International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.04,Issue.01
January-2015,
Pages:0107-0113

www.ijsetr.com

Security-Enabled Near-Field Communication Tag with Flexible Architecture Supporting Asymmetric Cryptography

P. M. SANDHYA[1], K. DHANUNJAYA[2]

[1]PG Scholar, Dept of ECE, Audishankara College of Engineering, Gudur, AP, India.
[2]Assoc Prof, Dept of ECE, Audishankara College of Engineering, Gudur, AP, India.

Abstract: This paper presents the design and implementation of a complete near-field communication (NFC) tag system that supports high-security features. The tag design contains all hardware modules required for a practical realization, which are: analog 13.56-MHz radio-frequency identification (RFID) front-end, a digital part that includes a tiny (programmable) 8-b microcontroller, a framing logic for data transmission, a memory unit, and a crypto unit. All components have been highly optimized to meet the fierce requirements of passively powered RFID devices while providing a high level of flexibility and security. The tag is fully compliant with the NFC Forum Type-4 specification and supports the ISO/IEC 14443A (layer 1–4) communication protocol as well as block transmission according to ISO/IEC 7816. Its security features include support of encryption and decryption using the Advanced Encryption Standard (AES-128), the generation of digital signatures using the elliptic curve digital signature algorithm according to NIST P-192, and several countermeasures against common implementation attacks, such as side-channel attacks and fault analyses. The chip has been fabricated in a 0.35-µm CMOS process technology, and requires 49 999 GEs of chip area in total (including digital parts and analog front-end). Finally, we present a practical realization of our design that can be powered passively by a conventional NFC enabled mobile phone for realizing proof-of-origin applications to prevent counterfeiting of goods, or to provide location-aware services using RFID technology.

Keywords: 8-B Microcontroller, Advanced Encryption Standard (AES), Elliptic Curve Cryptography, Elliptic Curve Digital Signature Algorithm (ECDSA), Embedded System, Implementation Security, Near-Field Communication (NFC), Radio-Frequency Identification (RFID), VLSI Design.

I. INTRODUCTION

Radio-Frequency Identification (RFID) is a wireless communication technique that has become increasingly important in the last decade. Applications such as electronic passports, logistics, animal identification, and car immobilizers already make use of this technology. A widely-used data-transmission standard based on RFID technology is near-field communication (NFC). With the integration of NFC functionality into the latest generation of mobile phones (Samsung Galaxy Nexus, HTC Ruby) a further spread of RFID technology is expected, paving the way for new applications. These new applications will have increased demand concerning the functionality provided by the RFID system, especially in the context of security and privacy. In a typical RFID system, a reader (e.g., a mobile phone) and a tag communicate remotely by means of an RF field. Most of the tags (more than 2 billion were sold in 2011) are so-called passive tags that also receive their power supply from the RF field. A passive tag is basically a microchip attached to an antenna. The simple design of passive tags allows them to be produced at low cost, which is important for applications where large numbers of tags are required. Tags used in future RFID applications will have to provide additional functionality, such as security and data-storage features. Moreover, the design of the tags must become more flexible to allow an easier adaption for new applications. Achieving these goals for passive tags by keeping the overhead in terms of power consumption (passive operation) and silicon area (directly influences the tag price) as low as possible is highly challenging.

A lot of effort has been made by the research community to allow cryptographic services on resource-constrained RFID tags. The most prominent services are strong authentication, using, for example, symmetric primitives like the advanced encryption standard (AES) [2], [3] or asymmetric primitives like elliptic curve cryptography (ECC) [4], [5]. The integration of asymmetric schemes is an especially big challenge for passive RFID tag designs as they need more resources (computational effort, memory, etc.) than symmetric schemes. When tags have to handle additional functionality, their control complexity also increases today's RFID tags use state machines fixed in hardware for handling their control tasks. However, this approach is no longer practical and even inefficient when the control complexity increases. Using a microcontroller approach instead is favorable and provides much more flexibility. Moreover, having a microcontroller on the tag

for handling the control tasks also allows reusing it for computing cryptographic operations.

In this paper, we present the design and implementation of a security-enabled NFC tag with flexible architecture. The so-called CRYPTA tag (**Cryp**tographic Protected **T**ags for new RFID **A**pplications) operates at a carrier frequency of 13.56 MHz and works fully passively. We target a low-area design that requires as little resources as possible such that the tag production does not exceed the practical limits of a possible commercial launch. The security-enabled NFC tag has a size of less than 50 kGEs and supports strong authentication features that are based on the AES-128 (symmetric cryptography) as well as on digital signing of data using the elliptic curve digital signature algorithm (ECDSA) over the prime field GF(p192) (asymmetric cryptography). The low-area goals have been achieved by heavily reusing existing hardware components, such as a common 8-b microcontroller or a common memory. Passive operation of the tag with conventional NFC-enabled mobile phones allows realizing security related NFC/RFID applications. Besides this, we also present a fully working prototype sample of our design fabricated on a 0.35-μm CMOS process technology. Our work contains multiple contributions that relate to the field of secure tag design, which are:

1. First low-resource RFID tag that supports asymmetric cryptography;
2. First combined low-area implementation of ECDSA and AES;
3. Flexible tag design based on a microcontroller for protocol handling and steering of the cryptographic module (including a design flow for program development); 4) first low-resource RFID tag with countermeasures against implementation attacks;
4. First prototype chip of a proof-of-origin tag;
5. Consideration of the whole tag life cycle including: production, personalization, and user application.

Among the contributions listed above, describing the design of a complete system, including all hardware components required for the practical chip fabrication of a security-enabled tag (including EEPROM and analog front-end which are often omitted in related work) is indeed the most valuable one. Moreover, we provide details of the design at a level that is hardly available in the published literature. The remainder of this paper is organized as follows. Section II provides Security Protocol Design. RFID Tag Architecture in Section III, Implementation results and a description of a prototyping sample are presented in Section IV. Conclusions are drawn in Section V.

II. SECURITY PROTOCOL DESIGN

In a security-enhanced RFID system, the level of security does not only rely upon the strength of the used cryptographic algorithms. The used protocols play a decisive role whether an attacker can successfully break into a system or not. Even if we use strong cryptographic algorithms, we need to ensure that the protocol is also secure. The protocol presented in this section allows the authentication of an RFID tag to a reader using the Advanced Encryption Standard (AES) as the cryptographic primitive. In RFID systems, the limited computing power and low-power constraints of the tags require special considerations concerning the used protocols. In addition to the available bandwidth for data transmission, attention should be paid to the compatibility to existing standards like the ISO/IEC 18000 or the Electronic Product Code (EPC).

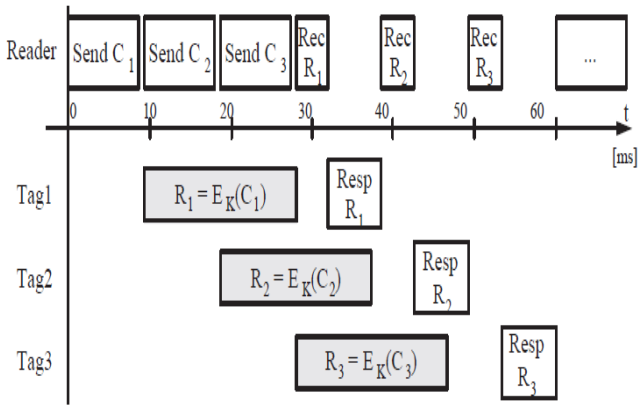

Fig.1. Interleaved challenge-response protocol in RFID systems.

The protocol is based on the unilateral authentication mechanism using random numbers presented. Integrating the presented challenge response authentication protocol into the ISO/IEC 18000 standard requires some additional considerations. In addition to the mandatory commands, which all tags must implement, custom commands can be specified. The two commands, integrated for authentication, are sending a challenge to the tag and requesting the encrypted value. These commands extend the existing standard although the basic functionality remains unchanged. Due to the low-power restrictions, the internal clock frequency of the RFID tag must be divided from 13.56 MHz to 100 kHz. The applied standard demands that a response must follow 320µs after a request. Otherwise, the tag has to stay quiet. This available time of 32 clock cycles at a frequency of 100 kHz is not enough for encrypting a challenge using the AES algorithm.

The solution to this problem is to modify the protocol as shown in Fig.1. The challenges and the responses to the tags are interleaved to each other. Normally, there are a lot of RFID tags to be authenticated in the environment of a reader. After retrieving all unique IDs of the tags using the inventory request and the anti-collision sequence, the reader sends a challenge C1 to Tag1. This tag immediately starts the encryption of the challenge without sending any response. In the meanwhile, the reader sends further challenges to the tags Tag2 and Tag3. They also start encrypting their challenges after reception. After finishing the encryption of EK (C1), Tag1 waits for the request to send the encrypted value R1 back to the reader. When the

reader has sent the three challenges, it sends a request for receiving the response from Tag1. The received value R1 is verified by encrypting the challenge C1 and comparing the result with the received value. The two other unanswered challenges are received using the same method. Then the reader starts from the beginning authenticating all other tags in the environment. This protocol was evaluated using high level models of the RFID communication channel and is a proof of concept for future research on authentication protocols in RFID systems. This interleaving challenge-response protocol has the advantage that each tag has at least 18 ms (1800 clock cycles at a clock frequency of 100 kHz) time for encryption. A maximum of 50 tags can be authenticated per second. If there are only few tags in the range of a reader, the reader can decide to make breaks of at least 18 ms instead of sending interleaved requests.

III. RFID TAG ARCHITECTURE

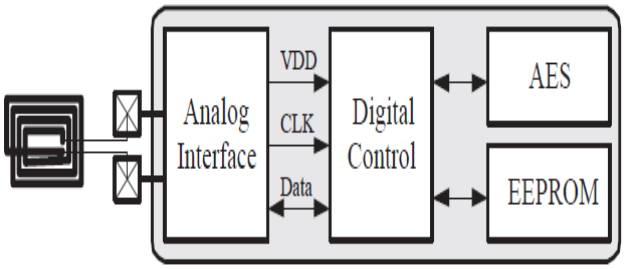

Fig.2. Architecture of an RFID tag.

The architecture of a security-enhanced RFID tag is sketched in Fig.2. It consists of four parts: analog frontend, digital controller, EEPROM, and AES module. The analog frontend is responsible for the power supply of the tag which is transmitted from the reader to the tag. Other tasks of the analog frontend are the modulation and demodulation of data and the clock recovery from the carrier frequency. The digital control unit is a finite state machine that handles communication with the reader, implements the anti-collision mechanism, and executes the commands in the protocol. Furthermore, it allows read and write access to the EEPROM and the AES module. The EEPROM stores tag-specific data like the unique ID and the cryptographic key. These data must be retained when the power supply is lost. The security-enhanced RFID tag calculates strong cryptographic authentication with an AES module which is designed for low power requirements and low die-size restrictions. The requirements concerning power consumption and chip area and a description of the AES module are presented in the following sections.

A. Requirements for RFID Tag Design

In order to achieve a significant economic benefit from using RFID systems, tags will need to be priced under US$ 0.10 for simple identification tags and a little bit higher for security-enhanced tags. Additionally to the aspect of low cost, the environmental conditions play a decisive role because contactless identification must work within a distance of a few meters. The limiting factors thereby are the available power supply for the tag and the signal strength for modulation and demodulation. The available power consumption for the digital part of the RFID tag (digital controller and AES module) is amounting to 20 µA. Estimating the current consumption of the digital controller to 5 µA, 15 µA remain for the AES module which should not exceed a chip area of 5,000 gates. Additionally, the number of authenticated tags per second is about 50. As presented in chapter 3, this leads to an available time slot of 18 ms for encrypting a 128-bit block of data. Our proposed AES architecture, which is presented in section 3.2, encrypts in about 1000 clock cycles. As a consequence, the clock frequency of the AES module can be reduced fewer than 100 kHz. This allows reaching the ambitious power consumption goal.

B. AES Architecture

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm which was selected in 2001 by the National Institute of Standards and Technology (NIST) as the Federal Information Processing Standard FIPS-197. It operates on blocks of data, the so called State, that have a fixed size of 128 bits. The State is organized as a matrix of four rows and four columns of bytes. The defined key lengths are 128 bits, 192 bits, or 256 bits. Our implementation uses a fixed key size of 128 bits. As most symmetric ciphers, AES encrypts an input block by applying the same round function. The ten round function iterations alter the State by applying non-linear, linear, and key-dependent transformations. Each transforms the 128-bit State into a modified 128-bit State. Every byte of the State matrix is affected by these transformations:

1. Sub Bytes substitutes each byte of the State. This operation is non-linear. It is often implemented as a table look-up. Sometimes the Sub Bytes transformation is called S-Box operation.
2. Shift Rows rotates each row of the State by an offset. The actual value of the offset equals the row index, e.g. the first row is not rotated at all; the last row is rotated three bytes to the left.
3. Mix Columns transforms columns of the State. It is a multiplication by a constant polynomial in an extension field of GF (28).
4. Add Round Key combines the 128-bit State with a 128-bit round key by adding corresponding bits mod 2. This transformation corresponds to a XOR-operation of the State and the round key.

The calculation of the 128-bit round keys works by applying the Key Schedule function. The first round key is equal to the cipher key. The computation of all other round keys is based on the S-Box functionality and the R-con operation. AES is a flexible algorithm for hardware implementations. A large number of architectures are possible to cover the full range of applications. AES hardware implementations can be tailored for low die-size demands in embedded systems or can be optimized for high

throughput in server applications. This flexibility of the AES algorithm was intended by its creators. They paid attention that the algorithm can be implemented on systems with different bus sizes. Efficient implementations are possible on 8-bit, 32-bit, 64-bit, and 128-bit platforms. Although many AES hardware architectures have been proposed, none of the reported architectures meets the requirements of an AES module for RFID tags regarding low die-size and low power-consumption requirements. Nearly all of these architectures have G-Bit throughput rates as optimization goal. This is contrarious to our needs where throughput is not of concern. Only a few published AES architectures do not optimize throughput at any cost. To name some: are FPGA implementations and ASIC implementations of AES which care about hardware efficiency. All these implementations do not unroll the AES rounds for sake of silicon size. The more S-Boxes are used, the less clock cycles are needed for encryption.

The encryption-only AES processor of a 128-bit architecture that utilizes 32 S-Boxes it is able to calculate one AES round in a single clock cycle. The compact 32-bit AES architecture is confident with four S-Boxes and takes eight cycles for one round. The FGPA implementations of N 32-bit architectures they also use four S-Boxes. Four S-Boxes suit a 32-bit architecture as each S-Box substitutes 8 bits. The Mix Columns operation and the Shift Rows operation are 32-bit operations too because they transform either four columns bytes or four row bytes of the AES State. The Add Round Key operation (128-bit XOR) can also be split-up into 32-bit operations. Implementing the AES algorithm as a 32-bit architecture allows quartering the hardware resources compared to a 128-bit architecture. This comes at the expense of quadrupling the time for an AES encryption. The lower amount of hardware resources has a positive side effect on the power consumption: a quarter of hardware resources consume only a quarter of power. This is an important feature for wireless devices where the average power consumption is an even more important quality aspect than the overall energy needed to encrypt one block.

The overall energy consumption of a 32-bit architecture might be worse than for 128-bit architectures. But RFID tags offer neither the silicon space nor are the electromagnetic field strong enough to power a 128-bit data path. The power requirements for RFID tags are even too restrictive to allow the operation of a 32-bit AES implementation. Therefore, we decided to implement the AES algorithm as an 8-bit architecture instead of a 32-bit architecture. This novel approach for a hardware implementation of the AES algorithm is motivated by two reasons. First, an 8-bit architecture allows decreasing the number of S-Boxes from four to one to save silicon resources. Second, 8-bit operations consume significantly less power than 32-bit operations do. A penalty of an 8-bit architecture is the increased number of clock cycles for encryption. In RFID authentication applications and encryption lasting for 1000 cycles does not deteriorate the authentication throughput when several tags are authenticated concurrently.

The architecture of the proposed 8-bit AES module is depicted in Fig.3. It is presumably the smallest hardware implementation of the AES algorithm.

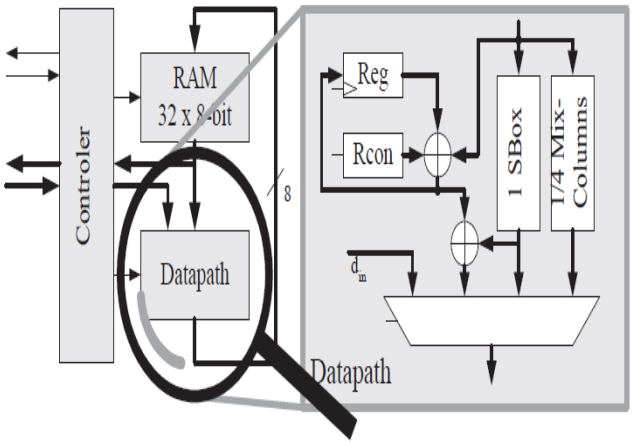

Fig.3. Architecture of the AES module.

The module consists basically of three parts: a controller, RAM, and a data path. The controller communicates with other modules on the tag to exchange data and it sequences the ten rounds of an AES encryption. Therefore, it addresses the RAM accordingly and generates control signals for the data path. The RAM stores the 128-bit State and a 128-bit round key. These 256 bits are organized as 32 bytes to suit the intended 8-bit architecture. 32 bytes are the smallest possible memory configuration for AES. The memory is single ported to ease silicon implementation. Modified States and calculated round keys overwrite previous values. As no spare memory is present for storing intermediate values, the controller has to assure that no State byte nor is a key byte overwritten if it is needed again during encryption. The RAM implementation is register based. It makes use of clock gating to minimize power consumption. The data path of the AES module contains combinational logic to calculate the AES transformations Sub Bytes, Mix Columns, and Add Round Key (see Fig.3). The Shift Rows transformation is implemented by the controller. During the execution of Sub Bytes the controller addresses the RAM such that the Shift Rows operation is executed.

The biggest part of the AES data path is the S-Box which is used for the Sub Bytes operation. There are several options for implementing an AES S-Box. The most obvious option is a 256 × 8-bit ROM to implement the 8-bit table lookup. Unfortunately, ROMs do not have good properties regarding low-power design. A more appropriate option is to calculate the substitution values using combinational logic as presented. We adapted the proposed combinational S-Box by omitting the decryption circuitry to suit our encryption-only AES. One feature of this S-Box is that it can be pipelined by inserting register stages. The S-Box

makes use of one pipeline stage. This shortens the critical path of the S-Box to seven XOR gates and lowers glitching probability. Moreover, the pipeline register is used as intermediate storage for a pipelined Sub Bytes operation: during the substitution of one byte, the next byte is read from the memory. The substituted byte is written to the current read address. By choosing the read addresses properly this procedure combines efficiently the Sub Bytes and the Shift Rows operation. Shift Rows degrades to mere addressing.

IV. IMPLEMENTATION RESULTS

We have implemented our flexible tag platform in VHDL and designed it toward low resource usage and low power consumption, i.e., by applying clock gating and operand isolation techniques. We implemented our design in a 0.35-μm CMOS technology using a semi-custom design flow with Cadence RTL Compiler as synthesis tool. Table I shows the chip-area results in terms of gate equivalents (GEs). In total, the chip needs 49 999 GEs, including analog front-end, FL, microcontroller, bus arbiter, CU, and memory. About 21% (10 763 GEs) is needed for the RFID analog front-end and the FL. The microcontroller needs around 19%, including instruction unit, ALU, PC, register file (about 65 GEs per register), and program ROM. The data path and the pattern sequencer of the CU take about 15% of the chip area, i.e., 7488 GEs (this number does not include the ROM for ECDSA, AES, and SHA-1 program and the needed constants). The highest amount of resources is required for the memory, i.e., about 44% of the total area, which equals to 22 027 GEs. The smallest component by far is the bus arbiter (responsible for the AMBA bus), consuming less than 1% of the total area.

TABLE I: AREA OF CHIP COMPONENTS

Component	GEs	%
Analog front-end	8100	16.20
FL	2663	5.33
8-b microcontroller		
Instruction decode unit, ALU, and PC	945	1.89
Register file (26 × 8-b)	1693	3.38
Program ROM (2027 × 16-b)	6764	13.53
Bus arbiter	319	0.64
CU		
Micro-code pattern sequencer	3880	7.76
Datapath (ECDSA, AES, and SHA-1)	3608	7.22
Memory unit		
EEPROM (256 × 16-b)	12 700	25.40
ROM (CU constants)	600	1.20
RAM macro (128 × 16-b)	8727	17.45
Total	**49 999**	**100.00**

The RFID front-end is clocked with 106, 212, or 448 kHz according to the specified data rate. The CU can be clocked at higher frequencies (0.847, 1.7, 3.3, or 6.68 MHz) in order to improve the performance (configured in an EEPROM register during tag personalization). At a frequency of 1.7 MHz, a digital signature can be generated within 505 ms, i.e., 863 109 clock cycles. Hashing a message needs 2.15 ms (3639 clock cycles) and AES needs 2.66 ms (no dummy rounds) and 9.16 ms (ten dummy rounds applied) which corresponds to 4529 and 15 577 clock cycles, respectively. At the highest frequency of 6.68 MHz, the ECDSA module needs 127 ms for generating a digital signature, which is sufficient for most applications having stringent response-time requirements.

TABLE II: DISTRIBUTION OF ROM CODE WITH RESPECT TO TAG FUNCTIONALITY

Tag Functionality	Code Size	
	[Instructions]	[%]
Protocol		
Generic subroutines	312	15.40
Block transmission	499	24.60
File management	331	16.30
Security features	119	5.90
Crypto services		
ECDSA-P192	487	24.00
AES-128	223	11.00
SHA-1	56	2.80
Total	**2027**	**100.00**

A. Program ROM

After developing and evaluating the program of the microcontroller with the Java-based ISS described in the assembler was used to transform the assembly code into synthesizable VHDL ROM code. Proper operation of the whole tag has been further verified through simulations with Cadence NC Sim and through tests on an FPGA RFID tag prototype that can communicate with different reader devices. The final ROM code for the microcontroller contains 2027 instructions (equals 4054 B of code). Subroutine calls are used whenever possible to keep code size small. Table II shows the distribution of the ROM code with respect to tag functionality. Most instructions of the ROM code, about 25%, are only used for handling the block-transmission protocol. Around 15% of the instructions are utilized for generic subroutines that provide a basic set of functions that are reused multiple times (e.g., routines for accessing the AMBA bus). File management and security features require about 22%. The program part for steering the CU needs 766 instructions, corresponding to about 38% of the total program ROM (24% for ECDSA-P192, 11% for AES encryption/decryption, and 2.8% for SHA-1).

Most of the instructions stored in the ROM relate to protocol handling, illustrating the high control complexity of our tag design. However, the code used for steering the CU also comprises mainly control instructions (e.g., for executing micro-code patterns). Analyzing the code in the ROM in detail shows that about 60% of the instructions are control operations (CALL, RET, BNZ, MICRO). Only 10% of the instructions relate to pure data-flow oriented

operations between one or two registers (XOR, ADD, ROT). The rest of the instructions belongs to operations between constants in ROM and registers, e.g., immediate load and compare instructions (MOVLF, XORLF).

B. Power Consumption

Power simulations of the system were conducted with the transistor-level SPICE simulator Synopsys Nanosim. The simulation for the microcontroller shows a mean power consumption of only about 10 μA for the 0.35-μm CMOS process technology when powered with a supply voltage of 2V and using a clock frequency of 106 kHz, i.e., for a default data rate of 106 kb/s. When higher data rates are selected, the power consumption increases accordingly (linearly with data rate). The CU, in contrast, consumes about 485 μA as total mean current measured at 847 kHz, i.e., the lowest frequency for the CU. More than 40% of that power is due to the memory unit, which is heavily used during scalar multiplication. The data path unit needs about 24%, the clock tree requires approximately 16%. Note that the overall power consumption of the system is already quite low due to low-power design techniques, such as clock gating and operand isolation. It meets the power requirements of most HF RFID systems and can be applied in different RFID or NFC applications. However, the power consumption value can be even further decreased by moving toward a more-advanced CMOS process technology, e.g., 0.18 or 0.13 μm. Using these technologies, the reading distance becomes even better and can be applied, e.g., in long range ISO/IEC 15 693 applications.

C. RFID-Tag Prototyping Sample

We manufactured our RFID-tag implementation on a multi-project wafer using the 0.35-μm CMOS process technology C35b4 from AMS AG. For ease of testability, a small serial debug interface has also been added that allows detailed analysis of the analog front-end and the EEPROM *(*e.g., reading/writing arbitrary values from/to EEPROM*)*. A photo of the manufactured chip is shown in Fig.4. After production, the chip has been integrated into a ceramic package and soldered on a small printed circuit board (PCB) to allow tests with real-world RFID-reader devices.

Fig.4. Photo of the manufactured RFID tag-prototype chip.

Fig.5. Proof-of-origin application using our RFID-tag prototyping sample and the Google Nexus S mobile phone.

The PCB contains an antenna with four windings that is connected to the analog front-end of the chip. An adjustable capacitor is used for matching of antenna and analog front-end. Fig.5 shows a photo of the PCB with the packaged chip. We successfully tested the RFID-tag sample with different commercially available RFID readers, including mobile devices featuring NFC capabilities. Using the Google Nexus S, for example, the tag can be powered fully passively and can reliably communicate with the phone up to 3 centimeters (at data rates up to 424 kb/s and frequencies up to 6.68 MHz for the CU). Using our flexible tag platform, different RFID and NFC applications have been realized, such as proof-of-origin authentication to thwart against counterfeiting goods, or to generate location aware signatures to prove that a person or object has been at a certain location in a specific moment in time. Several press releases have been published that demonstrate these demo applications.

D. Comparison with Related Work

Comparing our results with related work is rather difficult as only a handful of publications exist that deal with implementing security-enabled tags. Moreover, authors often give only a vague description of their designs regarding implementation details and provided functionality and presented tag designs for the ultrahigh frequency (UHF) range that contain an AES-128 implementation. The AES implementations used by them have an area requirement of about 6–7 kGEs. Moreover, the two tag designs cover only the baseband part, i.e., the digital circuit without EEPROM and analog front-end. A design that is better comparable to our work is the one of the authors presented an NFC tag, including EEPROM (4 kb, i.e., same size as ours), analog front-end and cryptographic unit with AES-128. Their NFC tag has a similar size (i.e., around 50 kGEs) than our design, but supports neither asymmetric cryptography (nor SHA-1) nor has it countermeasures against implementation attacks integrated. This illustrates the advantage of our design concept that provides not only high flexibility but also very low resource usage when considering all the implemented features.

V. CONCLUSION

In this paper, we presented a flexible NFC-tag architecture that provides enhanced security features using symmetric as well as asymmetric cryptography. As a main contribution, the work described an entire "real-world" RFID system, including all hardware components needed for a practical chip fabrication. During the work, several outcomes were obtained. First, our design showed that significant resources can be saved by applying a microcontroller-based architecture instead of using a finite-state machine-based controlling. The reason lies in the fact that the controller can be simply reused by many hardware components, such as the CU or the RFID FL that would require more area when implemented as individual hardware modules. For example, AES encryption and decryption has been realized with an area overhead of only 2387 GEs, which is lower than existing low-area AES implementations.

Furthermore, SHA-1 needs only 889 GEs because of reusing available memory and microcontroller components of the entire system. Next to these outcomes, we found that it is favorable to reuse the microcontroller for RFID protocol handling, e.g., handling ISO/IEC 14443 layer 4. This can be completely realized as a micro program, which reduces further chip-area requirements while increasing flexibility and assembly-based implementation convenience. Finally, we practically proved our design by fabricating the system as a prototyping sample that demonstrates the feasibility of a full-blown RFID/NFC tag supporting ISO/IEC 14443A layer 1–4, NFC Forum Type-4 features (including NDEF support), a flexible (programmable) 8-b microcontroller, memory (RAM, ROM, and EEPROM), analog frontend, and strong cryptography (ECDSA and AES) for less than 50 kGEs. In the future, we plan to further analyze our design regarding enhanced implementation attacks, such as side channel analysis and fault attacks. Moreover, we plan to implement additional demo applications to verify the applicability of our tag in different security-related scenarios.

VI. REFERENCES

[1] Thomas Plos, Michael Hutter, Martin Feldhofer, Maksimiljan Stiglic, and Francesco Cavaliere, "Security-Enabled Near-Field Communication Tag With Flexible Architecture Supporting Asymmetric Cryptography", IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 21, No. 11, November 2013.

[2] M. Feldhofer, S. Dominikus, and J.Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," in Proc. CHES, vol. 3156. Aug. 2004, pp. 357–370.

[3] P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D. Hämäläinen, "Design and implementation of low-area and low-power AES encryption hardware core," in Proc. 9th EUROMICRO Conf. Digit. Syst. Design, Sep. 2006, pp. 577–583.

[4] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls, and I. Verbauwhede, "Public-key cryptography for RFID-tags," in Proc. RFID sec, 2006, pp. 1–16.

[5] P. Tuyls and L. Batina, "RFID-tags for anti-counterfeiting," in Topics in Cryptology, vol. 3860, D. Point cheval, Ed. New York: Springer-Verlag, 2006, pp. 115–131.

[6] NFC Forum Type 4 Tag Operations - Technical Specification. (2007 Mar.) [Online]. Available: http://www.nfc-forum.org/specs.

[7] Identification Cards - Contactless Integrated Circuit(s) Cards – Proximity Cards - Part 3: Initialization and Anti-collision, ISO/IEC Standard 14443-3, 2001.

[8] Identification Cards - Contactless Integrated Circuit(s) Cards – Proximity Cards - Part 4: Transmission Protocol, ISO/IEC Standard 14443-4, 2008.

[9] Information Technology - Identification Cards - Integrated Circuit(s) Cards with Contacts - Part 4: Inter-industry Commands for Interchange, ISO/IEC Standard 7816-4, 1995.

[10] National Institute of Standards and Technology. (2001, Nov). FIPS-197: Advanced Encryption Standard, Gaithersburg, MD [Online]. Available: http://www.itl.nist.gov/fipspubs/.

[11] National Institute of Standards and Technology. (2009). FIPS-186-3: Digital Signature Standard (DSS), [Online]. Available: http://www.itl.nist.gov/fipspubs/.

[12] T. Plos and M. Feldhofer, "Hardware implementation of a flexible tag platform for passive RFID devices," in Proc. 14th Euro-micro Conf. Digit. Syst. Design, Aug. 2011, pp. 293–300.