Demo: The F_f Hardware Prototype for Privacy-Preserving RFID Authentication

Erik-Oliver Blass[1] Kaoutar Elkhiyaoui[1] Refik Molva[1] Olivier Savry[2] Cédric Vérilhac[2]
[1]EURECOM, Sophia Antipolis, France {blass|elkhiyao|molva}@eurecom.fr
[2]CEA/LETI, Grenoble, France {olivier.savry|cedric.verilhac}@cea.fr

ABSTRACT

In this demo, we present the realization and evaluation of a wireless hardware prototype of the previously proposed RFID authentication protocol "F_f". The motivation has been to get as close as possible to the (expensive) construction of a wafer and to analyze and demonstrate F_f's real-world feasibility and *functional correctness* in the field. Besides showing F_f's feasibility, our objective is to show implications of embedding authentication into an industry RFID communication standard. Apart from the documentation at hand, the **demonstrator** comprises the F_f RFID tag and reader prototypes and a standard EPC tag and reader. The hardware is connected to a laptop controlling the hardware and simulating attacks against authentication.

Categories and Subject Descriptors

B.7.m [**Hardware**]: Integrated Circuits—*Miscellaneous*

General Terms

Algorithms, Design, Security

1. INTRODUCTION

Privacy-preserving authentication of RFID tags is a challenging research problem due to tags' extreme hardware restrictions. With an expected target price of a few cents, tags feature only a couple of thousands "gates", i.e., electronic circuits. This renders complex cryptographic protocols infeasible. Recently, a plethora of research papers has been published that addresses RFID authentication. Papers typically design and evaluate solutions only theoretically, giving rough estimates for implementation costs and gate count. However, a real-world authentication protocol implementation has to be integrated into the state machine of the tag's wireless communication stack, e.g., ISO 18000-3 (EPC Global HF Gen 2) compatible. This is far from being trivial, as it requires a tag hardware ("chip") re-design and modifications to the RFID reader's communication stack. Such an integration has a serious impact on the resulting true hardware costs, tags' response times etc.

We present the F_f hardware implementation, comprising **1.)** the design of the F_f authentication part in hardware. Using a target technology of 130nm, synthesis results show a total chip area of $\approx 9,300\mu m^2$ ($\approx 1,500$ gate equivalents) for F_f. Execution time per authentication is $2.26ms$ at standard EPC 13.56MHz clock frequency.

2.) extending the finite state machine of an industry EPC Global HF Gen 2 RFID tag communication stack to embrace the F_f functionality. Therewith, area complexity for the complete RFID chip increases by a factor of ≈ 2.6 (total of $\approx 4,000$ gate equivalents). Total time per authentication increases to $\approx 10ms$ including *all* required communications.

3.) manufacturing an EPC compliant RFID tag for F_f. Together with a modified LETI RFID reader firmware, we have successfully verified our implementations in the field.

4.) developing a PC graphical user interface that connects to the LETI reader and visualizes the authentication and all data sent to and received from the tag. We demonstrate the different attacks an adversary might perform, such as eavesdropping, man-in-the-middle, etc. To compare to standard, identification, we also use our GUI to connect to an off-the-shelf EPC RFID reader, and we visualize communication with off-the-shelf tags. We show on the one hand, how standard EPC identification is prone to (privacy) attacks, and on the other hand, how F_f protects identification.

This paper also presents and discusses our "lessons learned" to contribute to and stipulate further research.

2. F_f OVERVIEW

Although the F_f protocol design is not subject of the paper at hand, we will briefly introduce its main concepts, necessary to understand the implementation and demo. For details, see original publications [1–3].

A tag T stores two keys $K_1, K_2 \in GF(2^{t \cdot l})$. For each valid tag in the system, the reader stores the tag's ID and its two keys. Both, tag and reader implement a function F_f and a linear feedback shift register (LFSR). F_f takes a key K (either K_1 or K_2) and a random number R as input and is computed using a small, lightweight fan-in function f. We rewrite $K, R \in GF(2^{t \cdot l})$ as $K = (k_1, k_2, \ldots, k_l)$, $R = r_1, r_2, \ldots, r_l$, with $k_i, r_i \in GF(2^t)$, and compute $F_f(K, R) := \sum_{i=1}^{l} f(k_i, r_i) \in GF(2^t)$.

Small fan-in function f is defined bitwise, but its exact design does not matter here.

An F_f authentication comprises two messages. The first message is sent from the reader to the tag, and the second message is the answer from the tag to the reader. In the first message, the reader only sends a random number N. The tag replies with the message $\{R, w_1, w_2, \ldots, w_q\}$. Here, R is a random number chosen by the tag, and the $w_i \in GF(2^t)$ are computed as $w_i := F_f(K_1, R_i^{a_i}) \oplus F_f(K_2, N_i)$. Taking N as a seed for the LFSR, the N_i are computed by iterating

CCS'11, October 17–21, 2011, Chicago, Illinois, USA.
ACM 978-1-4503-0948-6/11/10.

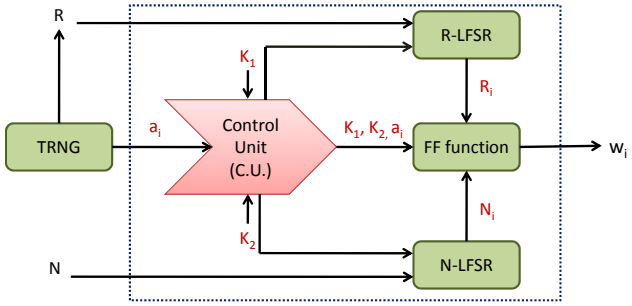

Figure 1: Conceptual blocks of F_f

the LFSR, respectively, $(l \cdot t)$ times. To compute value $R_i^{a_i}$, the tag initially loads R as a seed into the LFSR. For each w_i, the tag generates d values $R_i^1, R_i^2, \ldots, R_i^d$ by iterating the LFSR. It picks a random number a_i for each w_i and selects $R_i^{a_i}$. Therewith, the tag can compute w_i.

After receiving $\{R, w_1, w_2, \ldots, w_q\}$, the reader can for each w_i recompute all possible $R_i^1, R_i^2, \ldots, R_i^d$ and N_i. The reader iterates through the database containing all key pairs K_1, K_2. For the d possible $R_i^{a_i}$, the reader computes $w_i' := F_f(K_1, R_i^{a_i}) \oplus F_f(K_2, N_i)$. If none of these w_i' matches w_i for a given key pair, the reader can remove this tag from the database. Eventually, after evaluating all w_i, the reader "converges" to a single entry in the database. With high probability, the tag is therewith authenticated. The value of q, l, t, d and the size of N and R are security parameters in F_f. For good security, we set $q = 60, l = 64, t = 4, d = 8, |N| = |R| = 64$ [3].

3. CHIP DESIGN & SYNTHESIS

To implement F_f in hardware, we identify the conceptual blocks of F_f and thus define the general architecture as shown in Fig. 1. More precisely, F_f consists of:

Non-volatile memory. This stores the secret keys K_1 and K_2 each of size 256 bits.

A TRNG (true random number generator). The TRNG generates random numbers that are distributed according to the uniform distribution. The TRNG is implemented using high entropy noise from the radio interface. This generator is responsible for the randomness required on the tag side, i.e., R and $a_i \in [1, d], 1 \leq i \leq q$. Although being crucial for security, details of this TRNG's implementation are out of scope this paper, as this TRNG has been provided by hardware manufacturer LETI.

A linear feedback shift register R_LFSR. This LFSR takes as input the random number R generated by the TRNG. At each round i of the protocol, it generates $d = 8$ random numbers $R_i^j, 1 \leq j \leq d$, where $R_0^d := R$, $R_i^1 := \text{R\_LFSR}(\text{R}_{\text{i}-1}^{\text{j}})$, and $R_i^j := \text{R\_LFSR}(\text{R}_{\text{i}}^{\text{j}-1}), j > 1$.

A second linear feedback shift register N_LFSR. This LFSR takes as input the nonce N received from the reader. At each round round i, it generates a random number N_i defined as $N_i = \text{N\_LFSR}(\text{N}_{\text{i}-1})$ and $N_0 = N$.

A control unit C.U. With input K_1 and K_2 stored on the non volatile memory and a_i generated by the TRNG, the C.U. activates the two LFSRs and the F_f function module, and it controls the data flow between these blocks. Also, it keeps track of the computation state of the F_f module, e.g., to send a signal that indicates the end of the computation.

The F_f core function. The purpose of the module is to compute the w_i for each of the q rounds, such that $w_i = F_f(K_1, R_i^{a_i}) \oplus F_f(K_2, N_i)$. Instead of dedicating two functional blocks of F_f to compute w_i, we choose to use the same block and switching the inputs each time. While this saves area, it comes at a slightly larger time for computation.

Using this architecture, we have implemented the authentication protocol in VHDL.

Table 1: Synthesis Report with 13.56MHz [4]

		Register	Inverter	Logic	Buffer	Total
F_f	Instances	4	2	69	-	75
	Area [μm^2]	137.170	8.069	766.536	-	911.775
	Area %	15	0.9	84.1	-	100
C.U.	Instances	16	13	60	-	89
	Area [μm^2]	566.833	52.447	665.676	-	1284.956
	Area %	44.1	4.1	51.8	-	100
per LFSR	Instances	64	17	75	1	157
	Area [μm^2]	2194.714	68.585	1264.784	6.052	3534.135
	Area %	62.1	1.9	35.8	0.2	100

In order to evaluate the feasibility of implementing F_f on real world RFID tags, we have **synthesized** our VHDL implementation. The synthesizer used is Cadence RTL Compiler Version v08.10-p104_1 which synthesizes for a typical ASIC target of $130nm$ technology. The synthesizer outputs an area report from our VHDL implementation. The area report gives the silicon area in μm^2 of registers, inverters, logic elements, and the total area of the design. To derive the number of gate equivalents, we divide the total area by 6.0518 μm^2, which is the area to implement a NAND gate in ASIC $130nm$ using the ST-Microelectronics library HCMOS9GP/CORE9GPLL. The total area of our design is $\approx 9,300 \mu m^2$, i.e., $\approx 1,500$ gate equivalents. This is well below the often quoted figure of $2,000$ gate equivalents available for security [5]. F_f is executed in $2.26ms$ using EPC's clock frequency of 13.56MHz.

As in related work, the area report summarized in Table 1 does not take into account ROM used to store the secret keys K_1 and K_2. The synthesizer considers each memory element as a register which is expensive. Instead, we map the keys into ROM memory cells. We can estimate the area of our 512 bits ROM to 512 gates by mapping each bit stored to one gate equivalent. Interestingly, each LFSR occupies nearly 37% of the total area, compared to the F_f core function occupying less than 10%. An optimization to the current design could consist of using one single LFSR instead of two, however, this will lead to a slower execution time and therefore, to a slower response time from the tag.

4. INTEGRATION INTO EPC GLOBAL

After synthesis, we have integrated F_f into the EPC protocol stack. This automatically implies that F_f uses standard medium access protocol ("singulation", more on this later) and data exchange techniques. For integration and validation, we have used industry EPC compatible hardware. More precisely we have been provided with:

1.) A new RFID tag prototype: the LETI-EPSIS tag. This is an FPGA of type ALTERA EP3C16E144C8N with standard EPC wireless communication facilities. Moreover, we have been provided with an industry EPC finite state machine for wireless communication in VHDL – we have extended this by F_f and have downloaded into the tag.

2.) The LETI-LRF RFID reader. This is an ALTERA Cyclone II EP2C35U484C8 FPGA supporting a NIOS II embedded processor. We have used an EPC compliant reader protocol stack that we again extended for F_f.

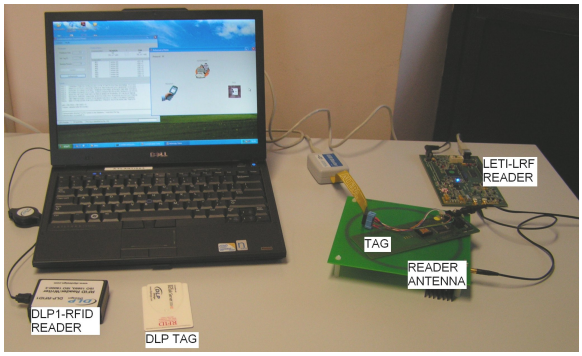

Figure 2: Hardware setup used for F_f integration

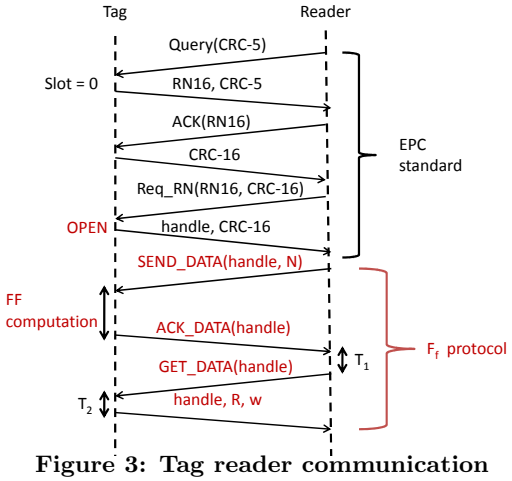

Figure 3: Tag reader communication

The tag and the reader communicate wirelessly using standard EPC Global HF Gen 2 (ISO 18000-3) mechanisms.

Integration details: The tag's finite state machine specifies the different EPC states (ready, acknowledged, reply, open, arbitrate, killed) of the tag and the tag responds to reader commands. Now, we integrate F_f as part of the EPC "open" state, cf., Fig. 3. Roughly speaking, the tag enters the open state once it is selected by the reader through EPC's medium access control. In this paper we omit the details of EPC's singulation protocol, however we point out that during singulation the tag is required to send its identifier (PC/XPC, EPC). EPC communication is query-response based. A tag cannot initiate communication, but has to wait for a reader query and can send a response. Therefore, we use these three communication primitives:

SEND_DATA: Sent by the reader to the tag. It carries random N along with "handle", a random session identifier.

ACK_DATA: Sent by the tag to the reader. It acknowledges that the tag has correctly received N and that it has finished computing its response.

GET_DATA: Sent by the reader to the tag. It asks the tag to send its response, i.e., tuple $(R, w_1 \mid\mid w_2 \mid\mid ... \mid\mid w_q)$.

After singulation as depicted in Fig. 3, the reader queries with SEND_DATA. Receiving SEND_DATA, the tag computes its response using the F_f function, i.e., computation of the w_i. Finally, the tag sends command ACK_DATA.

Synthesis of the chip including the EPC state machine revealed an area increase by a factor of ≈ 2.6 for the total chip and a total response time increase by a factor of ≈ 5.

5. GRAPHICAL USER INTERFACE

We have developed a graphical user interface for Windows that controls both the LETI-LRF reader and an off-the-shelf DLP-1 EPC reader. Therewith, wireless communication with F_f or EPC standard tags can be run. The GUI visualizes which information an eavesdropping adversary can deduce and allows, moreover, user ("adversarial") *interaction*: the user/adversary can modify exchanged values such as N, R or w_i to simulate man-in-the-middle attacks. In the GUI, the user can select between normal protocol execution and attacks and start either the authentication with F_f or simple identification with the DLP-1 reader. During authentication/identification, the GUI logs messages exchanged between tags and readers and displays reader computations, i.e., "converging" to a single database entry. Finally, the GUI displays whether the current tag has been correctly authenticated by the reader and whether the adversary has recognized that tag.

6. DISCUSSION

One interesting finding was that the computation part of F_f and the control unit only consumed $\approx 24\%$ of the chip area. The most expensive part of the authentication protocol is the two 64 bit LFSR, cf., Table 1. As registers ("buffers") are surprisingly expensive compared to computation, future protocol design should try trade-in buffer space against computation (inverse time-memory trade-offs). One has to keep in mind that increasing computations might eventually collide with EPC response time constraints.

Standard EPC communication uses *Aloha* singulation for layer 2 medium access control where the tag broadcast its identifier for further communication. Therewith, standard EPC Aloha medium access already spoils privacy, and no authentication protocol following singulation can preserve a tag's privacy anymore. Consequently, standard medium access before authentication needs to be privacy-preserving, too – which again will add complexity. Although similar issues have been reported for UHF's tree-walking singulation, these issues are still ignored in today's research.

The total tag response time has increased by factor 5 compared to the time required for only authentication. This might limit applicability of such authentication protocols in many scenarios where a high identification rate is important (scanning bunches of tags). Moreover, real-world bunch scanning will be much lower due to collisions, interference etc. Integration of F_f into the EPC stack increased chip area by a factor of 2.6. Surprisingly, this indicates that privacy-preserving authentication is by far not as complex as the standard EPC state machine.

[1] O. Billet and K. Elkhiyaoui. Two attacks against the F_f rfid protocol. In *Proceedings of INDOCRYPT*, pages 308–320, 2009.

[2] E.-O. Blass, A. Kurmus, R. Molva, G. Noubir, and A. Shikfa. The F_f family of protocols for rfid privacy and authentication. In *5th Workshop on RFID Security*, pages 1–15, Leuven, Belgium, 2009.

[3] E.-O. Blass, A. Kurmus, R. Molva, G. Noubir, and A. Shikfa. The F_f family of protocols for rfid privacy and authentication. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 8 (3):466–480, 2011. ISSN 1545-5971.

[4] A. Di Martino and V. Gehlen. Authentication and privacy in rfid-systems, 2010. `http://www.eurecom.fr/~blass/rfid-report.pdf`.

[5] S.A. Weiss and A. Juels. Authenticating pervasive devives with human protocols. In *Proceedings of CRYPTO'05*, pages 293–308, Santa Barbara, USA, 2005. ISBN 3-540-28114-2.