldots \rightarrow \mathsf{C}_{\mathsf{N}}$$
$$\mathsf{P}' \xrightarrow{E_{\mathsf{K}'}} \mathsf{C}'_1 \xrightarrow{E_{\mathsf{K}'\lll 16}} \mathsf{C}'_2 \xrightarrow{E_{\mathsf{K}'\lll 32}} \mathsf{C}'_3 \xrightarrow{E_{\mathsf{K}'\lll 48}} \mathsf{C}'_4 \xrightarrow{E_{\mathsf{K}'}} \mathsf{C}'_5 \xrightarrow{E_{\mathsf{K}'\lll 16}} \ldots \rightarrow \mathsf{C}'_{\mathsf{N}}$$

Fig. 3. Iterative encryption using eight related keys

The number of distinct population values in the sample is

$$V_{N,n} = \#\{j \in \{1, 2, \ldots, N\} : Y_j > 0\}.$$

The density function of $V_{N,n}$ *is*

$$Pr(V_{N,n} = j) = \binom{N}{j} \sum_{k=0}^{j} (-1)^k \binom{j}{k} (\frac{j-k}{N})^n$$

for $j = 1, \ldots, \min\{N, n\}$.

For each S-box, $N = 16$, and the number of constraints is $n = 18$. Table 3 shows the probability distribution of the number of recovered outputs of the S-boxes. The distribution takes its maximum value for 12 recovered outputs, with expected values of $\sum_{j=1}^{16} jPr(V_{16,18} = j) = 10.99 \approx 11$.

Table 3. The probability distribution of $V_{16,18}$

j	$Pr(V_{16,18} = j)$	j	$Pr(V_{16,18} = j)$
1	2^{-68}	9	$2^{-3.42}$
2	$2^{-47.09}$	10	$2^{-2.13}$
3	$2^{-34.34}$	11	$2^{-1.69}$
4	$2^{-25.20}$	12	$2^{-2.11}$
5	$2^{-18.25}$	13	$2^{-3.43}$
6	$2^{-12.85}$	14	$2^{-5.75}$
7	$2^{-8.70}$	15	$2^{-9.26}$
8	$2^{-5.59}$	16	$2^{-14.46}$

Differential Fault Attack on the PRINCE Block Cipher

Ling Song and Lei Hu

State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences,
Beijing 100093, China
{lsong,hu}@is.ac.cn

Abstract. PRINCE is a new lightweight block cipher proposed at the ASIACRYPT'2012 conference. In this paper two observations on the linear layer of the cipher are presented. Based on the observations a differential fault attack is applied to the cipher under a random nibble-level fault model, aiming to use fault injections as few as possible. The attack uniquely determines the 128-bit key of the cipher using less than 7 fault injections on average. In the case with 4 fault injections, the attack limits the size of key space to less than 2^{18}.

Keywords: lightweight cipher, PRINCE block cipher, differential fault attack.

1 Introduction

The idea of injecting faults during the execution of cryptographic algorithms to retrieve the key was first introduced by Boneh, DeMillo, and Lipton who succeeded in breaking a CRT version of RSA [4]. Later, Biham and Shamir adapted this idea to differential analysis on block ciphers and introduced the concept of Differential Fault Attack (DFA) [2]. Block ciphers implemented on smart cards and other low-end devices are vulnerable to such attacks, which exploit the links between right ciphertexts and the faulty counterparts. Typical examples are differential fault attack on AES, CLIEFIA and IDEA [15,6,7]. Usually the faults are injected by disturbing the power supply voltage, the frequency of the external clock, or by applying a laser beam, etc [1].

Recent years, many lightweight block ciphers have been proposed in the literature to provide cryptographic building blocks for resource constrained devices such as RFID tags. Among the best studied ciphers are PRESENT, KATAN, LED and PRINTCipher [3,5,9,11]. PRINCE is a novel lightweight block cipher proposed in 2012 [10], which is optimized with respect to latency when implemented in hardware. This is the first lightweight block cipher that takes latency as main priority.

In this paper, we present a differential fault attack on PRINCE under a random fault model which adds a random disturbance to a nibble of the state whose position cannot be predicted in advance. Our attack was inspired by the

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 43–54, 2013.

differential fault attacks on AES [13](especially [12,16]), since a typical round of PRINCE resembles that of AES. For AES, MDS plays an important role in its linear layer, while PRINCE only achieves an almost-MDS property. Based on the diffusion property of the linear layer of PRINCE, our attack was developed. With the knowledge of the difference distribution table of the Sbox it used, the number of survival keys using one fault injection can be evaluated statistically. However, the data complexity of finding the unique key is difficult to estimate due to the uncertainty of the diffusion property of the linear layer. Hence we obtain related results through experiments. The experiments show 4 faults are enough to break PRINCE practically and 7 fault injections uniquely determine the 128-bit key of PRINCE.

This paper is organized as follows: Section 2 briefly describes the PRINCE block cipher; in Section 3 we elaborate on our differential fault attack on PRINCE; Section 4 discusses the results; and finally, we conclude the paper in the last section.

2 Brief Description of PRINCE

PRINCE is a 64-bit block cipher with a 128-bit key. The key schedule is very simple. Specifically, the 128-bit key is split into two 64-bit parts:

$$k = k_0||k_1,$$

and extended to 192 bits by the following mapping:

$$(k_0||k_1) \rightarrow (k_0||k_0'||k_1) := (k_0||(k_0 \ggg 1) \oplus (k_0 \gg 63)||k_1).$$

During the encryption the first two subkeys k_0 and k_0' are used as whitening keys, while the third subkey k_1 is the key for a 12-round block cipher refered to as PRINCE$_{core}$. The highlevel structure of PRINCE is demonstrated in Fig. 1.

Fig. 1. The highlevel structure of PRINCE

The 12-round process of PRINCE$_{core}$ is depicted in Fig. 2. A typical round of PRINCE$_{core}$ consists of an Sbox layer, a linear layer and an addition layer. The intermediate computation result, called state is usually represented by a 64-bit vector or a 16-nibble vector.

Sbox-Layer. The cipher uses a 4-bit Sbox which is given as in Table 1. We denote the Sbox and its inverse by S and S^{-1} respectively.

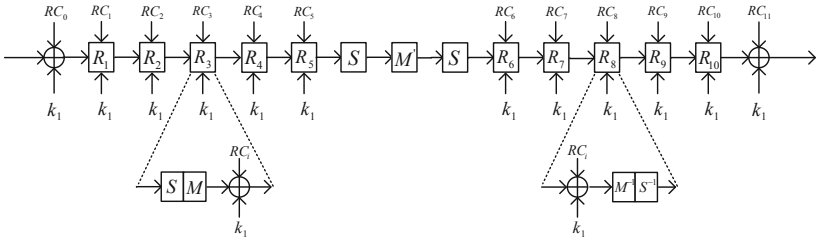


Fig. 2. PRINCE*core*

Table 1. The Sbox S of PRINCE

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4

Linear Layer. As can be seen from Fig. 2, the linear layer uses a matrix M or M' and is called M- or M'-mapping according to the matrix used. In the linear layer the 64-bit state is multiplied with M or M', both of which are 64×64 matrices and built from four 4×4 matrices. These four matrices are

$$M_0 = \begin{bmatrix} 0\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}, M_1 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}, M_2 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,0\,1 \end{bmatrix}, M_3 = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,0 \end{bmatrix}.$$

Two block matrices $\hat{M}^{(0)}$ and $\hat{M}^{(1)}$ of size 16×16 are generated as follows:

$$\hat{M}^{(0)} = \begin{bmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{bmatrix}, \hat{M}^{(1)} = \begin{bmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{bmatrix}.$$

Then, the 64×64 matrix M' is constructed as a block diagonal matrix with $\hat{M}^{(0)}, \hat{M}^{(1)}, \hat{M}^{(1)}, \hat{M}^{(0)}$ as its diagonal blocks. Note that M' is an involution matrix; that is to say, $M'M' = I$ is the identity matrix.

The M-mapping is the composition of the M'-mapping and a permutation SR, i.e. $M = SR \circ M'$. SR behaves like the AES shift rows and permutes the 16 nibbles of the state as $(a_0, a_1, \cdots, a_{15}) \rightarrow (a_0, a_5, \cdots, a_{11})$, where the subscripts are changed according to Table 2. Also, the inverse of SR is denoted by SR^{-1} for the sake of simplicity.

Addition. The 64-bit state is xored with the 64-bit subkey k_1 and a round-dependent constant as listed in Table 3. Note that the 12 round constants have a symmetric property: for all $0 \leq i \leq 11$, $RC_i \oplus RC_{11-i} = 0\text{X}c0ac29b7c97c50dd (=: \alpha)$. With this property and together with the fact that M' is an involution matrix, the encryption and decryption of the cipher have the following relationship:

Table 2. The SR operation of PRINCE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

$$D_{(k_0||k_0'||k_1)}(\cdot) = E_{(k_0'||k_0||k_1 \oplus \alpha)}(\cdot).$$

Thus, a same hardware implementation can fulfill both encryption and decryption operations of the PRINCE cipher.

Table 3. Round constants

RC_0	0000000000000000
RC_1	13198a2e03707344
RC_2	a4093822299f31d0
RC_3	082efa98ec4e6c89
RC_4	452821e638d01377
RC_5	be5466cf34e90c6c
RC_6	7ef84f78fd955cb1
RC_7	85840851f1ac43aa
RC_8	c882d32f25323c54
RC_9	64a51195e0e3610d
RC_{10}	d3b5a399ca0c2399
RC_{11}	c0ac29b7c97c50dd

More details about this cipher can be found in [10].

3 Attacking PRINCE

The sketch of our attack is inspired by recent differential fault attack against AES [12,16]. In these attacks the recovery of key bytes benefits from the optimal diffusion property of the linear layer, but in the case of PRINCE, the linear layer lacks of such optimal diffusion property. However, PRINCE's linear layer has its own properties, which can be utilized to develop a differential fault attack against PRINCE.

In this section the fault model is stated first. Then we describe our two important observations of PRINCE's linear layer. Exploiting these observations, our attack is elaborated afterwards.

3.1 Fault Model

We are dealing with random faults on a single nibble whose position cannot be predicted in advance. The effect of the introduced fault is to add an arbitrary nonzero nibble disturbance to the state. Thus, faults can be induced within

nibble-wise operations including Sbox substitution, SR operation and addition, which is beneficial to fault injections. This kind of nibble-wise fault model has also been used in [8].

Although PRINCE may not be implemented in a round-based fashion, we assume an attacker can typically predict when a particular round happens and induce a nibble fault at a specific round. Moreover, the time that certain events take place can often be determined by analyzing a suitable side channel leakage. Furthermore, we assume that an attacker can repeat the experiments with the same plaintext and key without applying external physical effects.

In the remaining part of this paper, a 16-nibble X is represented with $(X_0, X_1, \cdots, X_{15})$ and we always denote by (C, C^*) a pair of a right ciphertext C and its corresponding faulty ciphertext C^* for the same plaintext and key.

3.2 Observations

Before going to the details of our observations, we split the 16 nibbles of the state of PRINCE into four groups numbered from 1 to 4 as depicted in Fig. 3.

Fig. 3. Split the nibbles into four groups

Diffusion Property of the M'-Mapping. Set $X = (X_0, X_1, \cdots, X_{15})$ and $Y = (Y_0, Y_1, \cdots, Y_{15})$ to be the input and the corresponding output of the M'-mapping.

First, the M'-mapping diffuses the nibbles within groups. If only a certain group of X has nonzero nibbles, then only the same group of Y has nonzero nibbles. Hence the M'-mapping of the 64-bit state can be regarded as four small separate mappings M'_1, M'_2, M'_3, and M'_4, each of which diffuses the nibbles of the corresponding group.

Second, the M'-mapping achieves an almost-MDS property. If X has only one nonzero nibble, say X_2 (belongs to Group 1), Y will have at most 4 nonzero nibbles, all of which are located in the same group (Group 1). Concretely speaking, if the Hamming weight of X_2 is greater than 1, then all the four nibbles of Group 1 of Y are nonzero; otherwise, exactly three of them are nonzero.

Diffusion Property of the SR. Set $X = (X_0, X_1, \cdots, X_{15})$ and $Y = (Y_0, Y_1, \cdots, Y_{15})$ to be the input and the corresponding output of the SR operation. If X has a group of four non-zero nibbles, then Y will still have four non-zero nibbles, each of which is located in a different group. In a nutshell SR diffuses the nibbles over groups.

3.3 Principle of the Attack at the 11-th Round

Our attack is based on the induction of a nibble fault at the 10-th round under the model mentioned above. To explain the attack, first let us consider the scenario when there is a nibble disturbance at the 11-th round.

For the three operations marked in Fig. 4(a) where we want to inject a fault, the attack works the same in principle. Below we suppose faults are injected during the Sbox substitution of 11-th round.

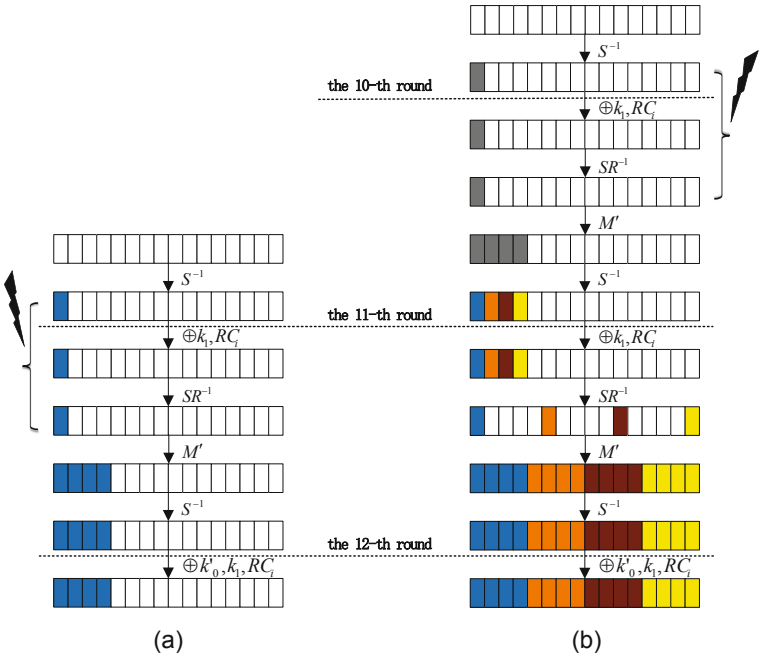


Fig. 4. Attack PRINCE with fault at 11-th round and 10-th round

Assume we get a right ciphertext C and its corresponding faulty ciphertext C^* for the same plaintext and key. The fault can happen at any position of the 16 nibbles. For the sake of simplicity, we take the first nibble as a faulty nibble and analysis for other positions are the same.

As illustrated in Fig. 4(a), the fault injected in the first nibble during the Sbox substitution of 11-th round influences the first group of the 16 nibbles of the final ciphertext due to the diffusion property of M'-mapping.

In this context, first four nibbles C_0, C_1, C_2, C_3 and $C_0^*, C_1^*, C_2^*, C_3^*$ are known, and so is the fact that the the bitwise Exclusive-or (XOR) differences of them stem from a single nibble induced by the fault.

Let us look into the first nibble. The C_0 and C_0^* are known. Given the input difference of the first Sbox Δ_0^{in}, with the knowledge of the difference distribution

table of the S^{-1} of PRINCE (see Table 4) in mind, the first nibble of $K = k'_0 \oplus k_1$ will be limited to one of 0, 2, or 4 choices by the following equation [14]:

$$S(C_0 \oplus K_0) \oplus S(C_0^* \oplus K_0) = \Delta_0^{\text{in}}.$$

Table 4. Difference distribution table of the S^{-1} used by PRINCE

Δin	Δout															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	2	0	2	0	0	0	0	0	0	2	4	2	0	0
2	0	0	0	0	2	2	2	2	2	2	0	0	0	0	2	2
3	0	0	4	0	4	2	2	0	0	2	2	0	0	0	0	0
4	0	2	0	0	2	0	0	0	4	0	2	4	0	0	0	2
5	0	0	0	2	2	2	2	0	2	0	4	0	2	0	0	0
6	0	2	0	2	0	0	2	2	0	0	0	0	2	0	4	2
7	0	0	2	0	0	2	0	0	0	0	4	2	0	2	2	2
8	0	4	2	2	2	0	0	2	2	0	2	0	0	0	0	0
9	0	2	0	2	0	2	2	0	2	2	0	0	0	4	0	0
A	0	0	0	2	2	0	0	4	0	2	0	0	2	2	0	2
B	0	2	0	2	0	2	2	0	2	0	0	2	2	0	2	0
C	0	0	0	2	0	2	0	0	0	4	0	2	2	0	2	2
D	0	0	4	0	0	2	0	2	2	2	0	0	0	2	2	0
E	0	0	2	0	0	0	4	2	0	0	0	2	2	2	0	2
F	0	0	0	2	0	0	0	2	0	2	2	2	0	2	2	2

To get information about all the first four-nibble of $K = k_0^{'} \oplus k_1$, we can guess $(\Delta_0^{\text{in}}, \Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_3^{\text{in}})$, i.e. the input difference of the first four Sboxes and then search the subkey information. Before searching, it is necessary to check whether the guesses satisfy the following two conditions which we call the M'-***Mapping Conditions***.

- Non-zero Δ_i^{in}s are valid differences that can lead to the right output differences.
- The preimage of $(\Delta_0^{\text{in}}, \Delta_1^{\text{in}}, \Delta_2^{\text{in}}, \Delta_3^{\text{in}})$ under the corresponding submapping of M'-mapping has only one nonzero nibble.

A guess cannot be called a right guess until it passes the M'-mapping Conditions. Below a right guess's four-nibble preimage under the corresponding submapping of M'-mapping is denoted by P.

Now consider the number of suggested four-nibble key values given one right four-nibble guess. Note that a pair of nonzero input/output differences of a single Sbox suggests 2 or 4 values (16/7 in average) for a subkey nibble, and zero differences suggest all possible values for a subkey nibble. As a result, one pair of input/output differences of four Sboxes suggests $2^4 \sim 2^{10}$ values for the four-nibble subkey used, where $2^{10} = 4 \cdot 4 \cdot 4 \cdot 16$ since there is at least three

nonzero differential nibbles. According to the property of the M'-mapping, the number of values suggested for a four-nibble subkey in average is

$$\frac{11}{15} \times \left(\frac{16}{7}\right)^4 + \frac{4}{15} \times \left(\frac{16}{7}\right)^3 \times 16 \approx 71.$$

Therefore, such a pair (C, C^*) reduces the size of the four-nibble subkey space from 2^{16} to $t_1 \cdot 71$, where t_1 is the number of right four-nibble guesses of the input difference of the Sbox layer.

If we want to uniquely determine the first four nibbles of $k_0^{'} \oplus k_1$, at least another fault leading to non-zero ciphertext difference at first four nibbles is needed. Hence, the whole nibbles of $k_0^{'} \oplus k_1$ can be uniquely determined with at lest $4 \times 2 = 8$ fault injections, each of which leaks some information of a group of nibbles of the 64-bit key.

After $k_0^{'} \oplus k_1$ has been recovered, the last round can be peeled off, and the attack is repeated on the reduced cipher to reveal k_1.

3.4 Attack Strategy at the 10-th Round

In this subsection our target fault attack on PRINCE is described on the basis of the principle elaborated in the previous subsection, aiming to use less faults as possible.

The attack scenario remains except that faults are injected one round earlier, as depicted in Fig. 4 (b). Set the first nibble to be the faulty nibble again (other cases work the same).

Suppose that the induced fault difference has a Hamming weight greater than 1 and the opposite case will be discussed later. As demonstrated by Fig. 4 (b), the difference keeps until it goes into the M'-mapping of the 11-th round. The M'-mapping spreads the difference to the whole group, and then the four nibble differences are changed by the S^{-1}. In order to make a distinction among the four nibble differences, we color them differently. The SR^{-1} of the 12-th round splits the four nibble differences into different groups, making each group have one and only one nonzero nibble of difference. After that, M'-mapping propagates differences within groups, resulting full difference in the ciphertext.

Given a pair (C, C^*) whose fault difference propagation follows the pattern depicted in Fig. 4(b), the analysis is sketched below.

1. For group $i, 1 \leq i \leq 4$, guess $(\Delta_{4i}^{\text{in}}, \Delta_{4i+1}^{\text{in}}, \Delta_{4i+2}^{\text{in}}, \Delta_{4i+3}^{\text{in}})$. For those that satisfy the M'-Mapping Conditions, store $(P_i, (\Delta_{4i}^{\text{in}}, \Delta_{4i+1}^{\text{in}}, \Delta_{4i+2}^{\text{in}}, \Delta_{4i+3}^{\text{in}}))$ in table T_i, where P_i is the four-nibble preimage of the corresponding guess.
2. After we get such four tables, search four-nibble subkey values using the items in Table T_i, $1 \leq i \leq 4$ as we do in Section 3.3.
3. Using the P_is in four tables $T_i, 1 \leq i \leq 4$, check whether the concatenations of $P_1||P_2||P_3||P_4$ satisfy the SR **Condition**, which is defined as the four nonzero nibbles need to gather together in a single group after the SR operation. For those concatenations that pass the SR Condition, record (P_1, P_2, P_3, P_4) in table D.

4. To get candidates of 64-bit key, concatenate the four-nibble subkey values suggested by (P_1, P_2, P_3, P_4), the items of D.
5. Inject more faults and repeat previous steps to reduce the space of the 64-bit key.

Since 71 four-nibble subkey values in average is returned by a pair of input/output differences of a group, $t_2 \cdot 71^4 = t_2 \cdot 2^{24.60}$ values of 64-bit key will be obtained with one fault injected at round 10, where t_2 is the number of items in list D. In this context, 64-bit key information are interrelated with one fault, and hence at least 2 fault are needed to recover 64-bit subkey information.

For the fault difference with Hamming weight equal to 1, less information can be obtained, since the fault difference propagates to only thress nibbles within the same group after the M'-mapping of 11-th round. The following SR^{-1} operation then scatters the three non-zero nibbles into different groups, resulting differences in only three groups of nibbles in the ciphertext. In this case, the 64-bit key space is reduced approximately to a size of $t_2 \cdot 71^3 \cdot 2^{16} = t_2 \cdot 2^{34.45}$.

Note that the induced fault difference has a Hamming weight greater than 1 with probability 11/15 and for the other case the probability is 4/15. Considering these two cases together, the number of possible values for the 64-bit key is reduced to

$$t_2 \cdot \left(\frac{11}{15} \cdot 71^4 + \frac{4}{15} \cdot 71^3 \cdot 2^{16} \right) = t_2 \cdot 2^{32.55}$$

with one fault injection.

Once the outer 64-bit subkey has been recovered, the last round can be peeled off and the inner 64-bit subkey will be retrieved in a more efficient way.

For the recovery of the inner 64-bit key k_1, the M'-Mapping Conditions can be applied to the items in D after Step 3. In other words, the difference of the states after decrypting two rounds back needs to have only one non-zero nibble[1]. In this way more wrong keys can be excluded.

For a group of four nibbles (all of them are nonzero or three of them are nonzero), it satisfies the M'-Mapping Condition with probability of

$$\frac{4 \times 15}{\frac{11}{15} \times 15^4 + \frac{4}{15} \times 15^3} = 2^{-9.31}$$

Consequently, using one fault injection the number of suggested values for the subkey k_1 is about

$$t_2 \cdot 2^{32.55} \cdot 2^{-9.31} = t_2 \cdot 2^{23.24}.$$

This is verified by the experimental results in the next section.

4 Results

In fact, random faults introduce differences with any Hamming weight. Due to the uncertainty of the fault difference and the almost-MDS diffusion property of

[1] This cannot be applied to the recovery of outer 64-bit subkey since the decryption needs subkey k_1 which is totally unknown.

M'-mapping, it is difficult to estimate accurately the number of fault injections needed to uniquely determine the 128-bit key. Through experiments on a PC, we not only verify the number of survival keys using one fault injection, but also evaluate the data complexity .

According to the experiments, t_1 ranges from 1 to 6 and has an average of 2.50, while $t_2 \in [1, 12]$ and achieves 1.88 on average. The result in Table 5, which is derived statistically from 1000 instances, shows that using 4.05 fault injections in average, the outer 64-bit key will be determined uniquely. In addition, if only 2 fault injections are used, the number of candidates for the outer 64-bit keys will be $2^{14.18}$. Similarly, the unique determination of the inner 64-bit key needs 2.56 fault injections in average and 2 fault injections reduce the size of the 64-bit key space to 2^3, as shown in Table 6. Note that $2^{34.76}$ and $2^{25.32}$, which are the experimental number of suggested values for the 64-bit subkeys using one fault injection, are very close to our estimates.

The time complexity mainly lies in the computation of Step 2 in the algorithm of Section 3.4. Since the computations for each group are processed separately and parallelly, the time complexity is very small, which is around 2^{16}.

Table 5. Recovery of outer 64-bit key of PRINCE

#faults	1	2	3	4	4.05
#survival	$2^{34.76}$	$2^{14.18}$	28.94	3.53	1

Table 6. Recovery of inner 64-bit key of PRINCE

#faults	1	2	2.56
#survival	$2^{25.32}$	8.03	1

5 Conclusions and Future Work

In this paper we broke PRINCE with differential fault attack under the random fault model which adds a nibble of disturbance to the state of the cipher. Experiments showed that the 128-bit key can be uniquely determined using 6.61 fault injections. Also, the key space can be reduced to a size of $2^{17.18}$ with 4 fault injections. To our knowledge, this is the first differential fault attack on PRINCE.

To prevent our attack, the designers and manufacturers need to guarantee that our fault model is impossible. Nowadays hardware and software countermeasures that can be used to counteract fault attacks had already been widely defined and successfully deployed, but any kind of them is costly for such a low-latency lightweight block cipher. In future work, some efficient countermeasures will be explored on one hand, and on the other we will mount differential fault attack against PRINCE under more practical models.

Acknowledgement. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The work of this paper was supported by the National Key Basic Research Program of China (2013CB834203), the National Natural Science Foundation of China (Grants 61070172 and 10990011), the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702, and the State Key Laboratory of Information Security, Chinese Academy of Sciences.

References

1. Bar-el, H., Choukri, H., Naccache, D., Tunstal, M., Whelan, C.: The Sorcerers Apprentice Guide to Fault Attacks. Proceedings of the IEEE 94(2), 370–382 (2006)
2. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
5. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
6. Chen, H., Wu, W., Feng, D.: Differential Fault Analysis on CLEFIA. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 284–295. Springer, Heidelberg (2007)
7. Clavier, C., Gierlichs, B., Verbauwhede, I.: Fault Analysis Study of IDEA. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 274–287. Springer, Heidelberg (2008)
8. Esmaeili Salehani, Y.: Side Channel Attacks on Symmetric Key Primitives. Master's degree thesis, Concordia University (2011)
9. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED Block Cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
10. Julia, B., Anne, C., et al.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
11. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
12. Mukhopadhyay, D.: An Improved Fault Based Attack of the Advanced Encryption Standard. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 421–434. Springer, Heidelberg (2009)
13. National Institute of Standards and Technology (NIST). Advanced Encryption standard. FIPS Publication 197 (2001), `http://www.itl.nist.gov/fipspubs/`

14. Nyberg, K.: Differentially Uniform Mappings for Cryptography. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 55–64. Springer, Heidelberg (1994)
15. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
16. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 224–233. Springer, Heidelberg (2011)

Multidimensional Meet-in-the-Middle Attacks on Reduced-Round TWINE-128

Özkan Boztaş[1,2], Ferhat Karakoç[1,3], and Mustafa Çoban[1,4]

[1] Tübitak BILGEM UEKAE, 41470, Gebze, Kocaeli, Turkey
{ozkan.boztas,ferhat.karakoc,mustafa.coban}@tubitak.gov.tr
[2] Middle East Technical University, Institue of Applied Mathematics, Cryptography Department, Ankara, Turkey
[3] Istanbul Technical University, Computer Engineering Department, 34469, Maslak, Istanbul, Turkey
[4] Sakarya University, Mathematics Department, Sakarya, Turkey

Abstract. TWINE is a lightweight block cipher designed for multiple platforms and was proposed at Selected Areas in Cryptography, 2012. The number of rounds of TWINE is 36 and the most powerful attack given by the designers is the impossible differential attack against 24 rounds of TWINE-128 whose time complexity is $2^{115.10}$ encryptions and data complexity is $2^{52.21}$ blocks. The best attack known so far is the biclique attack on the full round cipher with a time complexity of $2^{126.82}$ and data complexity of 2^{60}. However the time complexity of biclique attack is near exhaustive search and data needed for the attack is near the whole codebook.

In this paper we propose some meet-in-the-middle type attacks on reduced round TWINE-128. We show that meet-in-the-middle type attacks can be applied on more rounds than the best attack done by the designers while they claim that the first 5 rounds contain all the key bits for TWINE-128.

Our attacks are due to the slow diffusion of both the cipher and the key schedule algorithm. One of our attacks just use 2^{12} chosen plaintext-ciphertext pair with time complexity of 2^{124} to break 21 rounds of the algorithm. Also we propose another attack on 25 rounds of the cipher by using 2^{48} chosen plaintext-ciphertext pairs with the time complexity of 2^{122} and memory complexity of 2^{125} while the best attack proposed by the designers is for 24 rounds.

Keywords: TWINE, cryptanalysis, lightweight block cipher, Multidimensional Meet-in-the-Middle attack.

1 Introduction

Lightweight cryptographic algorithms rise for the need of the reduction of power, energy and area requirements in resource constraint platforms such as RFID tags. Some of the lightweight algorithms proposed recently are HIGHT [10], PRESENT [4], KATAN/KTANTAN [5], PRINTcipher [12], KLEIN [8], LED [9], Piccolo [14], and TWINE [15].

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 55–67, 2013.

TWINE supports two key lengths, 80 and 128 bits. The encryption functions are the same for both key lengths but the key schedules are different. We name the ciphers due to their key lengths TWINE-80 and TWINE-128. The authors didn't evaluate the cipher against meet-in-the-middle type attacks and they claim that it is difficult to mount such attacks on the full round since the first 3 (5) rounds contain all the key bits for TWINE-80 (TWINE-128). The most powerful attack given by the designers is the impossible differential attack against 24 rounds for TWINE-128 which time complexity is $2^{115.10}$ encryptions and data complexity is $2^{52.21}$ blocks. [15]. The best attack known so far is the biclique attack on the full round cipher with a time complexity of $2^{126.82}$ and data complexity of 2^{60} [6]. However the time complexity of the biclique attack is near exhaustive search and data needed for the attack is near the whole codebook.

Here we propose some attacks by using some meet-in-the-middle techniques. . First we propose a 19-round classical meet-in-the-middle attack whose data complexity is just 2 known plaintext-ciphertext pair and time complexity is 2^{124}. Then we propose a 21-round splice-and-cut [1] attack which needs 2^8 chosen plaintext-ciphertext pair and whose time complexity is 2^{124}. By using the Multidimensional Meet-in-the-Middle technique [17] we improve the attack on 25-rounds whose time complexity and $2^{125}$and use 2^{48} chosen plaintext-ciphertext pairs.

This paper is structured on 7 sections. The introduction section is followed by Section 2 which provides a short description of the TWINE algorithm. Section 3 provides an overview of the meet-in-the-middle techniques. In section 4, we present a 19 round basic meet-in-the-middle attack. Section 5 presents a way of choosing different subkeys from the key schedule for better attacks and gives an 21-round meet-in-the-middle attack. Section 6 presents 25-round Multidimensional Meet-in-the-Middle attack. Section 7 is devoted to the conclusion of the paper.

2 Notation and a Short Description of TWINE

2.1 Notation

The notations used through the paper are as follows:

A	: a bit string
$A(i)$	: i^{th} nibble of A. The left most nibble is $A(0)$.
$A(i, j, ..., k)$	: concatenation of i, j, ... , k^{th} nibbles of A.
$A \lll i$	: i-bit cyclic shift of A.
$A\|\|B$	: concatenation of A and B.
K_i	: 32 bit subkey value used in the i^{th} round
K^i	: 128 bit key value calculated in the i^{th} round of the key schedule.

2.2 TWINE

TWINE is a block cipher supporting two key lengths, 80 and 128 bits. The global structure of TWINE algorithm is a variant of Type 2 generalized Feistel

structure [16] with 16 4-bit sub-blocks. Each version of the algorithm consists of 36 rounds and has the same round function depicted in Figure 1.

In the round function, the key addition is applied before the S-box operation as seen in the figure and then the permutation is performed. As an example for the permutation layer, 0^{th} nibble before the permutation goes to 5^{th} position after the permutation as seen in the figure.

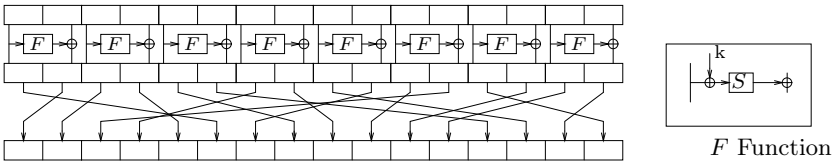


Fig. 1. i-th round of TWINE

The key schedules of both versions of TWINE consist of S-box operation, round constant addition and permutation operations.

TWINE-128 has the following key schedule which generates 36 32-bit round keys from the 128-bit master key.

1. $K^0 = K$
2. $K_1 = K^0(2, 3, 12, 15, 17, 18, 28, 31)$
3. for i=1,2,...,35 do the followings
4. (a) $K^i = K^{i-1}$
   (b) $K^i(1) = K^i(1) \oplus S[K^i(0)]$
   (c) $K^i(4) = K^i(4) \oplus S[K^i(16)]$
   (d) $K^i(23) = K^i(23) \oplus S[K^i(30)]$
   (e) $K^i(7) = K^i(7) \oplus (0||CON^i_H)$
   (f) $K^i(19) = K^i(19) \oplus (0||CON^i_L)$
   (g) $K^i(0, 1, 2, 3) = K^i(0, 1, 2, 3) \lll 4$
   (h) $K^i = K^i \lll 16$
   (i) $K_{i+1} = K^i(2, 3, 12, 15, 17, 18, 28, 31)$

We will represent the input to the i^{th} round function by S^i where $1 \leq i \leq 36$. Since the analysis is independent from the S-box and constants, herewith it is not considered necessary to go into further details. The key schedule for the 80 bit version is similar and will not be given here. For a complete description of the algorithm, one can refer to [15].

3 An Overview of the Meet-in-the-Middle Type Attacks

Meet-in-the-Middle attack was first introduced by Diffie-Hellman to attack double encryption of DES [7]. Although it has been more than 30 years since the first attack was introduced it received more attention when the method was used to attack hash functions in 2008. Many Meet-in-the-Middle techniques including

splice-and-cut, initial structure, partial matching and biclique techniques were developed and the latter led to the preimage attack on 50 rounds SHA-512 and the first attack on the reduced-round SKEIN hash function [11]. The biclique technique was also introduced to cryptanalyze block ciphers which gave the first attack on the full AES-128/-192/-256 [3].

Meet-in-the-Middle attacks depend on the fundamental idea that if the algorithms can be decomposed into two consecutive parts and computation of each part only involves partial information of the secret key then we can investigate the security level of each part separately and finally check if the results from different parts match. Since evaluating two small parts requires low computational complexity, the overall complexity to analyze the complete algorithm could decrease dramatically. The attack is illustrated in Figure 2. Suppose our cipher $E(k, P)$ can be decomposed into two consecutive parts $E_f(k_1, .)$ and $E_b(k_2, .)$ where k_1 and k_2 are subkeys used in E_f and E_b and f and b stand for forward and backward phases. The detailed steps of the attack are as follows.

1. For each guess of the common key bits of k_1 and k_2
   (a) Compute all possible values of $v = E_f(k_1, P)$ for all possible key bits. Collect v in a set T.
   (b) Compute $v' = E_b^{-1}(k_2, C)$ for all values of k_2 and check whether $v' \in T$. If so output the corresponding key pair (k_1, k_2) as a possible key.

The time complexity for (Step 1-a) in terms of complete encryptions/decryptions is $\frac{2^{|k_1|}.r_f}{r}$. Similarly the time complexity of (Step 1-b) is $\frac{2^{|k_2|}.r_b}{r}$. Note that r_f and r_b represent the round numbers from forwards and backwards. Total number of required plaintext-ciphertext pair is $\frac{|k_1+k_2|}{|v|}$.

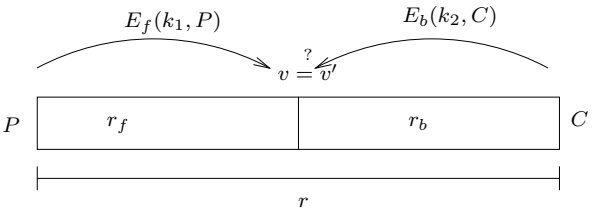


Fig. 2. General Meet-in-the-Middle Attack

Splice-and-Cut technique is first proposed by Aoki and Sasaki for performing Meet-in-the-Middle attacks to compute pre-images on the hash functions SHA-0 and SHA-1 [1]. In this technique we are not restricted only to inputs and outputs of the cipher. Suppose we choose an arbitrary value X for the intermediate state as in Figure 3. We can do partial decryptions to get the plaintext input of the cipher i.e $P = E_{b_2}^{-1}(k_1, X)$. As long as we have the access to decryption and encryption oracles we can perform chosen plaintext and ciphertext attacks on target ciphers. Notice that the original Meet-in-the-Middle attacks only require known plaintext and ciphertext pairs.

The data complexities of Splice-and-Cut attacks depend on minimum number of key bits we need to partially encrypt or decrypt X to get ciphertext C or plaintext P. The data complexities will increase but the overall time complexities may decrease since we can further divide ciphers into smaller parts.

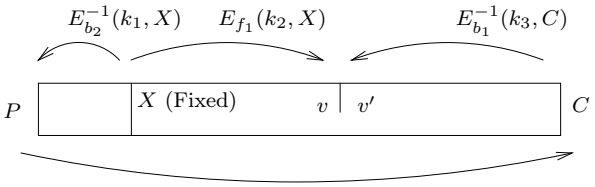


Fig. 3. Splice-and-Cut Attack

Another recent development of the meet-in-the-middle techniques is the Multidimensional Meet-in-the-Middle cryptanalysis [17]. In this method ciphers are first divided into consecutive sub-ciphers by guesses of certain intermediate states then meet-in-the-middle attacks are applied to these sub-ciphers separately. Finally, results of the attacks are combined to find correct keys.

Suppose we first guess an intermediate state g, and perform two meet-in-the-middle attacks on the sub-ciphers divided by g, as shown in Figure 4. The steps of the attack can be described as follows.

1. Compute $v_1 = E_{f_1}(k_1, P)$ for each possible k_1 , and gather all k_1 's into a set T_1 indexed by v_1 and k_{c_1} , where k_{c_1} is the common key of both k_1 and k_2 .
2. Compute $v_2' = E_{b_2}^{-1}(k_4, C)$ for each possible k_4, and put all k_4 's to a set T_2 indexed by v_2' and the common key k_{c_2} of k_3 and k_4 .
3. For each possible guess of g:
   (a) Compute $v_1' = E_{b_1}^{-1}(k_2, g)$ for each possible k_2, and use v_1' and k_{c_1} to find matched entries of k_1 in T_1. Then put matched (k_1, k_2) into a set Q.
   (b) Compute $v_2 = E_{f_2}(k_3, g)$ for each possible k_3, and then find matched entries of (k_3, k_4) in T_2. Check whether (k_3, k_4) are also matched entries of Q. If so, do brute-force testing on the matched key tuple (k_1, k_2, k_3, k_4) by using additional plaintext-ciphertext pairs. If all of these tests pass, output the tuple as the correct key.

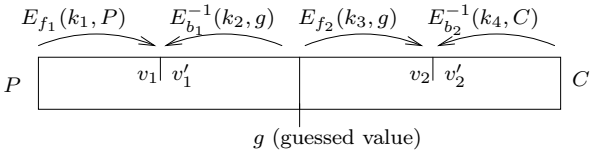


Fig. 4. Multidimensional Meet-in-the-Middle Attack

Time complexity of the attack is $2^{|k_1|} + 2^{|k_4|} + 2^{|g|}(2^{|k_2|+|k_3|})$. It consumes more memory compared to original meet-in-the-middle attacks since we need to save three sets T_1, T_2 and Q.

4 Basic Meet-in-the-Middle Attack on TWINE-128

In this section we present a basic meet-in-the-middle attack on 19 rounds TWINE-128 by using the slow diffusion of the key schedule.

4.1 An Analysis of the Key Schedule

In the first round of TWINE-128 we know that only 8 nibbles of the main key, namely $K^0(2, 3, 12, 15,$
$17, 18, 28, 31)$ are used. In the second round if we write the subkey nibbles in terms of K^0 again, only 8 nibbles of K^0, namely $K^0(0, 1, 6, 7, 16, 19, 21, 22)$, are used. Running the complete key schedule shows that it is not until the 29^{th} round that all the bits of the main key are used in the F function. See Figure 5 for the complete diffusion of the main key bits. Rows represent rounds and columns stand for K^0 nibbles. The black boxes represent the nibbles of the main key K^0 that are used in that round i.e the black nibble in position $(3, 5)$ means that 3^{rd} nibble of the main key K^0 is used in 5^{th} round. Using this, we can mount a 19 round meet-in-the-middle attack on the cipher.

4.2 19-Round Meet-in-the-Middle Attack

Consider the reduced round variant of TWINE-128 which consists of 19 rounds with the same key schedule. By using Figure 5 we see that $K^0(13)$ is not used in the first four rounds and $K^0(26)$ is not used in the rounds from 17 to 19 round TWINE-128. Without guessing $K^0(13)$, we can still calculate some state bits up to 6 rounds later, which is seen in Figure 6. The gray nibbles show the values that we cannot calculate. Note that $K^0(13)$ is used in rounds 5,9 and 10 and we don't need to guess this subkey to calculate the check point $S^{11}(4)$. From the decryption direction, similarly we can calculate the same check point $S^{11}(4)$ without guessing $K^0(26)$ which is used in 11^{th} and 16^{th} rounds.

Now we can mount a 19-round meet-in-the-middle attack on TWINE-128. We have $|k_1| = |k_2| = 124$ bits and $|v| = 4$ bits. Moreover k_1 and k_2 shares 120 common bits. Then we guess these bits first and then do meet-in-the-middle attack for the remaining bits to reduce the memory complexity. Thus the memory complexity is just $M = 2^4$ bits and time complexity is $T = 2 \times \frac{2^{124} \times 10}{19} \approx 2^{124}$. For the two plaintext - ciphertext pair, we have $2^{120}(2^4 \times 2^4 \times 2^{-8}) = 2^{120}$ remaining key bits which can be checked by exhaustive search.

5 Choosing Different K^i for Better Attacks

TWINE-128 key schedule is completely reversible. So we can choose any K^i for $i = 0, \ldots, 35$ to attack, because getting the value of the main key is easy by

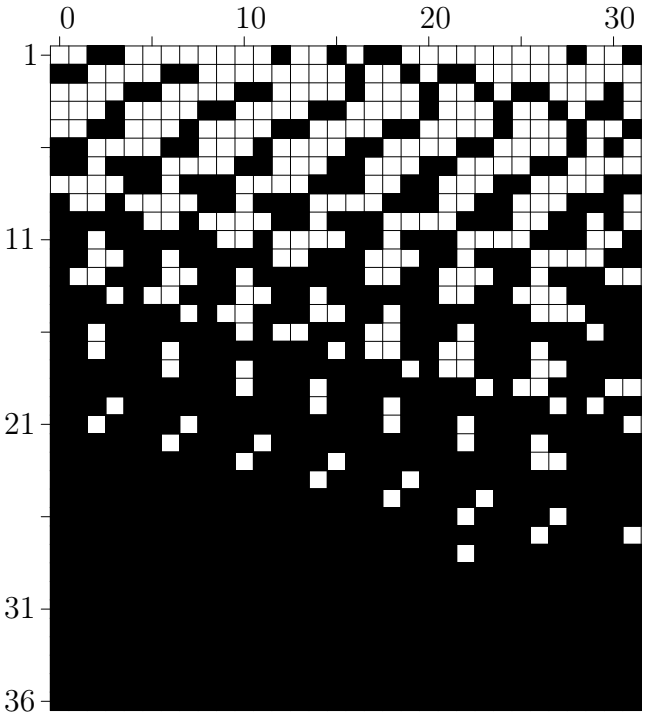


Fig. 5. Diffusion of the main key bits through the key schedule. Rows represent rounds and columns stand for K^0 nibbles.

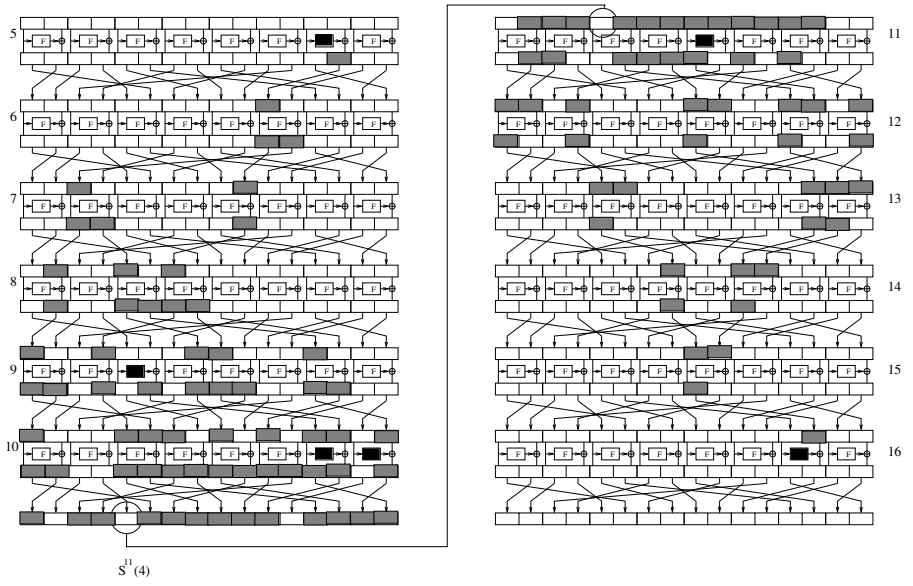


Fig. 6. 19 round Meet-in-the-Middle attack

backward computation. This can be advantageous since running the key schedule algorithm simultaneously for both forwards and backwards has a slower diffusion than running the key schedule in forwarding direction only. As an example if we choose K^8 and write the subkey bits in terms of K^8 the picture in Figure 5 now becomes like in Figure 7. The black boxes show the nibbles of K^8 that are used in that round. It seems that instead of guessing the main key bits, it is better to guess K^i for $i > 0$. The slow diffusion of the key schedule seems better now. Note that only the last round subkey uses all the nibbles of K^8.

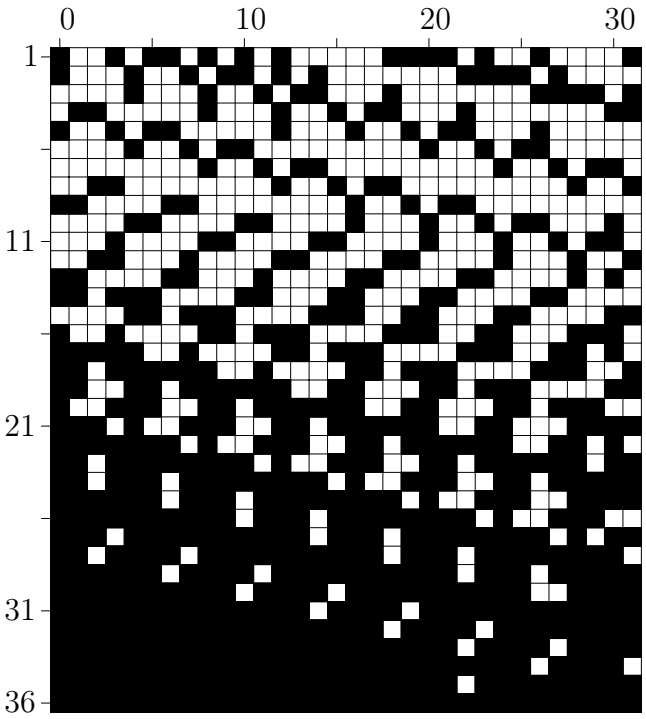


Fig. 7. Diffusion of the nibbles of K^8

5.1 21 Round Splice and Cut Attack by Guessing K^{10}

Now we will mount a 21 round splice and cut attack by guessing K^{10}. We see that $K^{10}(16)$ is not used between rounds $(3-7)$. For the decryption direction, $K^{10}(14)$ is not used in the last four rounds of 21-round TWINE-128. The check point is $S^{13}(5)$. Note that $K^{10}(16)$ and $K^{10}(14)$ are used in the second round as seen in Figure 8. So we can't choose the entire state (3^{rd} round input) as our intermediate state because we do not want to guess $K^{10}(14)$ in the decryption direction. We use a technique similar to the *ladder switch* presented in [2]. We choose the intermediate values in a ladder-type structure so that decryption without guessing $K^{10}(14)$ as well as encryption without guessing $K^{10}(16)$ are

both possible. So we choose a fixed value for $S^3(6)$ and $S^3(13)$. In order to decrypt this we also need the values $S^2(2)$ and $S^2(3)$ to be fixed. Now $S^1(7)$ and $S^1(10)$ can take all the possible values and the remaining plaintext nibbles can be fixed. Notice that we don't need to guess the value of $K^{10}(14)$ in the decryption of chosen intermediate values and $K^{10}(16)$ in the encryption process. For the encryption of X we need to guess 124 bits of the key and for the decryption of X we again need 124 bits to reach the check point $S^{13}(5)$. We have 120 common key bits. The attack is performed as follows.

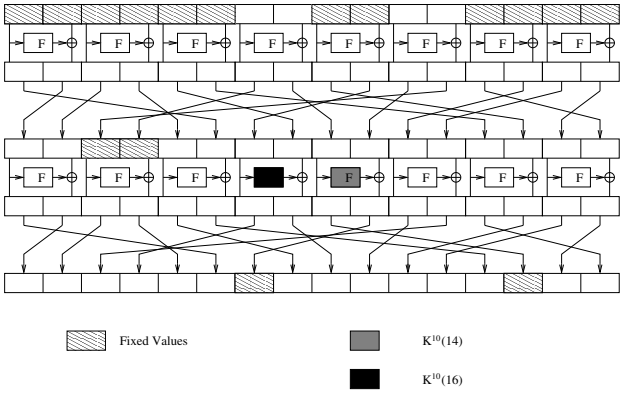


Fig. 8. Chosing X for the 21 round Splice-and-Cut attack

1. Choose a fixed value for $X = (S^2(2,3), S^3(6,13), S^1(0,1,2,3,4,5,8,9,12,13,14,15))$.
2. For each guess of the common 120 key bits,
   (a) Encrypt this fixed value and calculate the check point $S^{13}(5)$ and prepare a table consisting of the values of $S^{13}(5)$ and the key bits.
   (b) Decrypt this fixed value to get the corresponding plaintext.
   (c) Get the corresponding ciphertext value and continue the decryption until reaching the check point $S^{13}(5)$.
   (d) Check for a match in the table.

We have $|k_1| = 124$, $|k_2| = 124$ and $|v| = 4$. Moreover we have 120 common key bits. Thus the memory complexity is just $M = 2^4$ bits and time complexity is $T = 2^{120} \times \left(2^4 \frac{13}{21} + 2^4 \frac{8}{21}\right) \approx 2^{124}$. For the data complexity $S^1(7)$,$S^1(10)$ and $S^1(11)$ can take any values according to key guess and the other nibbles of the plaintext are fixed. So we need $D = 2^{12}$ chosen plaintext-ciphertext pair.

6 Multidimensional Meet-in-the-Middle Attack on 25 Round TWINE-128

Consider K^8 and notice that $K^8(1)$ is not used between the rounds (5-8) and $K^8(22)$ is not used in the last four rounds of 25-round TWINE-128. But there

is no check point here that can be efficiently calculable without guessing these subkeys $K^8(22)$ and $K^8(1)$. So the classical meet-in-the-middle attacks do not work here. Instead we will try to guess some state bits of the 15^{th} and 16^{th} round and do two meet-in-the-middle attacks for the rounds $1-15$ and $15-25$. As seen in Figure 9, $K^8(1)$ is used in the 4^{th} round and $K^8(22)$ is used in 2^{nd} and 4^{th} rounds. Therefore we try to fix the values in a clever way as in the previous section to avoid using these key values. Choose the constant value X such as

$$X = S^5(6,13), S^4(2), S^3(6,12,14), S^2(0,2,6,7,8,10,12), S^1(2,6,14)$$

Note that given the value of X we can decrypt it to get the plaintext values without guessing $K^8(22)$ and encrypt X to calculate the check points without guessing $K^8(1)$. Here is the explanation of the attack.

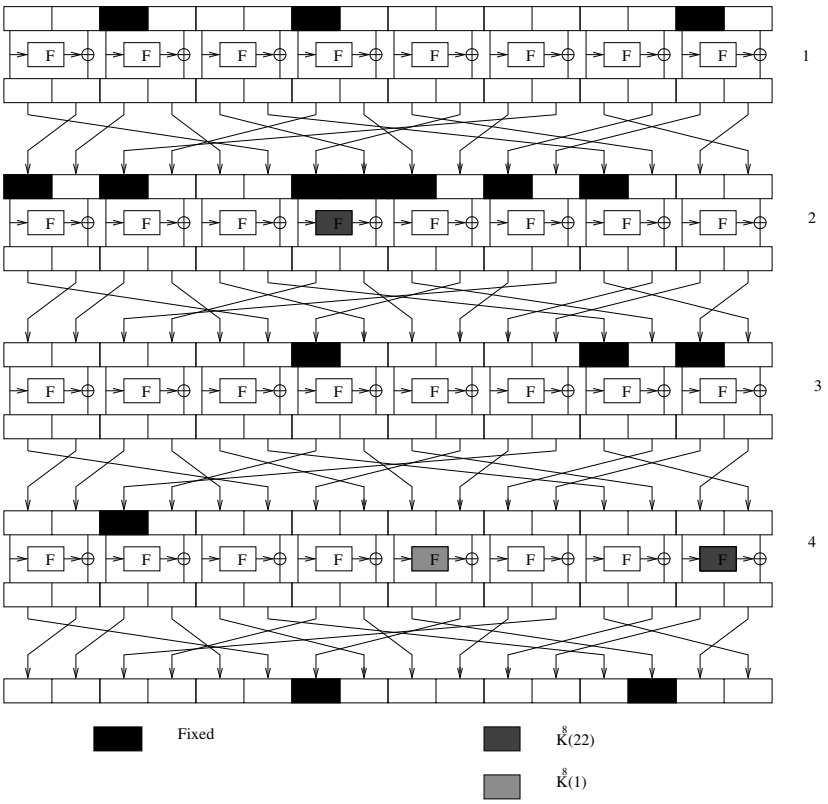


Fig. 9. Chosing the value X for 25 round Multidimensional Splice and Cut attack on TWINE-128

1. Using X compute $S^{13}(4,5,6,9,10,11,13,14,15)$ as the first check point and construct a table T_1, consisting of these values and key bits indexed by S^{13} values.
2. Decrypt X without guessing $K^8(22)$ and get the plaintext values. Get the corresponding ciphertext values and continue to decryption to get the value of the 18^{th} round's partial state bits $S^{18}(2,4,5,6,7,10,11,14)$ as the second check point. Construct a table T_2 in the same manner above.
3. Choose a fixed value of $g = S^{15}(0,1,\ldots,7), S^{16}(2,6,9,10,11,13,14,15)$.
   (a) Do a meet-in-the-middle attack between X and g and write the possible keys $(k_1,k_2)$to the table Q.
   (b) Do a meet-in-the-middle attack between C and g and check that the possible keys (k_3,k_4) are also in Q. If this is true then output (k_1,k_2,k_3,k_4) as a potential key. Try these values using another $P-C$ pair.
   (c) If no keys found, change the value of g.

For the first meet-in-the-middle attack between rounds $(1-15)$ notice that we only need to guess 60 key bits to reach the first check point as seen in Figure 10. Then $|k_1| = 124$ and $|k_2| = 60$ and $|v_1| = 36$. Using the bits of k_1 we can directly find 48 bits of k_2. So at the end of the first attack we have $2^{124+60-36-48} = 2^{100}$ (k_1,k_2) pairs. In the second meet-in-the-middle attack which is between C and the second check point we have $|k_3| = 64$, $|k_4| = 124$ and $|v_2| = 32$ bits as seen in Figure 11. Using the bits of k_4, 48 bits of k_3 can be computed. Therefore at the end of the second attack we have $2^{124+60-32-48} = 2^{104}$ (k_3,k_4) pairs. k_1 and k_4 shares 120 common bits. This gives an additional 120 bits filtering and finally we have $2^{100+104-120} = 2^{84}$ (k_1,k_2,k_3,k_4) key tuple. These key bits can checked trivially.

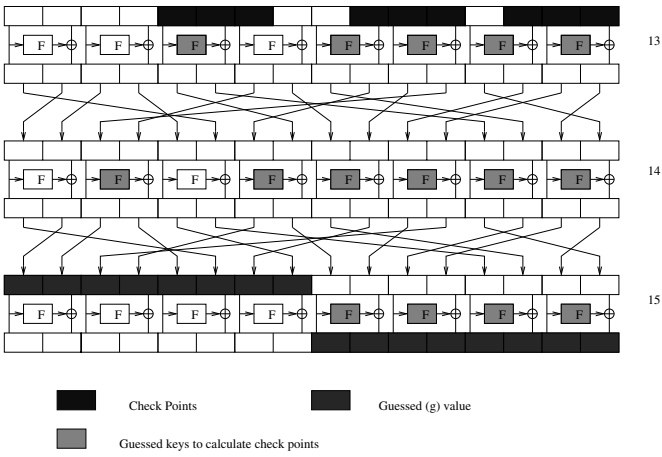


Fig. 10. Calculating the first checkpoint from g

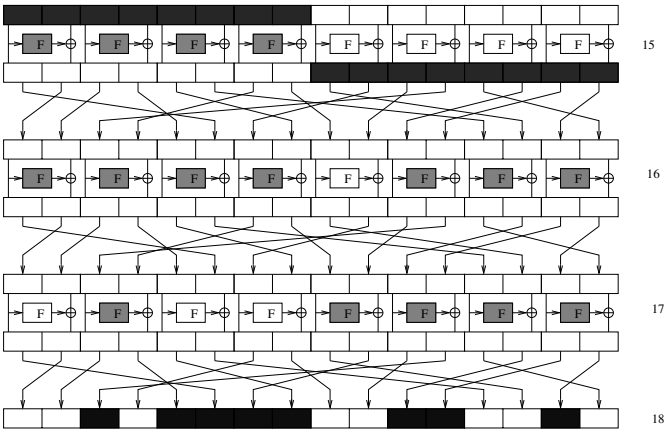


Fig. 11. Calculating the second checkpoint from g

For the memory complexity, we have two tables in step 2 and 3 both having the same size 2^{124}. For the time complexity, in step $4-a$ we are required to guess 60 key bits for all the values of g for approximately 2.5 round decryption. Then the time complexity of the first attack is $\frac{2^{64+60}\times 2.5}{25} \approx 2^{121}$. For the step $4-b$ we have to guess 64 key bits but notice that the values of $S^{16}(9)$ can be fixed. Again there is a 2.5 round encryption therefore the time complexity of the second attacks is also $\frac{2^{64+60}\times 2.5}{25} \approx 2^{121}$. Totally the time complexity is $2 \times 2^{121} = 2^{122}$ encryptions. The data complexity is $D = 2^{48}$ chosen plaintext-ciphertext pairs.

7 Conclusion

In this paper, we gave meet-in-the-middle type attacks on reduced round TWINE-128 algorithm. The attacks use the slow diffusion of both the cipher and the key schedule. The time complexity of the 25-round attack is approximately 2^{122} encryptions, memory complexity is around 2^{125} and 2^{48} chosen plaintext-ciphertext pairs are required. The most powerful attack given by the designers is the impossible differential attack against 24 rounds for TWINE-128 which time complexity is $2^{115.10}$ encryptions and data complexity is $2^{52.21}$ blocks [15].

References

1. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
2. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
3. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique Cryptanalysis of the Full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)

4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
5. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
6. Çoban, M., Karakoç, F., Boztaş, Ö.: Biclique Cryptanalysis of TWINE. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 43–55. Springer, Heidelberg (2012)
7. Diffie, M.E., Hellman, W.: Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10(6), 77–84 (1977)
8. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A New Family of Lightweight Block Ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
9. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: Preneel, Takagi (eds.) [13], pp. 326–341
10. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
11. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family. IACR Cryptology ePrint Archive 2011:286 (2011)
12. Knudsen, L.R., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: A Block Cipher for IC-Printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
13. Preneel, B., Takagi, T. (eds.): CHES 2011. LNCS, vol. 6917. Springer, Heidelberg (2011)
14. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: Preneel, Takagi (eds.) [13], pp. 342–357
15. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A Lightweight Block Cipher for Multiple Platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013)
16. Zheng, Y., Matsumoto, T., Imai, H.: On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 461–480. Springer, Heidelberg (1990)
17. Zhu, B., Gong, G.: Multidimensional Meet-in-the-Middle Attack and Its Applications to KATAN32/48/64. IACR Cryptology ePrint Archive, 2011:619 (2011)

An Implementation of the Hash-Chain Signature Scheme for Wireless Sensor Networks

Nadia Mourier, Reinhard Stampp, and Falko Strenzke

FlexSecure GmbH, Darmstadt, Germany
nadia.mourier@gmail.com,
reinhard.stampp@cased.de,
strenzke@flexsecure.de

Abstract. In this work we present the first implementation of the Dahmen-Krauß Hash-Chain Signature Scheme (DKSS) for short messages on a Wireless Sensor Node. We point out one error in the originally proposed scheme concerning the specification of the employed pseudo-random number generator and provide a corrected specification. We also give a new time-memory trade-off between signing time and private key size. We present performance results for various message-lengths, which are a parameter of this scheme, for two different choices for the block cipher used to build the one-way functions employed in this scheme, allowing comparisons with previous implementations of other public-key signature schemes. Furthermore, we describe our implementation of a synchronization protocol, needed in practical applications of this scheme whenever a node lacks the predecessor of a signature.

Keywords: public key signature scheme, implementation, wireless sensor nodes, embedded systems, hash-based signature scheme.

1 Introduction

A Wireless Sensor Network (WSN) is composed of a large number of sensor nodes and one or more base stations. The base station is usually a more powerful platform which connects the sensor nodes to the operator (for instance via the internet). The sensor nodes are composed of sensors, data processing and communication components [5]. They can be used for health, military or security applications, for instance they can monitor a given area and determine the type, concentration, and location of pollutants [3]. Such nodes are characterized by restricted power supplies, small memory sizes and limited energy reserve.

Realizing security in such networks poses different challenges than those of traditional wireless ad hoc networks. Indeed, knowing that the energy cost of receiving or transmitting a single bit of information is approximately the same as that of the CPU for the execution of up to a thousand cycles [15], and that a sensor nodes' battery lifetime is an important cost factor, the cryptographic algorithms and protocols must be adapted to this situation.

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 68–80, 2013.

A further consequence of the limited battery lifetime is that lengthy cryptographic computations should be avoided. Accordingly, in this application context, symmetric cryptography is still favoured over public key cryptography, as the former in general incurs a low computational cost compared to the latter.

The disadvantage of employing symmetric cryptography for a WSN lies in the physical insecurity of the sensor nodes: once an attacker gets physical control of a node he can extract the secret keys. One widely used approach to limit the damage that is done in such a scenario, is to have only two neigbouring nodes share a secret key for encryption and authentication. This approach bears basically two problems: the first is the difficulty of the key distribution as it is either done prior to the node deployment, then the network layout must be planned in advance or it is done after the deployment and thus vulnerable to man-in-the-middle attacks. Secondly, it has a big disadvantage, when it comes to the authentication of broadcast messages, i.e. messages sent from the base station to all or at least a large number of nodes: each node must then re-crypt respectively re-authenticate the message with its shared secret keys for the neighbouring nodes on the route. To avoid these problems of the key distribution public key cryptography can be used [9][18]. For instance Elliptic Curve Cryptography has been applied in such networks [4][19][14].

However, against the background of the in general great computational complexity of public key schemes, Dahmen and Krauß present a public key signature scheme [6] that is tailored to the application in WSNs, specifically to the authentication of broadcast messages. We refer to this scheme as the Dahmen-Krauß Signature Scheme (DKSS). In this application, the rather powerful base station platform generates the signatures and the nodes only have to verify them with the base station's public key. The most important traits of the DKSS scheme are:

- the signature size and the computational cost depend on the length of the messages that are signed. The scheme is only efficient very short messages: in [6] the estimated performance of DKSS verification is superior to ECDSA for message lengths of up to 22 bit.
- A verifier needs the $(i-1)$th signature to verify the ith signature.
- The total number of signatures that can be generated with one key pair is limited and influences the private key size and signing time.
- verification cost is much smaller than the signing cost

An important feature of this signature scheme is that it offers almost equally short signatures as ECDSA: for a security level of 80 bit and 16 bit messages, the signature length of DKSS is 336 bit and that of ECDSA is 320 bit [6].

The remainder of this paper is organized as follows: In Section 2, we introduce the DKSS, and give a brief description of the hardware and software of the chosen sensor node platform. Next, in Section 3, we present corrections and improvements we made to the originally proposed DKSS [6]. In Section 4, we give the performance results of our implementation under various parameter choices. Finally we give a conclusion of our work in Section 5.

2 Preliminaries

In this section we describe the DKSS scheme and the hardware and software platforms we used in our experiments.

2.1 The Short Hash Based Signature

We now introduce the DKSS in terms of key generation, signature generation and verification procedures. The scheme uses a variant of the Winternitz one-time signature scheme [7]. In contrast to the Merkle scheme [16], which uses a hash tree construction to build an l-time signature scheme based on the one-time signature scheme, the DKSS employs hash chains. This brings the advantage of shorter signatures, as the authentication path, that is part of Merkle-type signatures, is not needed. On the other hand, the disadvantage, which can however be tolerated in the WSN application scenario, is that a verifier needs to receive all signatures, as the preceding signature is always necessary for a verification.

Our description of the scheme deviates from the original description in [6] in one aspect, which is the specification of the PRNG, which cannot be realized as proposed by the authors. This is explained in detail in Section 3.1.

The DKSS is parameterized by three integers:

- As already mentioned, the number of signatures that can be generated by a single key pair is limited. This number is l.
- The bit length of messages that can be signed is given by $w = 2x \geqslant 2$ for some integer x.
- The security level is given by n.

Moreover for the following, let $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ and $g : \{0,1\}^{4n} \longrightarrow \{0,1\}^n$ be hash functions. For our implementation, we chose the functions f, g and PRNG in accordance to the proposals given in [6]:

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n \qquad g(x_1, x_2, x_3, x_4) = \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n$$
$$\text{with } k_3 = E_{k_2}(x_3) \oplus x_3$$
$$k_2 = E_{k_1}(x_2) \oplus x_2$$
$$k_1 = E_{IV}(x_1) \oplus x_1$$

Where $E_k(m)$ denotes the AES encryption of the message m with the key k, $\oplus$ denotes bitwise XOR, $\lfloor x \rfloor_n$ represents the truncation of the bit string x to n bits and IV is a fixed public value, for instance the zero string. Before applying the AES encryption, the message m is padded from the left with $128 - n$ zeroes.

Furthermore, let PRNG $:\{0,1\}^{128} \times \{0,1\}^{16} \times \{0,1\}^{16} \longrightarrow \{0,1\}^n$ be a pseudo random number generator which maps a fixed AES key and two variable numbers to an n bits pseudo random number, that means : $\mathrm{PRNG}(\mathrm{index}_1, \mathrm{index}_2) = \mathrm{rand}$. Our implementation of the PRNG function is :

$$\mathrm{PRNG}(\mathrm{index}_1, \mathrm{index}_2) = E_{K_{\mathrm{PRNG}}}\ (0 \ldots 0 \| \mathrm{index}_1 \| \mathrm{index}_2),$$

where K_{PRNG} is a secret pseudorandomly created AES key that is part of the private key.

The Key Pair Generation. Algorithm 1 shows how to generate one DKSS key pair with the final link z_l.

Algorithm 1. The DKSS Key Pair Generation

Require: the maximum number of signatures l, the Winternitz parameter w, and the security level n

1: // set the final link:
2: $z_l \leftarrow n$-bit random string
3: // Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
4: **for** $i = 1$ to l **do**
5: $\quad x_i[0] \leftarrow \text{PRNG}(i, 0)$
6: $\quad x_i[1] \leftarrow \text{PRNG}(i, 1)$
7: $\quad x_i[2] \leftarrow \text{PRNG}(i, 2)$
8: **end for**
9: // Calculate the one-time verification keys $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
10: **for** $i = l$ to 1 **do**
11: $\quad y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
12: $\quad y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
13: $\quad y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
14: $\quad z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$
15: **end for**
16: PubKey $\longleftarrow z_0$
17: PrivKey $\longleftarrow z_1$, K_{PRNG}
18: **return** (PubKey, PrivKey)

In Algorithm 1, the symbol $\parallel$ denotes the concatenation of bit strings, $f^k(x)$ means that the function f is applied k times to x, and $x_i[j]$ denotes the j-th entry of the array x_i. With those notations the first public key is the initial link z_0 and the initial private key is K_{PRNG} and the link z_1. The latter value is subject to change as signatures are produced by the key, i.e. if q signatures have been produced, the private key will contain z_{q+1}.

Furthermore, for efficient storage and computation, a hash chain traversal algorithm must be used for the on-demand production of the signature and verification keys. Concerning the choice of this algorithm, our implementation follows the proposal of the DKSS paper [6] which specifies the traversal algorithm presented in [23]. This means that the state of the traversal algorithm must also be part of the private key. Details about the memory demands of this algorithm are given in Section 3.1. Figure 1 gives a visualization of the hash-chain generation.

The Signature Generation. To generate the signature of the message $m = m_1 \| m_2 \in \{0, \ldots, 2^{\frac{w}{2}} - 1\}^2$ with the current link z_i, the signer uses Algorithm 2. At the end, he calculates the next link z_{i+1} of the hash chain with the above mentioned hash chain traversal algorithm. This requires to compute up to $\lceil \frac{1}{2} \log_2 l \rceil$ links per chain.

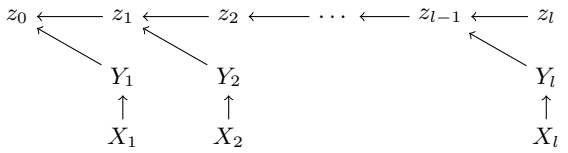


Fig. 1. Visualization of the generation of a hash chain during the key generation. The arrows denote the direction of the computation, i.e. application of the functions f and g.

Algorithm 2. The DKSS Signature Generation

Require: $m = m_1 \| m_2 \in \{0, \ldots, 2^{\frac{w}{2}} - 1\}^2$ the message to sign, $z_i \in \{0,1\}^n$ the current link

1: // Compute the one-time signature key $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
2: $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
3: $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
4: $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
5: // Calculate the checksum of the message m:
6: $c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$
7: // Compute the one-time signature $(\alpha_1, \alpha_2, \alpha_3)$:
8: $\alpha_1 \longleftarrow f^{m_1}(x_i[0])$
9: $\alpha_2 \longleftarrow f^{m_2}(x_i[1])$
10: $\alpha_3 \longleftarrow f^{c}(x_i[2])$
11: // Compute the next link z_{i+1} for the next signature and the updated state s_{i+1} of the traversal algorithm:
12: $(z_{i+1}, s_{i+1}) \longleftarrow \mathrm{hash\_traversal}(z_i,\ s_i)$
13: **return** The signature $\sigma = (i, \alpha_1, \alpha_2, \alpha_3, z_i)$

The Signature Verification. To verify a signature $\sigma = (i, \alpha_1, \alpha_2, \alpha_3, z_i)$, the verifier needs to know the last link z_{1-1} given in the previous signature, and to run Algorithm 3.

2.2 The Wireless Sensor Node Platform

As the hardware platform for our implementation, we chose the Tmote Sky platform, featuring a Texas Instruments MSP 430 16-bit microcontroller with 10 KB of RAM and 48 KB of flash memory, running at 8 MHz.

As operating system, we chose to use the Contiki OS [1]. This operating system is conceived for small memory microcontrollers, for instance the typical amount of RAM and ROM are 2 respectively 40 kilobytes. The Contiki OS is open source, highly portable, and can handle multi-tasking operations. Moreover, Contiki was the first operating system for sensor networks to supply TCP/IP communication or a cross-layer network simulator (Cooja) [8].

For the Contiki OS, there are three different simulation environments available; the MSPSim emulator [2], the Cooja cross-layer network simulator [11], and the NetSim process-level simulator [10], for software development and debugging. It also provides a file system, Coffee, which we use to store private

Algorithm 3. The DKSS Signature Verification

Require: $m = m_1 \| m_2 \in \{0, \ldots, 2^{\frac{w}{2}} - 1\}^2$ the message to verify, $\sigma = (i, \alpha_1, \alpha_2, \alpha_3, z_i)$ the signature, $z_{i-1} \in \{0,1\}^n$ the previous link

1: // Compute the one-time verification key $Y_i = (\beta_1, \beta_2, \beta_3) \in \{0,1\}^{(n,3)}$:
2: $c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$
3: $\beta_1 \longleftarrow f^{2^{\frac{w}{2}} - 1 - m_1}(\alpha_1)$
4: $\beta_2 \longleftarrow f^{2^{\frac{w}{2}} - 1 - m_2}(\alpha_2)$
5: $\beta_3 \longleftarrow f^{2^{\frac{w}{2}+1} - 2 - c}(\alpha_3)$
6: **if** $g(\beta_1 \| \beta_2 \| \beta_3 \| z_i) = z_{i-1}$ **then**
7: // The signature is accepted and the verifier stores the link z_i
8: **return** TRUE
9: **else**
10: **return** FALSE
11: **end if**

and public keys on the node. Application programs for Contiki are written in C programming language. We obtained our results with MSPSim, but also verified them partially on an actual Tmote Sky platform.

3 Modifications to the Originally Proposed DKSS Scheme

There are two important differences between the original paper [6] and our implementation. The first one is the correction of the PRNG function and the second one is the use of multiple chains in the hash chain traversal algorithm.

3.1 Correction of an Error in the PRNG Specification

The authors of [6] suggest that the PRNG function updates its seed ψ each time it is called. The PRNG is then defined by :

$$\text{PRNG} : \{0,1\}^n \longrightarrow \{0,1\}^n \times \{0,1\}^n$$

$$\text{PRNG}(\psi) = (\text{rand}, \text{seed}_{\text{in}}) = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n), \tag{1}$$

where rand is pseudorandom output and seed_{in} is the updated seed. This PRNG can produce outputs "rand" only "in one direction", it is not possible to produce previous outputs with the current seed.

The error of this approach has to do with the internal workings of the employed hash chain traversal algorithm [23]. This algorithm enables the on-demand creation of the l links z_i with a storage of only $\lceil 1/2 \log l \rceil$ so called pebbles, each consisting of a single z_i, and a computational cost scaling as $\lceil \log l \rceil$. Since each z_i is computed from a hash function with inputs z_{i+1} and Y_{i+1}, in this traversal algorithm, the pebbles can only be moved left in the spacial sense specified

by Figure 1. This would mean for the above specified PRNG, that it produces the X_i, which have to be created on demand for the computation of the Y_i, in descending order of i. However, for the signature generation, the X_i must be created in ascending order. In [6] the PRNG is specified so that it allows only for this direction, making the hash traversal impossible. This contradiction makes it obvious that the scheme can only be realized with a PRNG which is capable of creating the X_i in arbitrary order.

The PRNG proposed in [6] had the aim of achieving forward security, i.e., when an attacker captures a node and retrieves the private key, he still cannot forge signatures with a lower index than given by the current state of the private key, but only ones with a higher index. However, the forward security of the PRNG according to (1) was not very strong anyway, as it lost only a single bit of information from the seed in each call. Clearly, with the necessary changes forward security cannot be achieved at all.

3.2 Multiple Chains

In order to allow for a time-memory trade-off for the signature generation, we decided to allow the use of multiple chains. So for the same number of signatures l, we can use t chains of length $r = l/t$ for some integer t which divides l, instead of a single chain of length l. The algorithm will be faster since there is no need to process all chains, but only the one currently used.

Figure 2 shows how multiple chains are created and used with the one-time signature and verification keys. According to [23] the hash chain traversal algorithm has a cost of $\lceil \frac{1}{2} \log r \rceil$ in computation and $\lceil \log r \rceil$ in memory for each chain of length r. When r is a power of two, the rounding operator can be left out from these expressions.

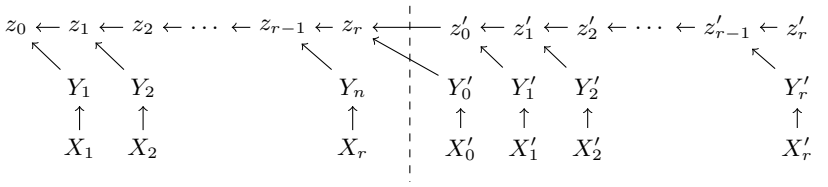


Fig. 2. Visualization of two hash chains

Using multiple chains of length $r = 2^x$ for some integer x modifies the algorithm cost: for two chains of length $\frac{l}{2}$ the computational cost becomes $\frac{1}{2} \log_2 \left(\frac{l}{2}\right) = \frac{1}{2} (\log_2(l) - 1) < \frac{1}{2} \log_2(l)$ and the memory requirement becomes $2 \log_2 \left(\frac{l}{2}\right) = 2 \log_2(l) - 2 > \log_2(l)$. In the general case, for t chains of length $\frac{l}{t}$ the computational cost is $\frac{1}{2} (\log_2(l) - \log_2(t)) < \frac{1}{2} \log_2(l)$ and the memory requirement becomes $t (\log_2(l) - \log_2(t)) > \log_2(l)$. Thus the use of several chains increases the memory cost but decreases the computational cost.

4 Implementation of the DKSS

In this section we describe our implementation of the DKSS scheme and give various performance results. We use $n = 80$ as security parameter throughout this section.

4.1 AES and XXTEA as Block Ciphers

In the original DKSS proposal [6], AES is suggested as the block cipher to be employed in the functions f, g and PRNG. However, in this work we also consider another block cipher: XXTEA. It is the correction of the original block cipher Tiny Encryption Algorithm (TEA). TEA was designed by David Wheeler and Roger Needham of the Cambridge Computer Laboratory and first presented in 1994 [21]. This algorithm is known for its simplicity of decryption and encryption. And this simplicity also involves some weaknesses: TEA is susceptible to a related-key attack. Thus, as the first correction, in 1997 TEA was modified to XTEA [20]. However, it turned out that a round reduced version of the algorithm is still susceptible to an impossible differential cryptanalysis [17]. Finally, in 1998 XXTEA was presented in an unpublished technical report [20]. These three algorithms are 64-bit block Feistel ciphers with a 128-bit key. XXTEA has minimal block size of 64-bit but offers the possibility to use larger block sizes that are multiples of 32 bits. XXTEA is still simple to implement (a few lines of code) but also vulnerable to a chosen-plaintext attack given in [22].

According to [12] XXTEA is a good cipher for WSN in terms of efficiency and performance. Since the memory demands of XXTEA are considerably smaller than those of an AES implementation, XXTEA may be the preferable choice in a memory-constrained application context.

Our AES implementation features only decryption and makes use of two 256 byte tables: one for the S-Boxes and the other for the MixColumn operation. The XXTEA implementation makes uses a 96-bit block size, as then the needed output size is $n = 80$ bit.

4.2 Execution Times and Memory Demands

Comparison of AES and XXTEA. Figure 3 gives the code sizes of the DKSS implementations with AES and XXTEA. Note that only the encryption is implemented, not the decryption. For comparison, also the code size of the ECDSA implementation given in [13] is shown. Obviously, with AES, the code sizes are approximately equal, for XXTEA, DKSS achieves smaller size than ECDSA. Table 2 shows the running times of signature generation and verification on the TMote Sky platform of DKSS for two values of the message size parameter w. XXTEA is slower by factor 1.4 for both operations.

Multiple Chains. We determined experimentally the improvement of the execution time achieved by the employment of multiple chains in the traversal algorithm as described in Section 3.2. We used the example parameter set $w = 14$

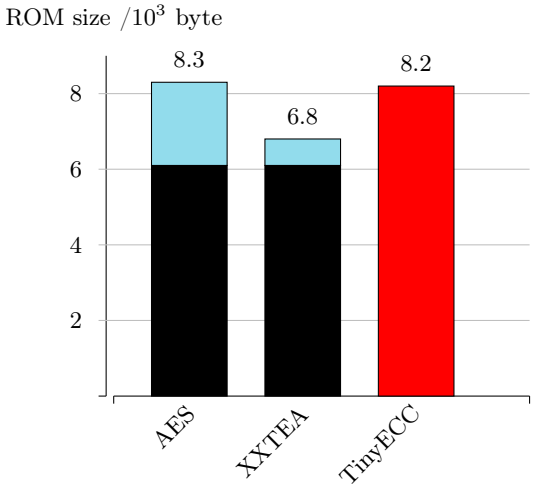


Fig. 3. Code size of the different implementation of the DKSS signature. The black area is the code size of the DKSS scheme without the block cipher. The ECDSA code size is taken from [13].

and $l = 1024$ and two different values for t, the number of chains, specifically $t = 1$ and $t = 8$. Since the verification time is not dependent on t, we only give the signature generation timings and the private key size; in fact the employment of multiple chains is not affecting the verifier in any way. The results are given in Table 1. Since the gain in speed is only approximately 10% the employment of $t > 1$ is mainly useful to benefit from otherwise unused memory, since the increase of the memory demand in this example is by a factor of about 5.3. For all performance results given in the following sections we always imply $t = 1$.

Table 1. Signature generation time and key size for the DKSS scheme depending on the number of chains for parameters $w = 14$ and $l = 1024$

	$t = 1$ chain with $r = 1024$	$t = 8$ chains with $r = 128$
Sign. Gen. time	5045 ms	4611 ms
Private Key	174 Bytes	924 Bytes

Dependency on the Message Length. Figure 4 shows running time of the signature generation for parameters $l = 1024$, $t = 1$ and varying values of w. For comparison, we included the signature generation time for ECDSA on the TMote Sky platform given in [14] in the figure. Obviously, the DKSS signature generation is only faster than ECDSA for messages shorter than 10 bits. This is a little lower than the expectation given in [6], which specifies a message size of 14 bits for this. However, there the authors assumed an optimized assembler implementation of AES on an 8-bit platform.

Table 2. Running time for the signature generation and verification of a single signature on a Tmote Sky in seconds

		Tmote Sky node	
Message bit length w		8 bits	16 bits
XXTEA	Sign	0.383	5.770
	Verify	0.098	1.394
AES	Sign	0.279	4.135
	Verify	0.071	1.007

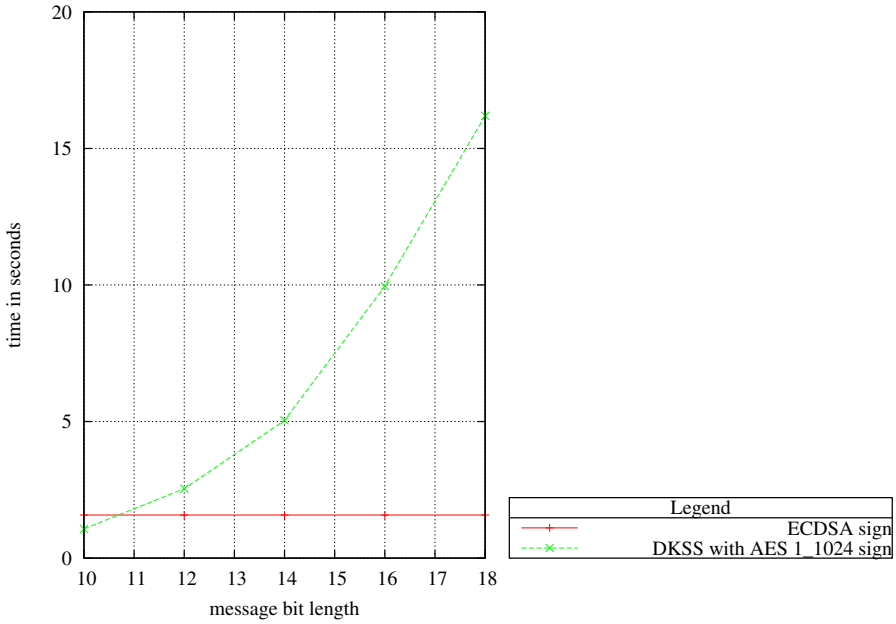


Fig. 4. Comparison of the signature generation times between the DKSS with $l = 1024$ and ECDSA (taken from [14]) running times on a Tmote Sky

The timings for the signature verification with AES and XXTEA are given in Figure 5. For AES, the largest message size for which DKSS is faster than ECDSA is 18, for XXTEA it is 16. These results are also worse than the expectation given in [6], which is 22 bit for AES.

4.3 Re-synchronization Protocol

In a real WSN, it may happen that a single node misses out on a signed message because of temporary reception problems. Then, in the DKSS scheme, this node would be incapable of verifying further signatures. To solve this problem we created a re-synchronization protocol as proposed in [6] where the node asks its neighbours to send the missing signatures. By this way the node can verify each new signature and at the end knows the last one sent by the base station.

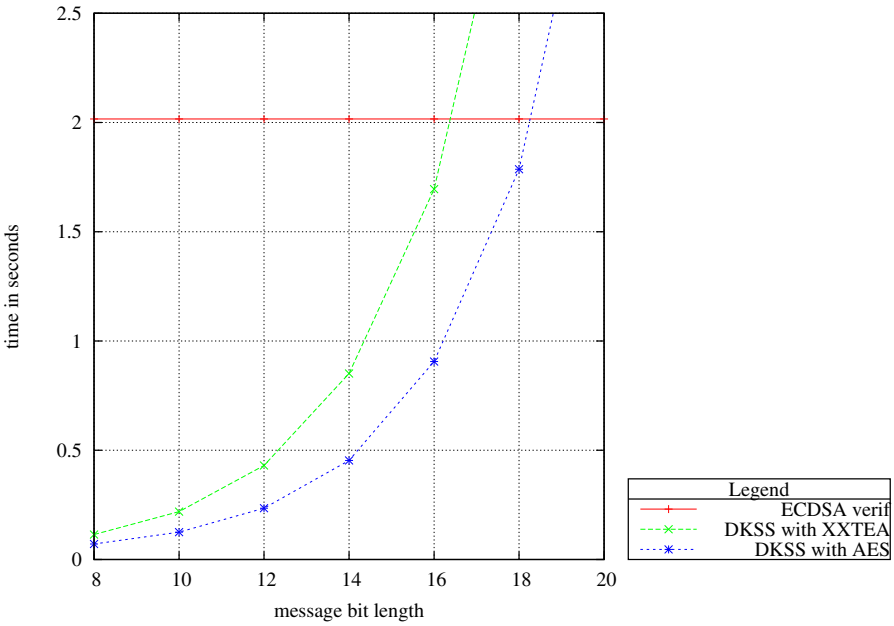


Fig. 5. Comparison of the signature verification time for DKSS with $l = 1024$ and ECDSA (taken from [14]) running times on a Tmote Sky

Since sensor nodes have a limited memory space they only save a few number of signatures and the base station, which is a platform with greater memory capacity, saves all of them. Thus if the node misses more signatures than its neighbours can store, the re-synchronization is made with the base station. In this latter case, it is more useful for the base station to provide the current link z_i directly, than sending all the missing signatures, as this unnecessarily incurs a significant energy consumption for a large number of nodes. To avoid an attacker being able to provide a false link z_i in this case, a means of authentication is needed, this can be done with a Message Authentication Code, i.e. each node must share an individual symmetric key with the base station. The code sizes given in Figure 3 already include the implementation of the synchronization protocol.

5 Conclusion

In this work we have given the first implementation of the DKSS scheme. On the theoretical part, we have provided a correction of the erroneous specification of the PRNG given in the DKSS publication [6]. We gave experimental results for a variety of parameters. Our timing results are worse than those expected by the authors of the original DKSS paper. However, there, the authors refer to an optimized assembler implementation of AES. The comparison with a previously published ECDSA implementation shows that the DKSS signature verification,

which is the more important operation in the broadcast scenario, is faster for messages of up to 18 bit size. This message size can be regarded as applicable for size-optimized broadcast message formats. Note that the broadcast protocol might very well allow for the splitting of longer messages into parts of bit size w, since the DKSS already implies an implicit message ordering. As long as incidents where broadcast messages span multiple DKSS messages are rare enough and the majority of broadcast messages requires only a single message of bit size w, the scheme remains efficient.

Furthermore, we gave a new time-memory trade-off for the scheme by employing multiple chains instead of a single one. The benefits in reduced timings is small, but may be useful on platforms where sufficient memory is available.

Concerning the code size, the DKSS has a certain advantage over ECDSA, which is enhanced if encryption has to be performed on the node as well: in this case, the cipher implementation has to be added to the ECDSA code, whereas in the case of DKSS at least encryption is already implemented.

However, one very important feature of this scheme is that its speed basically solely depends on the speed of the AES computation. This means that on a platform featuring AES hardware support, this scheme will outperform ECDSA also for larger message sizes. Note that hardware support for modular arithmetic, as it would be needed to speed up ECDSA (or the RSA signature schemes), are even less common on embedded platforms than AES coprocessors.

Acknowledgements. The authors would like to thank Erik Dahmen for general support and specifically for discussions concerning the correction of the PRNG error.

References

1. Contiki home page, `http://www.contiki-os.org/`
2. Mspsim home page, `http://sourceforge.net/apps/mediawiki/mspsim/index.php?title=Main_Page`
3. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38, 393–422 (2002)
4. Batina, L., Mentens, N., Sakiyama, K., Preneel, B., Verbauwhede, I.: Low-cost elliptic curve cryptography for wireless sensor networks. In: Buttyán, L., Gligor, V.D., Westhoff, D. (eds.) ESAS 2006. LNCS, vol. 4357, pp. 6–17. Springer, Heidelberg (2006)
5. Croce, S., Marcelloni, F., Vecchio, M.: Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. Comput. J. 51(2), 227–239 (2008)
6. Dahmen, E., Krauß, C.: Short hash-based signatures for wireless sensor networks. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 463–476. Springer, Heidelberg (2009)
7. Dods, C., Smart, N.P., Stam, M.: Hash based digital signature schemes. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 96–115. Springer, Heidelberg (2005)

8. Osterlind, A.D.F.: Contiki programming course: Hands-on session notes. Swedish Institute of Computer Science, Siena (2009)
9. Gaubatz, G., Kaps, J.-P., Ozturk, E., Sunar, B.: State of the art in ultra-low power public key cryptography for wireless sensor networks. In: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOMW 2005, pp. 146–150. IEEE Computer Society, Washington, DC (2005)
10. Netsim homepage, `http://tetcos.com/`
11. Cooja homepage, `http://www.sics.se/contiki/wiki/index.php/Cooja`
12. Jinwala, D., Patel, D., Dasgupta, K.: Optimizing the block cipher and modes of operations overhead at the link layer security framework in the wireless sensor networks. In: Sekar, R., Pujari, A.K. (eds.) ICISS 2008. LNCS, vol. 5352, pp. 258–272. Springer, Heidelberg (2008)
13. Lee, J., Kapitanova, K., Son, S.H.: The price of security in wireless sensor networks. Comput. Netw. 54(17), 2967–2978 (2010)
14. Liu, A., Ning, P.: Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks
15. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD 2003, pp. 491–502. ACM, New York (2003)
16. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
17. Moon, D., Hwang, K., Lee, W., Lee, S., Lim, J.: Impossible differential cryptanalysis of reduced round XTEA and TEA. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 49–60. Springer, Heidelberg (2002)
18. Piotrowski, K., Langendoerfer, P., Peter, S.: How public key cryptography influences wireless sensor node lifetime. In: Proceedings of the Fourth ACM Workshop on Security of ad hoc and Sensor Networks, SASN 2006, pp. 169–176. ACM, New York (2006)
19. Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.: NanoECC: Testing the limits of elliptic curve cryptography in sensor networks. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 305–320. Springer, Heidelberg (2008)
20. Wheeler, D., Needham, R.: Tea extensions (1997) (unpublished)
21. Wheeler, D., Needham, R.: Tea, a tiny encryption algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)
22. Yarrkov, E.: Cryptanalysis of xxtea. Cryptology ePrint Archive, Report 2010/254 (2010), `http://eprint.iacr.org/`
23. Yum, D.H., Seo, J.W., Eom, S., Lee, P.J.: Single-layer fractal hash chain traversal with almost optimal complexity. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 325–339. Springer, Heidelberg (2009)

An Adaptive Security Architecture for Location Privacy Sensitive Sensor Network Applications

Jiří Kůr and Vashek Matyáš

Masaryk University, Brno, Czech Republic
{xkur,matyas}@fi.muni.cz

Abstract. Security and location privacy in wireless sensor networks has drawn a lot of attention in recent years. Yet, the existing solutions are not likely to bring an optimal security/cost trade-off for real networks. They target a single attacker model and static application scenario requirements. However, both the attacker model and the requirements may evolve during the network lifetime. We propose an adaptive security architecture that allows for a dynamic adjustment of a security/cost trade-off. It consists of five security levels, each targeting a different attacker model. The security levels deploy both existing and novel security and privacy mechanisms. We also introduce a novel lightweight scheme for the dynamic security level interchange. The proposed architecture targets a particular application scenario where both traditional security and location privacy are concerned.

1 Introduction

A wireless sensor network (WSN) consists of tiny and cheap sensor nodes that monitor a physical phenomenon and report the measurements to a base station. Since information collected by a WSN is often sensitive, a certain level of security and privacy protection may be required. In this work, we propose an efficient adaptive security architecture for a given application scenario. Our proposal stems from the application security and privacy requirements. Beside traditional security services, it also supports location privacy of monitored subjects. The critical aspect of our architecture stems from the fact that it assumes multiple attacker models that alternate in time. This enables a dynamic optimization of the security/cost trade-off and brings energy savings.

It is a common practice to assume the strongest yet realistic attacker model when designing a security architecture. However, this approach may not be the best practice for highly constrained wireless sensor networks. The strongest attacker is usually present only for a certain fraction of network lifetime. For the rest of the lifetime, the network is threatened by weaker attackers. Thus it is not necessary to constantly protect the network against the strongest attacker. Furthermore, such constant protection may not be even feasible as it may consume too much energy to be active for the whole network lifetime. In this work, we assume that the attacker model changes during the network lifetime. The proposed architecture respects such changes and enables dynamically adjusting

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 81–96, 2013.

the security level to cover just the actual attacker model. This helps to save resources while keeping security at an adequate level. Our approach assumes that a network owner or the network itself is able to determine or at least estimate the actual attacker model. E.g., an intelligence service may issue a temporal warning of a higher attack risk due to a VIP person visit. Thus the network owner may expect a stronger attacker model. A network intrusion detection system (IDS) may detect an active attack and shift from a passive attacker level to an active one. Sometimes, even a network itself may sense an intruder [14] and adjust the actual attacker model. Our approach brings benefits only if the resources saved by the approach exceed the extra resources spent for adjusting the security level and deciding when to adjust the level. Thus it is beneficial especially in situations where there are big differences in costs of different security levels and/or where the cost of the approach application is low.

For our application scenario, we consider five different attacker models that result from our previous work. Thus the architecture provides five security levels that can be dynamically interchanged during the network lifetime. Each proposed security level targets a different attacker model and provides a different security/cost trade-off. The security levels build on both existing and novel security and privacy mechanisms. We do not aim to make the security levels indistinguishable for an attacker, because such levels would minimize the multi-level approach benefits. Furthermore, an internal attacker could reveal the current security level settings either. We also introduce a lightweight scheme that enables a dynamic security level interchange.

The proposed architecture provides – beside traditional security services – source location privacy. This imposes a challenge. Most existing location privacy protection mechanisms are designed to counter a passive attacker only, leaving the responsibility for active attackers to a network IDS. Yet, it has been shown that a privacy protection may render an IDS inefficient [8]. Thus one of our goals when designing the architecture was to include the privacy protection in a way that supports the concurrent operation of an IDS. For each proposed security level, we particularly stress information on which an IDS may build its decisions. The design of a particular IDS is left outside the scope of this paper. The proposed architecture is currently being implemented in order to provide detailed evaluation in both real and simulated scenarios as a part of our future work.

The roadmap of the paper is following – we discuss the application scenario, its requirements and threat model in Section 2. We describe the proposed architecture and security levels in Section 3. We then propose the lightweight mechanism for the security level interchange in Section 4. In the following Section 5, we discuss in detail the novel privacy mechanism used – dynamically changing pseudonyms. Section 6 describes related work. We conclude the paper in the last section.

2 Problem Statement

Our security architecture targets a specific application scenario. Thus it is able to cover actual security and privacy requirements with respect to the assumed attacker models. In this section we describe the application scenario and the requirements. Furthermore, we discuss the assumed attacker models and the attacker goals.

2.1 Application Scenario

We have a situation where a WSN is used by emergency services to secure a temporary perimeter and helps to organize an emergency operation. It monitors movements of an ambulance crew, police units, fire fighters and potential intruders inside the perimeter. The network is composed of a few hundreds of cheap lightweight nodes (potentially disposable) that are randomly spread over the protected area and that are deployed on demand. These nodes are combined with a few more expensive devices carried by the emergency crew. The nodes are equipped with motion detection sensors and any moving person in their neighborhood is immediately detected. Furthermore, the nodes are equipped with other environmental sensors such as acoustic, thermal and light sensors that may be helpful during the operation. The nodes are able to communicate with the devices carried by the emergency crew to a distance of 5 meters using, e.g., near field magnetic communication [3]. We assume that such communication is not detectable by an attacker unless she is in the communication distance. We base our assumption on the fact that a near field signal attenuates at $1/r^3$, where r is a distance from the transmitting antenna, whereas a far field signal, such as radio communication between sensor nodes, attenuates only at $1/r^2$. Due to a relatively short lifetime of the network when the emergency operation is in progress (e.g., several days) we can afford to drain battery power more aggressively and to ask for a high frequency of communication between nodes. Yet it is still not possible to apply the strongest security measures all the time.

2.2 Security and Privacy Requirements

The application scenario has the following high level security and privacy requirements.

- Location privacy of the emergency crew has to be protected, i.e., no unauthorized person should be able to infer the positions of the emergency crew or to track them. Otherwise an attacker could easily disturb the emergency operation, e.g., by leaking information to media or by a physical intervention.
- An unauthorized movement inside the monitored perimeter has to be reported to a base station within 3 seconds.
- Confidentiality of measured data has to be ensured. The measured data could provide an attacker with valuable information on a progress and a context of the emergency operation.

– Authentication, freshness and availability of the data are necessary for successful coordination of the emergency operation.

We revise the requirements with respect to the actual attacker model later in the subsections that describe the proposed security levels.

2.3 Threat Model

We consider five attacker models that result from our previous work within the project. The first model assumes no malicious actions and is included for reference. Each other model represents a basic attacker class with respect to attack costs, necessary equipment and impact on a network. Thus we cover the weakest attacker, the strongest attacker, a global eavesdropper that is often considered in a location privacy related literature and an attacker that is from our point of view most likely to appear in reality – internal, active and local.

We further specify each attacker model in detail prior to a related security level description in a corresponding subsection. We describe the basic characteristics from which the attacker models are composed below.

– *Local* – An attacker can overhear only a local area, i.e., her hearing range is similar to the hearing range of sensor nodes. Furthermore, she can locate the source of the transmission with reasonable precision. The attacker is located at a single spot at a time, i.e., is not distributed, and is able to move at the speed not exceeding 100 km/h.
– *Global* – An attacker can overhear all node-to-node and node-to-base station communication simultaneously for all the time. She is able to locate the source of the transmission with reasonable precision. However, she is not able to locate or overhear communication between nodes and devices carried by the emergency crew unless she is present in their communication range.
– *Internal* – An attacker is a legitimate member of a network. She has compromised at most 10% of sensor nodes and/or possesses secret keys from these nodes. If the attacker is also characterized as *local*, all the compromised nodes only come from a certain small subpart of the network. If the attacker is *global* instead, there are no constraints on the location of the compromised nodes.
– *External* – An attacker is not a legitimate member of a network and does not possess any key material.
– *Active* – An attacker may inject/modify/replay messages or she may perform jamming. Yet we assume that jamming can be detected by an IDS and reported to a base station. We leave the protection against jamming out of the scope of this work.
– *Passive* – An attacker takes no active actions. She neither injects/modifies/replays messages nor jams.

2.4 Attacker Goals

The attacker wants to take advantage of the network operation. She has one or more of the following goals. She wants to locate and track emergency crew in

the perimeter. This may help her, e.g., to evade the crew while moving in the perimeter, to physically attack the crew or to get information on the emergency operation progress. This goal can be achieved not only by interfering with the network, but also by the physical observation. Yet, in the low visibility, hilly terrain or forest such observation is difficult. Another attacker goal is to move within the perimeter undetected. Such goal could be achieved, e.g., by jamming or by other active actions. Therefore, we assume an IDS that could detect such actions and we design the security levels in a way that supports an IDS operation.

3 Security Architecture Description

The proposed security architecture consists of five security levels, each targeting a different attacker model and offering a different security/cost tradeoff. The levels can be dynamically interchanged to suit actual security and privacy requirements for a minimum cost.

We assume that nodes have established the following types of keys – pairwise link keys with all their direct neighbors, cluster keys for a local broadcast and pairwise keys with a base station. Such key distribution can be achieved, e.g., by the LEAP+ key establishment scheme [17]. We also assume that a routing topology is already established in a secure manner and has a tree-like structure. This can be ensured, e.g., by the INSENS routing technique [2].

The security levels are proposed in the following subsections. Each subsection respects the following structure – we first set up an attacker model for the level. Then, we summarize security and privacy requirements of the level. The actual security level specification is composed of an itemized description of security and privacy mechanisms that fulfill the requirements. We conclude the subsection with a summary of information available to an IDS. Yet, specification of a particular IDS is outside the scope of this paper.

3.1 Security Level 0

Security level 0 is a *reference level offering maximal efficiency and minimal privacy. No attacker is assumed. We have no security nor privacy requirements* except for a resilience against unintentional failures. Security level 0 employs no privacy and security protection at all. Packets are sent in clear and there is no authentication nor freshness guarantee. Even the weakest attacker can access all the information contained in the packets and reveal a location of the source nodes. The IDS has access to all information available.

3.2 Security Level 1

The aim of security level 1 is to protect the network against the weakest attacker – *external*, *passive* and *local*. There are three basic privacy requirements: we have to protect an identifier of the ultimate source node in the packet. Otherwise, the attacker could infer the packet source and the reported event location. Then,

we need to ensure application data confidentiality. This data may also reveal private information, e.g., that a particular unit is present in the perimeter. Finally, we have to protect the source location privacy against a traffic analysis attacks conducted by a local attacker. Security level 1 combines the following countermeasures:

- **Dynamically Changing Pseudonyms** are used to protect the ultimate source node identifier. These pseudonyms are calculated using a public one-way function, a pairwise key and a counter value shared between a source node and a base station. They change on per packet basis. Details on the pseudonym generation are discussed in Section 5.
- **End-to-End Encryption of Application Data** ensures application data confidentiality. The encryption is based on an end-to-end key shared between a source node and a base station. We propose to use a *counter mode* of operation that ensures semantic security and does not cause a ciphertext expansion. The end-to-end approach does not require a hop-by-hop re-encryption. Furthermore, the appearance of the packet remains unchanged during the forwarding process, which may help the IDS. Since a passive and external attacker is assumed in this level, a link layer hop-by-hop encryption could be used instead. This would enable data in-network processing, like aggregation.
- **Phantom Routing** [5] protects source location privacy against a local attacker. The phantom routing has two phases – in the first phase, a packet takes a random or directed walk for a given number of hops, in the second phase, the packet is routed via a shortest path to a base station. The routing information in the packet, i.e., the information on the routing phase and the number of remaining random hops, should be encrypted hop-by-hop on the link layer to increase the source location privacy.

The IDS has two main sources of information. Pseudonyms and encrypted data remain unchanged during forwarding process. Additionally, link layer information (e.g., forwarding node ID, receiver ID) is sent in clear. Since external attacker is assumed, neighboring nodes may provide their secret temporary keys to the IDS. Thus the IDS may extract additional useful information even from the encrypted parts of the packets. If the security level is switched, the nodes have to replace the provided temporary keys with new ones.

3.3 Security Level 2

Security level 2 targets an *internal*, *active* and *local* attacker. The privacy requirements cover the requirements of security level 1 – protection of the ultimate source node identifier, application data confidentiality, and protection of the source location privacy against a local attacker. Since the assumed attacker is active and may modify, inject or replay packets, we additionally need to ensure data authentication, freshness and integrity protection. Security level 2 encompasses the following security mechanisms:

- **Dynamically Changing Pseudonyms** protect the ultimate source node identifier similar to security level 1. Yet thanks to the added data authentication (see below), the pseudonyms can be shorter. For details see Section 5.
- **End-to-End Encryption of Application Data** ensures application data confidentiality in the same way as in security level 1. The end-to-end approach is particularly beneficial against an internal attacker. Such an attacker is not able to breach confidentiality of the data unless she has compromised the ultimate source node. This does not hold for the hop-by-hop encryption, thus we do not allow using hop-by-hop approach in security level 2. The *counter mode* in combination with data authentication also provides us with a weak data freshness.
- **Phantom Routing** protects source location privacy against a local attacker.
- **End-to-End Data Authentication** is necessary to counter an active attacker who could inject or modify false data. We propose to use CBC-MAC and the block cipher that is used for encryption. This enables a block cipher code reuse and saves memory of the nodes. The data to authenticate shall contain the pseudonym and the encrypted application data. We assume that the data to authenticate shall have a fixed length. Otherwise, the length has to be appended or XORed to the data as the secure use of CBC-MAC requires. The security of the authentication directly depends on the length of the MAC. This length can be taken as a security parameter. A reasonable level of security can be achieved with the length of 4 bytes.
- **Hop-by-Hop Packet Authentication** is optional and enables nodes to immediately filter out injected or corrupted packets. This does not hold for an internal attacker, who can inject packets via the compromised nodes. Yet even in this case the authentication is beneficial as it limits the possibilities and impact of the attacker.

The IDS has access to the same information as in the case of security level 1. Furthermore, it may exploit the hop-by-hop MAC and verify authenticity and integrity of the packet with respect to the last hop. In security level 2 we do not allow the neighboring nodes to provide any secret keys to the IDS, because an internal attacker could compromise the IDS.

3.4 Security Level 3

The aim of security level 3 is to protect the network against a global eavesdropper – *external*, *passive* and *global* attacker. The privacy requirements include a protection of the ultimate source node identifier, application data confidentiality, and protection of the source location privacy against a global attacker. The following countermeasures are employed:

- **Hop-by-Hop Packet Encryption** ensures application data confidentiality and protects the ultimate source node identifier. The encryption is based on a cluster key for a local broadcast that is shared between a sending node and all its neighbors. The hop-by-hop approach ensures that packet appearance,

including the source node identifier, is changed on every hop. Use of the cluster key enables the IDS to decrypt and analyze the packets. Since the attacker is external, we do not need to use an end-to-end encryption. Again, we propose to use the *counter mode* of operation.
- **Periodic Collection** [12] is employed to protect the source location privacy against a global attacker. The periodic collection makes the network traffic completely independent of the events detected and thus provides an event source unobservability together with a source location privacy against a global attacker [12]. In the periodic collection, nodes are equipped with the FIFO queue to buffer incoming real packets. Every node sends packets from the queue at a constant rate one packet per a predefined time interval. If a node has no packet to send, it creates a dummy one and sends it instead. Due to the hop-by-hop encryption the attacker is not able to distinguish real packets from dummy ones.

The IDS has access to the plain application data, source node identifiers and link layer information (e.g., forwarding node ID, receiver ID). It may also utilize the constant traffic rate.

3.5 Security Level 4

Security level 4 protects the network against all types of assumed attackers. The strongest attacker considered is *internal*, *active* and *global*. The privacy requirements of this level are the following: protection of the ultimate source node identifier; application data confidentiality, authentication, freshness and integrity protection; source location privacy against a global attacker. Security level 4 combines the following countermeasures:

- **Dynamically Changing Pseudonyms** are used in the short variant, similar to security level 2.
- **End-to-End Encryption of Application Data** is used in the same way as in security levels 1 & 2 to provide confidentiality against internal attacker and in combination with end-to-end data authentication also weak data freshness.
- **End-to-End Data Authentication**.
- **Periodic Collection** provides source location privacy against a global attacker.
- **Hop-by-Hop Packet Encryption** is necessary for a successful run of the periodic collection. It changes the packet appearance on every hop and cloaks the dummy packets used by the periodic collection. Cluster keys or pairwise keys can be used for the encryption. The use of pairwise keys provides a better protection against traffic analysis since only the nodes on the path are able to decrypt the packets. Yet, this limits the monitoring range of the IDS that also has to be on the path. Contrary to the pairwise keys, the use of cluster keys extends the IDS monitoring range to a whole neighborhood, but also helps the attacker. The attacker only needs to capture a single node to be able to decrypt the traffic in the node's whole neighborhood.

– **Hop-by-Hop Packet Authentication** is used in a similar way to security level 2.

The IDS may utilize the constant traffic rate created by the periodic collection. If the IDS has keys for hop-by-hop encryption, it gets access to the end-to-end encrypted data and pseudonyms that do not change hop-by-hop. It may also distinguish between real and dummy packets. Furthermore, the IDS may exploit the hop-by-hop MAC and verify authenticity and integrity of the packets with respect to the last hop. The monitoring range of the IDS is dependent on whether the cluster or pairwise keys are used for the hop-by-hop encryption and authentication.

3.6 Comparison

We summarize the security levels in Table 1. We provide a rough comparison of the energy costs of the security levels in order to plot the potential savings. Since energy costs caused by communication usually dominate energy costs of computations, we omit the energy spent on computations. Therefore the energy cost of a security level is dependent on the length of packets and on the number of transmitted packets. The additional lengths of the packets caused by security mechanisms are summarized in Table 1. The overhead is calculated as follows. The reference packet length with no overhead is achieved by security level 0. Packets on this level contain only information necessary for proper application operation. This information includes two byte ultimate source node identifier. If a short two-byte pseudonym is applied, it replaces the plain identifier and causes no overhead. The long four-byte pseudonym thus causes a two-byte overhead. Encryption in the counter mode does not cause ciphertext expansion and hence adds no extra bytes. Use of authentication adds extra four bytes for the message authentication code. When Phantom routing is used, up to one byte of routing information needs to be appended to every packet.

The packet overhead helps us to compare security levels that employ the same routing technique. Yet, the critical differences in energy costs of the security levels stem from the different routing techniques used. It is the routing that has the major effect as it influences the number of transmitted packets. We take the shortest path routing as a reference. The Phantom routing is a single path routing that introduces a random or directed random walk into the routing process. The number of extra packet transmissions can be thus approximated by the length of the walk. The length is a security parameter and for a network of a few hundreds of nodes the reasonable value would be between 5 and 20. Periodic collection targets global eavesdropper. Every node has to send a packet per predefined time interval τ no matter whether it has any real data to send. A total number of packets transmitted during a time period T in a network of N nodes can be estimated by $\frac{T \times N}{\tau}$ [12]. Thus, for given T and N, the overhead depends on the number of real data in the network and on the length of the interval τ. The τ value also influences latency of packets. Hence, in a setting where latency should be less than tens of seconds, the overhead introduced by

Table 1. Comparison of the security levels. Mechanisms in brackets are optional. Overhead values in brackets hold if optinal mechanism is enabled.

	Attacker model	Security mechanisms	Routing	Packet overhead
Level 0	none	none	shortest path	0 bytes
Level 1	external, passive, local	long pseudonyms, end-to-end encryption	Phantom routing	3 bytes
Level 2	internal, active, local	short pseudonyms, end-to-end encryption, end-to-end authentication, (hop-by-hop authentication)	Phantom routing	5 bytes (9 bytes)
Level 3	external, passive, global	hop-by-hop encryption	Periodic collection	0 bytes
Level 4	internal, active, global	short pseudonyms, end-to-end encryption, end-to-end authentication, hop-by-hop encryption, (hop-by-hop authentication)	Periodic collection	4 bytes (8 bytes)

the Periodic collection would be much bigger than the overhead of the Phantom routing.

4 Security Level Interchange

The described security levels can be changed on demand, to dynamically adjust the privacy provided. The most prominent way to change the security levels is to broadcast an authenticated and fresh *change-message* from the base station. The authentication and freshness of the change-message is necessary to counter the active attacker that could maliciously switch the network to the lowest security level. On the contrary, the confidentiality of the message is not necessary when the actual security level setting of the network is not secret information. Since the number of potential security levels is limited, the change-message to be broadcasted can be considered as a low entropy message. Thus we can employ a broadcast authentication scheme for low entropy messages such as LEA [11]. Yet LEA is still unnecessarily costly as we need to authenticate very low entropy messages, e.g., 3 bits.

We propose a lightweight mechanism for security level interchange based on Lamport's one-time passwords [9]. In our framework, five security levels can be set up. Thus a base station generates five one-way hash chains of length n and assigns each chain to a single security level. Let us denote $c_{i,n}$ the end point of the i-th hash chain and $c_{i,(n-j)}$ its j-th predecessor. Prior to the network deployment, all nodes are pre-loaded in an authenticated way with the end points of the hash chains. These end points act as public keys for the subsequent authentication

of change-messages. Each node also maintains a *change-counter* that counts the number of level interchanges since the network deployment. Now assume that the base station wants to set, e.g., security level 2 and it is a third interchange since the network deployment. The base station broadcasts a change-message of the form $(levelID = 2, changeNum = 3, signature = c_{2,(n-3)})$. Each node verifies the freshness and authenticity of the change-message upon its receipt. It checks if its own change-counter is lesser than the $changeNum$. It also repeatedly hashes the signature value $c_{2,(n-3)}$ and checks whether the result equals the corresponding public key $c_{2,n}$. The number of such hash applications is given by the $changeNum$. If both the checks succeed, the node updates its change-counter, sets appropriate security level and rebroadcasts the change-message.

In an efficient implementation, the node would also update the proper public key with the obtained signature value $c_{2,(n-3)}$ to reduce the number of hashing operations in the future. Also the $levelID$ and $changeNum$ values need not to be included in the change-message. This information can be derived by trial hashing of the signature value for a reasonable number of times. The exact number can be derived from the actual use of the hash-chains and never should be much above the current change-counter value. Also note that an attacker can mount a DoS power consumption attack by broadcasting a forged message forcing nodes into a trial hashing. Yet such an attack is limited by a radio range of the attacker, it is easily detectable and the impact is low as the power consumed by the useless computation is negligible when compared to the radio communication. Such DoS attacks are usually considered as a threat if a useless communication is performed by an attacked network or if the attack is hard to detect and the computation is extremely complex.

The scheme provides us with an authenticated broadcast of very low entropy messages. The number of different messages to be authenticated is equal to the number of hash chains used, i.e., to the number of end points stored by the nodes. The total number of messages to be authenticated is dependent on the length of the hash chains n. However, the chains can be very long, e.g., $n = 2^{30}$, as the length does not depend on nor influence the resources of the nodes. Using up such number of hashes would take much longer than the assumed network lifetime. The scheme also ensures weak message freshness due to the hash chain construction and the change-counter.

The flooding nature of the scheme ensures a robust propagation of the level interchange information. Thus after the flood, the network should be in a consistent state. A tricky situation happens if a message is being delivered in the middle of the security level interchange process. A node that receives the message should decide depending on a policy to either forward the message using the old security level, or drop the message. These options are used only if the security level was changed recently, otherwise the message sent in invalid security level is dropped. Alternatively, all the nodes that have initiated a message immediately before the change may resend their messages after the change. Besides the described on demand interchange, the security levels may change also at a preset time. This is extremely useful in situations in which the time intervals that

require different security levels are a priori known. Yet this approach requires a certain level of time synchronization across the network.

5 Dynamically Changing Pseudonyms

The identifiers of the ultimate source node have to be protected to ensure the source location privacy. We favor to use dynamically changing pseudonyms over end-to-end encryption. The encryption would require the base station to try large number of decryption keys. Furthermore, the message would have to carry an unnecessary redundancy to confirm a successful decryption with a proper key.

Whereas the use of other security mechanisms in our framework is straightforward or described in the referenced papers, the employment of dynamically changing pseudonyms is more complex and differs from the existing solutions. Therefore, we describe the details in this section. The pseudonyms are based on a key K_{AB} and a counter value shared between a node A and a base station B. First, a short term key K_{tAB} valid for a time interval t is derived from the key K_{AB}. Then, we use the short term key to generate a pseudonym $A_i = F(K_{tAB}, A, i)$, where A is a node ID, i is an actual counter value and F denotes a one-way function. The pseudonym A_i is valid for a message with a counter value i. The base station B possesses similar information to that of the node A. Thus, it can pre-compute expected values of pseudonyms and store them in a table indexed by the pseudonyms. This enables the base station to immediately identify the source node ID upon receipt of the packet. For every node ID, the base station stores pseudonyms for counter values $i, ..., i + c$, where i is an expected counter value and c denotes a constant. This enables a fast synchronization if a packet with an expected pseudonym is lost and an incoming packet has higher counter value than expected, yet lesser than constant c. The higher the constant, the better the ability to synchronize counter values. Note that the size of the table does not pose a problem since the table is only maintained and processed by a base station. Thus the constant c can be high enough, e.g., $c = 64$, to enable fast synchronization even if the packet loss ratio is high. For an example of such table see Table 2.

The length of pseudonyms is an important parameter. It depends on the number of nodes and the amount of traffic in the network. Should the length be too short, pseudonyms generated by two distinct nodes could often collide by accident. In such a case, the base station would not be able to uniquely identify the source node. We assume here that the message contains no redundancy. Thus, the decryption with a wrong key can result in a meaningful plaintext. We employ pseudonyms in two different situations. In the first situation, the pseudonym is used in combination with a MAC. In such a case, the length of the pseudonym can be relatively short, because the potential collision could be easily resolved by trial MAC verification. In the second situation, the message contains no MAC. Thus the pseudonym length has to be longer. Yet we propose to use a short pseudonym as in the first case and supplement it with a shortened MAC. This enables the base station to resolve the potential pseudonym collisions and

Table 2. An example of table stored at a base station. Index is on pseudonym column to enable fast search upon a packet is received. The expected pseudonyms for nodes A and C are in bold.

Pseudonym	Counter	Node ID	Short term key	Long term key
$\boldsymbol{A_3}$	3	A	K_{tAB}	K_{AB}
A_4	4	A	K_{tAB}	K_{AB}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_{3+c}	$3+c$	A	K_{tAB}	K_{AB}
$\boldsymbol{C_7}$	7	C	K_{tCB}	K_{CB}
C_8	8	D	K_{tCB}	K_{CB}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

slightly improves the security by adding the short MAC. Note that such MAC cannot be taken as a full and secure MAC since it can be, e.g., just 2B long. Yet it provides us with an additional integrity protection, while it does not increase communication cost and requires only a single MAC computation.

The use of pseudonyms and counter values enables a potential DoS attack by an attacker who blocks c successive messages from the same sender. In such a case, the sender would not be recognized by a base station again. However, there is a simple workaround. If a base station cannot recognize a sender of a message by the included pseudonym, it can perform a trial MAC verification with all keys available. This does not require a counter value nor a sender identity. Once a proper key and a sender identity are found, a counter value can be synchronized by a trial decryption. This workaround applies directly to security levels 2 & 4 where end-to-end authentication is involved. In security level 1, it can be used if the shortened MAC is used as proposed above. Security levels 0 & 3 do not use pseudonyms, thus synchronization is not needed.

6 Related Work

There are plenty of link layer security frameworks for WSNs [4, 6, 7, 10, 13, 16], yet only some of them enable adjusting the level of security. TinySec [6] is a link layer security architecture that offers three security modes – unprotected mode, TineSec-AE and TinySec-Auth. The unprotected mode offers no security service. TinySec-AE provides packet authentication and confidentiality. TinySec-Auth ensures packet authentication only. The security modes are applied on a per packet basis. The security mode used for a given packet is indicated by two bits in the packet header. An appropriate security mode for a packet is selected by a source node.

L^3Sec [7] is a link layer security architecture similar to TinySec. Unlike TinySec that offers authentication and confidentiality, L^3Sec supports three security services – authentication, confidentiality and replay protection. These services

can be enabled or disabled separately which results in eight different security modes in total.

SecureSense [16] is another configurable link layer security framework and offers even more flexibility than L^3Sec. SecureSense architecture enables composition of four basic security services – confidentiality, integrity, semantic security and replay protection. It also allows to select a strength of an encryption algorithm used. The enabled security services and the strength of the algorithm are indicated by a dedicated byte in every packet header.

TinySec, L^3Sec and SecureSense support a runtime composition of basic security services on a link layer. Different packets may implement different security services. Our architecture in turn also covers a network layer to provide source location privacy and end-to-end security services. Therefore, the security level is always applied throughout the whole network, not on a per packet basis. Furthermore, a security level is selected by a base station or a network operator instead of a source node.

FlexiSec [4] is a link layer security framework that offers the highest configurability. It enables a network owner to tailor the link layer security settings specifically to its network application requirements and a hardware platform used. This includes a selection of MAC lengths, encryption algorithms, modes of operation or methods for replay protection. Yet, the required security level has to be set prior to the network deployment and remains fixed during the network operation time.

An architecture that is close to our approach was proposed by de Oliveira et al. [1]. The authors extend the Manna network management framework for sensor networks [15] with a security management functionality. In their work, a manager running on a base station can dynamically enable or disable security levels that consist of selected security services. The considered security services include an end-to-end and hop-by-hop cryptography, a secure routing, key management, an IDS and revocation scheme and a secure data fusion. The approach is policy based, thus rules in a form *condition-action* are specified and followed by the manager. The condition contains information on intrusion provided by an IDS, the action represents an activation of a particular security level. Our work differs from this contribution in several ways. We propose an architecture and define particular security levels for a given application scenario, whereas de Oliveira's work presents only a general management architecture without any particular application or security levels in mind. We further propose a lightweight scheme that enables a secure and efficient dynamic security level interchange. The architecture of de Oliveira et al. does not discuss security of management messages and fully relies on the MannaNMP protocol that does not implement any security measure. This renders the whole architecture useless as an active attacker can easily activate the lowest security level available. Our architecture also includes location privacy protection, not only security related mechanisms, and also takes the presence of an IDS into consideration.

Rios and Lopez [14] have proposed the Context-Aware Location Privacy (CALP) approach. Their approach takes advantage of the ability of nodes to

detect the presence of a mobile attacker. Thus using CALP, a network may become aware of the attacker position and may use this knowledge to improve security/cost trade-off. Rios and Lopez [14] have applied CALP in routing to improve source location privacy in WSNs. If no attacker is present, packets may be efficiently routed via the shortest path. Yet if an attacker is detected, packets are de-routed around the detected attacker position.

7 Conclusions and Future Work

We proposed an adaptive security architecture for location privacy sensitive wireless sensor network applications. This architecture enables us to dynamically adjust the security level in the network according to the actual security and privacy requirements. Thus it may offer much better efficiency than a traditional fixed security setting. We showed the concrete instance of the architecture through the specification of the five security levels. These levels target a particular application scenario and multiple different attacker models. As a part of the architecture, we proposed a lightweight mechanism for broadcast authentication of very low entropy messages. This mechanism is used for a dynamic security level interchange. Also, we proposed a simple mechanism for node identifier protection that relies on dynamically changing pseudonyms. The proposed architecture is currently being implemented. We plan to experimentally evaluate the architecture and the security levels in both a real network and a simulator. We would also like to combine our scheme with a particular IDS to enhance the overall security of the network.

Acknowledgements. We are grateful to the anonymous reviewers for their suggestions that improved the paper. This work was supported by the project GAP202/11/0422 of the Czech Science Foundation.

References

1. de Oliveira, S., de Oliveira, T.R., Nogueira, J.M.: A policy based security management architecture for sensor networks. In: IFIP/IEEE International Symposium on Integrated Network Management, IM 2009, pp. 315–318 (June 2009)
2. Deng, J., Han, R., Mishra, S.: INSENS: intrusion-tolerant routing for wireless sensor networks. Computer Communications 29(2), 216–230 (2006)
3. Jiang, X., Liang, C.-J.M., Chen, K., Zhang, B., Hsu, J., Liu, J., Cao, B., Zhao, F.: Design and evaluation of a wireless magnetic-based proximity detection platform for indoor applications. In: Proceedings of the 11th International Conference on Information Processing in Sensor Networks, IPSN 2012, pp. 221–232. ACM, New York (2012)
4. Jinwala, D., Patel, D., Dasgupta, K.: FlexiSec: a configurable link layer security architecture for wireless sensor networks. Journal of Information Assurance and Security 4, 582–603 (2009)

5. Kamat, P., Zhang, Y., Trappe, W., Ozturk, C.: Enhancing Source-Location privacy in sensor network routing. In: ICDCS 2005, pp. 599–608. IEEE Computer Society, Washington, DC (2005)
6. Karlof, C., Sastry, N., Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, pp. 162–175. ACM, Baltimore (2004)
7. Krontiris, I., Dimitriou, T., Soroush, H., Salajegheh, M.: WSN link-layer security frameworks. In: Lopez, J., Zhou, J. (eds.) Wireless Sensor Network Security, pp. 142–163. IOS Press (2008)
8. Kůr, J., Matyáš, V., Stetsko, A., Švenda, P.: Attack detection vs. Privacy – how to find the link or how to hide it? In: Christianson, B., Crispo, B., Malcolm, J., Stajano, F. (eds.) Security Protocols XIX. LNCS, vol. 7114, pp. 189–199. Springer, Heidelberg (2011)
9. Lamport, L.: Password authentication with insecure communication. Commun. ACM 24(11), 770–772 (1981)
10. Luk, M., Mezzour, G., Perrig, A., Gligor, V.: Minisec: A secure sensor network communication architecture. In: 6th International Symposium on Information Processing in Sensor Networks, IPSN 2007, pp. 479–488 (April 2007)
11. Luk, M., Perrig, A., Whillock, B.: Seven cardinal properties of sensor network broadcast authentication. In: Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks, SASN 2006, pp. 147–156. ACM, New York (2006)
12. Mehta, K., Liu, D., Wright, M.: Location privacy in sensor networks against a global eavesdropper. In: IEEE International Conference on Network Protocols, ICNP 2007, pp. 314–323 (October 2007)
13. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. Wireless Networks 8(5), 521–534 (2002)
14. Rios, R., Lopez, J.: Exploiting context-awareness to enhance source-location privacy in wireless sensor networks. The Computer Journal 54(10), 1603–1615 (2011)
15. Ruiz, L.B., Nogueira, J.M.S., Loureiro, A.A.F.: MANNA: a management architecture for wireless sensor networks. IEEE Communications Magazine 41(2), 116–125 (2003)
16. Xue, Q., Ganz, A.: Runtime security composition for sensor networks (SecureSense). In: 2003 IEEE 58th Vehicular Technology Conference, VTC 2003-Fall, vol. 5, pp. 2976–2980 (October 2003)
17. Zhu, S., Setia, S., Jajodia, S.: LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Trans. Sen. Netw. 2(4), 500–528 (2006)

Secure and Lightweight Distance-Bounding

Ioana Boureanu[1], Aikaterini Mitrokotsa[2], and Serge Vaudenay[1]

[1] EPFL
CH-1015 Lausanne, Switzerland
http://lasec.epfl.ch
[2] University of Applied Sciences of Western Switzerland (HES-SO)
CH-1227 Geneva, Switzerland
katerina.mitrokotsa@hesge.ch

Abstract. Distance-bounding is a practical solution aiming to prevent relay attacks. The main challenge when designing such protocols is maintaining their inexpensive cryptographic nature, whilst being able to protect against as many, if not all, of the classical threats posed in their context. Moreover, in distance-bounding, some subtle security shortcomings related to the PRF (pseudorandom function) assumption and ingenious attack techniques based on observing verifiers' outputs have recently been put forward. Also, the recent terrorist-fraud by Hancke somehow recalls once more the need to account for noisy communications in the security analysis of distance-bounding. In this paper, we attempt to incorporate the lessons taught by these new developments in our distance-bounding protocol design. The result is a new class of protocols, with increasing levels of security, accommodating the latest advances[1]; at the same time, we preserve the lightweight nature of the design throughout the whole class.

1 Introduction

In [9], Brands and Chaum introduced the notion of distance-bounding (DB) protocols. The aim is to have a prover demonstrate his proximity to a verifier, and authenticate himself to this verifier. The proof of proximity can be an efficient deterrent against relay attacks [15]. DB protocols [23,25,34,37] generally consist of an *initialisation phase* (where the parties establish some short-term secret) and a *distance-bounding* phase. This latter phase is time-critical. It imposes very fast computation, typically of less than a single clock cycle per round, and the verifier measures the time-of-flight of the messages exchanged. This is how the verifier ascertains a distance-bound between him and the prover.

In the literature covering such protocols, the threat-model comprises three well-established types of attacks. The first is *distance-fraud* (DF), in which a prover tries to convince the verifier that he is closer than he really is. In the second type, *mafia-fraud* (MF), an adversary communicates with both a prover and a verifier which are far apart, and the adversary tries to convince the verifier that the prover would be close

[1] An earlier version of this line of work was presented in [7]. Also, some preliminaries and adjacent topics made the subject of an invited talk [8].

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 97–113, 2013.

enough to be granted privileges. Finally, in a *terrorist-fraud* (TF), an adversary is getting the necessary and sufficient help from a coerced, far-away prover in order to pass the protocol only during this corrupted run, but not in a later, coercion-free session. Generalisations of these frauds have also been described. In [13], Cremers *et al.* describe distance-hijacking as a mixture between distance-fraud and terrorist-fraud: one dishonest, far-away prover exploits several honest provers to gain privileges. Impersonation-fraud is presented in [16]; as its name suggests, one dishonest prover tries to impersonate an honest one.

The first DB protocols were not secure against terrorist-fraud [9,21,29,35]. Then, to name but a few, Bussard and Bagga [10], Hancke and Kuhn [21], Munilla and Peinado [29], Kim and Avoine [24], Reid *et al.* [34] proposed schemes addressing terrorist-fraud protection or mafia-fraud protection, or a better suitability to practice, etc. For instance, in [3], the TDB protocol by Avoine *et al.* addresses specifically the protection against terrorist-fraud, using threshold secret sharing schemes. Nonetheless, many attacks [1,27,28,31,30] onto DB protocols [25,35,37,33] continue to be published. To this end, Kim *et al.* stated [24] that there is no DB protocol, which can resist well against all three classical frauds and has only one-bit challenges/responses per iteration in the distance-bounding phase.

Recently, the first attempts to formalise DB have emerged. In [2], Avoine *et al.* give a semi-formal model for distance-bounding. Dürholz *et al.* [17] follow, with a more precise formalisation in which the expression TF appears possibly too strong (i.e., many protocols that are intuitively TF-resistant are shown insecure against TF in this model). At the same time, [6,4] expose some essential shortcomings of the DB design and of the security claims related to it (i.e., [6] exposes building blocks for DB, like pseudo-random functions (PRFs), that lead to DF and generalised MF attacks; [4] shows that public-key mechanisms may fail to provide TF).

In this paper, we attempt to take notice of all these recent developments: e.g., we strengthen the way PRFs are used in DB, we reinforce and take forward the manner in which secret sharing schemes can be employed to build TF-resistant DB protocols and, finally, we attempt to combine it all harmoniously in such a way that we obtain robust, yet lightweight DB.

2 Summary of DB Security: Status and Results

At this early stage, in Table 1 below, we present the security status of several, existing DB protocols, and announce two of our DB protocols to be presented herein. Namely, please notice our $\mathbf{SKI_{pro}}$ and $\mathbf{SKI_{lite}}$ protocols and their security guarantees by comparison to the other DB protocols in this table.

In this table, we assumed channels that are *not noisy*, though further in this paper we extend the analysis on our protocols to the case of noisy channels as well. Let us briefly explain some details from the table. Let n be the number of DB rounds, and $\nu < n$. Let t be the number of possible values for a challenge, i.e., classically $t = 2$. In the case of terrorist-fraud, we supposed along standard lines two facts: 1. for $n - \nu$ DB rounds, the adversary has got all responses, irrespective of the value of challenges; 2. for the other ν DB rounds, for each such round, the adversary knows the answers for $t - 1$ (out of t) values possible for a challenge.

Table 1. Probability of success of the best (known) attacks onto DB

Protocol	Success Probability				
	Key-Length	Distance-Fraud	Mafia-fraud	Terrorist-Fraud	MIM
Brands & Chaum [9]	n	$(1/2)^n$ cnf [19]	$(1/2)^n$ cnf [25]	1 cnf [25]	$(1/2)^n$
Bussard & Bagga [11]	n	1 cnf [4]	$(1/2)^n$	1 cnf [4]	$(1/2)^n$
Čapkun *et al.* (SECTOR) [12]	n	$(1/2)^n$ cnf [19]	$(1/2)^n$ cnf [25]	1 cnf [25]	$(1/2)^n$
Hancke & Kuhn [21]	n	$(3/4)^n$ cnf [19]	$(3/4)^n$ cnf [25]	1 cnf [25]	$(3/4)^n$
Reid *et al.* [34]	n	$(3/4)^n$ cnf [19]	$(3/4)^n$ cnf [26]	$(3/4)^{\nu}$ cnf [25]	$(3/4)^n$ or 1 cnf [4]
Singelée & Preneel [35]	n	$(1/2)^n$ cnf [19]	$(1/2)^n$ cnf [25]	1 cnf [25]	$(1/2)^n$
Tu & Piramuthu [37]	n	$(3/4)^n$ cnf [30]	$(9/16)^n$ cnf [30]	$(3/4)^{\nu}$ cnf [30]	1 cnf [25]
Munilla & Peinado [29]	n	$(3/4)^n$ cnf [19]	$(3/5)^n$ cnf [19]	1 cnf [19]	$(3/5)^n$
Swiss-Knife [25]	n	$(3/4)^n$ cnf [25]	$(1/2)^n$ cnf [25]	$(3/4)^{\nu}$ cnf [25]	$(1/2)^n$
Kim & Avoine [24]	n	$(7/8)^n$ cnf [19]	$(1/2)^n$ cnf [19]	1 cnf [19]	$(1/2)^n$
Nikov & Vauclair [32] *	$\bar{k}$	$1/k$ cnf [25]	$(1/2)^n$ cnf [25]	1 cnf [25]	$(1/2)^n$
Avoine *et al.* [3]	n	$(3/4)^n$	$(2/3)^n$	$(2/3)^{\nu}$	$(2/3)^n$
SKI$_{\mathbf{pro}}$	ℓ	$(3/4)^n$	$(2/3)^n$	$(2/3)^{\nu}$	$(2/3)^n$
SKI$_{\mathbf{lite}}$	ℓ	$(3/4)^n$	$(3/4)^n$	1	$(3/4)^n$

* In this case, $\bar{k}$ and k are additional parameters; this protocol requires heavy computations. The parameter ν is explained in the paragraph above.

From the table, we can already notice some similarities between the protocol in [21], by Hancke and Kuhn, and the simplest version of the **SKI** protocols to be introduced herein, namely **SKI$_{\mathbf{lite}}$** . Also, we can see a certain closeness between the Avoine *et al.* protocol [3] and a stronger version of our protocols, i.e., **SKI$_{\mathbf{pro}}$** . In that sense, what this line of work brings as a novelty is a more precise design of the protocols (i.e., there are design differences between the **SKI** protocols and its similar counterparts in the table). Our design is driven by very recent exhibited DB attack-techniques and classical frauds [20,4,6]. We also propose a more in-depth security analysis due to the same recent threats and a more attentive look into the DB security in noisy communications[2].

As the reader will see in our design choices presented in Section 5.3 and in the attacks we present in Section 5, we get our attack bounds (as per Table 1) by enforcing certain requirements on our DB building blocks. We hereby mention some of these enforcements: 1. the use of the PRF instance in the initialisation phase is masked, i.e., we use $f_x(\cdot) \oplus M$ for a randomly looking M, instead of just employing $f_x(\cdot)$; 2. the DB response-function is such that it uses the secret x in a way that it does not conflict with x keying $f_x(\cdot)$ in the initialisation phase; 3. a linear transformation is chosen at the initialisation phase to be applied on the secret x, before we use x in the response-function. These are the sort of design-amendments imposed by the recent, aforementioned attacks [20,4,6]. In fact, the very new attack-technique in [20] is not taken into account in Table 1. With our **SKI$_{\mathbf{pro}}$** protocol, we resist the TF by Hancke.

Structure. The rest of this paper is structured as follows. In Section 3, we reiterate what are the settings in which DB communications are taking place. In Section 4, we express the DB security requirements in these communication settings; to do so, we follow the well-established understandings of the classical frauds in the existing literature, and we also offer some generalisations. In Section 5, we introduce our protocol-schema, called **SKI**, explain most of its design, and present two instantiations of it, i.e., **SKI$_{\mathbf{pro}}$** and

[2] E.g., a recent attack-technique [20] by Hancke, described on page 106, reiterates the importance of considering noise in DB, bit-based computations.

$\mathbf{SKI}_{\mathbf{lite}}$. We then argue that these protocols protect against the frauds as they were described in Section 4. In Section 6, we conclude. In an appendix, we present other instantiations of our protocol schema (i.e., $\mathbf{SKI}_{\mathbf{shamir}}$ and $\mathbf{SKI}_{\mathbf{4}}$), varying in their security strength, but all remaining lightweight.

3 DB Communication

In what follows, we present the main, very straightforward guidelines of the settings in which DB protocols are considered to run. (The underlying communication and the threat model could actually be properly formalised, e.g., as an interactive system [18]. This is not our purpose herein, and it will be left for an extended version of this paper.)

DB protocols are run in natural communication settings :

- there is a generally accepted notion of time, e.g., there is a time-unit;
- a notion of measurable/quantifiable location and distance;
- the timed communication obeys the laws of physics.

All participants (provers, verifiers, adversaries) comply to the following:

- have the correct means/algorithms to run their part (e.g., an RFID tag, a reader, both, etc.);
- are fixed at some location;
- send messages with a destination through a broadcast, non-authenticated, asynchronous channel.

Furthermore, honest participants read messages that are intended for them, when these messages reach them. An attacker can change the destination of a message, aiming it to himself and can create his own messages and inject them into the communication. In the distance-bounding phase, the noise of the channel cannot be corrected by honest parties (i.e., the adversary may have extra technology to do so, but the honest parties cannot do so within the limits of time imposed).

NOTE: It is clear in this model that an adversary can do very limited man-in-the-middle attacks. If a verifier sends a message and expects a fast response back, this deters a man-in-the-middle (MiM) adversary to send the message further to a prover and await for the prover's response to convey to the verifier, i.e., as such responses would arrive to the verifier too late. In the same way, an adversary can get very limited, *online* help even from a coerced, but far-away prover.

4 DB: Protocols and Requirements

In line with the previous section, we present these requirements using natural language. (As before, it is worth mentioning that in a formal model for DB, these could be expressed, e.g., in the style of completeness/soundness requirements on interactive systems [18], with thresholds on the success/failure probabilities of different events or sequences of events. This is left for an extended version of this paper tackling formalisation and provable security aspects.)

4.1 Distance-Bounding Protocols

In general, let the provers be denoted by P and the verifiers by V. Let $\mathcal{A}$ denote the adversary and P^* designate dishonest provers. We assume that verifiers end the DB protocols by outputting one bit b denoting acceptance, i.e. $b = 1$, or rejection, i.e., $b = 0$. (I.e., this is in line with practice, where a LED turning green or red on an access point denotes granted or denied access, respectively). In the generalised MF presented in [4], it is this sort of return channel that facilitates the attacks (i.e., logically, intruders learn more information by looking also at whether the run was successful or not.). We proceed with the definition of a DB protocol.

Definition 1. ***Distance-Bounding Protocols.*** *A* distance-bounding (DB) protocol *is defined by an acceptable distance-bound, a prover P and a verifier V, each running probabilistic, efficient*[3] *algorithms, both sharing a long-term key x such that the following happens:*

- *the verifier's algorithm efficiently terminates on* any *interaction*[4]*;*
- *if the prover P is within the acceptable distance-bound from the verifier V, then the verifier V terminates successfully (almost always*[5]*).*

DB can take place in concurrent settings as well, i.e., there are several provers and several verifiers, sharing secrets in a pairwise manner, all running the same DB protocol in parallel. We can also think of the scenario where one prover and one verifier run the same protocol several times, in a sequential fashion. In the description of the security requirements to follow, we will also consider such multi-party extensions of the definition above.

4.2 Distance-Bounding Requirements

Let $\alpha, \beta, \gamma, \gamma' \in [0,1]$ be some variables (depending on some parameters, e.g., on the number of rounds in the distance-bounding phase), or let $\alpha, \beta, \gamma, \gamma' \in [0,1]$ be some fixed constants (e.g., pre-established security-tolerance limits). The security requirements of DB protocols are described below, and they depend on the values of these $\alpha, \beta, \gamma, \gamma'$.

Definition 2. α***-resistance to distance-fraud****: We say that a DB protocol is* α-resistant to distance-fraud *if any far-away, dishonest prover* P^* *which is running the protocol with a verifier V, on their shared secret, cannot make the verifier accept (i.e., output* 1*) with a probability greater than* α *(taken over the random choices made by V).*

[3] In theory, "efficient" denotes polynomial in some security parameters. In practice, one should be able to see clearly that these algorithms are computationally inexpensive.

[4] Even if the prover is dishonest, after a finite number of steps, a reader either accepts or rejects.

[5] In theory, "almost always" would entail some overwhelming probability in a security parameter. In practice, it means that there are some exceptional circumstances where the verifier would reject correct transcripts. I.e., in extremely noisy channels (which occur very rarely) the verifier would be bound to reject messages that originated correctly from the prover.

As we said before, depending on the security desired, one may take α to be negligible in a security parameter (e.g., $c^{\Theta(n)}$, where c is a constant in $(0,1)$ and n is the number of rounds and/or the key-length) or, simply a fixed value in $(0,1)$.

If we consider a multi-party setting (e.g., taking several runs, with far-away and close-by provers), then the DF-resistance as defined above captures the notion of distance hijacking in [13], i.e., an experiment in which a dishonest far-away prover P^* may use several other provers to get authenticated as if he was close to the verifier. The DF-resistance we assess in Section 5.5 can be extended to account for such a multi-party setting.

We move now to the resistance to mafia-fraud.

Definition 3. β***-resistance to MF****: We say that a DB protocol is* β-resistant to mafia-fraud *if an adversary $\mathcal{A}$ interfering up to his powers within the interaction between a far-away, honest prover P and a verifier V on their shared secret cannot make the verifier accept (i.e., output* 1*) with a probability greater than* β *(taken over the random choices made by P,V and $\mathcal{A}$).*

Of course, this definition of MF-resistance can be cast in a multi-party setting as well and it can also be generalised to a stronger MiM attack. For instance, in a multi-party setting, we consider that during a learning phase, the attacker $\mathcal{A}$ interacts, in parallel, with several provers and several verifiers and then —in an attack phase— $\mathcal{A}$ tries to win in a run in front of a verifier, which is far-away from several provers. (In a practical setting, it is as if an attacker would have cloned several tags and would make them interact with several readers with which they are registered. From such a multi-party communication, he can get potentially more benefits, faster.) In our security assessment in Section 5.5, the arguments can be easily extended to such a concurrent setting.

Definition 4. (γ,γ')***-resistance to TF****: We say that a DB protocol is* (γ,γ')-resistant to terrorist-fraud *if for any far-away, coerced prover P^*, it is the case that, below,* (1) *implies* (2)

— (1). *an adversary $\mathcal{A}$ interfering up to his powers with an interaction between P^* and a verifier V on their shared secret, where this interaction is successful with probability at least γ (over the random choices of V and $\mathcal{A}$),*

— (2). *$\mathcal{A}$ can later succeed on his own to make the verifier accept in a new protocol run with a probability greater than γ' (taken over the new random choices made by V and $\mathcal{A}$).*

This definition of TF-resistance can also be presented in a multi-party setting and generalised to a stronger threat. For instance, one first thing to imagine is a coercion-phase followed by a multi-party MF, i.e., a MiM phase as we mentioned after Definition 3. In fact, our assessment of TF-resistance made in Section 5.5 can be extended easily to such an enhanced threat.

5 The SKI Distance-Bounding Protocols

In the first part of this section, we present our protocols. In the second, we explain our design. In the third, we assess their resistance to frauds, upon the definitions and discussions in Section 3 and Section 4.

5.1 Protocols' Descriptions

We now propose a schema of DB protocols denoted ***SKI*** and presented in Figure 1, i.e., we use "schema" to denote that, at this stage, we leave under-determined the choice of the exact primitives to be used inside. Later in the section, by suggesting different instantiations of these primitives, we obtain a class of DB protocols, with varying levels of resistance to DB attacks. Nonetheless, from the weakest to the strongest of them, these protocols are lightweight.

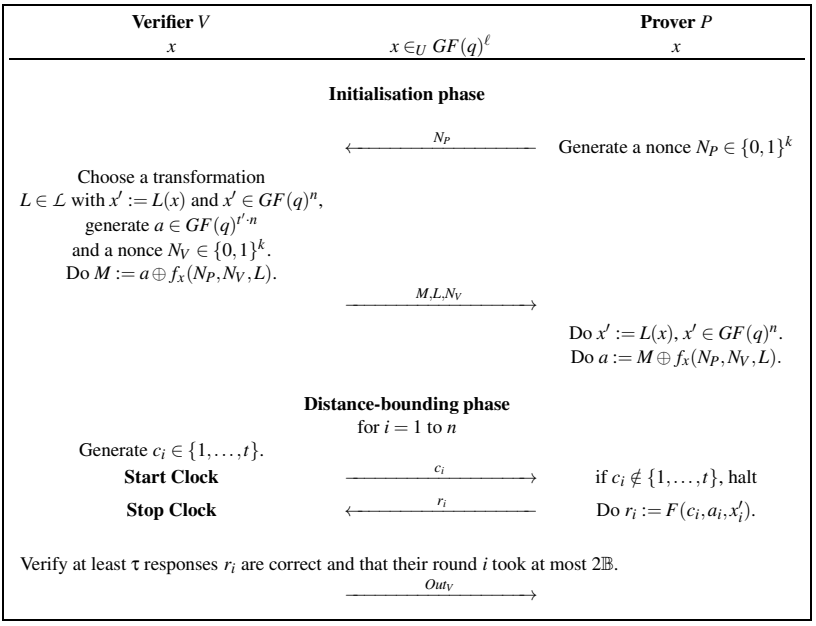


Fig. 1. The **SKI** schema of Distance-Bounding Protocols

Let s be a security parameter. The secret key x is a vector of ℓ elements over $GF(q)$, with $\ell \in \Omega(s)$, with q a constant giving the power of prime so that we work over $GF(q)$, the finite field with q elements. In some of the concrete examples to follow, we employ $q = 2$, i.e., we work over bits[6]. The **SKI** protocols are built using a PRF, denoted $(f_x)_{x \in GF(q)^\ell}$.

The prover selects a nonce N_P of k bits and sends it over to the verifier, for $k \in \Omega(s)$.

The verifier V first selects a nonce N_V also of k bits. Then, he picks a random linear transformation L from a set $\mathcal{L}$, set that is specified by the **SKI** protocol instance (as we will concretely see later). The parties compute $x' = L(x)$. We consider that the vector x' obtained out of x through L has length n, with $n \in \Omega(s)$.

[6] Irrespective of working over bits or not, we consider that the practicality of today's cheap RFID/NFC cards goes anyway beyond one-bit responses [36]. Moreover, pre-computation tables can be used to render online computation very efficient.

The distance-bounding phase will have n rounds with challenges taking t possible values, for a constant t. Another constant we use is t'. To give an anticipative intuition, t' is such that $t' \leq t$. In our main proposal, we use $t = 3$ and $t' = 2$, i.e., we keep the lightweight character.

The verifier finally picks a masking-vector M with $M \in GF(q)^{t' \cdot n}$. Further, the element $a = (a_1, \ldots, a_n)$ is established by V and it is sent encrypted into M as follows: $M = a \oplus f_x(N_P, N_V, L)$, with $M \in GF(q)^{t' \cdot n}$. (As we can see, **SKI** employs $f_x(N_P, N_V, L)$ as illustrated, with $f_x(N_P, N_V, L) \in GF(q)^{t' \cdot n}$.)

So, $c = (c_1, \ldots, c_n)$ is the challenge-vector with $c_i \in \{1, \ldots, t\}$, $r_i := F(c_i, a_i, x'_i)$ is the i-th response to the i-th challenge c_i, with $i \in \{1, \ldots, n\}$, $r_i \in GF(q)$ and F as specified below. In our concrete proposals, we use $t = 3$, or $t = 2$ for the lighter version.

By $\mathbb{B}$, we denote the maximal accepted time-of-flight of one challenge/response between P and V. Assume that messages travel uniformly with a speed of one space-unit/time-unit. Then, as usual, $\mathbb{B}$ is also the distance-bound acceptable between P and V.

The protocol ends with a message Out_V denoting the output of the verifier (i.e., the success/failure of the protocol). We tolerate communication noise. Thus, a successful run is that where at least τ out of n DB responses are correct and have been delivered within the time-bound $\mathbb{B}$. Later in the paper, it will be implied what bound on τ we need (in function of n and the probability of the communication noise) such that legitimate runs are not overruled, yet malicious runs are not validated.

As we anticipated already, all the variables and functions in **SKI** will be instantiated with small values and lightweight mathematical objects.

NOTE: To address noisy time-critical communications, we introduce the probability p_{noise} of one response being erroneous (à la [21]). The probability that at least τ responses out of n are of the correct kind is clearly given by:

$$B(n, \tau, 1 - p_{noise}) = \sum_{i=\tau}^{n} \binom{n}{i} (1 - p_{noise})^i p_{noise}^{n-i}$$

It is natural to choose τ (and other parameters) such that we operate with correct DB protocols, cnf. with Definition 1. I.e., the protocol is complete: honest communications succeed with high probability. Let us assess this. So, let $\varepsilon > 0$. If we force τ such that $\tau \leq (1 - p_{\mathsf{noise}} - \varepsilon)n$, then it implies $B(n, \tau, 1 - p_{noise}) \geq 1 - e^{-2\varepsilon^2 n}$ (due to the Hoeffding bound [22]), i.e., it implies the verifier accepting honest communications with a very good probability as n grows. Also, in practice, we may use a constant p_{noise} (i.e., hard-coded in the protocol implementation). This also entails employing τ as some parameter which is linear in terms of n (in order to have negligible probabilities of failure in honest executions).

A detailed analysis on optimising the selection of τ is provided in [14].

5.2 Towards Specific Building Blocks

We now continue with the instantiations of some objects in our **SKI** schema. Our choices of them will be explained shortly.

The response-function F. In the main body of the paper, we consider one generic such response-function F in which the i-th response ($1 \leq i \leq n$) is produced as follows:

$$\mathbf{F_{xor}}(c_i, a_i, x_i') = x_i' 1_{c_i=t} + (a_i)_1 1_{c_i \in \{t,1\}} + \ldots + (a_i)_{t'} 1_{c_i \in \{t,t'\}}$$

where $c_i \in \{1,\ldots,t\}$, $x_i' \in GF(q)$, $q \geq 2$, $(a_i)_j \in GF(q)$, $j \in \{1,\ldots,t'\}$, and 1_R is 1 if R is true and 0 otherwise.

In the appendix, we will present other possibilities for the response-function F.

The set of transformations $\mathcal{L}$. We can consider several sets $\mathcal{L}$ of transformations to be used in the PRF-instance of **SKI**'s initialisation phase. (Such a set is formally referred to as a *leakage scheme* and it is thuswise defined in [5].)

We consider $\mathcal{L}_{\mathsf{bit}}$ consisting of all L_μ transforms, where L_μ is defined using a vector $\mu \in GF(q)^\ell$ by

$$L_\mu(x) = (\mu \cdot x, \ldots, \mu \cdot x)$$

I.e., all coordinates of the vector $L_\mu(x)$ are set to the scalar product between μ and x.

We could consider other suitable[7] instances of $\mathcal{L}$, but this may entail a number of DB rounds greater than n (because of the noise involved). Or, if no noise is to be considered, we could employ $\mathcal{L} = \mathcal{L}_{\mathsf{classic}}$, i.e., the set containing a single function L which is the identity function. For the purpose of this paper (i.e., lightweight DB protocols), we restrict ourselves to using $\mathcal{L}_{\mathsf{bit}}$ as per the above. Also, if we do not view TF-resistance, then we can leave the $\mathcal{L}$ set empty.

5.3 SKI: Design Choices

Using a mask M. We chose to use a mask M, indirectly decided by V, due to the fact that just using $f_x(\ldots,N_P,\ldots)$ to calculate a can lead to DF attacks [6]. To mount such an attack, a corrupt P^* basically chooses a trapdoor N_P to bias the output distribution of $f_x(\ldots,N_P,\ldots)$. By using the mask M, we prevent such a P^* from reaching his goal.

The PRF f & the Response-Function F. Already note the $\mathbf{F_{xor}}$ is in fact carrying on from the TDB protocol [3], i.e., using secret-sharing ideas to protect against TF. Also, it preserves the lightweight trend.

Moreover, in **SKI**, the chosen f and F have to meet the following requirement. They are such that it is indistinguishable when L_μ is applied to the secret key x and when it is applied to another randomly selected $\bar{x} \in GF(q)^\ell$, even if we are given access to the other messages in the protocol, i.e., N_P, N_V and some results related to $f_x(N_p, N_V, L)$ and $L(\bar{x})$ as per the protocol, or even if we choose them adaptively as an adversary may do. This security-enforcement also has an impact on an additional property of the PRF f (i.e., on how its keys are used outside its calls). This design choice is motivated by the attacks in [6], where a trapdoor choice of N_P or N_V together with x being used in L_μ could lead generalised MF attacks.

The F-functions that we take (see the previous section) enjoy other properties that help attain security in front of DB frauds. E.g., similarly to [3], the F-functions are such

[7] "Suitable" denotes here compliant with deterring the TF in [20].

that knowing the complete table of the response-function F for a given c_i leaks x'_i, yet knowing only up to t' entries challenge-response in this table discloses no information about x'_i. Please refer to [5] for details.

The Set $\mathcal{L}$ of Transforms. The idea of the set $\mathcal{L}$ is that, when leaking some noisy versions of $L(x)$ for some chosen $L \in \mathcal{L}$, the adversary can reconstruct x without noise.

We introduced this transformation in order to protect against a TF observed by Hancke [20]. In this attack, a malicious prover could select a noise-vector e of Hamming weight $n - \tau$ and provide a slightly modified, but full table of all $c_i \mapsto F(c_i, a_i, x_i)$ functions. The modification in the table is as follows: if $e_i = 1$, then the output of $F(v, a_i, x_i)$ is flipped, where $v \in \{1, \ldots, t\}$ is one, randomly chosen value of the possible values for the i-th challenge c_i. Assuming that the adversary has a powerful device which can answer to V without noise, then this adversary passes with probability $\gamma = 1$. Then, an adversary –out of this full table– can reconstruct $x + e$. Then, x cannot be recovered efficiently by the adversary (whilst P^* substantially helped the adversary towards passing the protocol).

Recall that –in our protocols– the "master-secret" that f is applied upon for one value of c_i is not necessarily the shared key x, but instead it is x' with $x' = L(x)$, where L is the transformation chosen by V in the initialisation phase of the protocol and discussed above. As we said, this offers better protection against new types of threats: by introducing $L(x)$ instead of x inside F, then in Hancke's attack, the adversary can get to learn $L(x) + e$.

Imagine now a dishonest prover as above choosing a noise-vector e of Hamming weight (at most) t, with e possibly depending on x and a transformation L chosen in the current run initialisation phase. If $\mathcal{L}_{bit}$ is used as in our protocols, then in n rounds of the attack as per the above, an attacker $\mathcal{A}$ deduces $\mu \cdot x$, for all obtained μ in the round-transformations $L = L_\mu (L \in \mathcal{L}_{bit})$. The attacker does so by computing the majority of the vector x' that he learns[8] out of the responses. These values $\mu \cdot x$ can be correct with a high probability, if $t = HW(e)$ is not too close to n, i.e., t is at least less than $\frac{n}{2}$. (HW denotes the Hamming weight.) Then, $\mathcal{A}$ can solve a linear system to get x. Hence, leaking x makes this attack *not* a valid terrorist-fraud (since the dishonest prover helped $\mathcal{A}$ pass the protocol, but he also leaked x to this attacker). I.e., our protocol instance with $\mathcal{L}_{bit}$ resists the attack by Hancke [20].

5.4 The Main Instances of SKI

We now propose the most interesting instances of **SKI**: the first one protecting against all threats presented in Section 3 (and rendering the TF scenario by Hancke [20] hard to mount for some parameters, if not infeasible), and a second one, much more lightweight, not offering TF-resistance, but only DF- and MF-resistance. Of course, the spectrum of the class **SKI** is much larger, and we will touch upon that in our appendix.

- $\mathbf{SKI_{pro}}$: defined by the response-function $\mathbf{F_{xor}}$ above, with $q = 2$, $t = 3$, $t' = 2$, i.e., $F(c_i, a_i, x'_i) = (a_i)_{c_i}$ for $c_i \in \{1, 2\}$ and $F(3, a_i, x'_i) = x'_i + (a_i)_1 + (a_i)_2$, with $(a_i)_1, (a_i)_2, x_i \in GF(2)$, and $\mathcal{L} = \mathcal{L}_{\mathsf{bit}}$;

[8] We presume that if you know the full table of the response for a given c_i, then this leaks x'. Our F functions are like that.

– $\mathbf{SKI_{lite}}$: defined by a variant of response-function $\mathbf{F_{xor}}$ above (not depending on x'_i), with $q = 2$, $t = t' = 2$, i.e., $F(c_i, a_i, x'_i) = (a_i)_{c_i}$ for $c_i \in \{1,2\}$, with $(a_i)_1, (a_i)_2 \in GF(2)$, and $\mathcal{L} = \emptyset$. (In $\mathbf{SKI_{lite}}$, $\mathcal{L} = \emptyset$ since in the response-function F there is no x' used, as TF-resistance is not envisaged by this instance.)

Note, once more, that both protocols are very inexpensive computationally.

5.5 Security Analysis

In this section, we simply describe the best-known attack strategies for mounting DB frauds onto **SKI**. We report the security analysis for $\mathbf{SKI_{pro}}$, made in a symbolic form (i.e, on variables t, q, on the function F, etc). The analysis for $\mathbf{SKI_{lite}}$ is omitted, as it follows exactly the same principles (where eventually just the numerical values for t, q, or the expression of F would change). In an extended version of this work [5], we will give the formal proofs showing that these attacks are indeed the best attainable attacks against **SKI**, i.e., their probabilities of success can be shown to be the actual provable security bounds.

DF-resistance for $\mathbf{SKI_{pro}}$. Intuitively, to defeat DF-resistance, the dishonest, far-away prover P^* has to anticipate the challenge before it reaches him, to compute the response-function F with the challenge as one of the arguments, and to do so as early as possible. Then, he needs to send the resulting response pre-emptively. So, in real terms, this P^* is computing the preimage of a map $c_i \mapsto F(c_i, a_i, x'_i)$ and he gets more successful at mounting this fraud as this computable preimage gets larger. (Note that the size of this computable preimage depends on some random choice, i.e., on the value selected for a_i).

We recall that our response-function for $\mathbf{SKI_{pro}}$, taken on the i-th DB round, is as follows: $F(c_i, a_i, x'_i) = (a_i)_{c_i}$ for $c_i \in \{1,2\}$ and $F(3, a_i, x'_i) = x'_i + (a_i)_1 + (a_i)_2$. (Let us assume a fixed transformation L that gives x': e.g., as for $\mathbf{SKI_{pro}}$, one L_μ chosen in such a protocol round; this does not affect the rest).

So, the best case for P^* to invert $c_i \mapsto F(c_i, a_i, x'_i)$, i.e., to get the right answer on an "anticipated" challenge, is when $(a_i)_1 = (a_i)_2 = x_i$. In this case, he would know the answer, no matter which of the 3 values c_i actually takes, i.e., the preimage of the map $c_i \mapsto F(c_i, a_i, x'_i)$ has size 3. Over the choice of a_i, this would happen with probability $\frac{1}{4}$. If you look further at the response-function, you will note that it is impossible to invert the map onto one specific value of c_i, i.e., to make the aforementioned preimage have size 1. As the preimage can only have size 1, 2 or 3, then the prover narrowing his correct answers over a space of 2 values for the challenges can happen with probability $1 - \frac{1}{4}$.

So, the expected value of the size of this pre-image over the choices of a_i (i.e., the expected number of values for the challenge c_i that the prover could anticipate the answer for) is $(2 \times \frac{3}{4} + 3 \times \frac{1}{4}) = \frac{9}{4}$.

Remember that in $\mathbf{SKI_{pro}}$, in total, we have $t = 3$ values for any challenge. So, given x' fixed, each iteration has a probability to succeed equal of $\frac{9}{4} \times \frac{1}{3} = \frac{3}{4}$.

We note that there is no other mechanism that this prover P^* could pull through. For instance, since it is the verifier who chooses a and M, the distribution of the a_i's is

uniform, i.e., not influenced by a possible trapdoor choice of N_P from P^*. (This excludes the DF attacks in [6].)

So, we have just given the description of the best (mathematical) strategy of P^* to mount a DF, which passes with a probability of $(\frac{3}{4})^n$, if no noise is considered. (This is as reported on our initial table, Table 1, on page 99.).

If, in turn, we do consider noise, then the overall success probability is going to be $B(n,\tau,\frac{3}{4}) = \sum_{i=\tau}^{n} \binom{n}{i} (\frac{3}{4})^i (\frac{1}{4})^{n-i}$. Then, for $\tau > (\frac{3}{4}+\varepsilon)n$, we have $B(n,\tau,\frac{3}{4})$ less than $e^{-2\varepsilon^2 n}$, for some $\varepsilon > 0$ (by the Hoeffding bound [22]). □

MF-resistance for $\mathbf{SKI_{pro}}$. Assume a mafia-fraud attacker $\mathcal{A}$ taking part, up to his capabilities, in an interaction between P and V.

Assuming that the attacker does not learn anything conclusive during the initialisation phase (about x or how to respond in the DB phase)[9], the probability of him succeeding in this fraud rests only on giving (by chance) the correct answers to the challenges sent by V (before P does so); I.e., the probability of $\mathcal{A}$ of succeeding in this MF is given by

$$p = \prod_{1\le i\le n} \Pr_{c_i\in\{1,\ldots,t\}}[r_i \text{ being correct for } c_i | c_i \text{ is sent by } V].$$

Getting r_i correct for c_i can be attained in two distinct ways: 1. in the event $e1$ of guessing $c_i' = c_i$ and sending it beforehand to such a P_j and getting the correct response r_i, or 2. in the event $e2$ of simply guessing the correct answer r_i (for a challenge $c_i' \neq c_i$).

So the probability of success in one round is $\Pr[e1]+\Pr[e2] = \frac{1}{t} + \frac{t-1}{t} \times \frac{1}{q}$. In $\mathbf{SKI_{pro}}$, $t = 3$ and $q = 2$, so we get a concrete, overall p of $(\frac{2}{3})^n$, if no noise is considered within the communications. (This is as reported on our initial table, Table 1, on page 99.).

If we consider the noise of the channels, then we get $p = B(n,\tau,\Pr[e1]+\Pr[e2]) = B(n,\tau,\frac{2}{3})$. Then, for $\tau > (\frac{2}{3}+\varepsilon)n$, we have $B(n,\tau,\frac{2}{3})$ less than $e^{-2\varepsilon^2 n}$, for some $\varepsilon > 0$ (by the Hoeffding bound [22]). □

TF-resistance for $\mathbf{SKI_{pro}}$. We split the discussion in two analyses: I. for noiseless communication; II. for noisy communications.

[9] In fact, we can argue that this is the case in the **SKI** protocols. With high probability, there is no collision between the nonces N_P and N_V (if the space of the nonces is large enough, e.g., $2^{\Omega(n)}$). So, the output of the PRF instance f_x is not biased by the choices of these values. So, $\mathcal{A}$ learns nothing from this. Also, in $\mathbf{SKI_{pro}}$, it is not x that is used in F directly, but $L(x)$ is. Moreover, as we stated in the description of the design, the chosen F for $\mathbf{SKI_{pro}}$ is such that it is indistinguishable when it is applied to the secret key x and when it is applied to another randomly selected $x' \in GF(q)^\ell$, even if we are given access to the other messages in the protocol, i.e., N_P,N_V, or we choose them adaptively as $\mathcal{A}$ may do. (We leave the complete formalism and proof of this for an extended version of this paper [5].) So, due to this indistinguishability, it is as if the shared secret key x were not used outside the f-keying procedure. Given the above and the standard PRF assumption, it means that $\mathcal{A}$ seeing the output of f_x equates to him seeing the output of a real-random function, i.e., for $\mathcal{A}$, it is as if a were chosen at random. So, no "good" strategy comes out from the observed protocol transcript. So, there is no better strategy but what we say in the analysis above.

I. Let us assume first that there are noiseless conditions.

As it is traditional in TF analysis, let us assume that the dishonest prover P^* gives away information to help $\mathcal{A}$. Namely, suppose that: 1. for $n-\nu$ DB rounds, the adversary has got all responses, irrespective of the value of challenges; 2. for the other ν DB rounds, for each such round, the adversary knows the answers for $t-1$ (out of t) possible values for a challenge. Then, his best chances to succeed is to get from V the challenges that he knows how to answer to (in the ν "decisive" rounds), i.e., chances of $(\frac{t-1}{t})^\nu$.

(This translates in the case of noiseless conditions to the $(\frac{2}{3})^\nu$ bound, reported on our initial table, Table 1, on page 99, for $\mathbf{SKI_{pro}}$.)

II. Let us assume now that the communications are noisy.

For concreteness, let us assume that the threshold of noise acceptance is τ out of n.

As per the strong, new attack by Hancke [20], assume that the dishonest prover P^* chose a noise bit-vector e with $HW(e) = \frac{n}{2}$. Further assume that e deterministically depends on x and L, i.e., $e = g(x,L)$ for some function g (i.e., P^* does not choose e adaptively based on the transformation L and on x). Then, assume that this P^* leaked the response table with noise e. I.e., for each i with $e_i = 0$, P^* leaked the full $c \mapsto F(c,a_i,x'_i)$ table; for each i with $e_i = 1$, P^* leaked the table except that for some random c^*_i, for which the response-value $F(c^*_i,a_i,x'_i)$ was flipped. Clearly, the leakage property[10] of F makes sure that $\mathcal{A}$ learns L and $L(x)+g(x,L)$. Due to the structure of L, the latter is a vector of Hamming weight $\frac{n}{2}$. If g has some good property, this is indistinguishable from L and $L(x)+g(y,L)$ for x and y independent. But $g(y,L)$ perfectly randomizes $L(x)$. So, it does not help to give any information about x to $\mathcal{A}$. Without knowing x, $\mathcal{A}$ cannot predict any response in another session[11] with new nonces (to compute the a vector), so $\mathcal{A}$ has no advantage to succeed in the protocol. Therefore, for any strategy, γ' is negligible.

Concretely, the probability that P^* manages to help $\mathcal{A}$ succeed in the protocol during the terrorist-fraud is the probability that at least τ rounds give a correct answer. Clearly, the $\frac{n}{2}$ rounds for which $e_i = 0$ will be correct for sure. The others are correct with probability $\frac{t-1}{t}$. So, we have $\gamma = B(\frac{n}{2}, \tau - \frac{n}{2}, \frac{t-1}{t})$.

For $\tau - \frac{n}{2} > (\frac{t-1}{t} + \varepsilon) \times \frac{n}{2}$ and some positive ε, we have γ is lower than $e^{-2\varepsilon^2 \frac{n}{2}}$ (due again to the Hoeffding bound [22]). That is, for the latter, we need $\tau > \frac{5}{6}n + \varepsilon\frac{n}{2}$ (since our t is 3). This simply comes down to taking τ slightly bigger than $\frac{5}{6}n$. □

As we mentioned above, this analysis extends almost identically to $\mathbf{SKI_{lite}}$, i.e., up to some changes in values. As we saw, the attack bounds are obtained provided that the design-blocks inside **SKI** (i.e., the PRF f, the response-function F, their inter-play

[10] This holds as we assume that the response-function F is such that knowing the complete table of the response-function F for a given c_i leaks x'_i.

[11] As we saw in the proof for the MIM attack, there is no other advantage that this attacker can get on his own, in such runs.

within the rest of the design, the transformations $\mathcal{L}$, etc.) meet some requirements (e.g., f and F are such that within the protocol-exchanges it does not show the fact that F uses x inside, $\mathcal{L}$ contains linear transformations, M masks $f_x(\ldots)$, etc.). All these can be formalised further and then all these attack strategies can be transformed into proofs of provable security bounds for the whole **SKI** class, i.e., all conditioned by and parametrised in F, $\mathcal{L}$, f, t, t', q, ℓ, n, τ.

6 Conclusions

We note again the similar best-attack bounds stated in Table 1 for the protocol in [21], by Hancke and Kuhn, and our simplest version of **SKI**, namely $\mathbf{SKI_{lite}}$. The Swiss-Knife protocol [25] and the Avoine *et al.* [3] also seems to enjoy good security bounds for DF and MF, but they do not protect against the new TF attack by Hancke [20].

Moreover, it was shown in [6] that the conditions on the underlying primitives need to be strengthened for DF and generalised MF security to be indeed attained. This type of attacks as in [6] can be bypassed in the **SKI** protocols. I.e., when the PRF instance is used, it is masked with the randomly looking value M, computed on the right side of the protocol avoiding the DF susceptibility shown in [6]. In our best MF description for $\mathbf{SKI_{pro}}$, we explained the idea of choosing an f and an F that together with the protocol transcript make F look as if it is not using x; in our further work, we will give the formal details of how such a PRF f needs to come together with the response-function F to attain formally the avoidance of the generalised MF exposed in [6]. We remind that we also introduced a transform on x to be used inside the response-function F in order to deter (if not avoid) the recent TF attacks by Hancke [20]. So, by all this and beyond Table 1, we conclude that very compelling security—in accordance to the recent developments in DB—is now provided by (at least one of) the **SKI** protocols.

References

1. Aumasson, J.-P., Mitrokotsa, A., Peris-Lopez, P.: A Note on a Privacy-Preserving Distance-Bounding Protocol. In: Qing, S., Susilo, W., Wang, G., Liu, D. (eds.) ICICS 2011. LNCS, vol. 7043, pp. 78–92. Springer, Heidelberg (2011)
2. Avoine, G., Bingöl, M., Kardas, S., Lauradoux, C., Martin, B.: A Framework for Analyzing RFID Distance Bounding Protocols. Journal of Computer Security 19(2), 289–317 (2011)
3. Avoine, G., Lauradoux, C., Martin, B.: How Secret-sharing can Defeat Terrorist Fraud. In: Proceedings of the 4th ACM Conference on Wireless Network Security – WiSec 2011, Hamburg, Germany. ACM Press (June 2011)
4. Bay, A., Boureanu, I., Mitrokotsa, A., Spulber, I., Vaudenay, S.: The Bussard-Bagga and Other Distance-Bounding Protocols under Attacks. In: Kutyłowski, M., Yung, M. (eds.) Inscrypt 2012. LNCS, vol. 7763, pp. 371–391. Springer, Heidelberg (2013)
5. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: Provably Secure Authenticated Distance-Bounding (submitted)
6. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: On the Pseudorandom Function Assumption in (Secure) Distance-Bounding Protocols. In: Hevia, A., Neven, G. (eds.) LatinCrypt 2012. LNCS, vol. 7533, pp. 100–120. Springer, Heidelberg (2012)

7. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: On secure distance bounding (extended abstract). In: The Early Symmetric Crypto Seminar, ESC 2013, pp. 52–60 (2013) ISBN 978-99959-814-0-2
8. Boureanu, I., Mitrokotsa, A., Vaudenay, S.: Towards secure distance bounding. In: The 20th Anniversary Annual Fast Software Encryption, FSE 2013 (to appear, 2013)
9. Brands, S., Chaum, D.: Distance-Bounding Protocols (Extended Abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 344–359. Springer, Heidelberg (1994)
10. Bussard, L., Bagga, W.: Distance-Bounding Proof of Knowledge Protocols to Avoid Terrorist Fraud Attacks. Technical Report RR-04-109, EURECOM (May 2004)
11. Bussard, L., Bagga, W.: Distance-bounding proof of knowledge to avoid real-time attacks. In: Sasaki, R., Qing, S., Okamoto, E., Yoshiura, H. (eds.) Security and Privacy in the Age of Ubiquitous Computing. IFIP AICT, vol. 181, pp. 223–238. Springer, Boston (2005)
12. Čapkun, S., Buttyán, L., Hubaux, J.-P.: SECTOR: Secure Tracking of Node Encounters in Multi-hop Wireless Networks. In: ACM Workshop on Security of Ad Hoc and Sensor Networks - SASN, pp. 21–32. ACM (2003)
13. Cremers, C., Rasmussen, K.B., Čapkun, S.: Distance hijacking attacks on distance bounding protocols. In: IEEE Symposium on Security and Privacy, pp. 113–127 (2012)
14. Dimitrakakis, C., Mitrokotsa, A., Vaudenay, S.: Expected Loss Bounds for Authentication in Constrained Channels. In: Proceedings of INFOCOM 2012, Orlando, FL, USA, March 2012, pp. 478–85. IEEE press (March 2012)
15. Drimer, S., Murdoch, S.J.: Keep your enemies close: distance bounding against smartcard relay attacks. In: Proceedings of 16th USENIX Security Symposium, Berkeley, CA, USA, pp. 7:1–7:16. USENIX Association (2007)
16. Dürholz, U., Fischlin, M., Kasper, M., Onete, C.: A Formal Approach to Distance-Bounding RFID Protocols. In: Lai, X., Zhou, J., Li, H. (eds.) ISC 2011. LNCS, vol. 7001, pp. 47–62. Springer, Heidelberg (2011)
17. Fischlin, M., Onete, C.: Subtle kinks in distance-bounding: an analysis of prominent protocols. In: WISEC, pp. 195–206 (2013)
18. Goldreich, O.: Foundations of Cryptography, vol. 1. Cambridge University Press, New York (2006)
19. Özhan Gürel, A., Arslan, A., Akgün, M.: Non-uniform Stepping Approach to RFID Distance Bounding Problem. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cavalli, A., Leneutre, J. (eds.) DPM 2010 and SETOP 2010. LNCS, vol. 6514, pp. 64–78. Springer, Heidelberg (2011)
20. Hancke, G.P.: Distance-bounding for RFID: Effectiveness of 'terrorist fraud' in the presence of bit errors. In: RFID-TA, pp. 91–96 (2012)
21. Hancke, G.P., Kuhn, M.G.: An RFID Distance Bounding Protocol. In: SECURECOMM, pp. 67–73. ACM (2005)
22. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
23. Kapoor, G., Zhou, W., Piramuthu, S.: Distance Bounding Protocol for Multiple RFID Tag Authentication. In: Xu, C.-Z., Guo, M. (eds.) Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, EUC 2008, Shanghai, China, vol. 02, pp. 115–120. IEEE Computer Society (December 2008)
24. Kim, C.H., Avoine, G.: RFID Distance Bounding Protocol with Mixed Challenges to Prevent Relay Attacks. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 119–133. Springer, Heidelberg (2009)
25. Kim, C.H., Avoine, G., Koeune, F., Standaert, F.-X., Pereira, O.: The Swiss-Knife RFID Distance Bounding Protocol. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 98–115. Springer, Heidelberg (2009)

26. Mitrokotsa, A., Dimitrakakis, C., Peris-Lopez, P., Hernandez-Castro, J.C.: Reid et al.'s Distance Bounding Protocol and Mafia Fraud Attacks over Noisy Channels. IEEE Communications Letters 14(2), 121–123 (2010)
27. Mitrokotsa, A., Onete, C., Vaudenay, S.: Mafia Fraud Attack against the RČ Distance-Bounding Protocol. In: Proceedings of the 2012 IEEE RFID Technology and Applications (IEEE RFID T-A), Nice, France, pp. 74–79. IEEE Press (November 2012)
28. Mitrokotsa, A., Peris-Lopez, P., Dimitrakakis, C., Vaudenay, S.: On selecting the nonce length in distance-bounding protocols. The Computer Journal (2013)
29. Munilla, J., Peinado, A.: Distance Bounding Protocols for RFID Enhanced by Using Void-challenges and Analysis in Noisy Channels. Wireless Communications and Mobile Computing 8, 1227–1232 (2008)
30. Munilla, J., Peinado, A.: Security Analysis of Tu and Piramuthu's Protocol. In: New Technologies, Mobility and Security – NTMS 2008, Tangier, Morocco, pp. 1–5. IEEE Computer Society (November 2008)
31. Munilla, J., Peinado, A.: Attacks on a Distance Bounding Protocol. Computer Communications 33, 884–889 (2010)
32. Nikov, V., Vauclair, M.: Yet Another Secure Distance-Bounding Protocol. In: Proceedings of the Conference on Security and Cryptography (SECRYPT 2008), pp. 218–221 (July 2008)
33. Rasmussen, K.B., Čapkun, S.: Location Privacy of Distance Bounding. In: Proceedings of the Annual Conference on Computer and Communications Security (CCS), pp. 149–160. ACM (2008)
34. Reid, J., Nieto, J.M.G., Tang, T., Senadji, B.: Detecting Relay Attacks with Timing-based Protocols. In: ASIACCS 2007: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security, pp. 204–213. ACM (2007)
35. Singelée, D., Preneel, B.: Distance Bounding in Noisy Environments. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 101–115. Springer, Heidelberg (2007)
36. Toiruul, B., Lee, K.O., Kim, J.M.: SLAP - A Secure but Light Authentication Protocol for RFID Based on Modular Exponentiation. In: International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 29–34 (November 2007)
37. Tu, Y.-J., Piramuthu, S.: RFID Distance Bounding Protocols. In: Proceedings of the First International EURASIP Workshop on RFID Technology (2007)

A Other Instances of SKI

In this section, we present two more instances of **SKI**: one with even stronger security guarantees than what we have seen so far and another placing its assurances in between $\mathbf{SKI_{pro}}$ and $\mathbf{SKI_{lite}}$. For that, we first provide a new response-function.

Other instances of the response-function F. For the strongest version of **SKI**, we recommend the following response function.

$$\mathbf{F_{shamir}}(c_i, a_i, x_i') = x_i' + (a_i)_1\overline{c}_i + (a_i)_2{\overline{c}_i}^2 + \ldots + (a_i)_{t-1}{\overline{c}_i}^{t-1}$$

where $x_i' \in GF(q)$, $q \geq 4$, $c_i \in \{1,\ldots,t\}$ is mapped to $\overline{c}_i \in GF(q)^*$ by an arbitrary injective mapping, $(a_i)_j \in GF(q), j \in \{1,\ldots,t-1\}$;

It is obvious, given the expression of $\mathbf{F_{shamir}}$ (i.e., with x' inside it) that it is meant to protect against classical TF. If $x' = x$, then it may not protect against the newest TF scenario by Hancke [20].

Other instances of SKI. We present two more instances of **SKI** in descending order of their security strength:

- $\mathbf{SKI_{shamir}}$: defined by the response-function $\mathbf{F_{shamir}}$ above, with $q = 4$, $t = 3$, $t' = 2$, i.e., $F(c_i, a_i, x_i') = x_i' + (a_i)_1\overline{c}_i + (a_i)_2{\overline{c}_i}^2$, with $x_i, (a_i)_1, (a_i)_2 \in GF(4)$ and $\overline{c}_i \in GF(4)^*$; $\mathcal{L} = \mathcal{L}_{\mathsf{bit}}$
- $\mathbf{SKI_4}$: defined by the response-function $\mathbf{F_{xor}}$ above, with $q = 2$, $t = 4$, $t' = 3$, i.e., $F(c_i, a_i, x_i') = (a_i)_{c_i}$ for $c_i \in \{1,2,3\}$ and $F(4, a_i, x_i) = x_i' + (a_i)_1 + (a_i)_2 + (a_i)_3$, with $(a_i)_1, (a_i)_2, (a_i)_3, x_i \in GF(2)$; $\mathcal{L} = \mathcal{L}_{\mathsf{bit}}$.

As we can see, $\mathbf{SKI_{shamir}}$ is more secure even than $\mathbf{SKI_{pro}}$, with a response-function F, based on Shamir's secret-sharing scheme. In fact, recalling our DF-resistance analysis, this response-function F is more powerful when it comes to inverting the map $c_i \mapsto F(\cdot, a_i, x_i)$. Hence, the DF-resistance of this instance is better than the one of $\mathbf{SKI_{pro}}$. The opposite can be said about $\mathbf{SKI_4}$, by comparison to $\mathbf{SKI_{pro}}$.

Security of these instances. Doing an analysis similar to the one we did for $\mathbf{SKI_{pro}}$ in Section 5.5, and we consider noisy conditions, we can state the following bounds for the best mounted DF and MF attacks against these new instances:

	$\mathbf{SKI_{shamir}}$	$\mathbf{SKI_4}$
DF	$\alpha = B(n, \tau, \frac{5}{8})$	$\alpha = B(n, \tau, \frac{3}{4})$
MF	$\beta = B(n, \tau, \frac{1}{2})$	$\beta = B(n, \tau, \frac{5}{8})$

If, in turn, we do not consider noisy conditions then we get the following probabilities for the best-known TF attacks against these instances of **SKI**:

	$\mathbf{SKI_{shamir}}$	$\mathbf{SKI_4}$
TF	$(\frac{2}{3})^v$	$(\frac{3}{4})^v$

Cryptanalysis and Improvement of a Provably Secure RFID Ownership Transfer Protocol

Daisuke Moriyama

National Institute of Information and Communications Technology, Japan
dmoriyam@nict.go.jp

Abstract. Radio Frequency IDentifications (RFID) are useful low-cost devices for identification or authentication systems through wireless communication. The ownership of the RFID tag is frequently changed in the life cycle of the tag, it may fall in to the hands of a malicious adversary. The privacy problem in this situation is studied in the RFID ownership transfer protocol. However, almost all literatures provide only heuristic analysis and many protocols are broken. Elkhiyaoui et al. defined the security model for RFID ownership transfer protocols and proposed the detailed security proof to their protocol, but we show that their protocol does not provide enough privacy and cover the realistic attack. We investigate a suitable security model for RFID ownership transfer protocols and provide a new provably secure RFID ownership transfer protocol.

1 Introduction

Recently various technologies are developed to construct smart communications with digital data. Especially, Internet of Things (IoT) and Machine-to-Machine (M2M) architectures among them define that networked devices automatically communicate each other without human to accomplish high speed information transaction. One of the core building blocks to construct IoT or M2M systems is Radio Frequency Identification (RFID). The current RFID tags are used in logistics for product management, identification of animals, etc. RFID tags have high readability by wireless communication and parallel processing in comparison with barcodes. Thus RFID tag is useful device, but its privacy is a critical issue in commercial usage. So the RFID authentication protocols and ownership transfer protocols which focus on the life cycle are discussed in a few years [12–14, 16, 20].

The best way to support security and privacy is to provide security proof based on a cryptographic security model. The security model for canonical RFID authentication protocols is proposed by several researchers [10, 18, 21] and the main task for protocol designers is to provide a new protocol and describe a concrete security proof. However, when we focus on the RFID ownership transfer protocol, one of the applications of RFID authentication protocol, almost all literatures provide security and privacy with heuristic analysis only. They briefly describe the requirements for RFID ownership transfer or provide only intuition of the security proof for security and privacy, so many protocols are broken by the other researchers [5, 6, 11, 17, 19].

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 114–129, 2013.

To the best of our knowledge, the provably secure RFID ownership transfer protocol was only provided by Elkhiyaoui et al. in RFIDSec 2011 [7, 8]. They also introduced a security model for RFID ownership transfer protocols, but we remark that this model has the following strong restrictions:

- The malicious adversary cannot choose the target tag to attack
- The malicious adversary cannot obtain any protocol instructions when the tag resynchronizes to the RFID reader
- There is no malicious owner who participates in the ownership transfer protocol.

These are different from the security model for canonical RFID authentication protocols, the realistic usage and intuition of the privacy issue for ownership transfer, respectively.

In this paper, we analyze their protocol and illustrate a concrete attack to their protocol. Therefore, one can think that there is no provably secure RFID ownership transfer protocol. We provide a new security model for RFID ownership transfer protocols based on the existing construction of the RFID ownership transfer protocol. Moreover, we consider the additional requirements that both current and new owners can check the integrity of the peer. Therefore our protocol is resilient to the general impersonation attack in addition to the tag impersonation attack.

2 Security Model for RFID Ownership Transfer Protocols

In general, RFID ownership transfer protocols are proposed with the RFID authentication protocol. So we discuss how to formalize the suitable security model for RFID ownership transfer protocols based on the RFID authentication protocols and its operation.

In the canonical RFID authentication, there is an RFID reader $R \in \mathcal{R}$ ($\mathcal{R}$ is a set of RFID readers) which interacts with multiple RFID tags $\mathcal{T}$ to authenticate each other. Since the resource of the RFID tags is quite limited, only symmetric key primitives are used in many RFID authentication protocols and each tag shares a secret key with the RFID reader. Moreover, RFID tags are very cheap devices and it is difficult to assume the tamper resilience, so the key update mechanism is a desirable property to minimize the information leakage. Briefly speaking, the privacy for RFID authentication protocol requires that no adversary can obtain information about the identity of the target RFID tag from communication messages between the reader and tags.

In contrast, the RFID ownership transfer protocol consists of three phases: setup, authentication and ownership transfer. There are many RFID readers treated as owner, and each reader runs an authentication protocol with their own tags. The ownership transfer phase treats how to move the authority of the authentication. Since the RFID ownership transfer protocol includes authentication phase, we must cover the security and privacy issues for RFID authentication protocols in addition to that for the ownership transfer phase.

The requirements for RFID authentication protocol and ownership transfer protocol are correctness, security and privacy. The correctness for RFID ownership transfer protocol is that the owner and tag accept each other in the authentication phase and the new owner obtains the authorization of the authentication of the tag when the current owner agrees to transfer it in the ownership transfer phase.

2.1 Privacy

There are two fundamental privacy requirements for RFID ownership transfer protocols described in the previous literatures:

- Old ownership privacy: When the ownership is transferred, the past interactions of the tag should not be traced by the new owner, and
- New ownership privacy: When the ownership is transferred, the future interactions of the tag should not be traced by the old owner.

One can solve these privacy problems with the following operation.

S1. The current owner runs RFID authentication protocol with the intended tag (without active adversary) and resynchronizes and updates the shared secret key.
S2. The current owner sends tag's secret key and identity to the new owner.
S3. The new owner initializes the RFID tag and runs the setup algorithm for the RFID authentication protocol.

Then the new ownership privacy is trivially accomplished since the initialized key is independent from the previous one. Instead, we must define a suitable privacy requirement for the RFID authentication protocol to cover the old ownership privacy. Especially, tag's privacy must be preserved even when the secret key is revealed after the key updating algorithm. If the updated secret key gives no information about the past communication between the current owner and the tag, then the old ownership privacy can be satisfied by this resynchronization. Based on the above argument, we introduce a new security model for RFID ownership transfer protocols.

Let k be a security parameter. We consider the following privacy game between adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ and the challenger based on the indistinguishability-based privacy definition proposed in [10]:

$\underline{\mathsf{Exp}^{\mathsf{IND}\text{-}b}_{\Pi,\mathcal{A}}(k)}$
$\{(pk_i, sk_i)\}_i \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^k),\ pk := \{pk_i\}_i;$
$(R_j, t_0^*, t_1^*, st_1) \xleftarrow{\mathsf{R}} \mathcal{A}_1^{\mathsf{ReaderInit},\mathsf{Send},\mathsf{Corrupt},\mathsf{Result},\mathsf{OwnerTrans}}(pk, \mathcal{R}, \mathcal{T});$
$\mathcal{T}' := \mathcal{T} \setminus \{t_0^*, t_1^*\};$
$st_2 \xleftarrow{\mathsf{R}} \mathcal{A}_2^{\mathsf{ReaderInit},\mathsf{Send},\mathsf{Result}}(\mathcal{R}, \mathcal{T}', \mathcal{I}(t_b^*), st_1);$
$\pi_b \xleftarrow{\mathsf{R}} \mathsf{Execute}(R_j, t_b^*), \pi_{1-b} \xleftarrow{\mathsf{R}} \mathsf{Execute}(R_j, t_{1-b}^*);$

$b' \xleftarrow{\mathsf{R}} \mathcal{A}_3^{\mathsf{OwnerTrans}}(sk_{j+1}, \pi_b, \pi_{1-b}, st_2)$:
Output b'

The challenger runs the setup algorithm and sends all public parameter including public keys of all owners. Then the adversary can issue the following queries. $\mathsf{ReaderInit}(1^k)$ activates the reader and it outputs the new session ID, $\mathsf{Send}(m)$ allows the adversary to send arbitrary message to the reader and tags, $\mathsf{Corrupt}(t)$ responds tag's secret key, $\mathsf{Result}(sid)$ outputs the authentication result of the session. $\mathsf{OwnerTrans}(R_i, R_{i+1}, t)$ is the interactive ownership transfer algorithm to transfer the ownership of the tag t from current owner R_i to new owner R_{i+1}. After the interaction, the adversary chooses two tags t_0^* and t_1^* whose ownership is R_j. Then the challenger flips a coin b and allows the adversary to interact with the challenge tag t_b^* anonymously in the challenge phase. An intermediate algorithm $\mathcal{I}$ relays the communication message to accomplish the anonymous access. If the adversary finishes this interaction, the authentication phase is executed by the current owner and two tags through the $\mathsf{Execute}$ query for resynchronization. If we allow the adversary to interrupt or modify the communication message in this phase, the $\mathsf{Execute}$ query becomes meaningless so we assume that the adversary does not participate in this interaction. Instead, the adversary obtains these transcripts and the secret key of the new owner so as to transfer the ownership. Finally, the adversary output a bit b'. We assume that $t_0^* \neq t_1^*$ and the adversary cannot issue the $\mathsf{Corrupt}$ query to (t_0^*, t_1^*). The advantage of the adversary in the above game is defined by

$$\mathsf{Adv}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(k) = \left| \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}0}}_{\Pi,\mathcal{A}}(k) \to 1] - \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}1}}_{\Pi,\mathcal{A}}(k) \to 1] \right|.$$

Definition 1. *An RFID ownership transfer protocol Π holds IND-privacy if for any probabilistic polynomial time adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(k)$ is negligible in k.*

2.2 Security

We introduce the following three security requirements for the RFID ownership transfer protocol:

- The current owner and tag reject the session if an adversary modifies the communication message.
- Any illegitimate party who does not have the ownership of the tag cannot impersonate the current owner.
- The ownership should not be transferred to the unexpected party from the view point of the current owner.

The first issue is just the same as the security requirement for canonical RFID authentication protocols. We define the notion of matching session as follows. An entity has a matching session to the peer if the communication messages between them are successfully transferred until the entity decides the authentication result. The goal of the adversary is to establish a session such that the current owner or uncorrupted tag accepts without the matching session, The advantage of the adversary $\mathcal{A}$ is evaluated by $\mathsf{Adv}^{\mathsf{RAuth}}_{\Pi,\mathcal{A}}(k)$.

Definition 2. *An RFID authentication protocol Π holds basic security if* $\mathsf{Adv}^{\mathsf{RAuth}}_{\Pi,\mathcal{A}}(k)$ *is negligible for any probabilistic polynomial time adversary* $\mathcal{A}$.

When an RFID ownership transfer protocol is constructed by **S1-S3**, the above security is also necessary to evaluate the security for RFID ownership transfer protocols.

We think the other impersonation attacks must be covered in the ownership transfer protocol since a malicious adversary may impersonate the current or new owner to the other owner without any authorization. Let π' be the protocol transcript in the ownership transfer protocol. $(\mathcal{R}_j, \mathcal{R}_{j+1})$ denotes the current owner and new owner, respectively. Consider that each owner executes the ownership transfer protocol and outputs (b, b') which indicates the peer owner is legitimate or not. If one of the two coins becomes zero, the ownership transfer protocol is terminated. We consider the following game for the above security requirements:

$\underline{\mathsf{Exp}^{\mathsf{ROT}\text{-}c}_{\Pi,\mathcal{A}}(k)}$
$(pk_i, sk_i) \xleftarrow{\mathsf{R}} \mathsf{Setup}(1^k),\ pk := \{pk_i\}_i;$
$(R_j, t^*, st_1) \xleftarrow{\mathsf{R}} \mathcal{A}_1^{\mathsf{ReaderInit},\mathsf{Send},\mathsf{Corrupt},\mathsf{Result},\mathsf{OwnerTrans}}(pk, \mathcal{R}, \mathcal{T});$
$\pi \xleftarrow{\mathsf{R}} \mathsf{Execute}(R_j, t^*);$
If $c = 1$ — $(\pi', \cdot, b) \xleftarrow{\mathsf{R}} \mathsf{OwnerTrans}(\mathcal{A}_2(\pi, st_1), R_{j+1}, t^*)$:
If $c = 2$ — $(\pi', b, \cdot) \xleftarrow{\mathsf{R}} \mathsf{OwnerTrans}(R_j, \mathcal{A}_2(\pi, st_1), t^*)$:
Output b

Different from the privacy game, the adversary cannot obtain any secret information of the owner. The advantage of the adversary against the above game is defined by

$$\mathsf{Adv}^{\mathsf{ROT}\text{-}1}_{\Pi,\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{ROT}\text{-}1}_{\Pi,\mathcal{A}}(k) \to 1],\ \textit{and}$$
$$\mathsf{Adv}^{\mathsf{ROT}\text{-}2}_{\Pi,\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\mathsf{ROT}\text{-}2}_{\Pi,\mathcal{A}}(k) \to 1].$$

Definition 3. *An RFID ownership transfer protocol Π is resilient to owner impersonation attack if for any probabilistic polynomial time adversary $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{ROT}\text{-}1}_{\Pi,\mathcal{A}}(k)$ *and* $\mathsf{Adv}^{\mathsf{ROT}\text{-}2}_{\Pi,\mathcal{A}}(k)$ *are negligible in k.*

3 Elkhiyaoui et al.'s Ownership Transfer Protocol and Privacy Analysis

As explained in Section 1, the provably secure RFID ownership transfer protocol is only introduced by Elkhiyaoui et al. [8]. They described cryptographic security model and provided security proof for their protocol named ROTIV (RFID ownership transfer protocol with issuer verification). In addition to the basic ownership transfer property, ROTIV allows anyone to check the issuer of the RFID tag.

This is accomplished by the following mechanism. The issuer of the RFID tag signs the identity of the RFID tag with digital signature algorithm proposed by Boneh, Lynn and Shacham [3]. This signature is re-encrypted by the public key of the current owner, and the ciphertext is stored in the RFID tag. When someone wants to check the issuer of the tag, the current owner decrypts it and sends the digital signature. The detailed protocol specification is the following. Their protocol uses bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ which satisfies $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for any $a, b \in \mathbb{Z}_q$ and $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$. $(\mathsf{MAC.Gen}, \mathsf{MAC.Sign}, \mathsf{MAC.Ver})$ is the algorithm for message authentication code (MAC) and PRF is a pseudorandom function.

3.1 Protocol Description

Setup Phase. First, the issuer runs the following setup algorithm:

I1. Generate $g_1 \xleftarrow{\mathsf{U}} \mathbb{G}_1, g_2 \xleftarrow{\mathsf{U}} \mathbb{G}_2, z \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and compute $g_1' := g_1^z, g_2' := g_2^z$ where $|\mathbb{G}_1| = |\mathbb{G}_2| = k$ (k is a security parameter).
I2. Publish $pk_0 := g_2'$ and keep $sk_0 := (z, g_1')$ as the secret key.
I3. Choose $\alpha_j \xleftarrow{\mathsf{U}} \mathbb{Z}_q$, compute $(g_1'', g_2'') := g_1^{\alpha_j^2}, g_2^{\alpha_j})$, and send public key $pk := (g_1'', g_2'')$ and secret key $sk := \alpha_j$ to the j-th owner.
I4. Choose hash function $H : \mathbb{Z}_q \to \mathbb{G}_1$ and set $u := 1, v := H(t)^z$ where t is the identity of the RFID tag (v is the digital signature).
I5. Select the secret key for the MAC $k_0 \xleftarrow{\mathsf{R}} \mathsf{MAC.Gen}(1^k)$ and send $(s, s', x, y) := (k_0, k_0, t, v)$ to the first owner.

Upon receiving (s, s', x, y), the owner runs the following:

O1. Select $r_1 \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and set $u_1 := g_1^{r_1}, v_1 := (g_1'')^{r_1} \cdot y$.
O2. Send $c := (s, u_1, v_1)$ to the RFID tag.

Note that (u_1, v_1) is the ciphertext of the ElGamal encryption which the plaintext is y and its decryption key is maintained by the owner.

Authentication Phase. ROTIV is 3-move RFID authentication as follows:

A1. The owner generates $n_0 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and sends it to the RFID tag.
A2. The tag chooses $n_1 \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and computes $\sigma_1 := \mathsf{MAC.Sign}(s, (n_0, n_1, u_1, v_1))$. $(n_1, u_1, v_1, \sigma_1)$ is responded to the owner.
A3. The owner computes $y' := v_1/(u_1)^{\alpha_j^2}$ and verifies $y' = y$. If the verification fails, he terminates the protocol. Set $k_0 := s$ if $\mathsf{MAC.Ver}(s, (n_0, n_1, u_1, v_1), \sigma_1) = 1$ and $k_0 := s'$ if $\mathsf{MAC.Ver}(s', (n_0, n_1, u_1, v_1), \sigma_1) = 1$. Otherwise, the protocol is terminated. If either check is accepted, he selects $r_2 \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and computes $u_2 := g_1^{r_2}, v_2 := (g_1'')^{r_2} \cdot y$. That is, the decrypted signature is encrypted by the owner again (with another random coin). Then the MAC is computed as $\sigma_2 := \mathsf{MAC.Sign}(s, (n_1, u_2, v_2))$ and (u_2, v_2, σ_2) is sent to the tag. Finally, the secret key of the owner is updated to $s := k_j, s' := \mathsf{PRF}(k_0, n_0)$.

A4. The tag verifies $\mathsf{MAC.Ver}(k_0, (n_1, u_2, v_2), \sigma_2) = 1$ and terminates the protocol if it fails. Otherwise, it computes $s' := \mathsf{PRF}(k_0, n_0)$ and updates its secret key from (s, u_1, v_1) to (s', u_2, v_2).

Ownership Transfer Phase. The current and new owners are interacted as follows:

T1. The new owner chooses $n_0 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and sends n_0 to the tag.
T2. The tag runs **A2** and outputs $(n_1, u_1, v_1, \sigma_1)$ to the new owner.
T3. The new owner selects $r' \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and generates $A := u_1^{r'}$. $(n_1, u_1, v_1, \sigma_1, A)$ is sent to the current owner.
T4. The current owner verifies the RFID tag as the authentication phase. If the authentication is accepted, he sets $A' := A^{\alpha}$ and sends (s, t, y, A') to the new owner.
T5. The new owner checks

$$e(H(t), g_2') = e(y, g_2),$$
$$e(A^{\alpha}, g_2) = e(A, g_2'),$$
$$e(v_1, g_2)^{r'} = e(y, g_2)^{r'} e(A', g_2'').$$

When all equations hold, the new owner decides that the ownership is honestly transferred and runs **O1** and **O2**.

3.2 Privacy Problems

Passive Attack. Consider the situation that an malicious adversary activates t_0^* in the authentication phase. If the adversary eavesdrops the communication, (u_2, v_2) is sent from the current owner. The adversary sends t_0^* and another tag to the challenger in the privacy game and eavesdrops the communication. If the same message (u_2, v_2) is incoming from the challenge tag, the adversary decides that t_0^* is chosen as the challenge tag. Thus the adversary can distinguish which tag is chosen as the challenge tag. The main problem here is that (u_2, v_2) output by the reader in the first execution is sent from the tag in the next session.

Desynchronization Attack. The output message from the tag in the authentication phase can be parsed to random nonce n_1, MAC σ_1 and ciphertext (u_1, v_1). Whenever the tag accepts the current owner, this ciphertext is re-encrypted by the owner and the tag does not output the same ciphertext. However, if the adversary modifies the communication and sends a random message to the tag in **A3**, the reader authentication is failed and the secret key of the tag is not updated. Though the random nonce is chosen per session, the ciphertext cannot be updated by the tag and the adversary can observe that the same ciphertext is output from the tag when he activates the tag two times. If the adversary launches the message modification just before the challenge phase in the privacy game, he can learn which tag is chosen as the challenge tag and win the game.

Malicious New Owner Attack. ROTIV assumes that owners are always honest. On the other hand, the old ownership privacy requires that the privacy against the current owner must be hold even if the new owner is malicious. ROTIV specifies that the current owner responds $A' = A^{\alpha} = g_1^{\alpha r_1 r'}$ when the new owner sends $A = u_1^{r'} = g_1^{r_1 r}$ in **T4**. Recall that α is the secret key of the current owner. The new owner can derive $(A')^{-r'} = g_1^{\alpha r_1}$ from the response and check

$$e(v_1, g_2) = e(g_1^{\alpha r_1}, g_2^{\alpha}) \cdot e(y, g_2) = e(u_1^{\alpha}, g_2'') \cdot e(y, g_2)$$

for any ciphertext (u_1, v_1) which was transferred in the authentication phase with the current owner. This equation holds iff the decryption result of the ciphertext equals to y which is obtained in **T4**[1] . Thus the malicious new owner can learn when the ownership transferred tag executes the authentication protocol with the current owner.

4 Proposed Protocol

We introduce a generic construction of the RFID ownership transfer protocol based on the previous discussions. Let $\mathsf{PKE} := (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE}, \mathsf{Dec})$ be public key encryption algorithm and $\mathsf{SIG} := (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Ver})$ be digital signature algorithm. $f : \{0,1\}^* \to \{0,1\}^{4k}$ is a pseudorandom function (PRF). Note that the RFID tag does not execute public key encryption nor digital signature schemes. These schemes are only used in the ownership transfer of the tag.

Setup Phase. Each owner runs key generation algorithm of the public key encryption scheme $(\mathsf{pk}_j, \mathsf{sk}_j) \xleftarrow{R} \mathsf{PKE.Gen}(1^k)$ and digital signature scheme $(vk_j, sk_j) \xleftarrow{R} \mathsf{SIG.Gen}(1^k)$. $\{(\mathsf{pk}_j, vk_j)\}_j$ is published as the public parameter. When owner R_j obtains the ownership of tag t_i, he randomly chooses $s_i \xleftarrow{U} \{0,1\}^k$ and computes digital signature $\sigma \xleftarrow{R} \mathsf{SIG.Sign}(sk_j, t_i)$. R_j sends (s_i, σ) to the tag and keeps the secret key s_i and previous secret key $s_{i.old}$ (after the second invocation) in its own database.

Authentication Phase. Consider that the current owner is $R_j \in \mathcal{R}$ who interacts with the RFID tags.

A1. R_j chooses $r_1 \xleftarrow{U} \{0,1\}^k$ and sends $(\mathtt{auth}, r_1)$ to the tag.

A2. When the tag receives $(\mathtt{auth}, r_1)$, it chooses $r_2 \xleftarrow{U} \{0,1\}^k$ and computes $(k_0, k_1, k_2, u) := f(s_i, \mathtt{auth} \| r_1 \| r_2)$. Then the tag responds $(\mathtt{auth}, r_1, r_2, k_1)$ to R_j.

[1] Abyaneh insisted that ROTIV does not hold new ownership privacy in [1]. However, his analysis that checking $e(v_1, g_2) = e(h(t), g_2'') \cdot e(v_1/y, g_2)$ is useless to distinguish the challenge tag. This equation holds for any ciphertext v_1 regardless of the tag's identity.

A3. Upon receiving $(\texttt{auth}, r_1, r_2, k_1)$, R_j computes $(k'_0, k'_1, k'_2, u') := f(s_i, \texttt{auth}\|r_1\|r_2)$ and checks $k_1 = k'_1$ for $1 \leq i \leq \ell$. If this equation holds for an index i, R_j sets $s_{i.old} := s_i, s_i := k'_0$ and sends $(\texttt{auth}, r_2, k'_2)$ to the tag. Else if there is an index $i \in \{1, \ldots, \ell\}$ such that $k_1 = k''_1$ where $(k''_0, k''_1, k''_2, u'') := f(s_{i.old}, \texttt{auth}\|r_1\|r_2)$, R_j sends $(\texttt{auth}, r_2, k'_2 := k''_2)$ to the tag. If one of the above two equations hold, R_j accepts the session. Otherwise, R_j rejects the session and outputs $(\texttt{auth}, r_2, k'_2 \xleftarrow{\mathsf{U}} \{0,1\}^k)$ to the tag.

A4. When the tag receives $(\texttt{auth}, r_2, k'_2)$, it verifies $k'_2 = k_2$. If it holds, the tag accepts R_j and updates the secret key to $s_i := k_0$. Otherwise, the tag rejects the session and does not update the secret key.

Ownership Transfer Phase. Without loss of generality, we assume that the current owner of the tag is R_j and new owner is R_{j+1}. We assume that the secret key shared by the current owner R_j and the tag is resynchronized as denoted in the operation model.

T1. R_{j+1} chooses $r_1 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and sends $(\texttt{trans}, r_1)$ to the tag.

T2. When the tag receives $(\texttt{trans}, r_1)$, it chooses $r_2 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and computes $(k_0, k_1, k_2, u) := f(s_i, \texttt{trans}\|r_1\|r_2)$. The tag sets $c := u \oplus \sigma$ and sends $(\texttt{trans}, r_1, r_2, c, k_1)$ to R_{j+1}.

T3. Upon receiving $(\texttt{trans}, r_1, r_2, c, k_1)$, the new owner R_{j+1} generates $\sigma_{new} \xleftarrow{\mathsf{R}} \mathsf{SIG.Sign}(sk_{j+1}, r_1\|r_2\|c\|k_1)$ and sends $(\texttt{trans}, r_1, r_2, c, k_1, \sigma_{new})$ to the current owner R_j.

T4. When R_j receives $(\texttt{trans}, r_1, r_2, c, k_1, \sigma_{new})$, he computes $(k'_0, k'_1, k'_2, u') := f(s_i, \texttt{trans}\|r_1\|r_2)$ and checks $k_1 = k'_1$ for $1 \leq i \leq \ell$. Then R_j verifies $\mathsf{SIG.Ver}(vk_{j+1}, r_1\|r_2\|c\|k_1, \sigma_{new}) = 1$. This verification allows the current owner to verify whether the valid owner requests the ownership transfer. If both verifications hold and R_j decides to release the ownership of the tag, it encrypts the tag's identity t_i and secret key s_i as $c' \xleftarrow{\mathsf{R}} \mathsf{PKE.Enc}(\mathsf{pk}_{j+1}, t_i\|s_i)$. and sends $(\texttt{trans}, c')$ to R_{j+1}. Otherwise, R_j outputs $\perp$ and terminates the protocol.

T5. R_{j+1} decrypts c' with sk_{j+1} and obtains $t_i\|s_i$. Then R_{j+1} verifies the tag as **T4**. Moreover, R_{j+1} computes $\sigma := c \oplus u$ from s_i and verifies $\mathsf{SIG.Ver}(vk_j, t_i, \sigma) = 1$. If one of these verifications fails, R_{j+1} terminates the protocol. Otherwise, R_{j+1} sends $(\texttt{trans}, r_2, k'_2)$ to the tag.

T6. Upon receiving $(\texttt{trans}, r_2, k'_2)$, the tag checks $k'_2 = k_2$. If the verification holds, the tag allows the new setup from the new owner.

The RFID tag can proceed the authentication and ownership transfer phases in the same way except the string literal. So the implementation cost to add the ownership transfer is negligible from the view point of the RFID tag. The string literal is used to classify the authentication and ownership transfer and it prevents the reusing attack between them.

5 Security Proof

Theorem 1. *Assume that the public key encryption scheme is IND-CCA2 secure and digital signature scheme is secure against EUF-CMA. Then our protocol holds basic security and is resilient to the owner impersonation attack.*

Proof. The basic security is mainly provided by the security of the PRF. Any previous transcripts are rejected by the reader and tag because they randomly choose the nonce (r_1, r_2) for each session. Even when the adversary observes the ownership transfer interaction, the secret key of the RFID tag and its identity are encrypted by IND-CCA2 secure encryption so the ciphertext gives no information. Therefore if a modified message is accepted, this means that the security of the PRF is broken. The owner impersonation resilience is provided by the digital signature scheme. Recall that each owner verifies the signature generated by the other owner. If an adversary can successfully impersonate an owner, it directly implies that the accepted signature is a valid forgery against the digital signature.

When there is an adversary $\mathcal{A}$ who can break $\mathsf{Adv}^{\mathsf{ROT\text{-}1}}_{\Pi,\mathcal{A}}(k)$ with non-negligible advantage, we can construct an algorithm $\mathcal{B}$ which breaks the security of the digital signature. Upon receiving the verification key vk^*, $\mathcal{B}$ randomly selects and guesses the owner R_{j+1} who is impersonated by the adversary. $\mathcal{B}$ sets vk^* as the verification key of R_{j+1} and runs all the other setup operation by himself. $\mathcal{B}$ runs $\mathcal{A}$ internally and simulates all protocol flow with his secret except that R_{j+1} runs ownership transfer protocol. When $\mathcal{A}$ runs the ownership transfer protocol and issues $\mathsf{SendReader}(R_{j+1}, (\mathtt{trans}, r_1, r_2, k_1, c))$, then $\mathcal{B}$ sends $r_1\|r_2\|k_1\|c$ to the signing oracle. When σ_{new} is responded, $\mathcal{B}$ sends $(\mathtt{trans}, r_1, r_2, k_1, c, \sigma_{new})$ to the adversary. If the adversary issues $\mathsf{SendReader}(R_{j+1}, (\mathtt{trans}, c'))$, $\mathcal{B}$ runs **T5**-**T6**. If the verification holds and tag's identity is identified as t, $\mathcal{B}$ issues the signing oracle with input t to obtain σ, and a randomly chosen $s \xleftarrow{\cup} \{0,1\}^k$ and σ are kept as the tag's new secret key. After the above simulation, the adversary tries to impersonate the new owner and outputs $(\mathsf{trans}, r_1^*, r_2^*, k_1^*, c^*, \sigma_{new}^*)$. Then $\mathcal{B}$ outputs $(r_1\|r_2\|k_1\|c, \sigma_{new}^*)$ as a forgery to the digital signature scheme.

This signature can be verified by $\mathsf{SIG.Ver}(vk^*, r_1\|r_2\|k_1\|c, \sigma_{new}^*) = 1$. However, the probability that the adversary reuses the previous signature in the impersonation phase is negligible since the secret key shared by the current owner and tag is flesh. Thus the output message is different from the previous transcripts. If the signature verification is accepted, it means σ_{new}^* is a valid forgery against new message $r_1\|r_2\|k_1\|c$ where $\mathcal{B}$ does not issue it to the signing oracle. Therefore $\mathcal{B}$ can break the security of digital signature scheme if $\mathcal{A}$ can impersonate the new owner.

Next we discuss the situation that the adversary $\mathcal{A}$ who does not have the authorization of the tag can send the ownership to the new owner. Let c' be the ciphertext sent from the adversary. When the new owner decrypts as $t^*\|s^* := \mathsf{PKE.Dec}(sk_{j+1}, c')$, $(k_0^*, k_1^*, k_2^*, u^*) := f(s^*, \mathtt{trans}\|r_1\|r_2)$ and $\sigma^* := \mathsf{Dec}(u^*, c)$

are computed where (r_1, r_2, c) is given by the tag in the ownership transfer protocol. The new owner accepts the peer iff $\mathsf{SIG.Ver}(vk_j, t^*, \sigma^*) = 1$ holds.

Consider that s is the shared secret key between the current owner and the tag at the beginning of the ownership transfer protocol. We consider the following two cases.

Case 1 — $s^* = s$:
We construct an algorithm $\mathcal{B}'$ which can break the security of the PRF f. $\mathcal{B}'$ can access to the real function $f(s, \cdot)$ or truly random function RF. $\mathcal{B}'$ runs $\mathcal{A}$ and simulates all the protocol interactions except that R_j transfers t^*'s ownership. When they execute ownership transfer protocol, the adversary sends $(\mathtt{trans}, r_1)$ to the tag. Then $\mathcal{B}'$ randomly chooses r_2 and issues $\mathtt{trans} \| r_1 \| r_2$ to the oracle. Upon receiving (k_0, k_1, k_2, u), $\mathcal{B}'$ computes tag's response with k_1 and u. When the advesary sends c' to the new owner, $\mathcal{B}'$ decrypts it and obtains s^*. $\mathcal{B}'$ terminates the protocol and outputs 1 iff $(k_0, k_1, k_2, u) = f(s^*, \mathtt{trans} \| r_1 \| r_2)$ holds. This event happens when the adversary guesses the secret key against the PRF, thus $\mathcal{B}'$ can break the security of the PRF.

Case 2 — $s^* \neq s$: When Case 1 does not occur, the security of the digital signature algorithm is broken by an algorithm $\mathcal{B}''$. $\mathcal{B}''$ receives vk^* and assigns it to the verification key of R_j. The other secret key is honestly generated by $\mathcal{B}''$ and it simulates the protocol. When R_j is designated to the new owner of a tag or transfers the ownership of the tag, $\mathcal{B}''$ simulates the protocol transcript with the signing oracle. When the adversary impersonates the current owner and sends c' to the new owner, $\mathcal{B}''$ computes as the protocol specification and generates (t^*, σ^*). Finally, $\mathcal{B}''$ outputs it as the forgery of the digital signature.
Since $s^* \neq s$ is assumed in this case, deterministically defined σ^* is different from the past responses $\mathcal{B}''$ obtained from the signing oracle. Thus if this signature is valid and the new owner accepts, (t^*, σ^*) is a valid forgery to the digital signature scheme.

We note that the ciphertext sent from the current owner in the ownership transfer protocol gives no information of the secret key against the adversary for any security game. Thus the adversary cannot impersonate to the any party in our protocol. Therefore, our protocol is resilient to the owner impersonation attack.

Now, we prove that our protocol satisfies the basic security. Based on the above proof, we assume that the adversary $\mathcal{A}$ cannot receive the ownership sent from the current owner explicitly. Even in this case, the ciphertext itself may leak some useful information. So we show that the adversary does not obtain secret key of the tag from this ciphertext. Consider an algorithm $\mathcal{B}$ which try to break the IND-CCA2 security of the public key encryption scheme. $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{IND\text{-}CCA2}}(k)$ denotes the advantage of the algorithm $\mathcal{B}$ for breaking the public key encryption scheme. $\mathcal{B}$ assigns the public key to the current owner and simulates all the session except that the current owner runs ownership transfer phase as a current owner. When the authentication of the RFID tag t_i and verification of the new owner are

accepted, $\mathcal{B}$ sets $m_0 := t_i \| s_i$, chooses arbitrary message m_1 and sends them to the challenger (against the public key encryption scheme). Upon receiving the the challenge ciphertext c_b, $\mathcal{B}$ sends it to the new owner. $\mathcal{B}$ outputs 0 iff $\mathcal{A}$ successfully impersonates to the owner or tag. If the public key encryption scheme is IND-CCA2 secure, $\mathcal{A}$ does not realize whether $b = 0$ or not. Thus we can change the valid ciphertext to the random and the probability that $\mathcal{A}$ learn the secret key of the tag from the ciphertext is negligible.

When there is an adversary $\mathcal{A}$ who can break basic security with non-negligible advantage in any other case, we can construct an algorithm $\mathcal{B}$ which breaks the security of the PRF. $\mathcal{B}$ generates secret keys and honestly simulates the protocol except the ownership transfer protocol. $\mathcal{B}$ can access to the real function $f(s, \cdot)$ or truly random function RF. When the adversary sends $(\mathtt{auth}, r_1^*)$ in the first round and $(\mathtt{auth}, r_1^*, r_2^*, k_1^*)$ in the second round, $\mathcal{B}$ issues $\mathtt{auth} \| r_1^* \| r_2^*$ to the oracle and obtains (k_0, k_1, k_2, u). If $k_1^* = k_1$ holds, $\mathcal{B}$ terminates the simulation and outputs 1.

When the message output from the adversary is accepted, it means that the adversary can guess the output of the PRF. Thus $\mathcal{B}$ can break the security of the PRF. Note that if the adversary breaks the reader authentication, $\mathcal{B}$ checks $k_2^* = k_2$ instead of k_1^* where k_2^* is sent from the adversary in the third round.

Therefore, our protocol satisfies the basic security and owner impersonation attack.

Theorem 2. *Assume that the PRF is secure and the public key encryption schemes is IND-CCA2 secure. Then our protocol satisfies IND-privacy.*

Proof. To prove Theorem 2, we show that our protocol does not give any information about identity of the RFID tag in the authentication protocol nor ownership transfer protocol. First we show the former proposition with the following game transformation. We assume that the number of the owner and tag is denoted by n and ℓ, respectively. Suppose that the number of secret key update of the tag by one owner is at most q. S_i denotes the event that the adversary outputs 1 in Game i.

Game 0. This is the IND-privacy game between the adversary and challenger.
Game 1. The challenger randomly selects from $\{R_1, \ldots, R_n\}$ and guesses the owner of the challenge tag. If this guess is failed, the challenger terminates the game. Otherwise, Game 1 is proceeded as Game 0.
Game 2-(i, j)**.** We change the output value of the session as the following:

1. When the owner interacts with the tag whose index is less than i, then the outputs of the PRF is changed as uniformly random variables $(k_0, k_1, k_2, u) \xleftarrow{\mathsf{U}} \{0,1\}^{4k}$.
2. For the i-th tag,
   2-1. If the secret key update is less than j $(j' < j)$, the output variables from the PRF is changed as $(k_0, k_1, k_2, u) \xleftarrow{\mathsf{U}} \{0,1\}^{4k}$.
   2-2. If the secret key update is greater than or equal to j $(j' \geq j)$, these outputs are computed as $(k_0, k_1, k_2, u) := f(s'_j, r_1 \| r_2)$.

3. If the protocol is run with the tag whose index is greater than i, we proceed the game as Game 0.

Lemma 1. $\Pr[S_0] = n \cdot \Pr[S_1]$ *holds.*

The probability that the challenger correctly guesses the owner for the challenge phase is at least $1/n$. Therefore we have $\Pr[S_0] = n \cdot \Pr[S_1]$.

Lemma 2. *For any* $0 \leq j \leq q$*, there exists an algorithm* $\mathcal{B}_1$ *such that* $|\Pr[S_{2\text{-}(i,j)}] - \Pr[S_{2\text{-}(i,j+1)}]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_1,f}(1^k)$ *holds (for a fixed* $i = 1, \ldots, \ell$*).*

Proof. If there is a gap between Game 2-(i,j) and Game 2-$(i,j+1)$, we show there is an algorithm $\mathcal{B}_1$ which breaks the security of the PRF. The only difference between these games is whether the i-th tag which updates its secret key j times runs the PRF or these outputs are chosen by truly random.

Algorithm $\mathcal{B}_1$ can issue oracle queries to PRF $f(s_i, \cdot)$ or truly random function RF. $\mathcal{B}_1$ runs the setup algorithm and simulates the game as Game 1 except the sessions executed by t_i and the owner of the challenge phase. If t_i updates the secret key less than j, $\mathcal{B}_1$ always chooses random numbers in $\{0,1\}^{4k}$ instead of the output from f. If the number of key update is more than j, $\mathcal{B}_1$ computes the outputs of the tag as the specification of our protocol. When the key update is executed just j times, $\mathcal{B}_1$ proceeds the following. When the adversary $\mathcal{A}$ issues the $\mathsf{ReaderInit}(1^k)$ query, $\mathcal{B}_1$ generates $r_1 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and responds r_1 to the adversary. If the $\mathsf{SendTag}(t_i, r_1')$ query is issued, $\mathcal{B}_1$ selects $r_2 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and issues $r_1' \| r_2$ to the oracle. This response $(k_0, k_1, k_2, u) \in \{0,1\}^{4k}$ is used to compute the output message $(\mathtt{auth}, r_1', r_2, c, k_1)$ from the tag. When the owner receives (r_1, r_2', k_1'), $\mathcal{B}_1$ issues $r_1 \| r_2'$ to the oracle and authenticates the tag as **A3** by using the response from the oracle. If the authentication result is accepted, $\mathcal{B}_1$ outputs $(\mathtt{auth}, r_2, k_2)$ to the adversary. Otherwise, $\mathcal{B}_1$ randomly chooses $k_2 \xleftarrow{\mathsf{U}} \{0,1\}^k$ and outputs $(\mathtt{auth}, r_2, k_2)$. Finally, when the adversary $\mathcal{A}$ outputs b, $\mathcal{B}_1$ outputs the same bit.

If $\mathcal{B}_1$ accesses to the actual PRF, the above game is equivalent to Game 2-(i,j) from the view point of the adversary. Otherwise, if $\mathcal{B}_1$ issues the oracle queries to the truly random function, the above game is equivalent to Game 2-$(i,j+1)$. Thus we obtain $|\Pr[S_{1\text{-}(i,j)}] - \Pr[S_{1\text{-}(i,j+1)}]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_1,f}(1^k)$.

Note that $\Pr[S_1] = \Pr[S_{2\text{-}(1,0)}]$ and $\Pr[S_{1\text{-}(i,q+1)}] = \Pr[S_{1\text{-}(i+1,0)}]$ for any $1 \leq i \leq \ell$ since the output distributions of these games are equivalent, so we can change Game 0 to the final game. Since the output messages derived from the tag and owner are completed changed to random string, there is no useful information about the tag's identity in the authentication protocol.

The same argument is also applied to the ownership transfer phase since authentication process is quite similar to the authentication protocol. The additional information related to the tag's identity is only contained in the ciphertext c sent from the tag or c' sent from the current owner. When we proceed games as above, we can easily show that u is also uniformly chosen as a random string. Thus we can use u as a one-time pad and $c := u \oplus \sigma$ is (information theoretically) indistinguishable from random string. Moreover, it is clear that the ciphertext

gives no information to the adversary since the encryption scheme is IND-CCA2 secure. We can change the plaintext to generate c' to the random message whenever the ownership transfer protocol is executed. If this transformation is realized by the adversary, there is an algorithm $\mathcal{B}_2$ which breaks the IND-CCA2 security against the public key encryption scheme. Therefore the adversary cannot distinguish which tag is chosen in the challenge phase.

Remark that if the adversary breaks the basic security in the privacy game, the tag cannot resynchronize to the reader and it is easy to distinguish the tag from the result query. Thus we must add the advantage for the basic security to evaluate $\mathsf{Adv}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(k)$. Finally we have

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(k) \leq & n\ell(q+1)\cdot \mathsf{Adv}^{\mathsf{PRF}}_{\mathcal{B}_1}(k) + n(q+1)\mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathcal{B}_2}(k) \\ & + \mathsf{Adv}^{\mathsf{RAuth}}_{\Pi,\mathcal{B}_3}(k).\end{aligned}$$

for some algorithms $(\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$.

6 Additional Properties for RFID Ownership Transfer Protocols

In this paper, we concentrate on the basic requirements for RFID ownership transfer protocol: correctness, security and privacy only. The partial delegation of the authorization is proposed by several researchers [12,14], but this functionality is easily defeated by an active adversary (e.g. DDos attack). Another attractive requirement related to the ownership transfer is to support the rewinding of the authorization to the previous owner (authorization recovery [13]). If the ownership transfer protocol follows **S1**-**S3**, we can add the following operation to satisfy the rewinding process.

S4. The new owner keeps the secret key sent from the previous owner and rewrites it to the tag when he wants to return the authorization to the previous owner.

One may argue that the issuer verification described in [7] is removed in our protocol. But we can easily add this function since the new owner checks the digital signature of the current owner which is contained in the RFID tag. Consider that the tag permanently keeps the first owner's digital signature and treats it as the current owner's one. If we perform the ownership transfer phase and modify that the current owner only outputs the digital signature of the issuer, the issuer verification function is achieved. However, this information implicitly gives the identity of the tag when the issuer of the two tags is different. Thus another security model will be required and we leave it as an open problem in this paper.

7 Conclusion

We showed an attack to the previous provably secure RFID ownership transfer protocol. We introduced a new security model for ownership transfer protocol

based on the previous constructions. We provide generic construction for provably secure RFID ownership transfer protocol by using public key encryption scheme, digital signature scheme and pseudorandom function.

References

1. Abyaneh, M.R.S.: On the privacy of two tag ownership transfer protocols for RFIDs. In: ICITST 2011, pp. 11–14. IEEE (2011)
2. Billet, O., Etrog, J., Gilbert, H.: Lightweight privacy preserving authentication for RFID using a stream cipher. In: Hong, S., Iwata, T. (eds.) FSE 2010. LNCS, vol. 6147, pp. 55–74. Springer, Heidelberg (2010)
3. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
4. Canard, S., Coisel, I.: Data synchronization in privacy-preserving RFID authentication schemes. In: RFIDSec 2008 (2008)
5. Cai, S., Li, Y., Li, Y., Deng, R.H.: Attacks and improvements to an RFID mutual authentication protocol and its extensions. In: WiSec 2009, pp. 51–58. ACM (2009)
6. Erguler, I., Anarim, E.: Security flaws in a recent RFID delegation protocol. J. Pers. Ubiquit. Comp. 16(3), 337–349 (2012)
7. Elkhiyaoui, K., Blass, E.-O., Molva, R.: ROTIV: RFID ownership transfer with issuer verification. ePrint Archive 2010/634 (2010)
8. Elkhiyaoui, K., Blass, E.-O., Molva, R.: ROTIV: RFID ownership transfer with issuer verification. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 163–182. Springer, Heidelberg (2012)
9. Fernàndez-Mir, A., Trujillo-Rasua, R., Castellà-Roca, J., Domingo-Ferrer, J.: A scalable RFID authentication protocol supporting ownership transfer and controlled delegation. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 147–162. Springer, Heidelberg (2012)
10. Juels, A., Weis, S.A.: Defining strong privacy for RFID. ACM Transactions on Information and System Security 13(1) (2009)
11. Kardas, S., Akgün, M., Kiraz, M.S., Demirci, H.: Cryptanalysis of lightweight mutual authentication and ownership transfer for RFID systems. In: LightSec 2011, pp. 20–25. IEEE (2011)
12. Lim, C.H., Kwon, T.: Strong and robust RFID authentication enabling perfect ownership transfer. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 1–20. Springer, Heidelberg (2006)
13. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 276–290. Springer, Heidelberg (2006)
14. Ng, C.Y., Susilo, W., Mu, Y., Safavi-Naini, R.: Practical RFID ownership transfer scheme. J. Comput. Sec. 19(2), 319–341 (2011)
15. Ouafi, K., Phan, R.C.-W.: Traceable privacy of recent provably-secure RFID protocols. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 479–489. Springer, Heidelberg (2008)
16. Osaka, K., Takagi, T., Yamazaki, K., Takahashi, O.: An efficient and secure RFID security method with ownership transfer. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 778–787. Springer, Heidelberg (2007)

17. Peris-Lopez, P., Hernandez-Castro, J.C., Tapiador, J.M.E., Li, T., Li, Y.: Vulnerability analysis of RFID protocols for tag ownership transfer. J. Comp. Net. 54(9), 1502–1508 (2010)
18. Paise, R.I., Vaudenay, S.: Mutual authentication in RFID. In: ASIACCS 2008, pp. 292–299. ACM (2008)
19. Shaohui, W.: Analysis and Design of RFID Tag Ownership Transfer Protocol. In: Jiang, L. (ed.) ICCE 2011. AISC, vol. 110, pp. 229–236. Springer, Heidelberg (2011)
20. Song, B., Mitchell, C.J.: Scalable RFID security protocols supporting tag ownership transfer. ComCom 34(4), 1–27 (2011)
21. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)

An Efficient and Private RFID Authentication Protocol Supporting Ownership Transfer

Süleyman Kardaş[1,2], Serkan Çelik[1,2], Atakan Arslan[1], and Albert Levi[2]

[1] TÜBITAK BILGEM UEKAE Gebze, Kocaeli
[2] Sabancı University, Faculty of Engineering and Natural Sciences, İstanbul, TR-34956, Turkey

Abstract. **R**adio **F**requency **ID**entification (RFID) systems are getting pervasively deployed in many daily life applications. But this increased usage of RFID systems brings some serious problems together, security and privacy. In some applications, ownership transfer of RFID labels is sine qua non need. Specifically, the owner of RFID tag might be required to change several times during its lifetime. Besides, after ownership transfer, the authentication protocol should also prevent the old owner to trace the tags and disallow the new owner to trace old transactions of the tags. On the other hand, while achieving privacy and security concerns, the computation complexity should be considered. In order to resolve these issues, numerous authentication protocols have been proposed in the literature. Many of them failed and their computation load on the server side is very high. Motivated by this need, we propose an RFID mutual authentication protocol to provide ownership transfer. In our protocol, the server needs only a constant-time complexity for identification when the tag and server are synchronized. In case of ownership transfer, our protocol preserves both old and new owners' privacy. Our protocol is backward untraceable against a strong adversary who compromise tag, and also forward untraceable under an assumption.

Keywords: RFID, Privacy, Security, Ownership Transfer Protocol.

1 Introduction

Today, ubiquitous information and communication technology has been widely accepted by everyone that aspire to reach information anytime and anywhere. Radio-frequency identification (RFID) systems are one of the ubiquitous computing in which technology provides practical services to people in their daily life. RFID technology aims to identify and track an item or a person by using radio waves. It has been pervasively deployed in several daily life applications such as contact-less credit cards, e-passports, ticketing systems, etc.

A RFID system basically consists of several tags *(transponders)*, a set of readers *(interrogator)* and a back-end receiver. A tag contains a microchip which carries data and antenna. It is interrogated by a reader via its modulated radio signals. A RFID reader that is the central part of an RFID system, acquires

G. Avoine and O. Kara (Eds.): LightSec 2013, LNCS 8162, pp. 130–141, 2013.

the data of the tag and conveys it to the back-end system for further processing. Moreover, RFID tags can be categorized into three groups by using energy source such as active, passive and semi-passive or battery assisted tags. Passive RFID tags do not have internal energy sources. Instead, they use the radio energy transmitted by the reader [10]. Furthermore, RFID systems can also be grouped into three basic ranges by their using operating frequency: Low frequency (LF, 30-300 KHz), high frequency (HF 3-30 MHz) and ultra high frequency (300 MHz - 3 GHz) / microwave (>3 GHz) [9].

Nowadays, the number of RFID applications have been proliferating because of their productivity, efficiency, reliability and so on. Many companies also prefer low-cost tags with tiny sizes. This brings some computational and memory restrictions to RFID tags. On the other hand, RFID tags and readers communicate with each other over an air interface. This insecure channel and the limited capabilities of RFID tags cause security and privacy vulnerabilities. An adversary can do tag impersonating, tracking, eavesdropping, and denial of service (DoS) attack. Besides the vulnerabilities, a tag might be distinguishable in its life-span by an attacker. If it is once recognized by an adversary, it can be easily traceable. At that situation, there might be two attacks. (i) An attacker might track the previous interactions of the tag or (ii) he may track the future ones. These two attacks are called backward traceability and forward traceability, respectively. The protocol used for RFID system should provide not only resistance against passive attacks, replay attacks, cloning attacks but also resistance against active attacks. There are public-key cryptography solutions in the literature but none of them are convenient for the low-cost tags used in lots of applications because of their limitations. It needs to find much light-weight approaches. Therefore, many light-weight authentication protocols are proposed to defeat adversaries that deceive the capacity-restricted tags. But, designing light-weight cryptographic authentication protocols with basic cryptographic primitives (xor, hash function) is a challenging task [18].

Another significant problem is the changing ownership of an RFID tag several times during its life-cycle. For instance, tags are initially created and attached to objects by producers, then labeled objects are taken over to retailers, and finally consumers buy tagged objects from shopping malls [13]. The ownership of a labeled object may be frequently transferred from one party to another. At the moment of the transfer, both new and old owners have the same information about the tag. This might cause privacy problems. This transfer should guarantee that the old owner should no longer be able to trace the future interactions and the new owner should not be able to trace old interactions. Besides having secure authentication protocols by providing privacy, the performance of the entire system becomes an important issue. Therefore, designing authentication protocol without compromising security and privacy begets decreases the efficiency of the whole system. However, achieving both security and privacy properties, the computational complexity of the tag and the server side can vary dramatically from one protocol to another. Hence, while handling security and privacy issues, it is also important to realize it with less computational complexity.

In order to resolve these security and privacy issues, numerous RFID authentication protocols have been recently proposed [1, 4, 5, 7, 8, 11, 12, 14–17]. However, some of them are not compliant to ownership transfer. Also, none of them achieves constant-time complexity for identification while providing forward untraceability against old-owner and backward untraceability (forward secrecy) against the new owner.

Our Contributions. We propose an efficient, secure and private RFID mutual authentication protocol which needs constant-time complexity to identify a tag. Then, we utilize this protocol and achieve a secure and efficient ownership transfer. We prove that our protocol achieves forward secrecy against the new owner and forward untraceability against the old owner. Moreover, we also show that our protocol provides forward secrecy against a strong attack and forward untraceability under an assumption that the adversary misses one subsequent successful protocol between the reader and the compromised tag.

The outline of the paper is as follows. In Section 2, security and threat model, security and privacy concerns are discussed in RFID systems for ubiquitous networks. Section 3 describes our proposed protocol. In Section 4, analysis of our protocol is given in detail. In Section 5, we conclude the paper.

2 Adversarial Model

In this section we describe our adversarial model used in analyzing the proposed protocol, then define the privacy notions which are also used to be proved. Since the tags and the reader communicates over an insecure wireless channel, we consider Byzantine adversarial model [6].

- Each tag memory is not tamper resistant and vulnerable to physical attacks.
- Each tag/reader performs cryptographic hash operations.
- The reader and tags communicate over an insecure wireless channel and so an active attacker can intercept, modify and generate messages.
- The messages between server and readers are transmitted securely.
- The reader and the server are assumed to be trusted parties. They cannot be compromised.

Since the tags are not tamper resistant, we assume that a strong adversary can corrupt a tag and access to its persistent memory. In this case, the adversary should not be able link any current and past communication of the victim tags. This privacy notion is called backward untraceability. We define it more formally as follows.

Definition 1. *Backward Untraceability: An RFID scheme provides backward untraceability if $\mathcal{A}$ compromising $\mathcal{T}_i$ at time t cannot trace the past interactions of $\mathcal{T}_i$ that occurred at time $t' < t$.*

On the other hand, the strong adversary should not be able to trace the future interactions of the victim tag. This privacy notion, called forward untraceability, is described as follows.

Definition 2. *Forward Untraceability: An RFID scheme provides forward untraceability if $\mathcal{A}$ compromising $\mathcal{T}_i$ at time t cannot trace the future interactions of $\mathcal{T}_i$ that occurred at time $t' > t$.*

3 The Proposed Protocol

In this section, we propose a novel scalable RFID authentication protocol which is the enhanced version of the scheme presented in [12]. In our protocol, we achieve the constant-time complexity for the authentication of synchronized tags whereas the complexity in [12] is $\mathcal{O}(N)$ where N is the number of tag in the system.

The notations used in the protocol are defined. Then, the initialization and the authentication phases are described in detail. The protocol is summarized in Figure 1.

3.1 The Notations

- $\in_R$: The random choice operator that randomly selects an element from a finite set.
- $\oplus$, $||$: XOR operator and concatenation operator, respectively.
- h, H : A hash function s.t. $h : \{0,1\}^* \rightarrow \{0,1\}^n$, $H : \{0,1\}^* \rightarrow \{0,1\}^{2n}$. Both of them are one-way and collision resistant functions.
- N : The number of tags in the database.
- N_a, N_b : n-bit nonce generated by the reader and the tag, respectively.
- K : n-bit secret shared between the tag and the reader.
- val_1, val_2 : n-bit the server validator of the tag and the reader, respectively.
- K^{old_1}, K^{old_2} : Previous n-bit secret shared between the tag and the reader.
- val_1^{old}, val_2^{old} : Previous n-bit the server validator of the tag and the reader, respectively.
- L, S : The seed value of val_1 and val_2, respectively.
- r_1, r_2 : n-bit random bit strings produced by $h(N_a)$, $h(N_b, K)$, respectively.
- v_i : n-bit random bit strings produced by $h(K, r_1, r_2)$.
- M_1, M_2 : $M_1 = v_1 \oplus L$, $M_2 = v_2 \oplus S$.
- DB : Server database.
- γ : n-bit string.
- *state* : 1-bit string is 0 or 1.

3.2 The Registration Phase

For each tag T_i, the following steps have to be performed by the registrar (e.g. the tag manufacturer) before the authentication protocol:

1. The registrar generates three n-bit random nonce (K, S, L). It also computes $val_1 = h(L, K)$, $val_2 = h(S)$. Initially, K^{old_1} and K^{old_2} are both equal to K, S^{old} is equal to S, and val_1^{old} is equal to val_1. Finally, state is set to 0 and it computes hash of the shared secret key K, $\gamma = h(K)$.

2. The registrar creates an entry in its back-end database and stores $(K, S, val_1, K^{old_1}, K^{old_2}, S^{old}, val_1^{old}, h(K))$ in the entry.
3. The registrar assigns $(K, L, val_2, state)$ to the tag T_i.

3.3 The Authentication Phase

In our protocol (see Figure 1) each tag stores its own triple values K, L, val_2, γ,and *state* . The reader stores the K, S, val_1 for that tag. The steps are described below.

Step 1. A reader randomly generates an n-bit nonce N_a and computes hash of it $r_1 = h(N_a)$. Then it sends r_1 to the tag T_i.

Step 2. The tag T_i randomly generates a n-bit N_b nonce and computes hash of it, $r_2 = h(N_b, K)$. Then, it checks the state. If its own state is 0, it computes hash of the shared secret key K. If it is not, the tag randomly generates a n-bit γ nonce. Later, the tag uses a pseudo-random function that digests r_1, r_2 messages with shared secret key K to compute $v_1||v_2 = H(K, r_1, r_2)$. The length of each v_1 and v_2 are both equal to n. After that, the tag computes message M_1 by simply XORing v_1 with secret L. Finally, the tag sends r_2, M_1 and γ messages to the reader.

Step 3. The reader transfers N_a, r_1, r_2, M_1, and γ to the server.

Step 4. The server firstly searches in DB that there exists $h(K)$ equals to γ. The server performs an exhaustive search among all tags in the database. It computes $v_1||v_2 = H(K, r_1, r_2)$ and $h(M_1 \oplus v_1, K)$. The server checks whether $h(M_1 \oplus v_1, K^{old_1})$ is equals to val_1. If one match is found, then the server computes M_2 message by XORing v_2 with S and then sends M_2 to the reader. After that, it updates $K^{old_2} = K^{old_1}$, $K^{old_1} = K$, $S^{old} = S$, $val_1^{old} = val_1$, $K = v_2$, $S = N_a$, and $val_1 = r_2$. If no match is found, then the server performs another an exhaustive search among all tags in the database. In this time, it computes $v_1||v_2 = H(K^{old_1}, r_1, r_2)$ and it checks whether $h(M_1 \oplus v_1, K^{old_2})$ is equals to val_1^{old}. If one match is found, the server computes M_2 message by XORing v_2 with S and sends M_2 to the tag. After that, it updates $K = v_2$, $S = N_a$, and $val_1 = r_1$. However, if there is no match, the server generates an n-bit random bit string and sends it to the reader. The reason behind sending random bit string is that this prevents any attacker to validate M_1 for random nonce r_1 and r_2.

Step 5. The reader forwards M_2 to the tag T_i. Upon receiving M_2 message, T_i computes $h(M_2 \oplus v_2)$ and checks whether it is equal to val_2. If equal, then it updates $K = v_2$, $L = N_b$, and $val_2 = r_1$.

3.4 The Ownership Transfer

When the owner of the tags are required to change one party to another, the tags are first synchronized with the server. The server runs at least two successful authentication protocols with tags in a secure environment where no adversary

is allowed to perform any passive/active attacks. Then, all the tags and their related information are transferred to new owner. Once the new owner receives the information and tags, he/she runs at least one successful protocol between readers and the tags in a secure environment where a malicious adversary is not allowed.

During the ownership transfer, the old owner does not need to transfer the secret values of K^{old_2} and S^{old} of the tags to the new owner because the remaining secrets are enough to communicate with the synchronized tags.

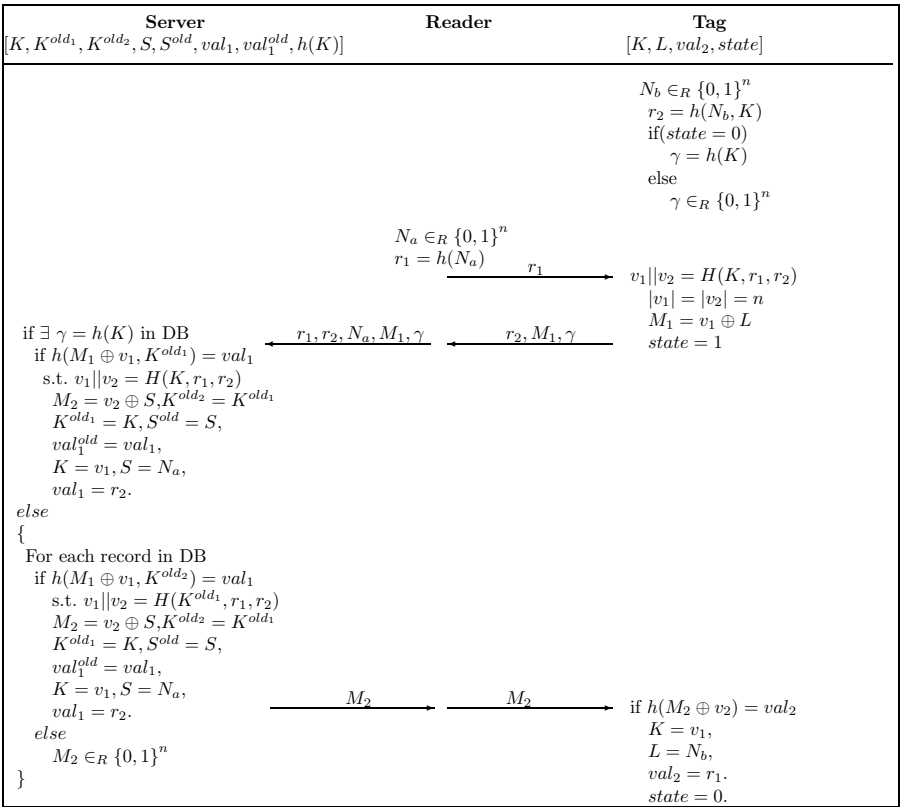


Fig. 1. The Proposed RFID Authentication Protocol

4 Security, Privacy, and Performance Analysis

In this section, we first describe the adversarial capabilities. Then, we analyze our ownership transfer protocol depicted in Figure 1 against passive and strong attacks.

In our model, we assume that each tag can perform cryptographic hash operations. The communication between server and readers are assumed to be

secure because they have no restriction on using SSL/TLS protocol. However, the reader and tags communicate over an insecure wireless channel and so an attacker can intercept, modify and generate messages. Also, each tag memory is not tamper-proof.

4.1 The Security against Timing Attacks

The proposed protocol is vulnerable to timing attacks [3]. An adversary can distinguish synchronized tags and un-synchronized tags by simply considering the response time of the server because the identification time for the latter tags requires much more than the former tags. This kind of attacks can be avoided by using distributed computation servers. Let us illustrate the solution. Assume that we have 2^{20} tags in the database and the server does only 2^{23} hash computation per second. Then, the time to identify an un-synchronized tag is $2^{20}/2^{23} = 0.125$s but for the synchronized tag is almost zero. For the solution, we can use multiple distributed server (say 16), then the identification time can be reduced to $0.125/16 = 7,8125$ms and when a synchronized tag is to be identified the server waits up to $7,8125$ms.

4.2 The Security against Passive Adversary

An offline passive adversary may want to know the contents of the secrets K and L stored in the tag $\mathcal{T}_i$. Then, the adversary simply eavesdrops the channels between a legitimate reader and $\mathcal{T}_i$ in order to get r_1, r_2, M_1, M_2 and γ. With these information and publish hash function H, she cannot obtain the secret K or L because of one-wayness of the hash function.

Moreover, the protocol also resists against replay attack because a challenge-response scheme is used in the protocol. In addition, for each session of the protocol a new pair of random numbers (r_1, r_2) are used. This prevents to use the same challenge-response values in other sessions.

Furthermore, our protocol is resistant against desynchronization even if' last flow of the protocol drops. Normally, this causes desynchronization of the tag secrets and the back-end server. However, this issue is resolved by storing previous tag secrets in the database. Hence the server can resynchronize with the tags in such a condition.

4.3 The Security against Strong Adversary

In this section, we will analyze the protocol depicted at Figure 1 in terms of backward and forward untraceability [2, 15, 19] against old owner, new owner, and a strong malicious adversary who can compromise a tag. As a starting point, we assume that at time t_i, the owner of the system is changed. We test backward untraceability for the new owner, denoted by $\mathcal{A}_n$, with assumption that $\mathcal{A}_n$ has had control over communications between reader and tags made before time t_i. Note that, the number of these communications is finite. Similarly, we test

forward untraceability against the old owner, denoted by $\mathcal{A}_o$. Also, we test these two privacy properties against a strong adversary $\mathcal{A}_s$ with assumption that $\mathcal{A}_s$ has ability of corrupting a tag and captures its secrets. Throughout the analysis, in order to make proofs more understandable, without loss of generality, we assume that there are only two tags in the system, namely $\mathcal{T}_0$ and $\mathcal{T}_1$. First of all, let us give the definitions of concepts mentioned above and the oracle that we use in the proofs of theorem given below.

Definition 3. *Oracle $\mathcal{O}^k$: The oracle chooses $b \in_R \{0,1\}$. If $b = 0$, $\mathcal{O}^k$ sends to the adversary the protocol transcript which was realized between tag $\mathcal{T}_0$ and the reader at time t_k. Similarly, if $b = 1$, the protocol transcript which was realized between tag $\mathcal{T}_1$ and the reader at time t_k is sent to the adversary by the oracle. At the end, the adversary sends the bit b' by after investigating the transcript sent. If $Pr[(b' = b) = 1] = \frac{1}{2} + \epsilon$, where ϵ is non-negligible, than the adversary wins.*

One can give simplified version of the oracle defined above as follows: At time t_i, $\mathcal{A}$ gets information of server and the tag $\mathcal{T}_0$. Then at time t_k, $\mathcal{O}^k$ chooses $b \in_R \{0,1\}$. The transcript sent to the adversary according to value of b same as above. Then, $\mathcal{A}$ returns $b' = 0$ if he thinks the transcript sent by oracle realized between reader and tag $\mathcal{T}_0$. Otherwise the adversary returns $b' = 1$. If $Pr[(b' = b) = 1] = \frac{1}{2} + \epsilon$, where ϵ is non-negligible, than the adversary wins.

Throughout the proofs given to the corresponding theorem, four subsequent successful protocol transactions are enough. Thus, without loss of generality, we assume that $i = 4$ is the time where server owner changed, i.e. at time t_4. Moreover, addition to the notations given at protocol steps, we use left subscript part to denote the time that it was used.

In order to obtain traceability capability of $\mathcal{A}_n$, we start studying with more powerful adversary $\mathcal{A}_c$, who has had all secrets of the server and tags at time t_i and observed all protocol transactions realized before given time.

Theorem 1. *The system has backward untraceability property for time t_k satisfying $k < i - 3$ for the adversary $\mathcal{A}_c$*

Proof. Since at time t_4, $\mathcal{A}_c$ knows the value of ${}_4val_1$ and this value equals to ${}_3r_2$, then at time t_3, $\mathcal{A}_c$ can traces $\mathcal{T}_0$. Moreover, as $\mathcal{A}_c$ knows the value of ${}_4S^{old_1}$, then she knows the value of ${}_3S$. Thus, ${}_2N_a$ value is known. Therefore, at time t_2, $\mathcal{A}_c$ can trace $\mathcal{T}_0$ as he can figure out the value of ${}_2r_1$ from $h({}_2N_a)$. Note that, after that point, $\mathcal{A}_c$ knows ${}_2r_2$ and ${}_2M_2$ and since ${}_2K = {}_4K^{old_2}$, the values of ${}_2v_1$ and ${}_2v_2$ are known. Hence, ${}_2S$ is known. So, $\mathcal{A}_c$ learns the value of ${}_1N_a$. From this knowledge, $\mathcal{A}_c$ calculates ${}_1r_1$. Therefore, $\mathcal{A}_c$ can trace $\mathcal{T}_0$ at time t_1, which means $\mathcal{A}_c$ also learns the values of ${}_1r_2$, ${}_1M_1$, ${}_1M_2$. Apart from these values, ${}_1L$ is also known. Note that, the only thing $\mathcal{A}_c$ knows about the transaction happened at time t_0 is ${}_0N_b$. Thus, the probability of $\mathcal{A}_c$'s finding the correct value of ${}_0r_2$ is $\frac{1}{2^n}$ since ${}_0K$ is not known and the range of hash function h is $\{0,1\}^n$. Similarly, finding correct values of ${}_0r_1$,${}_0M_1$,${}_0M_2$ is $\frac{1}{2^n}$. Thus, the probability that $\mathcal{A}_c$ distinguishes the transcript that the oracle sent is $\frac{1}{2} + \frac{1}{2^n}$. However, $\frac{1}{2^n}$ is negligible.

Therefore, if $\mathcal{A}_c$ has all secrets of the server and tags at time t_i, then the system has backward untraceability property for time t_k satisfying $k < i - 3$.

Remark 1. The values of K^{old_2} and S^{old} of tags are stored in server database in order to overcome synchronization problem. If the system is synchronized when ownership transfer is realized, then K^{old_2} and S^{old} values are not given to $\mathcal{A}_n$.

At the next part, we give a backward traceability result for an adversary $\mathcal{A}_{cR}$, which is like $\mathcal{A}_c$ with exception indicated at Remark 1.

Corollary 1. *The system has backward untraceability property for time t_k satisfying $k < i - 2$ for the adversary $\mathcal{A}_{cR}$.*

Remark 2. The privacy is the main aim that should be reached. Therefore, just before ownership transfer, $\mathcal{A}_o$ completes two successful protocol transactions with tags such that no part of the protocol transcripts are seen by $\mathcal{A}_n$.

Note that the adversary $\mathcal{A}_c$ with incapability explained at Remark 2 corresponds to the new owner, $\mathcal{A}_n$. Thus, we have the following corollary.

Corollary 2. *For the new owner, $\mathcal{A}_n$, the system has backward untraceability property for time t_k satisfying $k < i$.*

Theorem 2. *If $\mathcal{A}_o$ has all secrets of the server and tags at time t_i, then the system has forward untraceability property for time t_k satisfying $k > i$.*

Proof. Since ownership transfer occurs, $\mathcal{A}_o$ misses at least one of the subsequent successful protocol transactions between $\mathcal{A}_n$ and tags. We can get the best result if one subsequent successful transaction miss is assumed. In that case, $\mathcal{A}_o$ only knows values of ${}_5K^{old_1}$, ${}_4K^{old_2}$, ${}_5S^{old_1}$ and ${}_4val_1{}^{old}$. Since the attacker missed a subsequent successful transaction, the other values are unknown. Note that, $\mathcal{A}_o$ can find the value of ${}_4r_2$ with possibility of $\frac{1}{2^n}$ since the value of ${}_4N_b$ is not known. By similar arguments, $\mathcal{A}_o$ guesses the value ${}_4r_2$ with possibility of $\frac{1}{2^n}$. Although $\mathcal{A}_o$ knows the values of ${}_4S$ and ${}_4L$, as ${}_4v_1$ and ${}_4v_2$ are not known, $\mathcal{A}_o$ can figure out the values of ${}_4M_1$ and ${}_4M_2$ with possibility of $\frac{1}{2^n}$. Hence, the probability that $\mathcal{A}_n$ distinguishes the transcript that the oracle sent is at most $\frac{1}{2} + \frac{1}{2^n}$. However, $\frac{1}{2^n}$ is negligible.

Therefore, if $\mathcal{A}_0$ has all secrets of the server and tags at time t_i, then the system has forward untraceability property for time t_k satisfying $k > i$.

Our next result is about the adversary, $\mathcal{A}_s$, who can corrupt a tag and capture all secrets of the tag at any given time and follow all steps of the each successful protocol runs before and after the time that corruption occurs.

Corollary 3. *If $\mathcal{A}_s$ corrupts a tag at time t_j with $j \neq i$, then the system has backward untraceability for time t_k satisfying $k < j - 1$ and forward untraceability for time t_k satisfying $k > j + 1$ under the assumption that $\mathcal{A}_s$ misses the transactions occurred at time $j + 1$ and $j - 1$.*

Proof. Forward secrecy part is direct result of Theorem 2. Moreover, the backward secrecy result is derived from Remark 3

Remark 3. If $\mathcal{A}_s$ does not miss the transaction at $j-1$, then by the knowledge of ${}_jval_2$, he deduces the value of ${}_{j-1}r_1$. Thus, the values of ${}_{j-1}r_2$, ${}_{j-1}M_1$, ${}_{j-1}M_2$ are known to him. Thus, in this case, $\mathcal{A}_s$ can trace the corrupted tag at time t_{j-1}. However, no more traces are possible, because $\mathcal{A}_s$ knows only the value of ${}_{j-2}N_b$ about the transaction realized at time t_{j-2} and from the similar arguments given at proof of Theorem 1, the success probability that $\mathcal{A}_s$ traces the corrupted tag at time t_{j-2} is $\frac{1}{2} + \frac{1}{2^n}$ and $\frac{1}{2^n}$ is negligible.

Remark 4. If $\mathcal{A}_s$ does not miss any transaction after corruption occurs, then $\mathcal{A}_s$ can trace the corrupted tag forever.

Theorem 3. *The proposed protocol satisfies tag authentication under the assumption specified in Corollary 3.*

Proof. First of all, let us assume that the adversary has no corrupt tag capability. In this case, the adversary has to learn the value of either K or K^{old_1} to impersonate the tag. To learn the values of these variables, the adversary has to learn the value of v_1 of previous protocol transcript. However, to learn the value of v_1, the adversary has to figure out the value of K of previous runs or the value of L. However, the value of L is the chosen random N_b value of previous run. Thus, the adversary can only guess the value of L. Therefore, the values v_1, K and K^{old_1} are dependent each other. Thus, the only remained way for the adversary to impersonate the tag is to guess the value of v_1, K or K^{old_1} correctly. Since the space of these variables are large enough, the success probability of the adversary is negligible.

Moreover, since the tag authentication is investigated under the assumption Corollary 3, the system satisfies tag authentication for the case where the adversary can corrupt the tag.

4.4 Performance Issues

Considering memory storage for tag identifiers or keys and other information, our protocol requires $3n + 1$ bit ($3n$-bit for K, L, and val_2 and 1-bit for *state*) memory in tag side. Contrary to tags, server has no limited resource so we do not consider the server-side memory usage.

Concerning computational cost, our protocol requires at most 4 hash computation overhead for the tag. If the tags and the server are synchronized, the computational complexity at the server side is $\mathcal{O}(1)$. Otherwise, the complexity is at most $\mathcal{O}(N)$.

5 Conclusions

In this paper, we first proposed a secure and efficient an RFID mutual authentication protocol which is the revised version of the scheme presented in [12].

With the use of the authentication protocol, we achieve ownership transfer. We prove that our protocol provides forward untraceability against the old owner of the tags and backward untraceability against the new owner of the tags. Also, we show that our authentication protocol provides backward untraceability of a tag against an adversary who compromises the tag and forward untraceability under the assumption that the adversary misses at least one of the subsequent authentication protocol between the tag and the reader. Our protocol requires $\mathcal{O}(1)$ complexity to identify a synchronized tag.

References

1. Alomair, B., Clark, A., Cuellar, J., Poovendran, R.: Scalable RFID systems: a privacy-preserving protocol with constant-time identification. In: International Conference on Dependable Systems and Networks, pp. 1–10 (2010)
2. Avoine, G.: Cryptography in Radio Frequency Identification and Fair Exchange Protocols. PhD thesis, EPFL, Lausanne, Switzerland (December 2005)
3. Avoine, G., Coisel, I., Martin, T.: Time Measurement Threatens Privacy-Friendly RFID Authentication Protocols. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 138–157. Springer, Heidelberg (2010)
4. Burmester, M., de Medeiros, B., Motta, R.: Anonymous RFID authentication supporting constant-cost key-lookup against active adversaries. IJACT 1(2), 79–90 (2008)
5. Dimitriou, T.: A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. In: SECURECOMM 2005: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, pp. 59–66. IEEE Computer Society, Washington, DC (2005)
6. Dolev, D., Yao, A.C.: On the security of public key protocols. In: Proceedings of the 22nd Annual Symposium on Foundations of Computer Science, pp. 350–357. IEEE Computer Society, Washington, DC (1981)
7. Erguler, I., Anarim, E.: Practical attacks and improvements to an efficient radio frequency identification authentication protocol. Concurrency and Computation: Practice and Experience (October 2011)
8. Fernàndez-Mir, A., Trujillo-Rasua, R., Castellà-Roca, J., Domingo-Ferrer, J.: Scalable RFID Authentication Protocol Supporting Ownership Transfer and Controlled Delegation. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 147–162. Springer, Heidelberg (2012)
9. Finkenzeller, K.: RFID Handbook. John Wiley and Sons (2003)
10. Garfinkel, S., Rosenberg, B.: RFID: Applications, Security, and Privacy. Addison-Wesley (2005)
11. Ha, J., Moon, S.-J., Nieto, J.M.G., Boyd, C.: Low-Cost and Strong-Security RFID Authentication Protocol. In: EUC Workshops, pp. 795–807 (2007)
12. Kardaş, S., Levi, A., Murat, E.: Providing Resistance against Server Information Leakage in RFID Systems. In: New Technologies, Mobility and Security – NTMS 2011, Paris, France, pp. 1–7. IEEE Computer Society (February 2011)
13. Lim, C.H., Kwon, T.: Strong and Robust RFID Authentication Enabling Perfect Ownership Transfer. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 1–20. Springer, Heidelberg (2006)

14. Molnar, D., Wagner, D.: Privacy and security in library RFID: issues, practices, and architectures. In: CCS 2004: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 210–219. ACM (2004)
15. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic approach to 'privacy-friendly' tags. In: RFID Privacy Workshop. MIT, Massachusetts (2003)
16. Song, B., Mitchell, C.J.: RFID authentication protocol for low-cost tags. In: WiSec 2008: Proceedings of the First ACM Conference on Wireless Network Security, pp. 140–147. ACM (2008)
17. Song, B., Mitchell, C.J.: Scalable RFID Security Protocols supporting Tag Ownership Transfer. Computer Communication (March 2010)
18. Vajda, I., Buttyán, L.: Lightweight Authentication Protocols for Low-Cost RFID Tags. In: Second Workshop on Security in Ubiquitous Computing – Ubicomp 2003 (2003)
19. Van Le, T., Burmester, M., de Medeiros, B.: Universally Composable and Forward-secure RFID Authentication and Authenticated Key Exchange. In: Bao, F., Miller, S. (eds.) ACM Symposium on Information, Computer and Communications Security – ASIACCS 2007, Singapore, Republic of Singapore, pp. 242–252. ACM Press (March 2007)

Author Index