Graz University of Technology
IAIK
Institute for Applied Information Processing and Communications
Inffeldgasse 16a
A–8010 GRAZ

ECRYPT

Workshop on RFID and Lightweight Crypto

ECRYPT Workshop on RFID and Lightweight Crypto,

July 13-15, 2005,Graz University of Technology, Graz, Austria

http://www.iaik.tu-graz.ac.at/research/krypto/

Preface

The use of state-of-the-art cryptographic methods on RFID tags opens a new range of applications for these tags and for cryptography. The aims of the ECRYPT[1] workshop on RFID and Lightweight Crypto are to increase the awareness for cryptographic methods and solutions among RFID developers, and for the requirements of this heavily constrained environment among cryptographers.

The workshop brings together researchers and developers from industry and academia, in order to exchange novel ideas and experiences. The scope includes, but is not limited to, the following topics:

- Applications for RFID tags
- Cryptographic algorithms for constrained environments
- Cryptographic protocols adapted to RFID applications
- Low-power implementations

The workshop program consists of invited talks and contributed presentations. The workshop proceedings contain the revised articles that were accepted for presentation.

Thanks go to Vincent Rijmen for serving as program chair. Thanks go to Phong Nguyen, Christof Paar, Bart Preneel and Matt Robshaw for serving as program committee.

July 2005 Elisabeth Oswald

[1] The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Organization

Sponsoring Institutions

This workshop is organized by members of ECRYPT, the network of excellence in cryptology funded within the Information Societies Technology (IST) Programme of the European Commission's Sixth Framework Programme (FP6) under contract number IST-2002-507932.

The workshop is hosted by the Institute for Applied Information Processing and Communications at the Graz University of Technology in Graz, Austria.

The workshop is sponsored by Philips Semiconductors Styria and by Siemens CT Munich.

Table of Contents

Ciphers

A Scalable, Delegatable Pseudonym Protocol Enabling Ownership Transfer of RFID Tags

David Molnar, Andrea Soppera, and David Wagner

UC Berkeley, British Telecom Research, UC Berkeley

Abstract. The ability to link two different sightings of the same Radio Frequency Identification (RFID) tag enables invasions of privacy. The problem is aggravated when an item, and the tag attached to it, changes hands during the course of its lifetime. After such an *ownership transfer*, the new owner should be able to read a tag but the old owner should not. We address these issues through an RFID tag *pseudonym protocol*. Each time it is queried, an RFID tag emits a different pseudonym using a pseudo-random function and an increasing counter. Without consent of a special Trusted Center that shares secrets with the tag, it is infeasible to map the pseudonym to the tag's real identity. We present a scheme for RFID pseudonyms that works with legacy, untrusted readers, requires only one message from tag to reader, and is scalable: we require work only logarithmic in the number of tags for the Trusted Center to link readings.

Our scheme gives an exponential improvement over the previous pseudonym schemes of Ohkubo, Suzuki, and Kinoshita, and of Avoine and Oeschlin, which achieve $O(N)$ and $O(N^{\frac{2}{3}})$, respectively. Our scheme further allows for *delegation*, which gives an RFID reader the power to disambiguate a limited number of pseudonyms without further help from the Trusted Center. Next we show that our approach require little storage on an RFID tag and a small amount of over the air communication; we give example parameters for a deployment of one million tags in which each tag need store only 192 bits, make 4 PRF evaluations, and send 112 bits each time it is read.

Keywords: RFID, privacy, pseudonym protocol, cryptography.

1 Introduction

Radio Frequency Identification (RFID) technology holds great promise, but it also raises significant privacy concerns. The term RFID represents a family of emerging technologies that enable object identification without physical or visual contact. The main idea is to give a unique identity to every object by embedding a tag. A tag is a small chip, with an antenna, that stores a unique ID and other information which can be sent to a reading device. The reading device uses a database to link the ID with the object information stored in other databases.

The major security issues of these systems can be divided into two classes: threats to the integrity of the system and threats to the privacy of the user. The first threat is due to the fact that a tag does not provide any method to prove the claimed identity. In today's RFID systems, a tag always replies with the same ID, so it is hard to distinguish between a real and a fake tag. The second threat is associated with the nature of the tag/reader interaction. Tags can be read remotely and invisibly by any reader. This leads to unwanted consequences, such as the surreptitious tracking of objects and people through time and space. For instance, any party could use the RFID tags to track people's movements without authorization, since the ability to recognize an RFID tag allows for tracking items, and by extension, the people associated with them.

The seminal work of Weis, Sarma, Engels and Rivest proposed "hash locks," a kind of mutual authentication, as an answer to this problem. Infineon, one of the first producers of enhanced security RFID chips, incorporates a mutual authentication functionality based on a proprietary encryption algorithm with 64 bit keys to secure access to memory sectors. This class of tags are used today in item-level tagging operations that raise privacy concerns, such as library books [10]. Unfortunately mutual authentication cannot be considered the final answer. Mutual authentication

is overkill for many RFID applications, because in most cases we simply want to know the tag's identity. Writing to the tag or authenticating other commands from the reader are not necessary. In many supply chain applications of RFID, a large number of items will pass a reader in a short amount of time. Even if performing mutual authentication with a single item is fast, authenticating a reader to dozens of tags at once may be challenging.

We propose a cryptographic scheme that protects privacy while retaining many of the legitimate benefits of current RFID technology. The main idea is to introduce an RFID *pseudonym scheme* and to use a *trusted center* as a mechanism to delegate access to tag identifiers. Each time the tag is read, it generates a new "pseudonym" and sends this pseudonym to the reader. This response is generated using a 'tree of secrets' stored on the tag. The tree structure allows reader devices to determine the tag's identity with logarithmic work. Legacy readers, however, can simply pass upwards a pseudonym which they do not understand; the mapping from pseudonym to identity can occur anywhere in the network. Therefore, we do not need to change legacy readers.

Our scheme achieves two properties that are new for RFID protocols, as *controlled delegation* and *ownership transfer*. Delegation is the ability to give a reader the limited-time ability to determine when it has seen a particular tag. We can use delegation to limit the exposure if an adversary breaks into the reader; instead of losing the secrets for all tags for all time, we lose only what was delegated to that particular reader. Delegation also gives us a way to tolerate poor quality network connections between reader and Trusted Center. We also show how delegation gives us a way for Alice and Bob, who both trust the same Trusted Center but do not trust each other, to securely transfer an RFID-tagged item from one to the other. After the transfer, Bob has assurance that Alice can no longer read the RFID tag on the item, even though she could before. Our methods for ownership transfer require minimal or no online interaction by the Trusted Center itself.

In short, our scheme leaves the existing infrastructure of readers unchanged, while limiting complexity on the tag to the use of symmetric key cryptography and an increasing counter. "Contactless smart cards" proposed for United States passports from 2005 will have chips holding at least $64KB$ of data, along with optional 3DES based mutual authentication. Tags from Infineon and TAGSYS provide evidence that symmetric key cryptography is possible on RFID tags. Given these existing tags, along with recent work by Feldhofer et al. on small implementations of AES, we believe that our assumptions may be reasonable even for cheap supply chain tags in the near future [3].

2 Towards a Secure RFID Tag Protocol

In this section we outline the key aspects of the protocol that contribute to providing a robust solution to RFID security and privacy protection.

Pseudonym Scheme We wish to allow authorized readers to identify the RFID tag, while preventing unauthorized readers from determining anything about the identity of tags they interact with. Our approach is to build a RFID pseudonym protocol [12]. In our scheme, the RFID tag replies with a unique pseudonym that changes each time it is queried. The pseudonym is generated based on some secret key that is stored on the tag and known to authorized readers, so that authorized readers can identify the tag. However, without that secret, the pseudonym provides no information about the tag's identity. In particular, pseudonyms are unlinkable, so that unauthorized readers will be unable tell if two pseudonyms came from the same tag. In this way, possession of the secret key controls the ability to link sightings of the same tag.

The tag-reader protocol is very simple: the reader interrogates the tag, and the tag responds with its current pseudonym. Our use of pseudonyms allows the scheme to be compatible with legacy readers, because the reader does not need to know anything about the way that pseudonyms are

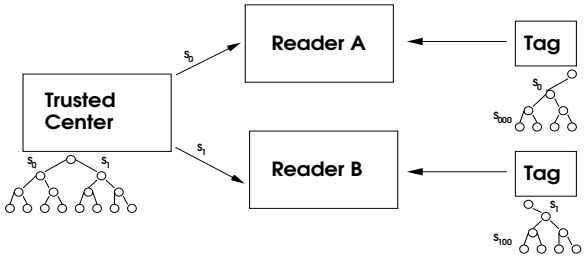


Fig. 1. The Trusted Center delegates access to two different Readers.

generated. Instead, the reader can provide the pseudonym it received to some other entity with the appropriate keys, and that other entity can recover the tag's identity from the pseudonym.

Privacy Control In many settings, we may wish to have a single party manage access to many tags. Thus, we assume the presence of a central trusted entity, which we call the Trusted Center(TC). When a tag is enrolled into the system, it is loaded with a secret key generated for it by the TC. The TC keeps a database listing for each tag with the secret key that was provided to that tag and any data that is to be associated with that tag (such as its identity, or access policy). Given any pseudonym from such a tag, the TC can determine the identity of the tag using the secret keys stored in its database.

Note that this also provides a simple way to delegate access to specific readers. In the future, a RFID infrastructure might consist of thousands or even millions of RFID readers deployed across the planet, and we need a way for legitimate readers to be allowed to read the tag. In a naive implementation, a TC for the tag would give a copy of the tag's secret key to each reader that is authorized to read the tag. However, this form of delegation is too coarse-grained, because the reader then permanently receives the ability to identify this tag for all time. We may not wish to place this much trust in every RFID reader that ever encounters the tag; the challenge is to provide controlled delegation, where a reader's ability to read a tag can be limited to a particular time period.

Controlled Delegation If readers are online, one simple approach is to have the reader simply act as a dumb relay, passing on the pseudonym from the tag to Trusted Center and letting the TC reply with the identity of the tag. In such a scheme, the TC can indeed authenticate the reader and check the privacy policy of the tag before responding to this reader's request. If a reader Alice wishes to determine a tag's ID, she must ask the TC. The TC can then decide whether Alice is allowed to see this information based on the tag privacy policy stored in the database. However, one limitation of this approach is that it requires a costly interaction between the reader and TC every time a tag is read. Because today's readers may repeatedly broadcast queries to all tags within range at a rate of 50 times per second or so, the burden on the TC and the database may be very high: if there are 10 tags within range, we require 500 round-trip interactions per second with the TC, multiplied times the number of readers. We instead focus on the problem of offline delegation.

In our scheme, the TC can compute a time-limited secret that only allows ability to disambiguate pseudonyms for a particular tag for a limited number of times. In particular, the TC computes a secret that allow to recognize the next q pseudonyms from this tag, where q is arbitrary and can be specified by the privacy policy. This secret can be communicated to Alice, the reader, through any channel, and thereafter the reader does not need to interact with the TC in any way.

In Figure 1 we show a diagram of how delegation works in our scheme with different RFID readers and the Trusted Center. Delegation is helpful for cases where readers have intermittent or

low-bandwidth connectivity. When a reader first sees a tag it is unable to recognize, the reader can send the pseudonym it received to the TC. If this reader is authorized this tag, the TC can return not only the tag's identity but also a secret that allows reading the tag for a limited time (say, for 1000 queries). Because tags typically repeatedly query their environment many times a second, this allows any arbitrary number of subsquent queries to be disambiguated locally at the reader, without requiring further interaction with the TC (until the query limit is exceeded). Thus, our scheme can still be used with online readers, and the ability to exploit the locality in tag sightings can be used to greatly improve the performance.

Ownership Transfer A related problem to delegation is that of *ownership transfer*, when Alice gives an RFID-tagged item to Bob. After the transfer of ownership, Bob should be able to read the item but Alice should not. Pseudonyms allow us to cleanly deal with ownership transfer from Alice to Bob. If Alice has not been delegated the ability to disambiguate pseudonyms, no further work is needed: the TC simply denies Alice's requests to disambiguate pseudonyms after Bob registers his ownership of the item. If Alice has been delegated linking ability, we have two methods for ensuring Alice can no longer link a tag after it is passed to Bob. First, a method we call *soft killing,* and second a method for securely incrementing a tag's leaf counter. We describe both methods in more detail in Section 6. Previous work on ownership transfer focused on a "recoding" technique with writeable RFID tags, in which a tag is overwritten with a new identifier that does not change between recodings. Therefore the RFID tag is still vulnerable to tracking and hotlisting until it is recoded [9]. Recoding also introduces the problem of managing secure access to the recoding operation in order to prevent other parties rewriting the tag.

Scalable Lookup A major technical challenge in the design of such systems is how to make them scalable to a large number of tags. Consider a TC with a database of N tags that receives a pseudonym to be disambiguated. Naively, one might check, for each of the N tags known to the TC, whether this pseudonym could have been generated by that tag. This results in O(N) work if the number of potential pseudonyms for each tag is limited. Ohkubo et al. introduce a pseudonym scheme for RFID that works in this manner [11].

To improve scalability Ohkubo et al. propose storing the expected next output of each RFID tag as an optimization, but this cannot be kept up to date unless the trusted authority is online for every tag read. Avoine and Oeschlin propose a time-space tradeoff technique that improves the complexity of the Ohkubo et al. protocol to $O(N^{\frac{2}{3}})$ time with a table of size $O(N^{\frac{2}{3}})$, but their protocol does not support delegation as ours does [2].

Instead, we design a scheme with logarithmic workload: in our protocol, the TC needs only do $O(\log N)$ work to disambiguate a pseudonym. The logarithmic complexity does not apply to readers who have been delegated access to a subset of tags: a reader can disambiguate each pseudonym in O(D) time, where D is the number of tags delegated to the reader. In practice we expect D will be small compared to the total number of tags; for example, D might be the number of tags in a single shipment of goods. Fortunately, since there is a great deal of locality in tag-reader interactions, most readers will only be associated with a small number of tags, so we expect this performance level to be more than adequate in practice. In Figure 2 we show a comparison to previous RFID privacy schemes.

3 Protocol Overview

The main idea of our scheme is to store a "tree of secrets" on the RFID tag. Our solution requires a pseudo-random function and a non-volatile counter on the RFID tag. Given recent results on AES

Scheme	T_{Reader}	S_{Reader}	T_{TC}	S_{TC}	# Msg	Comm	Delegation?
OSK [11]	O(N)	O(N)	NA	NA	1	O(1)	No
AO [2]	$O(N^{\frac{2}{3}})$	$O(N^{\frac{2}{3}})$	NA	NA	1	O(1)	No
MW [10]	$O(\log N)$	O(1)	NA	NA	$O(\log N)$	$O(\log N)$	No
Basic Scheme	O(D)	O(D)	$O(\log N)$	$O(2^{d_1})$	1	$O(\log N)$	Yes
Optimized Scheme	O(D)	O(D)	$O(\log N)$	O(1)	1	$O(\log N)$	Yes

Fig. 2. Comparison to previous RFID privacy schemes. Here T_{TC} and S_{TC} stand for the time and storage requirements of the Trusted Center, with the Reader requirements marked similarly. N is the total number of tags in the system, d_1 is the depth of the Trusted Center's tree, and D is the number of tags delegated to a particular reader. In practice, we expect $D \ll N$. The Optimized Scheme uses a PRF to generate the TC's tree of secrets and truncates the tag outputs, as described in Section 7.

implementation for RFID by Feldhofer et al., this appears reasonable for a large class of tags [3]. Each tag keeps a counter, which is incremented on each read. The counter stores the index of the next leaf of the tree to use. The path from root to leaf, combined with a random nonce, determines the tag's response to an RFID reader. After each response, the tag "updates" itself and its key material to the next leaf in the tree. A part of the tree from the root to a given depth (d_1) is shared between the Trusted Center and the tag alone. This shared key material allows the TC to determine the tag's identity with logarithmic complexity. Molnar and Wagner [10] showed how to use a "tree scheme" to reduce reader work in private mutual authentication from linear to logarithmic in the number of tags. Unfortunately, the scheme requires at least 3 and possibly as many as $O(\log N)$ rounds of communication between tag and reader, while we achieve one message from tag to reader. Further, their work does not support delegation, nor does it work with legacy readers. Our work uses a similar tree construction to achieve logarithmic work, but applies the scheme to a different problem, that of RFID pseudonyms.

Secrets from the tree below depth (d_1) may be given by the TC to an RFID reader. Because the tag evolves its key with each step, this delegated key material will "expire" after a certain number of tag reads. For simplicity, we will describe our protocol as if a random number generator exists on the RFID tag. In some RFID technologies, this may not be realistic; therefore we show in Section 7 how to replace this with an increasing counter. We will also limit the description to a binary tree of secrets, but in practice we will want to pick a tree with a high branching factor to make a tradeoff between reader work and tag communication. Again, we treat this in more depth in Section 7.

4 Notations and Background

- A pseudo-random functions (PRF) uses key k from a key space K on input M of length n-bits and output n-bits. $F : K \times \{0,1\}^n \to \{0,1\}^n$. We write $F_k(M)$.
- A pseudo-random generator (PRG) on input M of length k-bits is defined as: $G : \{0,1\}^k \to \{0,1\}^k \times \{0,1\}^k$. We write $G_{\{0,1\}}(M)$. By $G_0(M)$ we denote the first k bits output G on input M. By $G_1(M)$ we denote the next k bits output G on input M.
- Let $\{0,1\}^{\leq n}$ denote the set of bitstrings of length at most n. If $s \in \{0,1\}^*$ is a bitstring, let $s_{1..i}$ denote the first i bits of s, and let $\text{len}(s)$ denote the length of s (in bits).
- We also view $s_1 2^{n-1} + \cdots + s_{n-1}2 + s_n$. Each bitstring $s \in \{0,1\}^{\leq d}$ identifies a node in the tree; $s = 0$ and $s = 1$ are its left and right children, respectively.
- If $f : S' \to T$ is a function and $S \subseteq S'$, let $f|_S : S \to T$ denote the function f restricted to S. When given a function $h : \{0,1\}^{\leq d_1} \to K$ defined on $\{0,1\}^{\leq d_1}$, we extend it to a function defined on all of $\{0,1\}^*$ as needed by defining $h(sb) = G_b(h(s))$ for every $s \in \{0,1\}^{>d_1}, b \in \{0,1\}$.

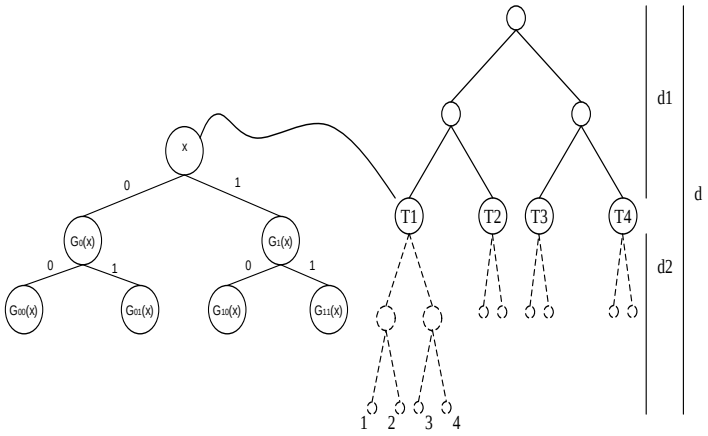


Fig. 3. An example tree of secrets for four tags in our RFID pseudonym scheme. The nodes drawn with solid lines correspond to secrets shared only between the tags T1,...,T4 and the Trusted Center. Each of these secrets is drawn uniformly at random and independently of each other. The dashed line nodes are secrets in *delegation trees*, where child nodes are derived by the GGM construction of applying a pseudo-random generator to the parent. On each read, a tag updates its state to use the next leaf in the delegation tree for its next pseudonym. To delegate limited-time access to a Tag, the Trusted Center can give out subtrees of the delegation tree; for example, the immediate parent of 1 and 2 allows learning T1's identity in time periods 1 and 2, but not in time periods 3 and 4.

We define a rooted full binary tree of depth d with k-bit string stored in the nodes and edges labeled 0 or 1. The tree stores random k-bit strings in all nodes $\leq d1$. In the nodes of succeding levels it stores k-bit string computed by applying G as follows. If a k-bit string is stored in an internal node v, then $G_0(v)$ is stored in $v's$ left son and $G_1(v)$ is stored in $v's$ right son. If $s \in \{0,1\}^d$ is a bitstring representing the position of a node v at the leaf. Let $s_{1..d-1}$ denote the position of $v's$ parent. The ancestor path from leaf to the root is defined by the nodes in position: $(s_{1..d-1}), (s_{1..d-2}), ..., (s_{1..1})$ and the function $h(s_{1..i})$ represents the k-bit string value of the node in position $s_{1..i}$.

5 Our Protocol

We assume a central Trusted Center role that can authenticate and authorize readers. Each Tag has a unique ID, which we would like to keep secret from a Reader unless the Reader's request meets a privacy policy associated with the Tag. The Reader interacts with a Tag and learns a one-time pseudonym p. Then the Reader asks the Trusted Center to identify the Tag. We first describe the basic "tree of secrets" used to generate the one-time pseudonym, including a description of the setup phase. We then describe the process through which a tag responds to the reader. Next we describe the mapping from pseudonym to tag identity, focusing on the problem of delegation. Later we will show how our protocol enables a secure transfer of ownership without need to rekey the tag.

Tree of Secrets To ensure our privacy goal the pseudonym needs to be updated whenever a tag response is generated. Our protocol is based around a tree of secrets of depth $d = d_1 + d_2$ as shown in Figures 3. Each node represents a cryptographic secret of length k-bit.

The first d_1 levels contain node secrets that are chosen independently of each other. The Trusted Center maintains the tree and generates these secrets at system initialization time using the algorithm TC.GenTC. The TC associates each tag with one node at depth d_1 and the following

Tag State:
c, a counter in $\{0,1\}^d$. Initialized to 0.
h, where $h = H|_S$ for some set $S \subseteq \{0,1\}^{\leq d_1}$.

Algorithm Tag.Respond():
1. Pick $r \in_R \{0,1\}^k$ uniformly at random.
2. Set $p := (F_{h(c_{1..1})}(r), F_{h(c_{1..2})}(r), \ldots, F_{h(c_{1..d})}(r))$.
3. Set $c := c + 1$.
4. Return (r, p).

Fig. 4. Algorithms and state for the RFID tag.

property will always hold: each tag knows all the keys from its node at depth d_1 up to the root node, but not other nodes in the tree. Secrets above d_1 in the tree are shared only between a Tag and the Trusted Center; a Reader will not have access to these secrets. We model the secret generation as a random function H kept by the TC and generated during TC.GenTC. All provisioning is done by the TC, which also ensures no tags are given the same secrets at level d_1. This algorithm is shown in Figure 5 (see TC.EnrollTag). The TC at enrollment time also records each tag's real identity ID, which may be an arbitrary string.

The next d_2 levels of the tree contain node secrets that are derived using a GGM tree construction [4]: each node is labelled with a secret, and the secrets for its children are derived by applying a PRG. Knowing a secret in the tree allows computation of the secrets for every descendent, i.e. the subtree rooted at that node, but nothing else. As shown in Figure 3, if we denote a secret x stored in a node v at depth d_1 then $G_0(v)$ is stored in $v's$ left son and $G_1(v)$ is stored in $v's$ right son. Let $s = s_1 2^{n-1} + \cdots + s_{d-1}2 + s_d$ be a binary string. The value of a node at depth d is $G_{s_d}(G_{s_{d-1}}(\ldots(G_{s_{d1}}(x))))$. These secrets are shared between a Tag and the TC and can be shared with a reader during the delegation process.

Tag Response Having access to subtrees of secrets is important for a Reader, because these subtrees allow the reader to map the Tag's pseudonym (r, p) to an ID without needing the TC. Each Tag T keeps a counter c. A Tag responds to a query from the reader by generating a random number r and sending a pseudonym

$$(r, p) = (r, (p_1, \ldots, p_d)) = (r, F_{h(c_{1..1})}(r), F_{h(c_{1..2})}(r), ..., F_{h(c_{1..d})}(r))$$

where the $h(c_{1..i})$ values represent secrets along the path in the tree of secrets from the root to the Tag's current leaf $T.c$. The Tag then increments the counter c. In practice, the counter value might be 64 bits.

Pseudocode for computing the response is shown in Tag.Respond. Each leaf value c corresponds to a new pseudonym of the tag. Below we discuss how the TC and the Reader can use their trees of secrets to map the pseudonym (r, p) to the Tag's ID. Note that because the counter c increments, the Tag will use a different path of secrets, and therefore a different pseudonym, for every Reader response: this is what ensures that the Reader's subtree of secrets will "expire" after a certain number of tag reads. The complexity of Tag.Respond depends on the overall depth of the tree, however, not directly on the size of the counter. By varying the branching factor and depth of the tree, we can trade off between the complexity of Tag.Respond and the complexity for the reader; we return to this in more depth in Section 7.

Mapping and Delegation To map a pseudonym p to the Tag's identity, the TC starts at the root of the tree of secrets. Then the TC performs a depth-first search over the tree, looking for the

TC State:
$H : \{0,1\}^{\leq d_1} \to K$, a function.

Algorithm TC.GenTC():
1. Let $H : \{0,1\}^{\leq d_1} \to K$ be a random function, i.e., pick $H(s) \in_R K$ uniformly at random for each bitstring s of length at most d_1.

Algorithm TC.EnrollTag(ID):
1. Find the smallest integer $t \in \{0,1\}^{d_1}$ that hasn't been assigned to any other tag. Assign t to this tag.
2. Set $S := \{t_{1..j} : 0 \leq j \leq d_1\}$.
3. Return $(t\, 0^{d_2}, H|_S)$ as the state for this tag.

Algorithm TC.Delegate(L, R):
1. Set $S := \{x_{1..d_1} : L \leq x \leq R\}$. Return $H|_S$.

Algorithm TC.IdentifyTag(r, p):
1. Return DFS($r, p, 1, \epsilon$), where ϵ denotes the empty bitstring.

Algorithm DFS($r, p = (p_1, .., p_d), i, s$):
1. Set $ids := \emptyset$.
2. If $F_{H(s\,0)}(r) = p_i$ then set $ids := ids \cup \text{DFS}(r, p, i+1, s\,0)$.
3. If $F_{H(s\,1)}(r) = p_i$ then set $ids := ids \cup \text{DFS}(r, p, i+1, s\,1)$.
4. Return ids.

Fig. 5. Algorithms and state for the Trusted Center.

Reader State:
$h : S \to K$, for some $S \subseteq \{0,1\}^{\geq d_1}$, with S initialized to $\emptyset$.

Algorithm Reader.IdentifyTag($r, p = (p_1, .., p_d)$):
1. For each $s \in S$, do:
2. If $F_{h(s)}(r) = p_{\text{len}(s)}$, then return s.
3. return $\perp$.

Fig. 6. Algorithms and state for the Reader.

path in the tree that matches the response p. At each node s, the TC can check whether the left child $s\,0$ or the right child $s\,1$ matches entry p_i in the response by checking whether $F_{s0}(r) = p_i$ or $F_{s1}(r) = p_i$, respectively. Pseudocode is shown in Figure 5 (see TC.IdentifyTag). Then the TC can map from the identity of the tag's current node to the tag's real identity ID. Based on ID, the identity of the Reader, and a privacy policy, the TC can then decide whether to reveal ID to the reader. This provides a mechanism for enforcing a privacy policy regarding which readers are allowed to learn which Tag IDs.

With this approach, the TC must be online for every tag read, which may incur too much overhead for the TC. Our protocol also allows for "offline delegation" the TC to delegate access to a certain interval of pseudonyms to the Reader. This allows the Reader to perform the mapping itself from a pseudonym (r, p) to the Tag's identity ID, but only if the Tag's counter value is in a prescribed interval $[L, R]$ (for some $1 \leq L \leq R \leq 2^d$).

Recall that each leaf of the tree corresponds to a different pseudonym for a tag. To delegate access to leaves in an interval $[L, R]$, the TC first determines the set S of all x_i such that i is of length d_1 and x_i falls in the interval $[L, R]$. The TC then sends $H|_S$ to the Reader along with the Tag's identity. Pseudocode is shown in TC.Delegate. In terms of our tree, the set S corresponds

to the minimal set of nodes that cover the interval $[L, R]$. Now, when the Reader sees the Tag's pseudonym (r, p), the Reader no longer needs to communicate with the TC. Instead, the Reader computes $F_{h(s)}(r)$ for all $s \in S$, which it can do because it has access to $H|_S$. If the Reader finds a match between the tag response and an s value, then it has learned the Tag's identity. Pseudocode for the Reader's computation is shown in Figure 6 (see Reader.IdentifyTag). After the Tag updates itself past the leaf R, however, the Reader can no longer map the Tag's pseudonym (r, p) back to the Tag's identifier ID. This is because the counter Tag.c will have updated past the subtree of secrets known to the Reader. At that point, the Reader must re-apply to the TC for more access.

During the depth-first search, the TC determines which node at level d_1 is currently in use by the Tag. This requires $2d_1$ evaluations of a PRF. Because each tag has at least one node at level d_1 of the tree and none of these values are shared between tags, this requires only $O(\log N)$ evaluations of the PRF. If the TC further wishes to learn the exact counter value used by the Tag, this requires another $2d_2$ evaluations of a PRF.

The Reader, by contrast, must check every value in its delegated subset S to see if it finds a match with an entry of the Tag's response. This takes time $O(D)$, where $D = |S|$. In the case of large subsets S, the Reader could apply the precomputation technique of Avoine and Oeschlin to achieve time $O(D^{\frac{2}{3}})$ with a table of size $O(D^{\frac{2}{3}})$ [2].

6 Ownership Transfer

Ownership transfer in RFID is the following problem: Alice gives an RFID tag to Bob. How do we prevent Alice from later reading the RFID tag? This problem is crucial for limiting the trust required in readers which may need to read tags at some point in the tag's lifetime.

In the case that Alice has not been delegated access to the RFID tag, ownership transfer in our model is simple. The Trusted Center is notified of the transfer and updates a privacy policy associated with the tag. Afterwards, Alice requests access to the tag's ID. The TC then checks the privacy policy, sees Alice no longer owns the item, and denies access. In case Alice has already been delegated access to the tag, we introduce two methods for ownership transfer.

Soft Killing. Bob queries the TC and learns how many leaves were delegated to Alice. Suppose this number is k. Bob then reads the tag $k + 1$ times. The tag will then have updated past Alice's access, so she will no longer be able to disambiguate the tag's pseudonyms. Notice that even if Bob knows how many leaves were delegated to Alice, he still cannot distinguish a tag delegated to Alice from any other tag without Alice's help; this is because the tag will emit a new, pseudorandom, pseudonym on each read. Therefore knowing the number of leaves delegated to Alice does not hurt the privacy of our protocol.

The benefit of soft killing is that it does not require shared secrets between the tag and reader. The downside is that soft killing requires many tag reads. Soft killing also opens up the possibility for a denial of service attack if an adversary reads the tag many times; Alice can recover from this by simply asking the Trusted Center to delegate more access.

Increasing The Tag Counter. We allow Bob to increase the counter on a tag from c to c'. Bob does so by sending the Tag a random seed r, after which Bob and the Tag can perform mutual authentication and establish a secure channel with the shared secret $F_{h(c)}(r)$. Bob then sends c' to the tag. We require that $c' > c$, so Bob can only increase the tag's counter, not decrease it. Alternatively, Bob can send the Tag a similar message identifying a subtree; the tag then updates itself to the least leaf in that subtree. By doing so, Bob can "leapfrog" the tag over Alice's delegated

leaves and be sure that Alice can no longer read the tag. Increasing the counter requires only one read, but also requires that Bob share a secret with the tag. Notice that the Trusted Center need not be involved at all in the transaction in this case.

7 Security Analysis and Optimizations

Threat Model. We now outline the security goals and threat model for our scheme. First, our protocol provides *privacy* for RFID tag readings: without specific permission by the Trusted Party, a reader cannot determine the tag identity from the pseudonym or otherwise link different readings of the same tag. Privacy should hold even when the adversary is allowed to ask for delegated access to tags of its choice. In particular, an adversary cannot map a tag's pseudonym to the tag's ID unless it has been specifically delegated access to the tree leaf currently used by that tag. Second, we provide *replay-only security against impersonation attack*. In an impersonation attack, an adversary wishes to pretend it is a legitimate RFID tag without knowing that tag's secrets. Because a pseudonym protocol uses only one message from tag to reader, it necessarily falls victim to a replay attack in which an adversary records a tag's pseudonym and replays it later to an RFID reader. We want a protocol where replay is the "worst" an adversary can do: without the secret keys of a tag, an adversary cannot generate valid tag pseudonyms it has not yet seen. We believe this limited replay-only security is tolerable, as duplicate readings of the same pseudonym can be detected and handled by a back-end database correlating RFID information.

We say an *adaptive radio-only* adversary is allowed to query tags of its choice in the order of its choice. We also assume the adversary can use the legitimate reader as an oracle to learn the "true" identity of any given pseudonym. In Appendix A we formalize this adversary's capabilities with a left-or-right definition of privacy. Even stronger than this is a *tag-breaking* adversary, which can compromise tags of its choice. We show in Appendix A that our protocol is private against an adaptive radio-only adversary, even if the adversary is allowed to ask for delegated leaves. Our protocol further achieves replay-only security against impersonation attack even against a tag-breaking adversary.

In contrast, we lose some privacy in the case that an attacker can compromise a tag. This is because two tags may share secrets if they share the same path in the tree. The likelihood that two randomly chosen tags share a secret is controlled by the branching factor of our tree of secrets. At one extreme, a tree with one level and a branching factor of N gives each tag a different secret. At the other, a binary tree of depth d means that two randomly chosen tags have a $(\frac{1}{2})^k$ chance of sharing k secrets. Each deployment can pick the branching factor that makes an acceptable tradeoff between privacy loss under tag compromise and reader work; our scheme still yields benefits from delegation even at high branching factors.

From PRFs to Hash Functions. Throughout we have assumed the use of Pseudo-Random Functions for generating tag responses. Because of the structure of our protocol, however, a *Hash function* would suffice. It remains an open question whether Hash are more efficient to construct than PRFs in practice on RFID devices and there may be not practical differences in the context of low-cost RFID tags [13].

Truncating PRF Values. Instead of sending full PRF values in a tag response, we could send truncated versions. This approach reduces communication overhead at the cost of causing potential misidentification. Regarding the size of the communication, if the output of the PRF used to produce the elements of the tag output is k' bits, then the output message length will be size$(r) + k'd$ bits.

Number of Tags	Tag Storage	Communication	Tag Compute	Reader Compute
2^{20}	192	112	5	$5 \cdot 2^{10}$
2^{30}	256	116	6	$6 \cdot 2^{10}$
2^{40}	320	120	7	$7 \cdot 2^{10}$

Fig. 7. Concrete resource use of our scheme for some example parameters. We use a branching factor of 2^{10} in all cases, use a 64-bit r value with truncation, and we assume tags will be read at most 2^{20} times. Tag and reader computation are both measured in expected number of PRF evaluations.

Instead of sending full PRF outputs, we could send truncated versions. This approach raises the costs of identifying the tag only slightly since we might navigate more than $2d$ branches.

We define a truncation function $R_a : X \to X \bmod 2^a$ where $1 < a < k'$ is the length in bit of the truncated output. When truncation is applied a Tag responds to a query generated from a reader with $(r, p) := (r, R_a(F_{h(c_{1..1})}(r)), ..., R_a(F_{h(c_{1..d})}(r)))$.

While "Truncating PRF Values" improves the communication efficiency, it necessarily introduces false positives in the identification process. By using truncation a tag cannot be positively identified from a single node secret, but must be probabilistically identified using multiple nodes. In the following analysis, for simplicity, we consider a tree with branching factor 2.

- Probability of False Positive. False positives or misidentifications occur when a message identifies at least a leaf when and only when the function R_a is applied. An upper bound of the probability to get a false positive is given by $2^{-(a-1)(d)}$.
  The exact probability can be calculated by the recursion: $P_0 = 1$ and $P_{n+1} = 2pP_n - p^2P_n^2$, where $p = 1/2^a$. Again by recursion we can show that $0 \leq P_n \leq (2p)^n$. We can therefore limit the probability of false positive to ϵ by choosing a so that $(2p)^n \leq \epsilon$, or in other words, $a \geq 1 - \ln \epsilon/(n \ln 2)$.

- Search Complexity. There are two main usages for tag identification: to recognize valid messages from tags owned by the TC and rejecting messages from unknown tags. In terms of PRF computations a non-truncated scheme has a complexity of $2d$ in the first case and just over 2 in the second. A truncated scheme has the same order of complexity for values of $a > 4$ as we show below.
  We define D_n as the average number of tests to check that a message is not in a tree of depth n. Again D_n can be defined by recursion as: $D_1 = 2$, $D_{n+1} = 2 + 2pD_n(1 - P_n)/(1 - pP_n)$. By recursion we can show that $0 \leq D_n \leq 2/(1 - 2p)$.
  We then define V_n as the number of tests necessary to authenticate a valid message from the tag. $V_n = n + 1/2(D_1 + ... + D_{n-1})$. For $a \geq 2$, we can show that $V_n \leq (1 + d/2)n + 1$. So the number of operations to validate a message is of the order $(1 + d/2)n$ which converges towards $2n$ for high enough values of a. We conclude that four-bit truncation leads to a probability of false positive of 4.5×10^{-44} for $d = 48$ and search complexity of $2.13 \cdot d$. Overall, the number of verification operations does not increase while the communication overhead is greatly reduced by a factor of a/k'.

Branching Factor and Concrete Examples. As we have noted, in practice we would employ trees with branching factors much larger than 2. A larger branching factor reduces the depth of the tree, and therefore reduced tag storage and communication, at the cost of more computation for the Trusted Center and Reader. For example, consider an RFID system with 2^{20}, or about one million, tags each of which will be read at most 2^{20} times. We construct a five-layer tree of

secrets with branching factor $1024 = 2^{10}$ at all levels. Each tag stores three 64-bit secrets s_1, s_2, s_3, with the third secret being the root of a GGM tree that covers the final two tree levels. For each pseudonym, the tag runs a PRF twice to obtain s_4, s_5, the secrets completing the path to its current leaf. Total tag storage is $3 \cdot 64 = 192$ bits and total tag computation is 5 applications of the PRF. If we truncate the tag's responses to 4 bits except for the leaf value, then use a 64-bit r and 32-bit truncation of the leaf value, the tag's total communication is $64 + 16 + 32 = 112$ bits. The work for the reader, on the other hand, is only $5 \cdot 2^{10}$ applications of the PRF. We show concrete parameters for this and some other examples in Figure: 7.

8 Conclusions

Several large-scale deployments of RFID devices already exist, and more are on the way. Unless we address the privacy issues in RFID now, we will find ourselves with legacy RFID devices that do not support privacy at all. At the same time, mechanisms for RFID privacy must be as lightweight as possible, both for the tag and for the supporting infrastructure. We have shown an RFID pseudonym scheme that requires only a PRF and counter on tag, and achieves logarithmic work for the Trusted Center to identify a tag. We argued that these assumptions are reasonable for some current RFID deployments, such as passports and libraries, and showed that we can support large deployments with modest tag state and communication. Further, we can delegate the ability to identify RFID tags, a novel primitive for RFID that eases ownership transfer between mutually distrusting parties and allows limiting the trust placed in a single RFID reader. By using our protocol, we can enhance the privacy properties of a wide range of current and future deployments of RFID technology.

References

1. Gildas Avoine. RFID privacy: A multilayer problem. In *Financial Cryptography*, 2005.
2. Gildas Avoine and Philippe Oeschlin. A scalable protocol for RFID pseudonyms. In *IEEE PerSec*, 2004.
3. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In *CHES*, 2004.
4. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
5. ICAO. PKI for machine readable travel documents offering ICC read-only access, version 1.1, October 2004.
6. Ari Juels. Minimalist cryptography for RFID tags, 2003. `http://www.rsasecurity.com/rsalabs/staff/bios/ajuels/publications/minim%alist/index.html`.
7. Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard systems. Cryptology ePrint Archive, Report 2005/052, 2005. `http://eprint.iacr.org/`.
8. Elena Malykhina. RFID tests are positive for cvs and pharmaceuticals. Information Week, `http://informationweek.com/story/showArticle.jhtml?articleID=48800464`.
9. David Molnar, Ross Stapleton-Gray, and David Wagner. Killing, recoding, and beyond, 2005. Chapter in RFID Applications, Security, and Privacy, Simson Garfinkel and Beth Rosenberg eds., To appear 2005, Addison/Wesley.
10. David Molnar and David Wagner. Security and privacy in library RFID: Issues, practices, and architectures. In *ACM CCS*, 2004.
11. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Cryptographic approach to a privacy friendly tag. In *RFID Privacy Workshop, MIT*, 2003.
12. Michael Pastore. Electronic passports enter home stretch, October 2004. InsideID, `http://www.insideid.com/credentialing/article.php/3420781`.
13. Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Computing*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212, 2004.
14. Rick Whiting. Drug makers 'jumpstart' tagging of bottles, July 2004. Information Week, `http://www.informationweek.com/story/showArticle.jhtml?articleID=256002%13&tid=5978`.
15. Junko Yoshida. Tests reveal e-passport security flaw, August 2004. EE Times.
16. Kaan Yuksel. Universal hashing for ultra-low-power cryptographic hardware applications, 2004. Master's Thesis, WPI.

A Security Analysis

A.1 Definitions

Exp-NoDelegation():
1. Call IN.GenTC(1^k).
2. For $i := 1$ to N, do:
3. Set $T_i :=$ IN.EnrollTag().
4. Choose $b \in_R \{0,1\}$ uniformly at random.
5. Set $b' := A^{\text{Reader},\mathcal{O}_b(i,j)}$.

Exp-Delegation():
1. Call IN.GenTC(1^k).
2. For $i := 0$ to N, do:
3. Set $T_i :=$ IN.EnrollTag().
4. Choose $b \in_R \{0,1\}$ uniformly at random.
5. Set $b' := A^{\text{Reader},\text{IN.Delegate},\mathcal{O}_b(i,j)}$.

Fig. 8. Experiments for definition of private RFID pseudonyms, with and without delegation.

Experiment-ReplayOnly-TagCompromise():
1. Call IN.GenTC(1^k).
2. For $i := 0$ to N, do:
3. Set $T_i :=$ IN.EnrollTag().
4. Return $A^{\text{Reader},\text{IN.Delegate},\text{IN.IdentifyTag},\text{BreakTag},\mathcal{O}_b(i,j)}$.

Fig. 9. Experiment for definition of replay-only security of RFID pseudonyms under tag compromise attack.

We define privacy for our RFID pseudonym protocol by the experiments shown in Figure 8. In these experiments, we define a sequence of tag oracles $T_1, \ldots, T_N$. On a query, an oracle runs Tag.Respond , updates the tag's state, and returns the corresponding result. Our experiments use a "left-or-right" style of definition in which the adversary is given a special oracle $\mathcal{O}_b(i,j)$, where $b \in \{0,1\}$. If $b = 0$, the oracle $\mathcal{O}_b$ will behave like the tag oracle T_i on query (i,j), otherwise $\mathcal{O}_b$ will behave like the tag oracle T_j. The adversary, based on its interaction with $\mathcal{O}_b(i,j)$ and other tag oracles, must decide whether $b = 0$ or $b = 1$. The adversary may make as many queries as it wishes to the special oracle $\mathcal{O}_b$.

Notice that the adversary has access to an individual tag oracle T_i by querying the left-or-right oracle with (i,i). This access does not allow for trivial distinguishing because tag oracles are stateful and update their counters on each invocation. Further note that while we have provided the adversary with an oracle for the RFID reader, in this experiment interaction with the reader cannot help the adversary, as the reader produces no output.

When we do not allow delegation, we use Exp-NoDelegation and define the advantage of such an adversary as $Adv^{NoDelegation}A := |\Pr[b = b'] - \frac{1}{2}|$. We say that a pseudonym protocol is (t, q, ϵ)-private against no-delegation adversaries if the advantage of an adversary A that runs in time at most t and makes at most q queries is at most ϵ.

We can also define privacy against an adversary that asks the Trusted Center for delegated access to tags. We model this by augmenting the adversary with an IN.Delegate oracle that gives the adversary access to the Trusted Center's Delegate functionality, with a special restriction on the queries of the adversary. We will stipulate that the adversary may not query $\mathcal{O}(i,j)$ with a tag T_i such that the adversary previously queried the value $T_i.ctr$ of that reading to IN.Delegate but not $T_j.ctr$ or vice versa.

Similarly, if the adversary has queried $\mathcal{O}_b(i, j)$, we prohibit the adversary from calling IN.DELEGATE on a range of leaves used by i in its state at that query but not j or vice versa. We enact similar restrictions for the cases of i and j whose leaf values have both been queried to IN.DELEGATE. These restrictions rule out "trivial" tag distinguishing, in which A distinguishes a tag for which it has legitimate access from IN.DELEGATE from a tag for which it has no such access, or distinguishes two tags for which it has legitimate access.

We also consider security against *impersonation attacks*. In an impersonation attack, the adversary wishes to falsely convince a reader that a tag is present. Our protocol admits a replay attack, because an adversary can record a tag's response and replay it later to a reader. Replay, however, is the worst the adversary can do. Even if the adversary can choose tags, compromise them, and learn all secrets of the chosen tags, the adversary cannot create non-replayed pseudonyms for any of the remaining tags. We formalize this notion, which we call *replay-only security against impersonation under chosen tag compromise* by the experiment in Figure 9.

In the experiment, BREAKTAG is a special oracle that on query i returns the internal state for the tag oracle T_i, including the tag's secret key $T_i.TK$. We define the advantage of A as the probability that A queries IN.IDENTIFYTAG with a value v not previously returned by T_j that causes IN.IDENTIFYTAG to return j for some j not previously queried to BREAKTAG or IN.DELEGATE. We define (t, q, ϵ) replay-only security under chosen tag compromise to mean that an adversary running in time t and making more than q queries has advantage at most ϵ.

A.2 Proofs of Security

We now prove that our pseudonym scheme satisfies the definition of privacy in the previous section both with and without delegation. We also show our scheme provides replay-only impersonation security under tag compromise attack.

Theorem 1. *Suppose $\{F_s\}_{s\in K}$ is a $(t_{PRF}, q_{PRF}, \epsilon_{PRF})$-PRF and G is a $(t_{PRF}, \epsilon_{PRG})$-PRG. Then our pseudonym protocol for a tree of depth d and delegation tree of depth d_2 is (t', q', ϵ') private against a no-delegation adversary, with $t' = d(t_{PRF}) + d_2(t_{PRG})$, $q' = (q_{PRF}/d)$, and $\epsilon' = q'^2 d(\epsilon_{PRF} + 2\frac{q^2}{2^k} + q'd\epsilon_{PRG})$.*

Proof. The main idea of the proof is to first analyze the protocol as if all r values that appear in pseudonyms are distinct and all secrets in the tree are generated randomly.We then bound the probability that the pre-conditions fail to hold.

Lemma 1. *(Pseudonym Indistinguishability (PI) Lemma). Let a_i and b_i be the secrets for tag oracles T_a and T_b, and assume that for all i, we have a_i and b_i chosen uniformly from K. Then the response of T_a, $A := (r_1, F_{a_1}(r_1), \ldots, F_{a_d}(r_1))$ is $(t_{PRF}/d, 2d \cdot \epsilon_{PRF})$-indistinguishable from the response of T_b $B := (r_2, F_{b_1}(r_2), \ldots, F_{b_d}(r_2))$ over $r_1, r_2 \leftarrow_R \{0,1\}^k$, provided $r_1 \neq r_2$ and $a_i \neq b_j$ for all i and j.*

Proof. (Lemma) We build hybrids between A and B. At some hybrid ℓ we distinguish between $F_{a_\ell}(r_1)$ and $F_{b_\ell}(r_2)$. This contradicts the PRF Which-Key Lemma. The security loss is a factor of d, which when combined with the factor of 2 from the Which-Key Lemma gives us a total of $2d \cdot \epsilon_{PRF}$.

Lemma 2. *(Precondition Lemma). The probability that the preconditions of the PI Lemma are not met over q' queries is at most $2\frac{q'^2}{2^k} + q'd \cdot \epsilon_{PRG}$.*

Proof. (Lemma) Let BAD be the event that not all the preconditions of the PI Lemma are satisfied. We see that $\Pr[BAD] = \Pr[RCOLLIDE] + \Pr[KEYCOLLIDE] + \Pr[GGMFAIL]$. Here $RCOLLIDE$ is the event that some r appears twice as a response to an adversary query. By the Birthday Paradox, after q queries, $\Pr[RCOLLIDE] \approx \frac{q'^2}{2^k}$. By a similar argument, the event $KEYCOLLIDE$ that two keys in the tree of secrets are equal is at most $\frac{(q'd)^2}{2^k}$.

The event $GGMFAIL$ is the event that the adversary can distinguish the PRFs in a tag's response keyed with elements of a GGM tree g_i from PRFs keyed with truly random values t_i. We bound the probability of $GGMFAIL$ by building hybrids between $(r_1, F_{t_1}(r_1), \ldots, F_{t_d}(r_1))$ and $(r_2, F_{g_1}(r_2), \ldots, F_{g_d}(r_2))$. As we walk from hybrid to hybrid, there must be some ℓ such that A distinguishes $F_{t_\ell}(r_1)$ from $F_{g_\ell}(r_2)$. This contradicts the security of the PRG. We lose a factor of d for a total probability of $d\epsilon_{PRG}$ for each query, a total of $q'd\epsilon_{PRG}$.

Combining the PI Lemma and the Precondition Lemma for each of the adversary's q' queries, we see that the responses of the oracle $\mathcal{O}_1(i,j)$ and $\mathcal{O}_0(i,j)$ are $(t, q'd\epsilon_{PRF})$-indistinguishable from each other for all q'^2 pairs of i and j with probability at least $2\frac{q^2}{2^k} + qd\epsilon_{PRG}$. Therefore our protocol is $(t', q', q'^2 d\epsilon_{PRF} + 2\frac{q'^2}{2^k} + q'd\epsilon_{PRG})$-private. Because each query has d PRF invocations and d_2 PRG invocations, we have $q' = q_{PRF}/d$ and $t' = dt_{PRF} + d_2 t_{PRG}$

Theorem 2. *Suppose $F : K \times \{0,1\}^k \to \{0,1\}^{k'}$ is a $(t_{PRF}, q_{PRF}, \epsilon_{PRF})$-PRF and G is a (t_{PRG}, q_{PRG})-PRG. Then our protocol is (t', q', ϵ') private for a tree of secrets of depth d against an adversary with access to a delegation oracle, with $t' = d(t_{PRF} + t_{PRG})q', q' = q_{PRF}/d$, and $\epsilon' = q'^2 2\epsilon_{PRF} + q'd(2\epsilon_{PRF} + \epsilon_{PRG}) + 2\frac{q'^2}{2^k}$.*

Proof. The main idea of the proof is to show that the tree values obtained by the adversary via IN.DELEGATE do not give the adversary much advantage at distinguishing PRFs keyed with the remaining, unrevealed values g_i in the same subtree. We express this in the following lemma:

Lemma 3. *(GGM Resilience Lemma). Let $t_1, \ldots, t_{d_2}$ and g_1 be chosen uniformly from $\{0,1\}^k$. Let $g_1, \ldots, g_{d_2}$ be a path of secrets in a GGM tree with root g_1. Let $v_1, \ldots, v_\ell$ be a subset of secrets in the same GGM tree such that no g_j is a descendant of v_i for all i and j. Then for all v_i and g_i, $|\Pr[A^{F_{t_1},\ldots,F_{t_{d_2}}}(v_1, \ldots, v_\ell) = 1] - \Pr[A^{F_{g_1},\ldots,F_{g_{d_2}}}(v_1, \ldots, v_\ell)]| \leq \epsilon''$, where $\epsilon'' = d(2\epsilon_{PRF} + \epsilon_{PRG})$.*

Proof. (Lemma) We build a hybrid sequence of oracles between $F_{t_1}, \ldots, F_{t_{d_2}}$ and $F_{g_1}, \ldots, F_{g_{d_2}}$. Because A distinguishes the extreme hybrids, there must be some ℓ for which A distinguishes the oracle sequence $F_{t_1}, \ldots, F_{t_\ell}, F_{g_{\ell+1}}, \ldots, F_{g_{d_2}}$ from $F_{t_1}, \ldots, F_{g_\ell}, F_{g_{\ell+1}}, \ldots, F_{g_{d_2}}$. Therefore A distinguishes F_{t_ℓ} from F_{g_ℓ}.

Now there are three cases. In case 1), if $\ell = 1$, then by the PRF Which-Key Lemma we see that F_{t_1} and F_{g_1} are distinguishable with probability at most $2\epsilon_{PRF}$, as both t_1 and g_1 were chosen uniformly at random.

In case 2), there exists a v_i such that v_i is a descendant of g_ℓ. Let α be the bit-string which encodes the sequence of G_0 and G_1 applications which produce v_i when applied to g_ℓ. Then we can build a distinguisher A^2_{PRG} that contradicts the security of G. On on input m, A^2_{PRG} computes $G_\alpha(m)$ and runs $A^{F_{(m)}}(G_\alpha(m))$. We see that if G is an ϵ_{PRG}-secure PRG, then F_{t_ℓ} and F_{g_ℓ} are distinguishable with probability at most ϵ_{PRG}.

In case 3), no v_i is a descendant of g_ℓ. Let v_j be the value that minimizes the edge distance between g_ℓ and a, where a is the least common ancestor of v_j and g_ℓ. By the argument of case 2), a is distinguishable from a random value with probability at most ϵ_{PRG}. Therefore g_ℓ is indistinguishable from random, and so F_{t_ℓ} and F_{g_ℓ} are distinguishable with probability at most ϵ_{PRG}.

Putting it together, we obtain a bound of $d(\max(2\epsilon_{PF}, \epsilon_{PRG})) \leq d(2\epsilon_{PRF} + \epsilon_{PRG})$.

Now consider the adversary's queries to $\mathcal{O}_b(i, j)$. If the adversary has asked for delegated values in the same GGM tree as $T_i.ctr$ or $T_j.ctr$, we apply the GGM Resilience Lemma. Otherwise we apply the bound on $GGMFAIL$ from the Precondition Lemma. As before, we apply the Birthday Paradox to bound the probability that an r re-appears in any query of the adversary and the probability any keys are repeated. Finally we can apply the PI Lemma to see that the responses of $\mathcal{O}_0$ are indistinguishable from the responses of $\mathcal{O}_1$ for all q'^2 pairs of i and j. This yields the result.

Theorem 3. *Suppose $F : K \times \{0,1\}^k \to \{0,1\}^{k'}$ is a $(t_{PRF}, q_{PRF}, \epsilon_{PRF})$-PRF. Then our protocol is (t', q', ϵ') replay-only secure under chosen tag compromise, with $t' = dt_{PRF}q', q' = q_{PRF}/d$, and $\epsilon' = d\epsilon_{PRF} + \frac{(q'd)^2}{2^k}$.*

Proof. Assume for contradiction that there exists an adversary A that breaks replay-only security under chosen tag compromise. Let $S := \{s_1, \ldots, s_{break}\}$ be the secrets obtained by A from calls to the BreakTag oracle. Let $W := (r, x_1, \ldots, x_d)$ be A's successful query to the IdentifyTag oracle and j be the value returned by that oracle. By the definition of IN.EnrollTag , there exists some $s_j \notin S$ and some ℓ such that $F_{s_j}(r) = x_\ell$, assuming all keys in $IN.H$ are distinct, which occurs with probability approximately $(1 - \frac{(qd)^2}{2^k})$. By the construction of our experiment, r never appeared in a response from tag j and so $F_{s_j}(r)$ has not appeared in a response to an adversary query. Therefore x_ℓ predicts an output of F_{s_j}, contradicting the fact that F_{s_j} is a PRF.

Mutual Authentication Protocol for Low-cost RFID

Jeongkyu Yang[1], Jaemin Park[2], Hyunrok Lee[2], Kui Ren[3], and Kwangjo Kim[2]

[1] Korea Minting & Security Printing Corporation,
35 Gajeong-dong,Yuseong-gu Daejeon 305-713, Korea.
jkyang@komsco.com

[2] Information and Communication University (ICU),
103-6 Munji-Dong, Yuseong-Gu, Daejeon, 305-714, Korea.
{jaeminpark, tank, kkj}@icu.ac.kr

[3] Worcester Polytechnic Institute (WPI),
100 Institute Road, Worcester, MA 01609, U.S.A.
kren@ece.wpi.edu

Abstract— *Radio frequency identification (RFID) is the latest technology to play an important role for object identification as a ubiquitous infrastructure. However, current low-cost RFID tags are highly resource-constrained and cannot support its long-term security, so they have potential risks and may violate privacy for their bearers. To remove security vulnerabilities, we propose a robust mutual authentication protocol between a tag and a back-end server for low-cost RFID system that guarantees data privacy and location privacy of tag bearers. Our protocol firstly provides reader authentication and prevent active attacks based on the assumption that a reader is no more a trusted third party and the communication channel between the reader and the back-end server is insecure like wireless channel. Also, the proposed protocol exhibits forgery resistant against simple copy, or counterfeiting prevailing RFID tags. As tags only have hash function and exclusive-or operation, our proposed protocol is very feasible for low-cost RFID system compared to the previous works. The formal proof of correctness of the proposed authentication protocol is given based on GNY logic.*

Keywords: RFID, tag, reader, back-end server, authentication protocol

1 Introduction

Radio Frequency Identification (RFID) is currently considered as the next generation technology that is mainly used to identify massive objects and will be a substitution for an optical bar code system in the near future. The typical RFID system consists of Radio Frequency (RF) tags, or transponders, and RF tag readers, or transceivers [8, 12]. A back-end server is usually included in RFID system as an individual component [4, 11, 12, 14]. The micro-chip equipped on a tag has a unique identification information and is applicable for various fields such as animal tracking, supply chain management, inventory control, *etc.*

The existing RFID systems are vulnerable to many security risks and imply potential privacy problems, since the implementation of well-known cryptographic algorithms remains hard due to the restricted computational power and the memory size of a low-cost RFID tag [3, 4, 6, 11, 12, 14]. User privacy issues are considered as a big barrier for the proliferation of RFID system applications since the data of a tag can be transmitted by an illegal interrogation without its bearer's attention.

To remove security vulnerabilities, an authentication protocol for RFID systems can be considered as a security measure. As discussed in [1, 3, 11, 14], one of the important issues to provide the security services under RFID environment is to design an authentication protocol keeping the low computational power of RFID tags in mind. In this paper, we propose a robust mutual authentication protocol that fits the low-cost RFID system environment. Our protocol meets the privacy protection for tag bearers, which requires confidentiality, anonymity, and integrity in the cryptographic point of view. The proposed protocol is robust enough against the active attacks such as the man-in-the-middle attack, and the replay attack as well as the data loss [11, 12, 13]. Our protocol is based on mutual authentication between a tag and a back-end server, and provides authentication for the reader in a special case the reader is no more regarded as the trusted third party (TTP). We consider forgery resistance against the attacker who copies or counterfeits a prevailing RFID tag.

The remainder of the paper is organized as follows: In Section 2, we introduce RFID system primer and

its related works, and then propose new authentication scheme in Section 3. We discuss the security proof of our scheme and suggest its security and performance in Section 4 and Section 5, respectively.

Finally, we conclude this paper in section 6.

2 Related Works

A hash function is a powerful and yet computational efficient cryptographic tool. Based on the one-wayness of hash function together with authentication process for low-cost RFID system are currently considered as the proper solution in the aspect of security requirements and hardware implementation for low-cost RFID tags. According to [9], a hash function can be implemented with only about 1.7 K-gate.

Weis *et al.* [14] introduced two hash-based authentication schemes; hash-lock scheme and extended hash-lockscheme. Their schemes mutually authenticate a tag and a back-end server, and try to provide the user privacy protection features such as anonymity on a tag's data. However, their proposed protocols are neither private nor secure against eavesdroppers since the attacker can track $metaID$ and $(r, f_s(r) \oplus ID)$ and impersonate the tag to a legitimate reader. Extended hash-lock scheme also has an implementation issue like a random number generator into each tag.

Recently, Henrici and Müller [4] proposed a simple and efficient authentication protocol for low-cost RFID system. Their protocol is based on a hash function embedding in a tag and a random number generator on a back-end server to protect the user information privacy, the user location privacy, and the replay attack. Their scheme also provides a simple method for the data loss. However, this protocol cannot resist against the man-in-the-middle attack. The attacker can be located between a legitimate tag and a legitimate reader and obtain the information from the tag. Thus, the attacker easily can be authenticated by the legitimate reader before the next session.

In the previous schemes, a reader is generally regarded as a TTP without the loss of security. However, the wireless communication channel between a reader and a back-end server can be considered as the insecure channel. Thus, an adversary can impersonate as a legitimate reader. Previous schemes cannot prevent the man-in-the-middle attack when a reader is no more a TTP. Besides, previous results did not clearly denote the linkage between the authentication information and the tag, so forgery is easily enabled with the passive eavesdropping.

3 Our Proposed Protocol

3.1 Notations

We use the notations as summarized in Table 1 to describe the protocol throughout the paper. Like [4], we adopt the similar database structure and the same mechanism to prevent the data loss.

Table 1: Notations

$\mathcal{T}$	RF tag, or transponder.
$\mathcal{R}$	RF tag reader, or transceiver.
$\mathcal{B}$	Back-end server, it has a database.
D	A database of $\mathcal{B}$.
C	Chip serial number that is embedded into $\mathcal{T}$.
$E_k()$	Symmetric-key encryption function with the key, k.
$D_k()$	Symmetric-key decryption function with the key, k.
$h()$	One-way hash function.
$h_k()$	Keyed hash function with the secret key k.
ID	Temporary identification value of $\mathcal{T}$, it is used to make the shared secret k_2 randomized.
ID'	Temporary value to be used to make the shared secret k_1 randomized.
k	Secret key shared between $\mathcal{R}$ and $\mathcal{B}$.
k_1	Shared random secret between $\mathcal{T}$ and $\mathcal{B}$.
k_2	Shared random secret between $\mathcal{T}$ and $\mathcal{B}$.
RNG	Random Number Generator in $\mathcal{R}$.
r	Random number generated by RNG.
S	Keyed one-way hash value of $h_k(r)$.
$\oplus$	Exclusive-or (XOR) function.
$\stackrel{?}{=}$	Verification operator to check whether the left side is valid for the right side or not.
$\leftarrow$	Update operator from the right side to the left side.
T_1	A field for the shared random secret, k_1.
T_2	A field for the shared random secret, k_2.
AE	A field for the pointer linking a pair of records.
CN	A field for the chip serial number, C, of $\mathcal{T}$.
$DATA$	A field for all other application related data of $\mathcal{T}$.

3.2 Assumptions and Attacking Model

Our protocol works under the natural assumption that $\mathcal{T}$ has a hash function, XOR gate, and the capability to keep state during a single session. The widely acceptable low-cost RFID tags likely require the usage of passive tags [12, 14]. To design our proposed protocol, we assume the low-cost RFID tag is passive and has a re-writable memory like EEPROM with reasonable size like EPC Class 2 of EPC Global [13]. In Crypto 2004, Biham *et al.* [5, 15] showed that collision of SAH0, MD4, MD5, HAVAL-128, and RIPEMD in a special case is easily found. With this in mind, we expect that the cryptographic hash function used in our protocol has the desirable security like preimage resistance, second preimage resistance, and collision avoidance. In our protocol, we assume $\mathcal{T}$ has a hash function. In [9], a hash function unit with block size of 64-bit can be implemented with only about 1.7 K-gate, so it is also assumed that there will be the practical implementa-

tion of hash function for the low-cost RFID tag with the desirable security. Like [4, 11], we assume that $\mathcal{T}$ only has its authentication related information. A tag also has a memory for keeping values of ID, k_1, and k_2 to process mutual authentication. The simple structures for the database record and the tag memory are shown in Figure 1. Other required data of $\mathcal{T}$ for an application are stored in the database of $\mathcal{B}$.

In the previous schemes [4, 14], they assumed that $\mathcal{R}$ is a TTP and the communication channel between $\mathcal{R}$ and $\mathcal{B}$ is secure. However, we assume that $\mathcal{R}$ is not a TTP and the communication channel is insecure like the current wireless network. We also assume that k is the secret key for keyed hash function shared between $\mathcal{R}$ and $\mathcal{B}$, and $\mathcal{R}$ and $\mathcal{B}$ has enough capability to manage the symmetric-key cryptosystem and sufficient computational power for encryption and decryption.

To solve the security risks and privacy issues, the following attacking model must be assumed and prevented [4, 12, 13, 14]. However, in our protocol, we do not consider a physical attack like detaching RFID tag physically from a product because it is hard to carry out in public or on a wide scale without detection. We consider the following attacks:

- *Man-in-the-middle attack*: The attackers can impersonate as a legitimate reader and get the information from $\mathcal{T}$, so he can impersonate as the legitimate $\mathcal{T}$ responding to $\mathcal{R}$. Thus, the attacker easily can be authenticated by the legitimate $\mathcal{R}$ before the next session.

- *Replay attack*: The attackers can eavesdrop the response message from $\mathcal{T}$, and retransmit the message to the legitimate $\mathcal{R}$.

- *Forgery*: The simple copy for the information of $\mathcal{T}$ by eavesdropping is enabled by the adversary.

- *Data loss*: The protocol can be damaged from the denial-of-service(DoS) attack, power interruption, and hijacking.

3.3 Security Requirement

To protect the user privacy, we consider the following requirement in cryptographic point of view [13, 11].

- *Data Confidentiality*: The private information of $\mathcal{T}$ must be kept secure to guarantee user privacy. The information of $\mathcal{T}$ must be meaningless for its bearer even though it is eavesdropped by an unauthorized $\mathcal{R}$.

- *Tag Anonymity*: Although the data of $\mathcal{T}$ is encrypted, the unique identification information of $\mathcal{T}$ is exposed since the encrypted data is constant. An attacker can identify each $\mathcal{T}$ with its constant encrypted data. Therefore, it is important to make the information of $\mathcal{T}$ anonymous.

- *Data Integrity*: If the memory of $\mathcal{T}$ is rewritable, forgery and data modification will happen. Thus, the linkage between the authentication information and $\mathcal{T}$ itself must be given in order to prevent the simple copy for $\mathcal{T}$. On the other hand, the data loss will happen from the DoS attack, power interruption, message hijacking, *etc.* Thus, the authentication information between $\mathcal{T}$ and $\mathcal{B}$ must be delivered without any failure, and the data recovery must be provided.

Besides, we must consider and evaluate the following security feature in the design of RFID authentication protocol.

- *Mutual authentication and reader authentication*: In addition to access control, the mutual authentication between $\mathcal{T}$ and $\mathcal{B}$ must be provided as a measure of trust. By authenticating mutually, the replay attack and the man-in-the-middle attack to both $\mathcal{T}$ and $\mathcal{B}$ is prevented. $\mathcal{B}$ also must authenticate $\mathcal{R}$ to avoid the man-in-the-middle attack by an illegitimate $\mathcal{R}$ over the insecure channel.

3.4 Protocol Design

The overall protocol is shown in Figure 1. The detailed procedures for each step are described.

3.4.1 Initial Setup

1) Each $\mathcal{T}$ is given two fresh random secrets and a database, D, of $\mathcal{B}$ also stores them as the shared secret. The temporary used two shared secrets are k_1 and $k_2 \in_U \{0,1\}^l$. $\mathcal{T}$ has a hash function and a XOR function. $\mathcal{T}$ does not need to have the additional storage for its serial number, C, since C is unique and permanently embedded into each $\mathcal{T}$ [8]. The initial identification data, $h(k_1)$, k_1, and k_2 are initially stored into ID, k_1, and k_2 of each $\mathcal{T}$'s memory, respectively.

2) $\mathcal{R}$ has a RNG with a keyed hash function, generates a fresh random nonce, $r \in_U \{0,1\}^l$, and calculates $h_k(r)$ for every session. $\mathcal{R}$ and $\mathcal{B}$ manage the secret key k for keyed hash function. We simply denote $h_k(r)$ by S.

3) The database, D, of $\mathcal{B}$ manages a record pair for each tag consisting of $\langle T_1, T_2, AE, CN, DATA \rangle$ like [4]. AE is not set since no associated entry exists initially at this moment. CN, keeps

the unique chip serial number, C, for each $\mathcal{T}$. $\mathcal{B}$ has a hash function and a keyed hash function to verify $\mathcal{T}$ and $\mathcal{R}$, respectively. The pair of records point each other with the pointer field, AE.

3.4.2 Detailed Description

We describe the proposed protocol according to the sequence of message exchange and also discuss the security goals that are achieved during the execution of each protocol message.

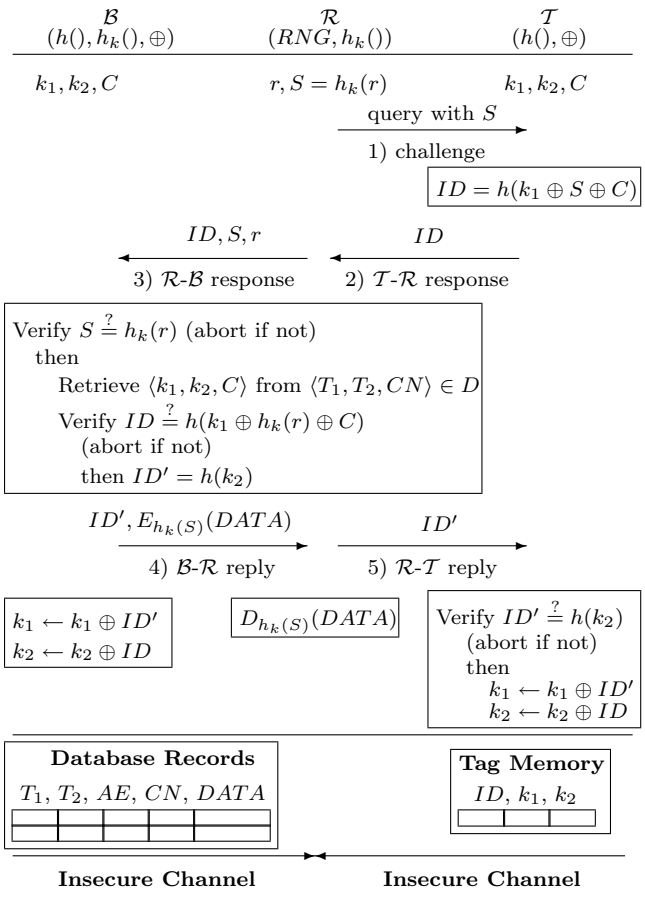


Figure 1: Proposed Authentication Protocol

Step 1 (Challenge) In this step, $\mathcal{R}$ usually applies a collision-avoidance protocol like the secure binary tree walking [2, 13] or the standard protocols of ISO 18000-3 MODE [7] to singularize $\mathcal{T}$ out of many. $\mathcal{R}$ generates a fresh random nonce, r, and randomizes it with the keyed one-way hash function, $S = h_k(r)$. $\mathcal{R}$ sends S to the queried $\mathcal{T}$. The key, k, is shared by $\mathcal{R}$ and $\mathcal{B}$, and S is used to authenticate the validity of $\mathcal{R}$. With S, the man-in-the-middle attack is prevented against an active attacker. It is also used to detect the illegitimate $\mathcal{R}$ by $\mathcal{B}$ after step 3.

Step 2 ($\mathcal{T}$-$\mathcal{R}$ Response) When queried, $\mathcal{T}$ sends ID to $\mathcal{R}$. ID is the output of one-way hash function and used as the identification information. ID has two purposes: One is to verify the legitimate $\mathcal{R}$ with S, and another is to prevent the forgery with C by the passive eavesdropping. ID is randomized with the shared secrets, k_1 and k_2 for every read attempt.

Step 3 ($\mathcal{R}$-$\mathcal{B}$ Response) $\mathcal{R}$ simply forwards ID to $\mathcal{B}$. At the same time, $\mathcal{R}$ also transmits S and r to prevent the man-in-the-middle attack and to detect the illegal $\mathcal{R}$. Within this step, $\mathcal{B}$ authenticates $\mathcal{R}$ and $\mathcal{T}$ consequently with ID.

At first, $\mathcal{B}$ verifies whether the forwarded r is valid or not by comparing S with $h_k(r)$. k is the shared secret key only between $\mathcal{R}$ and $\mathcal{B}$, so $\mathcal{B}$ can detect the illegal $\mathcal{R}$ and discards the forwarded message. So, the man-in-the middle attack by the illegitimate $\mathcal{R}$ and a passive eavesdropper can be prevented.

If $\mathcal{R}$ is valid, $\mathcal{B}$ retrieves the records corresponding to ID and get k_1, k_2, and C from T_1, T_2, and CN, respectively. Then, $\mathcal{B}$ authenticates $\mathcal{T}$ with ID. $\mathcal{B}$ calculates $h(k_1 \oplus h_k(r) \oplus C)$ and compares with ID.

Since $\mathcal{B}$ initially stores the chip serial number, C, $\mathcal{B}$ can evaluate the linkage between the forwarded authentication information ID and $\mathcal{T}$ itself in order to prevent forgery. Forgery can be detected and prevented by $\mathcal{B}$ at this moment.

At the same time, $\mathcal{B}$ can detect and prevent the man-in-the-middle attack since S is used as the factor of the man-in-the-middle attack detection. Similarly, the replay attack can be also detected and prevented simultaneously.

If $\mathcal{B}$ successfully finishes the authentication process, $\mathcal{B}$ generates ID' with its one of shared random secrets k_2. ID' will be used to make the shared secret, k_1, anonymous in the remaining steps.

The database of $\mathcal{B}$ generates a new record to consist of a pair of records and updates with the corresponding record. AE has the value to point the pair of records each other. When errors or the data loss in message for the current session happens the database of $\mathcal{B}$ can refer to the record of the previous session pointed by AE of the current session. Thus, the protocol is reliable for the data recovery against the data loss.

Step 4 ($\mathcal{B}$-$\mathcal{R}$ Reply) $\mathcal{B}$ encrypts the $DATA$ using $h_k(S)$, the randomly created shared secret key between $\mathcal{B}$

and $\mathcal{R}$. Then, $\mathcal{B}$ replies ID' and $E_{h_k}(S)(DATA)$. Then, $\mathcal{B}$ makes its shared two keys, k_1 and k_2, randomized simply by Xoring. The same process will be applied to the next step for making the corresponding shared secrets of $\mathcal{T}$ to be anonymous. After this step, the corresponding decryption process, $D_{h_k}(S)(DATA)$, is processed by $\mathcal{R}$ to get $DATA$. Thus, $DATA$ of $\mathcal{T}$ is securely obtained only by the legitimate $\mathcal{R}$ although the adversary eavesdrops the reply messages on the insecure channel.

Step 5 ($\mathcal{R}$-$\mathcal{T}$ Reply) Like step 3, $\mathcal{R}$ forwards ID' to the corresponding $\mathcal{T}$. Then, $\mathcal{T}$ processes the mutual authentication. $\mathcal{T}$ verifies the forwarded ID', calculates $h(k_2)$ and compares it with ID'. If matched, the mutual authentication is finally succeeded, and $\mathcal{T}$, as the last process, updates the shared secrets k_1 and k_2 simply exclusive-ors with ID and ID', respectively. Otherwise, $\mathcal{T}$ will not updates them in a case the replay attack to $\mathcal{T}$ occurs.

4 Correctness

In this section, we prove the correctness of the proposed protocol based on GNY logic [10]. Specifically, the correctness means that after the protocol execution, the communication parties, $\mathcal{T}$ and $\mathcal{B}$, believe that they are sharing two fresh secrets, k_1 and k_2, and ensure that this belief is confirmed by the other side. In addition to this, two entities, $\mathcal{R}$ and $\mathcal{B}$ should believe that they share the secret keys in a case the communication channel between the two entities is insecure.

In the forthcoming description, we use the conventional notations as follows: T, R, and B are entities, $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{B}$, respectively; K_1^i and K_2^i are shared secrets for i-th session between $\mathcal{T}$ and $\mathcal{B}$. $H()$ is a one-way hash function and $H_K()$ is a one-way keyed hash function; N_R is a random nonce generated by $\mathcal{R}$; K is a shared secret for $H_K()$ and K_{RB} is a shared secret for conventional encryption;m is data; other notations like T1, P1, F1, *etc.* follow the logical postulates of GNY logic [10].

4.1 Formalized Protocol

The conventional notations of the generic type of protocol are not convenient for manipulation in a logic. In this section, we, at first, simplify the protocol and describe it as a generic type. Then, we formalize the generic type of the protocol for verification goals as shown in Table 2.

Table 2: Generic Type of Protocol

Protocol Generic Type:
Msg. 1 $R \rightarrow T : H_K(N_R)$
Msg. 2 $T \rightarrow R : H(K_1^i \oplus H_K(N_R)), H(K_1^i \oplus H_K(N_R) \oplus C)$
Msg. 3 $R \rightarrow B :$
$H(K_1^i \oplus H_K(N_R)), H(K_1^i \oplus H_K(N_R) \oplus C), H_K(N_R), N_R$
Msg. 4 $B \rightarrow R : H(K_2^i), \{m\}_{K_{RB}}$
Msg. 5 $R \rightarrow T : H(K_2^i)$

Formalized Protocol:
Msg. 1 $T \triangleleft \star(H_K(N_R)) \rightsquigarrow R \models R \overset{K}{\longleftrightarrow} B$
Msg. 2 $R \triangleleft \star(H(K_1^i \oplus H_K(N_R))) \rightsquigarrow T \models \phi(H(X))$
Msg. 3 $B \triangleleft \star(H(K_1^i \oplus H_K(N_R))) \rightsquigarrow B \models R \overset{K}{\longleftrightarrow} B$
Msg. 4 $R \triangleleft \star(H(K_2^i), \{R \overset{K_{RB}}{\longleftrightarrow} B\}_{K_{RB}}) \rightsquigarrow B \models R \overset{K_{RB}}{\longleftrightarrow} B$
Msg. 5 $T \triangleleft \star(H(K_2^i)) \rightsquigarrow T \ni K_2^i$

Table 3: Goals of the Correctness Proof

1. $B \models T \mid\sim \sharp(H(K_1^i \oplus H_K(N_R)))$	2. $T \models B \mid\sim \sharp(H(K_2^i))$
3. $R \models R \overset{K}{\longleftrightarrow} B$	4. $B \models R \overset{K}{\longleftrightarrow} B$
5. $R \models R \overset{K_{RB}}{\longleftrightarrow} B$	6. $B \models R \overset{K_{RB}}{\longleftrightarrow} B$

4.2 Proof Goals and Assumptions

The proof goals of correctness are shown in Table 3. The first two goals, (1) and (2), are for the shared secrets. Those beliefs are to state that two entities shared secrets each other exchange fresh messages. The goals (3-6) are about shared keys between two entities. (3) and (4) are for a keyed hash function to guarantee the validity of reader, and (5) and (6) are for message encryption and decryption based on the symmetric key cryptosystem.

Table 4 shows the initial assumptions for our protocol. Assumptions (1-4) state that $\mathcal{T}$ has a hash function, $\mathcal{B}$ has a hash functions and a keyed hash function, $\mathcal{R}$ has a RNG and a keyed hash function, and the random nonce N_R of $\mathcal{R}$ and the keyed hash value $H_K(N_R)$ are fresh. The next six assumptions (3-8) are for two fresh shared secrets, K_1 and K_2, between $\mathcal{T}$ and $\mathcal{B}$. Assumptions (9) and (10) are based on the assumptions (1-8) and $\mathcal{R}$ must be a trusted entity in the viewpoint of $\mathcal{B}$ since the authentication messages from $\mathcal{T}$ are transmitted via $\mathcal{R}$. The abilities for verifying the hashed authentication message transmitted from $\mathcal{T}$ by $\mathcal{B}$ and from $\mathcal{B}$ by $\mathcal{T}$ respectively are based on assumptions (11-14). Assumptions (15-20) mean that both entities, $\mathcal{R}$ and $\mathcal{B}$, trust each other with those keys, K and K_{RB}.

4.3 Verification

In this section, the formal proof of our protocol is stated. The proof based on GNY logic is processed with the assumptions of Table 4. We strictly follow

Table 4: Initial Assumptions for Proof

1. $T \ni H(X)$	2. $R \ni H_K(X)$
3. $B \ni (H(X), H_K(X))$	4. $T \models \sharp(N_R)$
5. $T \ni (K_1^i, K_2^i)$	6. $B \ni (K_1^i, K_2^i)$
7. $T \models \sharp(K_1^i, K_2^i)$	8. $B \models \sharp(K_1^i, K_2^i)$
9. $T \models T \overset{K_1^i, K_2^i}{\leftrightharpoons} B$	10. $B \models T \overset{K_1^i, K_2^i}{\leftrightharpoons} B$
11. $T \models B \ni (K_1^i, K_2^i, C)$	12. $B \models T \ni (K_1^i, K_2^i, C)$
13. $T \models B \Rrightarrow T \overset{K_1^i}{\rightleftharpoons} B$	14. $T \models \sharp(H(K_2^i))$
15. $T \models R \Rrightarrow B \mid\sim H(K_2^i)$	16. $B \models T \Rrightarrow T \overset{K_2^i}{\rightleftharpoons} B$
17. $R \ni (K, K_{RB})$	18. $R \models R \xleftrightarrow{K, K_{RB}} B$
19. $B \ni (K, K_{RB})$	20. $B \models R \xleftrightarrow{K, K_{RB}} B$
21. $B \models R \Rrightarrow R \xleftrightarrow{K, K_{RB}} B$	22. $R \models B \Rrightarrow R \xleftrightarrow{K, K_{RB}} B$

the logical postulates of [10]. We refer n is the number of list and denote the list of proof goals of Table 3 by Gn, the list of assumptions of Table 4 by An, and the verification steps by (n). The extensions to messages are the precondition and are valid since they hold when messages are sent as are evident from the initial assumptions.

Message 1 $T \triangleleft \star(H_K(N_R)) \rightsquigarrow R \models R \xleftrightarrow{K} B$

1. $T \triangleleft H_K(N_R)$ /*By T1*/
2. $T \ni H_K(N_R)$ /*By P1*/
3. $T \models \sharp(H(N_R))$ /*By F1*/
4. $T \models \sharp(H_K(N_R))$ /*By (2),F10*/

Message 2 $R \triangleleft \star(H(K_1^i \oplus H_K(N_R))) \rightsquigarrow T \models \phi(H(X))$

5. $R \triangleleft H(K_1^i \oplus H_K(N_R))$ /*By T1*/
6. $R \ni H(K_1^i \oplus H_K(N_R))$ /*By P1*/
7. $R \models \sharp(H(K_1^i \oplus H_K(N_R)))$ /*By F10*/
8. $R \models \phi(H(K_1^i \oplus H_K(N_R)))$ /*R6*/
9. $R \models \sharp(H(K_1^i \oplus H_K(N_R)))$ /*For (7), by A18,(5),(6),(8),I1*/
10. $R \models R \xleftrightarrow{K} B$ /*By A20,A22,J1*/

Message 3 $B \triangleleft \star(H(K_1^i \oplus H_K(N_R))) \rightsquigarrow B \models R \xleftrightarrow{K} B$

11. $B \triangleleft H(K_1^i \oplus H_K(N_R))$ /*By T1*/
12. $B \ni H(K_1^i \oplus H_K(N_R))$ /*By P1*/
13. $B \models \sharp(H(K_1^i \oplus H_K(N_R)))$ /*By A3,A6,F10*/
14. $B \models \phi(H(K_1^i \oplus H_K(N_R)))$ /*For (12), by R6*/
15. $B \models R \mid\sim H(K_1^i \oplus H_K(N_R))$ /*For (13), by A3,A6,A20,(11),(13),(14),I1*/
16. $B \models R \xleftrightarrow{K} B$ /*For A18,A21, by J1*/
17. $B \models T \mid\sim H(K_1^i \oplus H_K(N_R))$ /*For (13), by A3,A6,A10,(11),(13),I3*/
18. $B \models T \mid\sim \sharp(H(K_1^i \oplus H_K(N_R)))$ /*For (17), by (13),F1*/

Message 4 $R \triangleleft \star(H(K_2^i), \{R \xleftrightarrow{K_{RB}} B\}_{K_{RB}}) \rightsquigarrow B \models R \xleftrightarrow{K_{RB}} B$

19. $R \triangleleft (H(K_2^i), \{R \xleftrightarrow{K_{RB}} B\}_{K_{RB}})$ /*By T1*/
20. $R \triangleleft H(K_2^i)$ /*By T2*/
21. $R \ni H(K_2^i)$ /*By P1*/
22. $R \models \sharp(H(K_2^i))$ /*By P1*/
23. $R \ni (H(K_2^i), \{R \xleftrightarrow{K_{RB}}\}_{K_{RB}})$ /*For (19), by P1*/
24. $R \models B \mid\sim (H(K_2^i), R \xleftrightarrow{K_{RB}} B)$ /*For (19), applying A18,I1*/
25. $R \models B \mid\sim R \xleftrightarrow{K_{RB}} B$ /* By I7*/
26. $R \models R \xleftrightarrow{K_{RB}} B$ /*By A20,J1*/

Message 5 $T \triangleleft \star(H(K_2^i)) \rightsquigarrow T \ni K_2^i$

27. $T \triangleleft H(K_2^i)$ /*By T1*/
28. $T \ni H(K_2^i)$ /*By P1*/
29. $T \models \sharp(H(K_2^i))$ /*By A7,F10*/
30. $T \models B \mid\sim H(K_2^i)$ /*By A5,A9,(27),I3*/
31. $T \models B \mid\sim \sharp(H(K_2^i))$ /*By (29),F1*/

As shown above, the proof goals of Table 3 are accomplished by verification steps (10) for G3, (16) for G4, (18) for G1, and (26) for G5, respectively. We omit the proof for G6 since, for the encrypted message with the key, K_{RB}, there is no further message exchange after this step. That is, the encrypted message of the entity, B, is replied to R and decrypted by R.

5 Evaluation

5.1 Security Analysis

We evaluate our protocol in the view point of the security requirement.

Our protocol guarantees the secure mutual authentication only with the hashed messages, $ID = h(k_1 \oplus S \oplus C)$, $ID' = h(k_2)$, and $S = h_k(r)$, and $\mathcal{T}$ does not store user privacy information. Thus, data confidentiality of tag owners is guaranteed and the user privacy on data is strongly protected. In every session, we use the fresh random nonce as the keys between entities. These keys are randomized and anonymous since they are updated for every read attempt. Thus, tag anonymity is guaranteed and the location privacy of a tag owner is not compromised, either. Based on the mutual authentication, our protocol guarantees the data integrity between $\mathcal{T}$ and $\mathcal{B}$. By using the pair of database records and managing AE as we described in the authentication step 3, our protocol provides the data recovery against the data loss during the authentication processes.

To give the forgery resistance feature, we exclusive-or the embedded chip serial number, C, of $\mathcal{T}$ to the authentication information, ID. C is initially embedded during the chip manufacturing. Whenever $\mathcal{T}$ generates ID, it refers to C, so we can come up with the linkage between ID and $\mathcal{T}$ itself. $\mathcal{B}$ keeps each tag's chip serial number initially and authenticates the ownership of the authentication information for $\mathcal{T}$.

Through the authentication step 1 to step 3, $\mathcal{R}$ sends S to $\mathcal{T}$ and S, r to $\mathcal{B}$ for preventing the man-in-the-middle attack. $\mathcal{B}$ can verify S with the calculation of the keyed hashed value of r transmitted from $\mathcal{R}$. Also, the man-in-the-middle attack by $\mathcal{R}$ as an illegitimate

Table 5: Comparison between Protocols

Protocol	HLS [14]	EHLS [14]	HBVI [4]	Our Scheme
User data confidentiality	×	△	△	◯
Tag anonymity	×	△	△	◯
Data integrity	△	△	◯	◯
Mutual authentication	△	△	△	◯
Reader authentication	×	×	×	◯
Man-in-the-middle attack prevention	△	△	×	◯
Replay attack prevention	△	△	◯	◯
Forgery Resistance	×	×	×	◯
Data Recovery	×	×	◯	◯

†† Notation
◯ satisfied △ partially satisfied
× not satisfied

Table 6: Computational Loads and Required Memory

Protocol	Entity	HLS [14]	EHLS [14]	HBVI [4]	Our Scheme
No. of Hash Operation	$\mathcal{T}$	1	2	3	2
	$\mathcal{B}$	¬	n	3	2n
No. of Keyed Hash Operation	$\mathcal{R}$	¬	¬	¬	1
	$\mathcal{B}$	¬	¬	¬	1
No. of RNG Operation	$\mathcal{T}$	¬	1	¬	¬
	$\mathcal{R}$	¬	¬	¬	1
	$\mathcal{B}$	¬	¬	1	¬
No. of Encryption	$\mathcal{B}$	¬	¬	¬	1
No. of Decryption	$\mathcal{R}$	¬	¬	¬	1
Number of Authentication Steps		6	5	5	5
Required Memory Size	$\mathcal{T}$	$1\frac{1}{2}L$	$1L$	$3L$	$2\frac{1}{2}L$
	$\mathcal{R}$	¬	¬	¬	$1\frac{1}{2}L$
	$\mathcal{B}$	$2\frac{1}{2}L$	$1\frac{1}{2}L$	$9L$	$8L$

†† Notation
¬ not required n number of tags
L size of required memory

reader is detected and prevented on the insecure channel between $\mathcal{R}$ and $\mathcal{B}$. The $DATA$ of the corresponding $\mathcal{T}$ is not compromised since it is encrypted by $\mathcal{B}$ and decrypted by $\mathcal{R}$ with the randomly generated secret key, $h_k(S)$, from S of $\mathcal{R}$. The key freshness is also guaranteed for each session. The replay attack for $\mathcal{T}$ and $\mathcal{B}$ is detected and prohibited through the step 3 for $\mathcal{B}$ and the step 5 for $\mathcal{T}$. Table 5 shows the comparison of the security requirements and the possible attacks.

5.2 Performance Analysis

We analyze the performance of the proposed scheme in forms of the following overheads: 1) computation, 2) storage, 3) communication, and 4) cost.

• **Computational Overhead.** $\mathcal{T}$ requires only a hash calculation and a XOR operation and needs three hash calculation. However, the cost of hash calculation at the server side is $2n$, where n is the number of tags. Compared to [4], the cost of our protocol has overheads for $\mathcal{B}$. Meanwhile, in [4], the anonymity of tag is guaranteed only after the authentication is successfully completed. Therefore, the location privacy of tag bearers is compromised until the next session is successfully started. To make the output of $\mathcal{T}$ anonymous for the current session, $\mathcal{B}$ should check for every records of D to authenticate each tag like EHLS [14]. However, note that the reduction of this cost should be needed for the admirable performance.

On the other side, our protocol seems to have encryption and decryption overheads for $\mathcal{R}$ and $\mathcal{B}$. However, those cryptographic tools are needed to secure $DATA$ on the insecure channel. We assume that $\mathcal{R}$ and $\mathcal{B}$ have enough computational power to process encryption and decryption based on the symmetric-key cryptosystem.

• **Storage Overhead.** To compare with the previous protocols, we assume the sizes of all components are L bits, and a RNG and a hash function are $h, h_k : \{0,1\}^* \rightarrow \{0,1\}^{\frac{1}{2}L}$ and $r \in_U \{0,1\}^L$, respectively. In our protocol, $\mathcal{T}$ only has a hash function and XOR function, and the size of the memory is $2\frac{1}{2}L$. Thus, the proposed protocol is light-weight and practical. We exclude the comparison for the application-specified data, $DATA$ since the size of $DATA$ depends on applied applications.

• **Communication Overhead.** The proposed protocol accomplishes mutual authentication between $\mathcal{T}$ and $\mathcal{B}$ requiring five rounds. As we denote in the previous section, some protocols [14, 11] requires three or six rounds. However, their protocol have synchronization problem on authentication data between $\mathcal{T}$ and $\mathcal{B}$. Five rounds is mostly acceptable for a minimum number of mutual authentication in RFID environment. Therefore, the proposed protocol is feasible in the sense of communication overheads.

• **Cost Overhead.** [11, 13] claimed that the number of gates available for security generally cannot exceed 2.5-5 K-gate. In our protocol, only one hash function unit and the storage for XOR operation are needed. If we assume the gates for XOR operation needs several tens of gates, the number of expected gates is less than 2 K-gate. Therefore, the proposed protocol is feasible and practical for low-cost RFID environment.

Table 6 shows the comparison of the computational loads and the required memory size for a single session with previous results [4, 14].

6 Concluding Remarks

In this paper, we proposed a robust RFID mutual authentication protocol for the low-cost RFID environ-

ment that is computationally light-weight and anonymously interact between entities. The proposed protocol basically fits the low-cost RFID system environment. The tag only has a hash function with the shared two fresh random secrets of small memory size. With this minimal cryptographic primitive, our protocol provides the mutual authentication between the tag and the back-end server and anonymously interacts. Our protocol is robust enough since it protects the replay attack and man-in-the-middle even when the reader is not a trusted third party and the communication channel is insecure. We add the linkage feature between the tag and its authentication data, so forgery is prohibited. All authentication messages are randomized and the tag only has its unique identification data, so the user data privacy and the location privacy is guaranteed. The formal proof of correctness for the proposed protocol was discussed based on GNY logic.

Acknowledgement

I am grateful to Prof. Adi Shamir of the Weizmann Institute in Israel for his precious comments and kind advice on our protocol during his visit to ICU during Asiacryt2004.

References

[1] Auto-ID Center, "860MHz-960MHz Class I Radio Frequency Identification Tag Radio Frequency & Logical communication Interface Specification Proposed Recommendation Version 1.0.0", Technical Report MIT-AUTOID-TR-007, Nov. 2002.

[2] A. Juels, R.L. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy", *Proc. of 10th ACM Conference on Computer and Communications Security*, pp.103-111, Oct. 2003.

[3] A. Juels and R. Pappu, "Squealing euros: Privacy protection in RFID-enabled banknotes", In Rebecca N. Wright, editor, *Financial Cryptography FC'03*, LNCS, vol.2742, pp.103-121, Le Gosier, Gaudeloupe, French West Indies, Jan. 2003.

[4] D. Henrici and P. Müller, "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers", *PerSec'04 at IEEE PerCom*, pp.149-153, Mar. 2004.

[5] E. Biham and R. Chen, "New results on SHA-0 and SHA-1", Presented at the rump session of *Crypto 2004*.

[6] G. Avoine, "Privacy issues in RFID banknotes protection schemes", *In Sixth Smart Card Research and Advanced Application IFIP Conference* - CARDIS, Toulouse, France, Aug. 2004.

[7] ISO/IEC JTC 1/SC 31/WG 4. "Information technology AIDC techniques - RFID for item management Air interface, Part3: Parameters for air interface communications at 13.56 MHz", Version N681R, Apr. 2004.

[8] K. Finkenzeller, "RFID Handbook Second Edition", Wiley & Sons, 2002.

[9] K. Yüksel, "Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications", Master's Thesis, Dept. of Electronical Engineering, WPI, 2004.

[10] L. Gong, R. Needham, and R. Yahalom, "Reasoning about Belief in Cryptographic Protocols", *1990 IEEE Computer Scociety Synopsis on Research in Security and Privacy*, pp.234-248, 1990.

[11] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to Privacy-Friendly Tags", *RFID Privacy Workshop 2003*, MIT, MA, USA, Nov. 2003.

[12] S. Sarma, S. Weis, and D. Engels, "RFID Systems and Security and Privacy Implication", Auto-ID Center, 2002.

[13] S. Weis, "Security and Privacy in Radio-Frequency Identification Devices", Master's thesis, MIT, 2003.

[14] S. Weis, S. Sarma, R. Rivest, and D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", *Proc. of the 1st Security in Pervasive Computing*, LNCS, vol.2802, pp.201-212, 2004.

[15] X. Wang, X. Lai, D. Feng, and H. Yu, "Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD", Presented at the rump session of *Crypto 2004*.

Symmetric Authentication for RFID Systems in Practice

Sandra Dominikus, Elisabeth Oswald, Martin Feldhofer
Institute for Applied Information Processing and Communications,
Graz University of Technology,
Inffeldgasse 16a, 8010 Graz, Austria
Sandra.Dominikus@iaik.tugraz.at

***Abstract*— Radio Frequency Identification (RFID) systems will soon become an important part of everyday life. Security mechanisms for RFID systems, such as authentication and encryption, are therefore of utmost importance.**

In this article we present a concept for strong symmetric-key authentication, which can be implemented with Class 2 RFID tags in practice. By using strong authentication threats such as forgery of tags, unwanted tracking of tags, and unauthorized memory access of tags can be defeated. We describe five authentication protocols, based on symmetric-key encryption, with different security features all conforming to state of the art standards such as ISO/IEC 18000-3. We calculate the timing of the proposed protocols with a sophisticated Java simulation tool. By using interleaved versions of the authentication protocols together with the AES module described in [3] we show that authentication is possible for Class 2 RFID systems in practice.

I. Introduction

Radio Frequency Identification (RFID) is an emerging technology. The main idea behind it is to give a digital identity to every object in a particular environment by attaching a so called RFID tag. An RFID tag is a small microchip, with an antenna, holding an unique ID and other information which can be sent over radio frequency. The information can be automatically read and registered by RFID readers. The data received by the RFID reader can be subsequently processed by a back-end database. A typical RFID system is shown in figure 1.

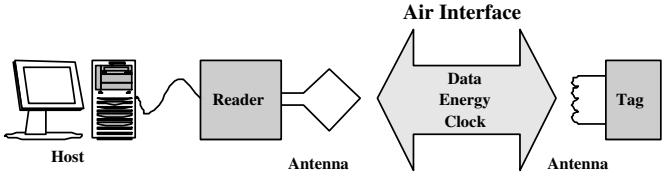


Fig. 1. Model of an RFID System.

The range of possible applications vary with the capability of the tag and are separated by different classes. Class 0 and Class 1 RFID tags are used as barcode replacement and are read-only or can be programmed only once in the field, respectively. Inventory maintenance which is used in the supply chain management can be automated using such tags. They are cheap (approximately 5 Cents) and can be used on item-level on nearly every product.

This paper is focusing on more advanced tags (Class 2) which also have a rewritable memory and additional hardware resources. Such tags cost about 50 Cents and the available silicon area is about 10,000 gates. The applications for more advanced RFID systems are manifold but especially high-value products like pharmaceutical and branded goods can be protected against security vulnerabilities.

Strong authentication mechanisms can solve uprising security problems in RFID systems and therefore give protected tags an added value. In this paper, we propose authentication protocols for RFID systems based on the ISO/IEC 9798-2 standard [4]. These protocols allow to protect high-value goods against adversary attackers. Additionally, we show that these protocols are feasible for nowadays restriction concerning data rates and compliance to existing standards as well as the requirements concerning chip area and power consumption.

This paper is structured as follows. In section II we give an overview over security issues. Section III deals with related articles in the field of RFID. The following section deals with authentication protocols in general, and in section V the proposed protocols are described in detail and compared.

II. Security Aspects in RFID Systems

Unfortunately, RFID systems are susceptible to remote attacks. The attacker can work at a certain distance to the reader and the tag and is not bound to normative limitations. The three main security threats are unwanted tracking, forgery, and unauthorized memory access. An unsecured tag gives its ID to every standard reader which sends a valid command. In that way, everybody possessing a reader can track a person wearing an RFID tag and collecting personal data about this individual. For example, it is not acceptable that medical or financial information leaks from an item a person is carrying.

Forgery of tags is also a problem when tags are used to proof the origin of a product. High-value branded goods like CDs, medicaments, or even immobilizer systems in cars can be protected using security-enhanced RFID tags. Recently this was a topic because a car immobilizer system using a proprietary algorithm was broken [1].

The unlimited access to tag's memory is a further point of attack when reading sensitive data from the memory. For example, Employers can read tags attached to medicaments of his employees or medical or financial data stored on the tag can

be read by an unauthorized party. Unauthorized write access can lead to forgery of data, which is especially a problem, when the data are sensitive.

III. Related Work

Most of the work done so far regarding RFID security deals with privacy protection and securing low-cost tags conforming the EPC standard [2]. In this paper, we will focus on Class 2 tags that are not intended to be used in low-cost applications. Nevertheless, we give an overview over the existing work. The first approach to avoid tracking was to introduce a ”kill” command, which deactivates the tag permanently. This measure was proposed to prevent tracking of consumers. The consumer can decide, whether the tag should be ”killed” at the cash desk or not. This is indeed an effective measure for privacy protection, but as the tag loses its RFID capability, the consumer cannot use the tag for RFID applications after the cash desk.

Another proposed measure was shielding of the tag, which is also very effective, but not always possible (e.g. when wearing a tagged jacket or the like). Juels et. al. introduced the so called blocker tag [9], which is able to simulate a whole range of IDs. In that way, the reader cannot determine the ID of the “real” tag. This method works with a particular anticollision algorithm and can also be used for malicious purposes (e.g. denial-of-service attack).

Other approaches employ hash functions to secure the RFID communication. One suggestion was a so called hash-lock scheme [13]. The tag only sends its ID to the reader if it knows a special key, which is unique for the tag. The back-end database has to hold the keys for all tags to find out the correct one for one particular tag. Furthermore, the ”real” ID and the key are transmitted, so tracking is possible. An extension was proposed, where only randomized data is sent, but this approach requires lots of calculations from the back-end database and is not suited for a large number of tags. A similar approach was suggested by NTT Labs [11], which also uses hash functions and a large back-end database.

In 2003, RSA Labs. published a method which was called re-encryption and used for securing banknotes [8]. Here, the authenticity of banknotes can be proven by a signature which is printed on the banknote and has to be read out optically. With the optical information, one gets writing access to the tag’s memory. A random number is chosen and this number is written into the memory, then it is encrypted with the public key of a law enforcement agency and also written on the memory. The law enforcement agency can always verify the presence of a particular banknote, whereas an attacker, after a re-encryption, cannot detect the same banknote. In that way unauthorized tracking is prevented.

Other approaches are silent tree walking (used for a particular anticollision algorithm) [13], one-time pad schemes (where the tag and the reader have to exchange lists of one-time pads) [7], global and private IDs, or lightweight authentication protocols [12].

Cryptographic methods are at the moment mainly used in car immobilizers and access control devices. Most these methods used nowadays are proprietary, because they are less costly than standard algorithms. It is a popular belief that standard cryptographic algorithms are too slow. Proprietary algorithms typically do not undergo a public evaluation by the cryptographic community; their security is unknown. They run the risk that they can easily be broken, like it was shown for the DST algorithm used in RFID car immobilizers [1]. Furthermore, proprietary algorithms lead to closed systems. Interoperability is therefore impossible. We however, wanted to tackle the RFID security problem in a more general way. A promising first step for authentication using standard mechanisms was presented in [3]. An AES hardware module is described, that fits the RFID constraints. Consequently, it is possible to employ standard symmetric authentication methods for RFID systems. In the next section, we briefly go through authentication in general.

IV. Authentication Protocols

During authentication, one entity proves its identity to another entity. Strong authentication protocols, such as challenge-response protocols (standardized in ISO/IEC 9798) are widely used in practice today. In challenge-response protocols, one or several messages are exchanged between the party who wants to prove their identity (the prover) and the party who wants to verify the identity (the verifier). In a typical scenario, the verifier challenges the prover with an unpredictable value that is used no more than once (the nonce). The prover is required to return a response that is depending on the nonce and on a secret.

Using strong authentication for RFID systems leads to significant security enhancements. If readers are required to authenticate themselves to tags, attacks such as unwanted tracking and unauthorized memory access are rendered infeasible. If tags are required to authenticate themselves against readers forgery of tags is prevented.

As argued in the previous section, it is advantageous to use standardized protocols and algorithms because they have been rigorously cryptanalyzed and are widely used. Hence, systems based on standardized protocols and algorithms are more likely to be secure and interoperable. Standardized challenge-response protocols are defined upon symmetric-key and asymmetric-key cryptographic primitives.

When using symmetric methods, both reader and tag possess the same key. It is possible to use symmetric encryption algorithms or MACs (message authentication codes). The drawback of symmetric authentication methods is that every party works with the same secret key. If the key of one party is compromised, the whole system becomes insecure, and key establishment and distribution is costly. Symmetric authentication methods are best suited for closed systems, where all devices are under the control of one central instance. Nevertheless, also open system applications are possible, but for each application the appropriate key establishment and management methods have to be considered.

Asymmetric authentication methods are well suited for open systems. Each party possess a public and a private key. With the private key one party can sign or encrypt a message. With the public key, which is open to everybody, each other party can verify the encryption or the signature and can in that way verify the identity of the message sender. Asymmetric methods are in general more time- and power-consuming than symmetric algorithms and are therefore out of question for RFID systems today.

Strong symmetric-key cryptographic primitives include encryption primitives such as AES [10] which allow extraordinary compact implementations [3]. This module is feasible for RFID tags and conforms the timing and energy requirements of an RFID system. In the following section, we show how to base strong authentication on such a compact implementation of a strong symmetric-key encryption primitive.

V. Suggested Protocols and Performance

A. Simulation of Protocols

For one tag in the reader's field, the timing of a protocol flow can be exactly determined by using the timing information of the applied communication standard. If RFID systems work with more than one tag and more than one time slot, the order of tag responses and therefore the protocol flow can no longer be exactly determined. The answering sequence and the time, when an unique ID is found depend on the tag IDs and also the protocol flow depends on it. This behavior of a real world RFID system is modeled with a sophisticated simulation tool written in Java. The communication between the classes is thread-based and is therefore well suited to simulate the communication in real world RFID systems. Figure 2 shows the concept of this Java model. Each component of an RFID system corresponds to a class in Java.

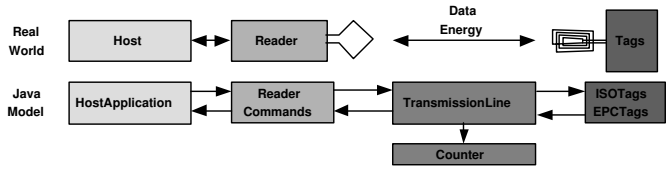


Fig. 2. Java Model of RFID System.

The Reader class provides reader requests, which are used by the HostApplication class to perform a protocol flow, programmed by the user. Here, the authentication protocols are embedded. The TransmissionLine class broadcasts the reader requests to all tags and evaluates the answers. The resulting response is passed over to the Reader and the HostApplication. Attached to the TransmissionLine is a Counter, which is able to exactly calculate the elapsed time during a protocol flow. With this counter, the performance of the different protocols can be evaluated. Before startup, the user can determine the system specification parameters, like the number of tags in the field and the dynamic behavior and so on.

At startup, a random ID is created for each tag and the protocol is executed until each tag is in quiet state. A protocol flow can be repeated with the same parameter setting. For one tag, the execution time is equal for all iterations. Due to the random IDs and the resulting arbitrary protocol flow, flows with more than one tag result in different execution times. From this values, an average protocol execution time can be determined.

B. ISO-18000 Standard Reference Protocol

The ISO/IEC 18000-3 standard [6] defines physical and procedural communication mechanisms between reader and tag using a frequency of 13.56 MHz. The reader talks first and sends a request. Tags, that are addressed can answer to this request. A request consists of an SOF (Start of Frame), flags, command code, parameter and data field, CRC value, and an EOF (End of Frame). Responses to requests also consist of an SOF, flags, parameter and data field, CRC value, and an EOF. In this paper, we limit ourselves to a description of some relevant concepts. For a more detailed description of physical parameters and communication mechanisms see [6].

Before sending requests to particular tags, the reader has to determine, which tags are in the field. For this purpose, it sends an inventory request, which is a mandatory request in ISO/IEC 18000-3. In most applications, there is more than one tag present which sends a response, and collisions are likely to occur. With an anti-collision mechanism, the reader has to find out the ID of each tag. Then, various requests can be sent by addressing one tag with its unique ID. A stay quiet request (also mandatory) has to be sent to the tag to remove the tag from the anti-collision procedure.

The minimal protocol to get the IDs of the present tags is a inventory request, followed by a stay quiet request for the identified tags. This protocol is used as reference protocol for the authentication protocols described in the next subsection. Figure 3 shows the reference protocol flow for one tag. This protocol requires in total 10.228 ms (this was calculated with timing constants from the ISO standard: 1-out-of-4 coding, 1 subcarrier, and 100% modulation).

R: inventory	Time to answer	T: ID	Time to next request	R: stay quiet	Time to next request
1.62368 ms	0.3209 ms	3.92704 ms	0.3092 ms	3.73824 ms	0.3092 ms

Fig. 3. ISO 18000 Reference Protocol.

First the reader starts with an inventory request without mask (SOF, 5 bytes, EOF). This lasts 1.62368 ms (this value was calculated with timing constants from the ISO standard: 1-out-of-4 coding, 1 subcarrier, and 100% modulation). Then the tag has to wait 0.32090 ms before sending the answer (TTA = time to answer). The tag transmits its ID (SOF, 12 bytes, EOF), which requires 3.92704 ms. The next request can be at the earliest 0.3092 ms after getting the tag response (TTR = time to next request). The stay quiet request from the reader (SOF, 12 bytes, EOF) needs 3.73824 ms. Here, no answer is expected and the reader has to wait 0.3092 ms before sending the next request (TTR). So, this protocol requires in total 10.22826 ms.

With more than one tag and/or the 16 slots variant (where tags have 16 time slots available to answer to an inventory

request) of the anticollision protocol, the timing can no longer be exactly determined, because the answering sequence and the time, when an unique ID is found depends on the IDs of the tags in the field. Therefore, the timing is estimated with the Java model described in subsection V-A. The average execution time for the reference model is about 359 ms which was evaluated running 40 simulations.

C. Authentication Protocols for RFID

By knowing the IDs of the tags in the field, the reader can send custom requests (the message formats can be found in [6]) to a tag by addressing it with its ID. In that way, the following authentication procedures are embedded into the protocol. All presented authentication protocols work with the AES-128 algorithm [10]. The goal of the new protocols is to prevent the security threats for RFID systems. These threats are forgery of tags, prevention of unwanted tracking, and unauthorized access to the tag's memory.

Protocol 1: Tag Authentication with Known ID. Here, the tag authenticates itself against a reader. The origin of the tag can be proved and forgery is prevented. The protocol works as follows (we denote the concatenation of values by $|$):

$$\begin{aligned} Reader \rightarrow Tag &: InventoryRequest \\ Tag \rightarrow Reader &: ID \\ Reader \rightarrow Tag &: AuthRequest \mid ID \mid R_R \\ Tag \rightarrow Reader &: E_K(R_R \mid R_T^\star) \mid R_T^\star \\ Reader \rightarrow Tag &: StayQuietRequest \end{aligned}$$

The reader sends an inventory request, the tag answers with its ID. The reader sends an authentication request, addressed with the ID of the tag (8 bytes). It contains a nonce, generated by the reader (R_R, 8 bytes). The tag encrypts the nonce (maybe padding is necessary) with the secret key and sends the result back to the reader, which can then verify the result. At last, the reader sends a stay-quiet request.

In our protocol, only the first 8 bytes of the encryption result are sent, see [5]. Nevertheless, the number of bytes can be changed in other applications. To avoid chosen-plaintext attacks, i.e. that an attacker can fix the value of R_R and can therefore control the input for the encryption, the tag can itself generates a nonce (R_T, 8 bytes) to "hide" the challenge. The use of R_T is optional and marked with *.

The whole protocol for one tag requires 20.940 ms (optional variant: 23.357 ms). For 20 tags, 16 slots, and 40 simulations, the average protocol execution time was 583 ms (optional variant: 624 ms). The tag needs to do AES encryption, and requires a nonce for the optional variant.

Protocol 2: Tag Authentication with Nonce. This protocol can be used for proof of origin of the tag and additionally prevents tag tracking. The tag takes part in the anticollision algorithm with a random ID (R_T, 8 bytes), which is generated when the tag enters the reader field and stays constant during the whole time when the tag is in the field. All addressed requests are done with the random number R_T used in the anti-collision procedure. In that way, tracking can be prevented.

$$\begin{aligned} Reader \rightarrow Tag &: InventoryRequest \\ Tag \rightarrow Reader &: R_T \\ Reader \rightarrow Tag &: AuthRequest \mid R_T \mid R_R \\ Tag \rightarrow Reader &: E_K(R_R \mid ID) \\ Reader \rightarrow Tag &: StayQuietRequest \end{aligned}$$

The protocol works like the above described protocol, but uses a random ID for tracking prevention. The challenge from the reader (R_R, 8 bytes) is concatenated with the tag's ID for encryption. All 16 bytes of the result have to be sent, because the reader has to extract the tag's ID by decrypting the answer.

The protocol for one tag requires 23.357 ms. For 20 tags, 16 slots, and 40 simulations, the average protocol execution time was 622 ms. The tag needs AES encryption and a nonce generation mechanism. Only an authorized reader (knowing the secret key) can reveal the tag's ID.

Protocol 3: Reader Authentication. This method is used for authenticated access to the tag's memory. The tag requests an authentication from the reader before it reveals its ID. The tag takes part in the anti-collision algorithm with a random ID (R_T, 8 bytes). All further requests are addressed with R_T which prevents tracking of the tag. The authentication of the reader works again using the encryption algorithm. After authorization of the reader, the tag sends its ID in plain text and grants the reader access to the memory. Attackers can get the ID by passively listening to the communication, although they are not able to initiate it. Another problem could be hijacking of an authorized connection. It has to be analyzed if this is a realistic security threat for real-world applications.

$$\begin{aligned} Reader \rightarrow Tag &: InventoryRequest \\ Tag \rightarrow Reader &: R_T \\ Reader \rightarrow Tag &: ReaderAuth \mid R_T \mid E_K(R_T \mid R_R^\star) \mid R_R^\star \\ Tag \rightarrow Reader &: ID \\ Reader \rightarrow Tag &: StayQuietRequest \end{aligned}$$

When answering to the inventory request, the tag indicates with a flag that the reader has to authenticate itself. The reader answers to the challenge (R_T, 8 bytes) and sends a request to reveal the tag's ID. To avoid a chosen-plaintext attack, the reader can generate a nonce R_R and combine it with R_T before answering the challenge (optional).

The protocol for one tag requires 20.940 ms (optional variant: 23.357 ms). For 20 tags, 16 slots, and 40 simulations, the average protocol execution time was 580 ms (optional variant: 624 ms). The tag needs a nonce generator and AES encryption.

Protocol 4: Mutual Authentication. In mutual authentication, both parties authenticate themselves against each other. All three security threats (unwanted tracking, unauthorized memory access, and forgery) can be prevented. Like in the

former protocols the tag answers the inventory request with a nonce (R_T, 8 bytes), and requests authentication from the reader. The reader answers the challenge and sends another challenge (R_R, 8 bytes) for the tag. The tag answers the reader's challenge and both are authenticated. The ID is never sent in plain, so unwanted tracking is prevented.

$$
\begin{aligned}
Reader \rightarrow Tag &: InventoryRequest \\
Tag \rightarrow Reader &: R_T \\
Reader \rightarrow Tag &: MutualAuth \mid R_T \mid E_K(R_T \mid R_R) \mid R_R \\
Tag \rightarrow Reader &: E_K(R_R \mid ID) \\
Reader \rightarrow Tag &: StayQuietRequest
\end{aligned}
$$

The protocol for one tag requires 25.774 ms. For 20 tags, 16 slots, and 40 simulations, the average protocol execution time was 680 ms. The tag needs to do AES encryption, and requires a nonce generator.

Protocol 5: Mutual Authentication with Key Exchange. This protocol is an extension of protocol 4. A secret key is exchanged to guarantee further secure communication.

$$
\begin{aligned}
Reader \rightarrow Tag &: InventoryRequest \\
Tag \rightarrow Reader &: R_T \\
Reader \rightarrow Tag &: MutualAuth \mid R_T \mid E_K(R_T \mid SK_R) \\
Tag \rightarrow Reader &: E_{SK}(SK \mid ID) \\
Reader \rightarrow Tag &: StayQuietRequest
\end{aligned}
$$

The secret key SK contains 16 bytes, whereas SK_R is only 8 bytes long. SK has to be derived from SK_R, for example $SK{=}SK_R \; xor \; R_T \mid SK_R$. In the last response, the tag confirms the secret key. This key can be used for further (encrypted) communication. The protocol for one tag requires 25.774 ms. For 20 tags, 16 slots, and 40 simulations, the average protocol execution time was 676 ms. The tag needs to do AES encryption and decryption, and requires a nonce generation mechanism.

Protocol Performance and Usage of Different Keys. The first three presented protocols all require at most 23.357 ms for one tag and about 625 ms for 20 tags. Protocol 4 and 5 require 25.774 ms for one tag and about 680 ms for 20 tags. The overhead factor (compared with the reference protocol) lies between 1.7 and 2.5. The byte count of nonces can be varied, depending on the security requirements of a system. All described protocols work with one global key. To prevent this, different secret keys could be used for example for different product classes. The tag holds only one key and can indicate the use of the correct key to the reader, which can hold various keys. One or more bytes can provide the selection of the correct key. Further investigation has to be done regarding key management and nonce generation, but this is out of scope of this paper.

VI. Interleaved Authentication Protocol

The performance data mentioned above is only true, if the used cryptographic primitive (AES) is able to calculate its result during the fixed time to answer (TTA = 0.3209 ms, defined in ISO/IEC 18000-3 [6] as the interval between the end of a request an the start of a response). For the module described in [3], which meets energy and size constraints for use on RFID tags, this assumption does not hold. The calculation time for AES with this module needs about 10 ms. Nevertheless, authentication for RFID is possible. We present a method how authentication can be done with a cryptographic primitive, which requires more calculation time than TTA. For this purpose, authentication is split into two parts. The first part is the authentication request, which tells the tag to encrypt the challenge and does not expect any response. The second part is the response request, which collects the authentication response, if the result is available.

For the tag authentication with known ID, the interleaved version of the authentication protocols works like shown in figure 4 (the inventory request and stay quiet request are omitted in the picture). During the idle time, the reader is not

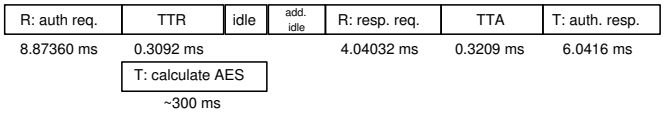


Fig. 4. Interleaved Protocol with 1 Tag

active. The additional idle time indicates the period, where the reader is idle and the tag has already stopped its calculation. In the best case (when the response request is sent right after the calculation of AES has stopped, i.e. the additional idle time is 0), the authentication protocol for one tag requires about 40 ms (with inventory and stay quiet). This time is strongly determined by the calculation time for the AES. For one tag, the timing overhead is very large, but with more than one tag, the reader can use the idle time (while the tag calculates the AES) to send authentication requests (or other requests) to other tags. The other tags start their calculation and the calculation results can be collected (after a certain time) one after another. This mechanism is outlined in figure 5. AR

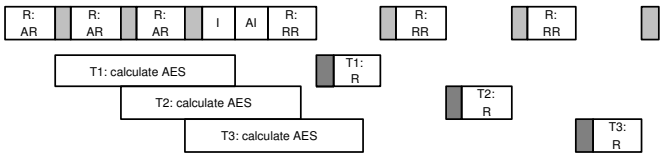


Fig. 5. Interleaved Protocol with 3 Tags

means authentication request, RR means response request and R is the tag's response. I is the idle time, where the reader is inactive, AI is the additional idle time, where neither the reader nor the tag is active. The grey areas indicate TTA or TTR. The calculation of the AES is pipelined. The shown model assumes, that the time for collecting one answer exceeds the offset between the calculation start for two tags, i.e. that the response for the next response request is ready before the response request is sent. This assumption holds in most

of the cases: the offset between two tags is $AR{+}TTR$ and the time for the response collection is $RR{+}TTA{+}R{+}TTR$. The number of data bytes for AR and $RR{+}R$ are in general similar, but the overhead for communication is higher in the second term (TTA and communication overhead bytes of R). The stay quiet requests for all tags are issued after the collection of the authentication results.

While the tags are calculating the AES result, they do not listen to the reader. So, they are excluded from the inventory process. Therefore, the reader can even send an inventory request during this time to get new IDs from other tags. It has to be made sure, that the inventory request is issued before the tags stop their calculation. The reader has to decide, when to start to collect the results from the tags. It can send a response request (polling) and if it gets no response, it can decide to send other requests and try again after a certain time. It is up to the reader, to find the optimal strategy for handling interleaved protocols.

Nevertheless, an estimation of the performance for interleaved protocols can be given. The total time (T) for handling the authentication requests for a certain number of tags (N), where $N{*}(AR{+}TTR){<}AES{+}AR$ is calculated like this:

$$T{=}AR{+}AES{+}AI{+}N{*}(RR{+}TTA{+}R{+}TTR). \quad (1)$$

For $N{*}(AR{+}TTR){>=}AES{+}AR$ the following equation is valid:

$$T{=}N{*}(AR{+}TTR){+}AI{+}N{*}(RR{+}TTA{+}R{+}TTR) \quad (2)$$

AR is the time for the authentication request. AES is the time required for AES calculation of one tag (10 ms for the AES module described in [3]). RR is the response request (4.040 ms). R is the tag's response.

The worst case for timing occurs, when only one tag has to be authenticated. The best case for the total protocol execution time occurs, when $AI{=}0$ and the total idle time of the reader can be filled with authentication requests, i.e. if $N{*}(AR{+}TTR){>=}AES{+}AR$. In protocol 1, this condition holds for $N{>=}3$ ($AES{=}10ms$). The total time for execution of the interleaved authentication protocol 1 for 3 tags is about 45 ms. Non-interleaved authentication for 3 tags takes about 32 ms, i.e. that the interleaved version of protocol 1 takes about 1.41 times the time of the non-interleaved version. For the worst case (only one tag is authenticated) the timing ratio is about 2.3. The more tags are authenticated in parallel, the better is the timing ratio between interleaved and non-interleaved version, especially for cryptographic algorithms with very long calculation times. For $N{*}(AR{+}TTR){>=}AES{+}AR$ the timing ratio reaches its minimum. Figure 6 shows the timing ratios

	Prot. 1, 3	Prot. 2	Prot. 4, 5
Min.	1.41	1.33	1.28
Max.	2.30	2.07	2.55

Fig. 6. Interleaved Protocol Performance.

(interleaved version vs. non-interleaved version) for the best-case and worst-case estimations of the five protocols. In protocols 4 and 5 the tags need to calculate the AES algorithm two times. Therefore, the AES requires 20 ms. The presented figures only consider the authentication part of the protocols and omit the inventory and stay quiet requests. The overhead factor for the interleaved version lies between 1.28 and 2.55 (for AES with 10 ms for protocol 1, 2, and 3 and AES with 20 ms for protocol 4 and 5) depending on the protocol and the number of parallel authentication processes.

VII. Conclusion

In this paper we have challenged the believe that strong authentication is infeasible with RFID tags in practice. By using strong authentication threats such as forgery of tags, unwanted tracking of tags, and unauthorized memory access of tags can be defeated.

We have described five symmetric-key authentication protocols (based on ISO/IEC 9798-2) with different security features all conforming to ISO/IEC 18000-3. Our protocols are based on symmetric-key encryption. Therefore, they can be used with the AES algorithm, which can be implemented such that the energy constraints of Class 2 RFID systems are met. In order to comply to the timing constraints (in particular to the TTA), we have suggested interleaved versions of the authentication protocols. With a sophisticated Java simulation tool we have analyzed a system with one reader and several tags performing the authentication protocols. This simulation has shown that there is only a small penalty for authentication in a real world scenario. The interleaved protocols take on average 1.28 times to 2.55 times longer than a non-interleaved authentication protocol. The non-interleaved authentication protocols take between 1.7 to 2.5 times longer than the reference protocol without authentication. Summarizing, we have presented a concept for strong symmetric-key authentication for Class 2 RFID tags, which can be implemented in practice today.

References

[1] S. Bono, M. Green, A. Stubblefield, A. Juels, A. Rubin, and M. Szydlo. Security Analysis of a Cryptographically-Enabled RFID Device, 2004. http://www.rfidanalysis.org/DSTbreak.pdf.

[2] EPCglobal. 13.56 MHz ISM Band Class 1 Radio Frequency (RF) Identification Tag Interface Specification, February 2003. Available online at http://www.epcglobalinc.org/.

[3] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In *CHES 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2004.

[4] International Organization for Standardization (ISO). ISO/IEC 9798-2: Information Technology - Security Techniques - Entity authentication mechanisms - Part 2: Mechanisms using symmetric encipherment algorithms, 1993.

[5] International Organization for Standardization (ISO). ISO/IEC 9797-1: Information Technology – Security Techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher, 1999.

[6] International Organization for Standardization (ISO). ISO/IEC 18000-3: Information Technology AIDC Techniques — RFID for Item Management, March 2003.

[7] A. Juels. Minimalist cryptography for low-cost rfid tags. in submission, 2003.

[8] A. Juels and R. Pappu. Squealing Euros: Privacy protection in RFID-enabled banknotes. In *Financial Cryptography 2003, Revised Papers*, volume 2742 of *Lecture Notes in Computer Science*, pages 103–121. Springer, 2003.

[9] A. Juels, R. L. Rivest, and M. Szydlo. The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy. In *ACM Conference on Computer and Communication Security, 2003, Proceedings*, pages 103–111. ACM Press, 2003.

[10] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at http://www.itl.nist.gov/fipspubs/.

[11] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to a privacy friendly tags. In *RFID Privacy Workshop, MIT*, 2003.

[12] I. Vajda and L. Buttyn. Lightweight authentication protocols for low-cost RFID tags. In *2nd Workshop on Security in Ubiquitous Computing*, October 2003.

[13] S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In *Security in Pervasive Computing, 2003, Revised Papers*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2003.

8-Bit Microcontroller System with Area Efficient AES Coprocessor for Transponder Applications

Mark Jung[1], Horst Fiedler[2], and Reneé Lerch[1]
[1] Fraunhofer Institute of Microelectronic Circuits and Systems
Finkenstrasse 61, 47057 Duisburg, Germany
[2] Integrated Systems Institute at the University of Dortmund
Emil-Figge-Strasse 68, 44227 Dortmund, Germany

Abstract

Apart from pure identification systems, transponder systems with embedded sensors gain increasing importance. In contrast to low-end Radio Frequency Identification (RFID) systems these transponders usually require more complex protocols which need to be adjusted to a specific application. In this contribution the demanded flexibility is achieved by using the 8-bit microcontroller IMS3311. In order to guarantee privacy and security of transferred information, it is essential that transmitted data is encrypted by a strong algorithm. Therefore, the Advanced Encryption Standard (AES) is used. The main focus of this paper is the development of an area-efficient AES coprocessor which accelerates the processing of the controller's ciphering tasks. The AES unit requires only 2168 logic gates and takes less than 650 clock cycles for either encryption or decryption of a 128-bit block.

1 Introduction

The market for Radio Frequency Identification (RFID) transponder systems has grown considerably in the last years, as they are introduced into various new application areas. Such transponder systems usually consist of a reader device and the RFID chip or tag.

In the most simple implementation, a tag holds an identification code which can be requested by a reader. Data is transmitted via an electromagnetic field which is generated by the reader device. Possible applications for low-end RFID tags are the identification of objects, animals or persons. Hence, the tags can be used as replacement of barcodes in shops, for the observation of cattle in agriculture or for the identification of luggage at the airport. Often, these tags contain additional object information, e.g. the date of expiry in barcode chips.

More complex transponder chips require bidirectional data transmission and complex protocols between reader and transponder. These chips can also contain integrated

sensors, and they often provide in-field programmability of an internal EEPROM. Possible applications for complex tags range from temperature measurements in refrigerated transports to tire pressure sensors in vehicles.
Different applications require different protocols. In order to enhance the flexibility of a transponder and consequently reduce development cost, it is important that diverse protocols can be processed by the same chip. The needed flexibility can be provided by a small and efficient microcontroller system. In this contribution the 8-bit µC architecture IMS3311 is used, which was developed at the Fraunhofer IMS in Duisburg and is well-suited for the requirements of transponder systems.

RFID tags are called passive transponders if they obtain their energy from the reader's electromagnetic field [13]. Passive tags are distinguished from transponders with separate energy sources like batteries or solar cells, which are referred to as active transponders. For both active and passive transponders low power dissipation is important, as the energy provided by field, battery or solar cell is limited.

There are several fields that are very sensitive with respect to privacy and security, e.g. chips with biometric identification data or chip implants in medical technology. It is necessary that communication between reader and tag is encrypted within these systems. A reasonable choice of a cryptographic algorithm is the Advanced Encryption Standard (AES).
Especially if the tag remains in the electromagnetic field for a short time, it is important that enciphering of data is accelerated by a cryptographic unit. A reduced number of cycles for the execution of the protocol can also be used to lower clock frequency and power dissipation, respectively.
The gate count of the system and thus the AES unit needs to be minimized in order to reduce unit cost and power dissipation. This paper focuses on the development of a novel area-efficient AES coprocessor, which accelerates the microcontroller system in encryption and decryption of data and avoids that those tasks become a bottleneck within the transmission protocol. By sharing storage resources with the controller the coprocessor significantly reduces its gate count in comparison to prior architectures.

Section 2 gives an overview of the current state of research. Furthermore, it introduces the general architecture of the IMS3311 µC. The architecture of the AES coprocessor and the most significant strategies for reduction of chip area are presented in section 3. A comparison of the proposed architecture with prior implementations is presented in section 4. Finally, concluding remarks concerning the developed system are given.

2 Architecture Considerations

Figure 1 represents the architecture of a transponder chip. The main part consists of the microcontroller system, an analog frontend and an antenna. In addition, the system can be extended by an EEPROM, sensors, a battery or an oscillator. The focus of this paper will be the optimization of the µC system, which contains the controller IMS3311, the area-efficient AES coprocessor, a RAM and a program ROM. After an overview of the state of research, the controller system is introduced.

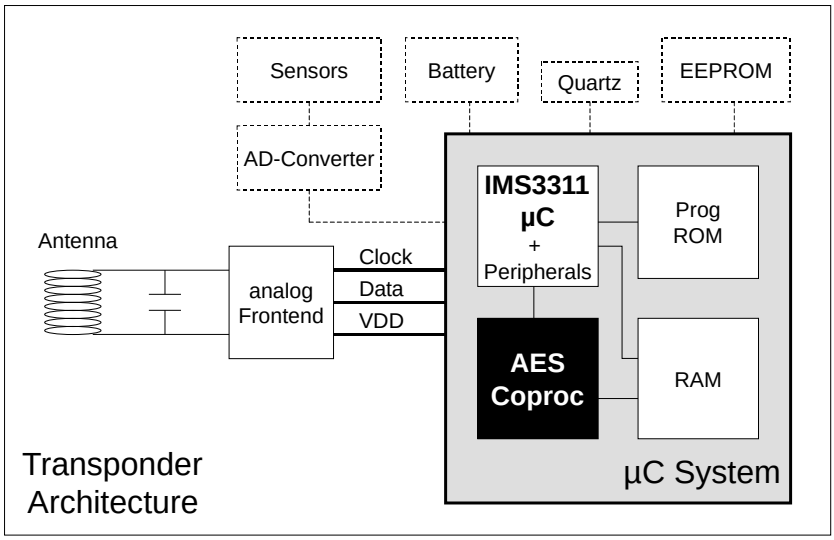


Figure 1: Transponder chip architecture with cryptographic coprocessor.

2.1 Related Work

Research and industry offer several stand-alone AES ASICs and IP cores [1] [5] [9] [14]. Few AES units are implemented in combination with a µC. Furthermore, the existing controller systems with on-chip AES unit usually are 32-bit architectures which have too high power and area requirements for RFID applications. One of these complex architectures is the 32-bit Freescale MCF5235 [7] that contains a cryptographic coprocessor supporting AES, DES and Triple-DES.
There are even few stand-alone AES cores that meet the requirement of a low gate count. Feldhofer [5] recently published the implementation of an AES encryption-only core which is optimized for the operation in RFID tags. However, the system proposed in [5] does not support complex bidirectional protocols associated with sensor transponders. The system developed in our contribution bridges the gap between costly 32-bit µC architectures and specialized but inflexible ASICs.

2.2 IMS3311 µC Architecture

The small and efficient controller IMS3311 [6] is instruction set compatible with the Motorola M68HC11 and well-suited for transponders or embedded systems. Its register file contains two 8-bit accumulators, as well as two 16-bit index registers, a program counter and a stack pointer. Beside the regular 8-bit operations, the instruction set of the IMS3311 supports several operations which work with 16-bit operands.

The 8-bit data bus connects the peripherals in a ring, which has the advantage over the tri-state bus concept that the load is distributed over multiple drivers. Furthermore, the ring bus architecture can be synthesized simply. Possible

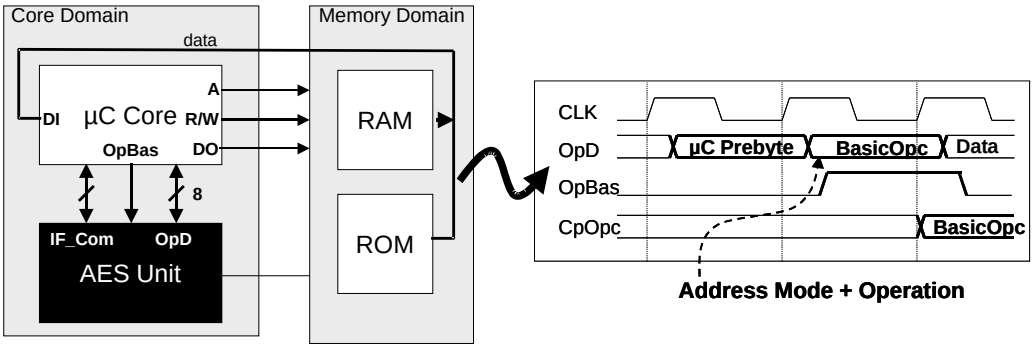


Figure 2: µC interface to AES coprocessor.

peripheral devices on the ring bus are interfaces like SPI, SCI, PIO or CAN. In addition, the system usually includes a clock controller and a JTAG test controller.

In order to allow the extension of the controller with a cryptographic unit, the standard IMS3311 had to be equipped with a slim coprocessor interface. This interface constitutes a significant difference between the IMS3311 and the M68HC11. The microcontroller provides special opcodes for coprocessor instructions. Thus, the AES unit is controlled by dedicated cryptographic instructions. The simplified interface between controller and acceleration unit is presented on the left side of figure 2, while the transfer of one instruction is depicted on the right side. Data and instruction bytes are transmitted via the *OpD* (opcode and data) bus to the coprocessor. Whenever the controller receives a new byte from its data bus, it automatically forwards this byte to the *OpD* bus with the next rising clock edge. One of these bytes is the µC pre-byte, which contains the information if the current instruction is a coprocessor command. This byte is exclusively decoded by the controller.
In the following cycle the instruction's basic opcode byte is transmitted by the *OpD* bus to the AES unit. It is stored in the internal coprocessor register *CpOpc* after the rising clock edge, if the basic opcode strobe signal *OpBas* is asserted high. The byte is subdivided into the following nibbles: One nibble contains information about the address mode for load/ store instructions or the processing of a branch instruction, while the other decides about the current operation (e.g. "decrypt" or "load"). The address mode bits are evaluated by the microcontroller. This allows the following address modes to be used for coprocessor instructions:

- immediate,
- direct,
- extended,
- indexed and
- push to/ pop from stack.

After the basic opcode byte and address bytes, following data bytes may be loaded into the coprocessor, or data from the coprocessor may be stored into memory.

```
//+++++++++++++++++ ENCRYPTION +++++++++++++++++//
State = Plaintext;
for(i=0; i<9; i=i+1)
{
   AddRoundKey(State, ExpKey[i]);
   SubBytes(State);
   ShiftRows(State);
   MixColumns(State);
 }
AddRoundKey(State, ExpKey[9]);
SubBytes(State);
ShiftRows(State);
AddRoundKey(State, ExpKey[10], Ciphertext);
```

Code Example 1: Equivalent encryption structure of the AES.

3 AES Acceleration Unit

Based on the description of the µC's interface to the AES unit, the internal structure of the coprocessor is developed. Since RFID chips demand a very small die size, the main methods for the reduction of chip area are introduced. As area reduction considers the algorithm's structure and operations, an overview of the AES is given.

3.1 Advanced Encryption Standard

The AES [4] is a block cipher with an input and output block length of 128 bits. The algorithm is capable of using keys of a length of 128, 192 and 256 bits. For the presented implementation, a key-length of 128 bits is chosen which is assumed to be secure against state-of-the-art cryptanalytic attacks. The key and input block length of 16 bytes implies a round number of 10.

The encryption structure used for the presented implementation is described by the pseudo code in code example 1. This representation is equivalent to but more regular than the representation in the standard, as for a hardware implementation just MixColumns has to be replaced by AddRoundKey in the final round. For decryption the operation order has to be reversed and the presented operations have to be replaced by their inverse operations. The AddRoundKey operation is the same for both encryption and decryption. For decryption the SubBytes operation is replaced by InvSubBytes, MixColumns by InvMixColumns and ShiftRows by InvShiftRows.
The algorithm consists of the following operations:

- The AddRoundKey step performs a bitwise XOR operation of the current state and the round key.
- SubBytes replaces each byte of a state by applying an S-Box. The S-Box is the non-linear step of the algorithm. InvSubBytes consists of the inverse S-Box which is again applied byte-wise to the current state.
- The MixColumns step performs a bricklayer permutation on four bytes of the state. These bytes form a state's column. Each MixColumns operation

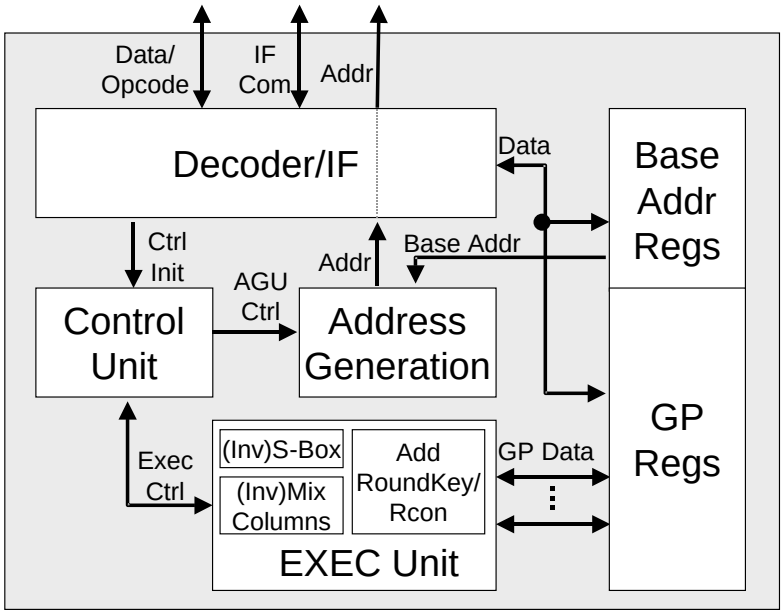


Figure 3: Internal structure of the AES coprocessor.

is a modular multiplication with a fixed polynomial. InvMixColumns is the inverse operation of MixColumns. These operations are usually represented as multiplication with a 4×4 matrix. An optimized implementation with a combined circuit for encryption and decryption is demonstrated in section 3.4.

- ShiftRows is a byte transposition step which cyclically shifts four selected bytes of a state. These four bytes are called a row. InvShiftRows is the inverse operation of ShiftRows.

The key expansion generates a 16-byte round key for each of the rounds. The round keys can be either generated on-the-fly during ciphering or they can be expanded before execution of the cipher. On-the-fly expansion uses less memory. However, in this contribution key expansion is done *before* encryption or decryption for several reasons: Round key generation prior to ciphering leads to a noticeably higher throughput, since the key can remain in memory for the next block. Furthermore, on-the-fly key generation for decryption starts with the last round key and is developed back to the symmetric key, so the last round key would have to be generated and stored before decryption anyway. Finally, for encryption and decryption the same expanded key and thus the same round key generation algorithm can be used.

3.2 Coprocessor Architecture

Figure 3 depicts the internal coprocessor structure. The decoder and interface unit controls the communication with the IMS3311 core and its memory. In addition, it generates initiate signals for the control unit. In particular, these signals determine the currently valid operation, so a possible decoder output is ”InitEncrypt”.

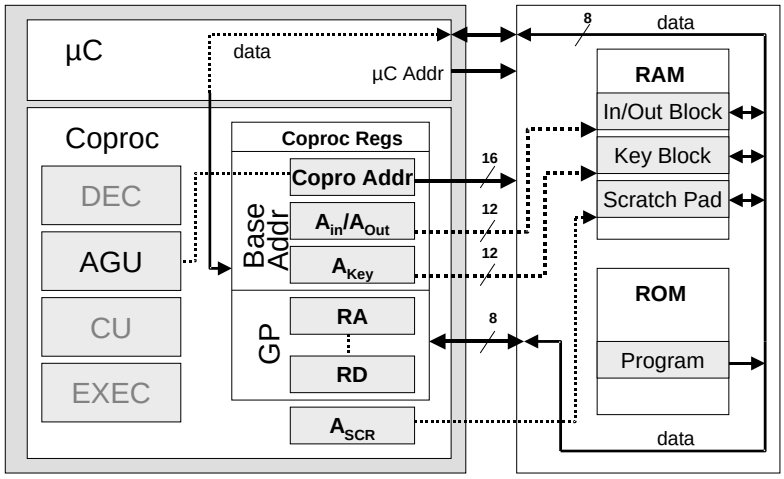


Figure 4: Direct access to microcontroller memory.

The control unit generates an adequate control sequence for the current instruction using hard-wired logic. The core of the control unit forms a global finite state machine (FSM) which initiates further local FSMs for the execution of the AES operations. Since the local FSMs are independent from each other, some of the AES operations can be processed in parallel.
The output signals of the state machines initiate the address generation and the execute unit. While the execute unit implements the combinatorial logic for all AES operations, the address generation unit provides an interface to the microcontroller's internal memory.
The register file consists of the general purpose (GP) and the base address registers. General purpose registers store the temporary results, whereas base address registers can be regarded as pointers to the controller's memory (see section 3.3).

3.3 Coprocessor Interface to µC RAM

The minimization of the gate count is essential for commercially viable realizations of transponder chips. A main approach of this contribution is to minimize the number of general purpose registers and thus flipflops without loosing too much of the performance. Especially flipflops contribute heavily to the chip's gate count, as they use an area of about six gate equivalents (GE). If the complete input and key blocks had to be stored in internal flipflops, this would already require 256 flipflops - more than the IMS3311 microcontroller needs itself. Hence, an input block is not processed in parallel as in [14], but it is encrypted and decrypted in blocks of 32 bits. Temporary results are stored in RAM.

This strategy has already been applied to FPGA designs [2]. Furthermore, the AES core introduced in [5] uses an internal RAM. However, a small internal RAM is very expensive. Especially in small memories the column and row decoders contribute significantly to memory area. As a result small RAMs tend to be uneconomical.

This leads to the main advantage of our acceleration unit: The coprocessor does not need internal memory, but shares RAM and address space with the controller. Memory access is controlled by the coprocessor's internal address generation unit. In order to achieve higher flexibility, it provides several base address registers, as illustrated in figure 4. These registers serve as pointers to RAM or ROM addresses and can be used to determine the position of the input, output or key block. Start addresses are aligned to 16-byte boundaries which is the size of one block. The base address registers are loaded by a coprocessor load instruction making use of the controller's address modes. In addition, there are constant pointers to the RAM, e.g. for the start address of the 16-byte scratch pad block for temporary results.

The address generation unit allows several further features, which enhance throughput and programming convenience: Two additional instructions, ECRI ("encrypt and increment") and DCRI ("decrypt and increment"), are provided for ciphering of successive blocks. They support encryption and decryption of large blocks by executing just one instruction repeatedly. At the end of these instructions, the input and output base address registers increment and automatically point to the next block in memory.

The most compact interface between controller, coprocessor and RAM supports *sequential* processing of coprocessor and µC instructions. The µC stops data bus accesses during coprocessor activity. The AES unit becomes bus master and gets access to ROM or shared RAM. After the coprocessor has terminated the current instruction, the µC becomes bus master again and loads the next instruction from memory.
With few additional gates this sequential interface can be extended to a *hybrid* interface. The hybrid interface additionally allows processing of controller and coprocessor instructions in *parallel*. Special instructions allow switching between parallel and sequential behavior. Furthermore, the processing status of the current coprocessor instruction can be requested by the µC. The results in section 4 are based on the sequential interface. A system with a hybrid interface requires 208 additional gates.

3.4 Optimized Implementation of the AES Operations

As mentioned in section 3.2, the AES unit processes only four bytes at once, namely those bytes that are needed for one 32-bit (Inv)MixColumns operation. The required bytes are loaded from the memory's input array in the first round and alternating from the memory's scratch pad or output array in the other rounds. During DataFetch they are sequentially loaded into the four general purpose registers $R_A - R_D$. The ShiftRows operation can already be processed by DataFetch, if loading of $R_A - R_D$ considers the cyclic byte shifts (also discussed in [2]). The timing diagram in figure 5 presents the flow for one column of a regular encryption round. After DataFetch the steps AddRoundKey, SubBytes and MixColumns are executed in a pipeline-like structure. The temporary results after a column calculation are written to the 16 byte scratch pad or output array during the WriteBack state.

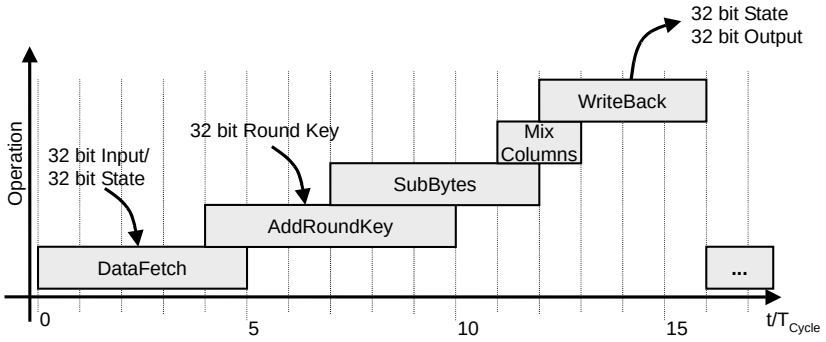


Figure 5: Timing diagram of a regular 32-bit AES encryption round transformation.

Usually the *SubBytes* and *InvSubBytes* steps are performed by using 256 byte LUTs for each operation. If the LUTs were located in the µC's program ROM, this operation occupies the µC's data bus, which is already the bottleneck, for 16 more clock cycles per round. Hence, the S-Box and the inverse S-Box are implemented using hard-wired logic. Rijmen [12] proposes a method to implement the S-Box and the inverse S-Box by using combinatorial logic, which is based on the design of logic circuits for the multiplicative inverses in composite fields (see also [10] [11]). In this contribution the logic for the multiplicative inverses in $\mathcal{GF}((2^4)^2)$ is shared by encryption and decryption. Both substitute operations are pipelined and executed in two clock cycles. Four (Inv)SubBytes instances would lead to an enormous gate count, so the substitute operations are processed byte-wise sharing one instance.

MixColumns and *InvMixColumns* are the only steps that process four bytes in parallel. While MixColumns is executed within one clock cycle, InvMixColumns is processed in two cycles. An 8-bit implementation of MixColumns already takes 7 cycles [5]. InvMixColumns is much more complex and is assumed not to take less cycles. That is why the proposed AES unit processes four bytes in parallel. For the InvMixColumns operation Barreto's transformation [3] is used:

$$\begin{bmatrix} \texttt{0E} & \texttt{0B} & \texttt{0D} & \texttt{09} \\ \texttt{09} & \texttt{0E} & \texttt{0B} & \texttt{0D} \\ \texttt{0D} & \texttt{09} & \texttt{0E} & \texttt{0B} \\ \texttt{0B} & \texttt{0D} & \texttt{09} & \texttt{0E} \end{bmatrix} = \begin{bmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{bmatrix} \times \begin{bmatrix} \texttt{05} & \texttt{00} & \texttt{04} & \texttt{00} \\ \texttt{00} & \texttt{05} & \texttt{00} & \texttt{04} \\ \texttt{04} & \texttt{00} & \texttt{05} & \texttt{00} \\ \texttt{00} & \texttt{04} & \texttt{00} & \texttt{05} \end{bmatrix} \tag{3.1}$$

The InvMixColumns matrix can be decomposed into the MixColumns matrix and a matrix which just contains the `04` and `05` elements. The implementation of the combined operations is presented in figure 6. Each of the GP registers $R_A - R_D$ contains one column byte. In the first cycle of InvMixColumns (IMC), multiplication with the matrix on the right side is performed, while in the second cycle multiplication with the MixColumns (MC) matrix takes place. Results are stored in $R_A - R_D$ again. A constant multiplication with `02` is called *xtime()* function in [3] and requires 3 XOR gates, whereas the constant multiplication with `04` needs 5 XORs. Only the operations with the grey gates were added to the MixColumns circuit in order to support InvMixColumns as well, which increases circuit size by

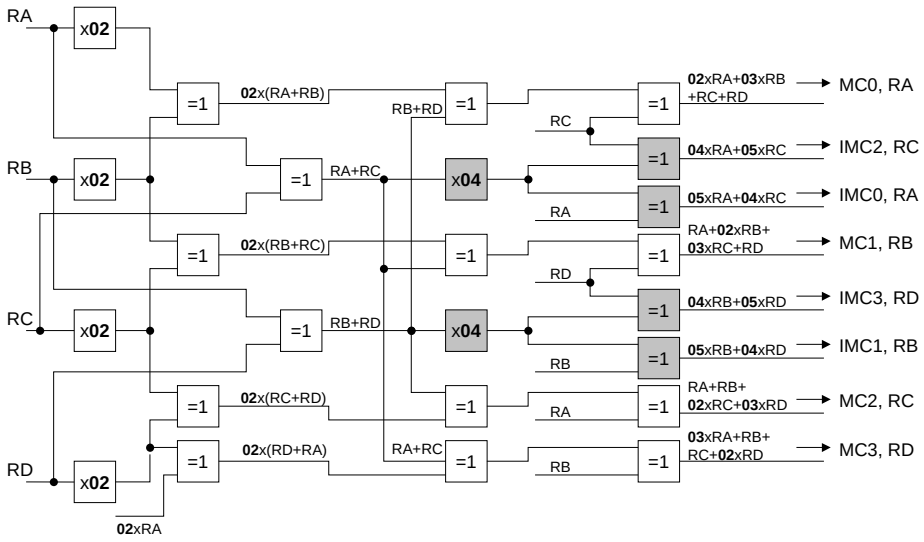


Figure 6: Implementation of the MixColumns and InvMixColumns step.

just 42 gates. The combined column implementation only takes 166 XOR gates. In comparison, the optimized solution introduced by Zhang and Parhi [15] requires 264 XOR gates.

4 Implementation Characteristics

The introduced µC and coprocessor modules were synthesized on a 0.5µm CMOS 3-metal-layer process. Table 1 summarizes the synthesis results. The total results were achieved by a flat synthesis, while the values of the internal coprocessor blocks result from a hierarchical synthesis.
The entire circuit requires only 2168 gates for encryption, decryption and key expansion. Since the coprocessor shares the RAM with the microcontroller core, it does not require an internal RAM. After the termination of a coprocessor instruction, the µC can write to the RAM addresses, which were used by the coprocessor, again. The RAM area used by the AES unit is listed in brackets in table 1. AES unit and microcontroller together have a logic gate count of less than 5300 GE. Additional area for microcode and program ROM, RAM and the desired peripherals has to be added.

A comparison of the proposed coprocessor with other AES cores is shown in table 2. The developed AES coprocessor is the only solution that offers encryption, decryption and key expansion with a low gate count. Feldhofer [5] introduced an encryption-only core which requires only a small amount of logic gates. The high gate count of the used internal RAM results in a total number of 3595 GE. The other AES units are optimized for higher throughput. Hence, their large gate count makes them inapplicable for the operation in transponder systems.
Only the presented solution is implemented in combination with a flexible 8-bit µC system. A software implementation of the AES algorithm is described in [8].

Characteristic	Unit/ Block	Value
Gate Count	Coproc AES total	**2168 GE**
	Coproc Execute Unit	54.8 %
	Coproc Register Unit	22.0%
	Coproc Decoder/ Control Unit	13.6%
	Coproc Address Generation	9.6%
	µC RAM area used by Coproc	(921 GE)
	Coproc AES + used µC RAM area	(3089 GE)
	IMS3311 µC Core	3110 GE
	Microcode ROM (2 KB)	2456 GE
Cycle Count	Encryption (128-bit)	**645**
	Decryption (128-bit)	**645**
	KeyExpansion (128-bit)	**363**

Table 1: Characteristics of the developed system.

The used Freescale M68HC05 architecture is comparable to the IMS3311. For encryption, it needs about 10 times more clock cycles than the AES unit proposed here, for decryption it is about 15 times slower.

5 Conclusions

This contribution introduced a microcontroller system that is optimized for the operation in low-cost applications. It is well-suited for RFID transponder chips that are used for data logging tasks. The implemented system contains the 8-bit microcontroller IMS3311 and an area-efficient AES coprocessor which supports both encryption and decryption. An instruction based coprocessor interface was introduced, which extends the µC's instruction set with commands for the AES acceleration unit. In addition, it was shown how the microcontroller's resources can be shared in order to minimize coprocessor area. The proposed AES implementation requires only 2168 gates and takes less than 650 clock cycles for encryption or decryption of a 128-bit input block.

AES Unit	Supports E/D/RK[1]	Gate Count	Cycle Count E/D	Cycle Count (RK+E)/(RK+D)
Proposed AES Unit (incl. Shared RAM)	+/+/+	2,168 GE (3,089 GE)	645/645	1008/1008
Feldhofer [5]	+/–/+	3,595 GE	–/–	1016/–
Amphion CS5265 [1]	+/+/+	25,000 GE	–/–	44/44
Verbauwhede [14]	+/–/+	173,000 GE	–/–	10/–

Table 2: AES core comparison considering gate count and clock cycles.
[1]E=Encryption, D=Decryption, RK=Round Key Generation

References

[1] Amphion: *CS5265/75 AES Simplex Encryption/Decryption Cores*, data sheet, http://www.amphion.com

[2] P. Chodowiec, K. Gaj: *Very Compact FPGA Implementation of the AES Algorithm*, Cryptographic Hardware and Embedded Systems (CHES), Springer, 2003, pp. 319-333

[3] J. Daemen, V. Rijmen: *The design of Rijndael*, Springer, 2002

[4] Federal Information Processing Standards Publication (FIPS): *Announcing the Advanced Encryption Standard (AES)*, Publication 197, National Bureau of Standards, U.S. Department of Commerce, 2001

[5] M. Feldhofer, S. Dominikus, J. Wolkerstorfer: *Strong Authentication for RFID Systems Using the AES Algorithm*, Cryptographic Hardware and Embedded Systems (CHES), Springer, 2004, pp. 357-370

[6] Fraunhofer IMS Duisburg: *IMS3311, System Integration Manual*, data sheet, 2001

[7] Freescale: *Coldfire MCF5235 Reference Manual*, data sheet, http://www.freescale.com

[8] G. Keating: *Performance Analysis of AES Candidates on the 6805 CPU Core*, Proc. of the 2nd AES Candidate Conference, 1999, pp. 109-114

[9] A.K. Lutz et al.: *2 Gbit/s Hardware Realizations of Rijndael and Serpent: A Comparative Analysis*, Cryptographic Hardware and Embedded Systems (CHES) 2002, Springer 2003, pp. 144-158

[10] C. Paar: *Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields*, VDI, 1994

[11] C. Paar, M. Rosner: *Comparison of Arithmetic Architectures for Reed-Solomon Decoders in Reconfigurable Hardware*, IEEE Symposium on Field-Programmable Custom Computing Machines, 1997

[12] V. Rijmen: *Efficient Implementation of the Rijndael S-Box*, URL: http://www.esat.kuleuven.ac.be/

[13] S. Sarma, D. Brock, D. Engels: *Radio Frequency Identification and the Electronic Product Code*, IEEE Micro, November 2001

[14] I. Verbauwhede, P. Schaumont, H. Kuo: *Design and Performance Testing of a 2.29 Gb/s Rijndael Processor*, IEEE Journal of Solid-State Circuits, 2003, pp. 569-572

[15] X. Zhang, K.K. Parhi: *Implementation Approaches for the Advanced Encryption Standard Algorithm*, IEEE Journal, 2002

Electromagnetic Side Channel Analysis of a Contactless Smart Card: First Results

Dario Carluccio, Kerstin Lemke, Christof Paar

Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany
www.crypto.rub.de
{carluccio, lemke, cpaar}@crypto.rub.de

Abstract. Radio frequency (RF) based cryptographic tokens have been a booming market in the past years. These low-cost devices are in use for security services such as ticketing, access control, and electronic payment. The security services often require that the owner of the device is not able to read out or modify the secret data (cryptographic keys and unique IDs) stored.
Side channel cryptanalysis includes smart methods to extract cryptographic keys just by observing its physical leakage during computation. EM radiation that is emitted by crypto devices can be used for Differential Electromagnetic Analysis (DEMA). Previous work on EM analysis has been done on self programmed microcontrollers and FPGAs. For our analysis we use a Mifare DESFire card that is supplied by an RF interface. We present the measurement set-up and reconsider the properties of electric and magnetic field coupled antennas. As preparation step for DEMA, the authentication procedure of the DESFire card was partially revealed. By using the efforts described in this work DEMA was not successful to extract cryptographic keys. Further directions are given that yield to more powerful conditions for applying DEMA against RF based crypto devices.

1 Introduction

Radio Frequency Identification (RFID) systems are heavily promoted and continue to become more and more pervasive. An RFID system consists of an RFID transponder (RFID card) and an RFID card reader (see Figure 1). The RFID transponder is supplied by the card reader.

In the security context, privacy aspects (see, e.g, [9]) are widely discussed as consumer tracking becomes feasible using RFID based technology. In the main security related works RFID is related to extreme low-cost devices that store unique IDs and transfer them without any cryptographic mechanisms in place.

In [4] an AES hardware implementation is presented for strengthening RFID transponders by using strong cryptographic mechanisms. Widespread commercial RFID transponders are mainly equipped with weaker cryptographic mechanisms. DST transponders produced by Texas Instruments that are used for

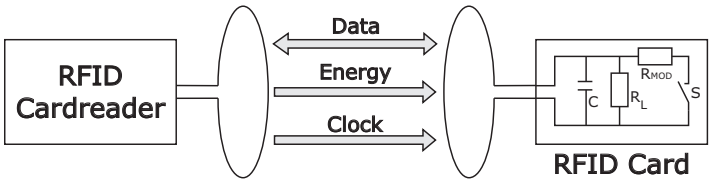


Fig. 1. RFID-Transmission

automotive immobilizer systems turned out to have an insufficient key length once the proprietary cryptographic algorithm is known ([3]). Other commercial products that are widely used for applications such as ticketing are also equipped with a proprietary cipher which makes an independent security analysis hard.

In many applications (e.g., ticketing) the RFID transponder is handed over to the end user. Therefore, an end user is potentially an adversary who aims to write tickets at the non-volatile memory of the RFID transponder without paying. In this case, the adversary has physical access to the cryptographic module and can apply a full bunch of implementation attacks. Among them side channel attacks are promising as they act in a passive way and do not lead to a physical destruction of the RFID transponder.

It was first shown by [6] that EM emanation of a smart card can be used as side channel information. The terms "Simple ElectroMagnetic Analysis" (SEMA) and "Differential ElectroMagnetic Analysis" were introduced, in analogy to "Simple Power Analysis" and "Differential Power Analysis" in [7]. Other contributions follow as, e.g., [1].

In this work we present an EM analysis at a cryptographic RFID transponder. For our analysis we use a Mifare DESFire smart card [8]. To our knowledge, this is the first work in public literature that presents DEMA results at a cryptographic RFID transponder.

2 EM-Analysis

Our focus is to measure the EM-emanation in the near field of the chip as the EM-leakage of a RFID tag is very weak. It will not be promising to measure the radiated EM-wave in the far field also because of surrounding noise.

2.1 Antennas

We used a probe for the electric field, which is mainly built of a small copper-plate with the size of 4x4 mm. This is nearly the same size as the DESFire chip.

Antennas that couple to the magnetic field are mainly coils. During previous work some antennas of this type have been built. An isolated copper wire with a

thickness of 0.2 mm has been turned around a small cylinder with a diameter of 2 mm used as a shaft. Afterwards the shaft was removed and so the air-core coil was constructed. The two ends of the coils have been sold to a copper core of a RG-58 coaxial-cable with an impedance of 50Ω. We tried out different coils with the same wire and the same inner diameter, but with different winding numbers from 40 up to 800.

In this work we used the RF U-2 near field probe, from the company *LANGER EMV-Technik GmbH, Bannewitz, Germany*[1]. This probe is composed of a small coil in a plastic mount attached to a calibrated amplifier.

3 Authentication Protocol

The authentication procedure is implemented in the card reader and it is not known in the public. So our first goal was to analyze the authentication procedure.

For communication between the card reader and the DESFire card a standardized RFID protocol is used. The reader uses a modified Miller code to send data to the card. The card responds using a load modulation according to the Manchester code. For the details of the communication protocol we refer to [5].

A card reader device which has built-in support for the DESFire card was bought together with some Mifare DESFire cards[2] from ACG[3].

Some authentication sequences have been performed with different keys that can be loaded into the reader and the card.

The antenna signal from the reader was recorded using a digital scope. The recorded data was demodulated in software and so we could extract the plain communication data between reader and DESFire card.

Using this data we were able to discover the authentication procedure for different keys in reader k_R and card k_C as far as it is relevant for DEMA testing. In the following procedures $Enc(x;k)$ denotes an encryption of data x with a single DES (or triple DES) key k. Accordingly $Dec(x;k)$ denotes a decryption.

1. To initiate the procedure the card reader sends a request to the card.
2. The card generates a 64-bit random number R_C and encrypts this number with its own secret key k_C. This encrypted block 0; $B_0 = \mathsf{Enc}(R_C; k_C)$ is sent to the reader.
3. The reader generates 64-bit random data R_R and decrypts this with its secret key which becomes block 1; $B_1 = \mathsf{Dec}(R_R; k_R)$.
   Afterwards, the reader decrypts the 8 byte block 0 received from the card with its key k_R. This results is $R_{C1} = \mathsf{Dec}(B_0; k_R)$ (which becomes R_C if $k_C = k_R$). Now the reader rotates R_{C1} by 8 bits to the left. The result of this operation is XORed with B_1 and afterwards decrypted with k_R. This

[1] http://www.langer-emv.de
[2] for further information refer to [8]
[3] ACG Identification Technologies GmbH, Walluf, http://www.acg.de

result becomes B_2. Finally, the reader sends these two 64-bit blocks back to the card:

$$B_1 : \mathsf{Dec}(R_R; k_R)$$
$$B_2 : \mathsf{Dec}(\mathsf{RotLeft}(\mathsf{Dec}(B_0; k_R), 8) \oplus B_1; k_R)$$

4. Now the card examines if the card reader works with the same key by performing the following computation:

$$R'_C = \mathsf{RotRight}((\mathsf{Enc}(B_2; k_C) \oplus B_1), 8)$$

   If this results in R_C, the card has successfully verified the cryptogram. If $R'_C \neq R_C$ then the card sends a failure response to the reader.
5. If the keys on the card and the reader have been chosen different, at this point the communication terminates.

This authentication procedure is summarized in the following Figure.

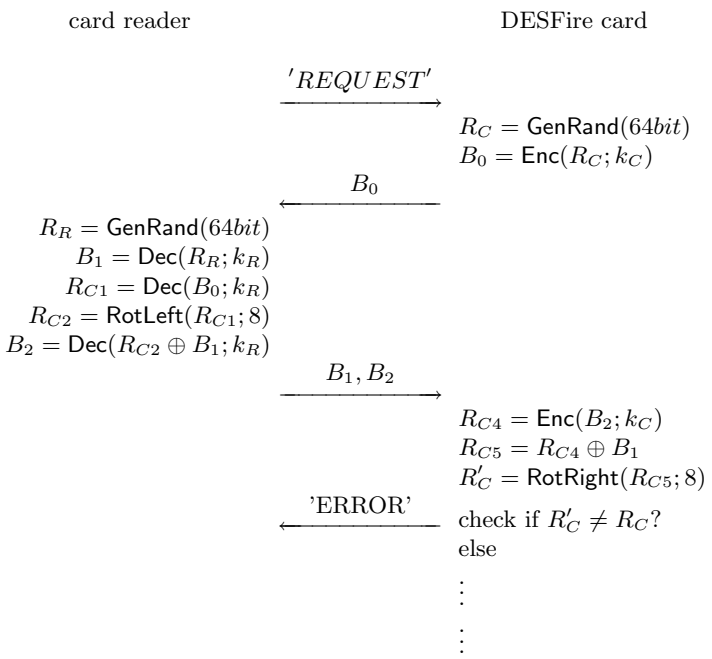


Fig. 2. Authentication Protocol

Note that the card reader does only perform DES decryption whereas the card only computes DES encryption.

For a differential side channel analysis either the plaintext or the ciphertext must be known. We chose a 'real-life' scenario with different keys in the RFID card and reader. For differential analysis we recorded B_1 and B_2 that are sent

to the card. We treat these data as plaintext data for the Triple-DES encryption that is performed by the card. This Triple-DES encryption is targeted by DEMA.

4 Experimental Analysis

Our previous experiments in EM analysis showed that for optimal results the antenna must be placed exactly on top of the chip. As the chip is embedded in the smart card it is not possible to see where the chip is located. The antenna and the chip are sealed inside the card material. The card material is not transparent to strong light neither the chip can be palpated.

4.1 X-Ray Analysis

To analyze the physical arrangement of the DESFire card, we made an X-Ray photograph of the card.

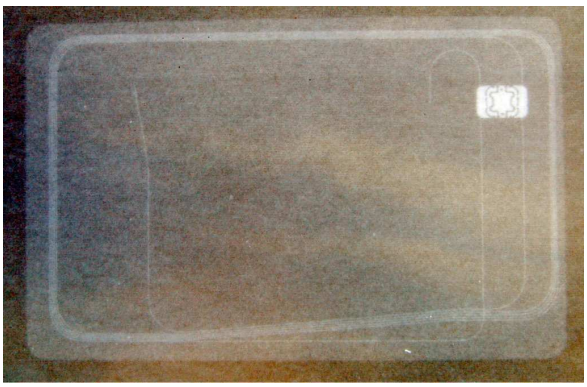


Fig. 3. X-Ray Photograph of a DESFire card

So we localized the position of the chip and the geometry of the antenna in the card and it was possible to place the antenna exactly on top of the DESFire chip, to receive the EM emanation of the DESFire chip.

The first EM experiments showed that the EM field, which is generated from the reader dominates in the measurements. The comparative small EM emanation from the DESFire chip is suspended.

4.2 Dissolving the Card and Separating Antenna and Card

To separate the reader's EM field from the card the DESFire chip must be isolated from the cards antenna. To access the chip we dissolved the card using Trichloroethylene $C_2\,H\,Cl_3$ at a temperature of 100° C. Trichloroethylene dissolves the outer layers from the card and the chip can be removed without damage.

Then we built an antenna according to the geometry of the antenna that is visible on the X-Ray of the DESFire card. Additionally we elongated the cable to connect the chip by a distance of 30 cm. After soldering the chip to the antenna, the chip was placed away from the reader's field, so that the emitted field does not significantly disturb the measurements anymore.

Figure 4 shows the antenna set-up used for EM analysis.

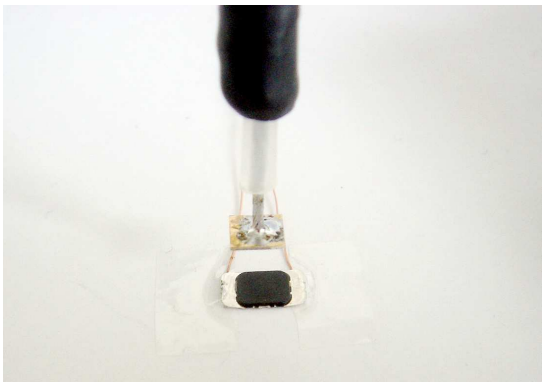


Fig. 4. Near field probe (cu-plate) on top of the chip during operation.

This results in two benefits: first it is possible to position the antenna more accurate at the chip, second the measurement antennas can be placed closer to the chip and so the signal amplitude increases, which results in a better signal-to-noise ratio (SNR).

4.3 EM Analysis

We recorded up to ten thousand measurements at a sampling rate of 1 GHz using a digital scope and the RF U-2 near field probe. We observed two distinct execution times: one is about 390 μs, the other is about 460 μs. The structure of the EM emanation turned out to be very uniform (see Figure 5).

The correlation method that is described in [2] was used to test for DEMA leakage. Until now, DEMA was not successful to reveal cryptographic keys.

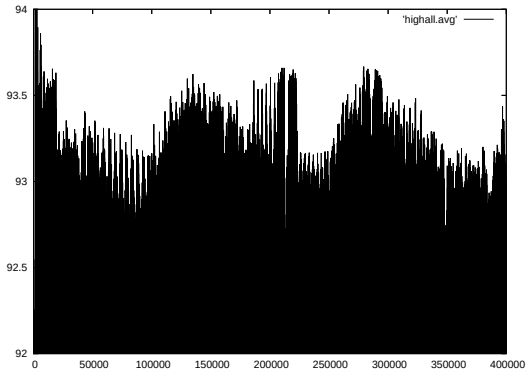


Fig. 5. Mean EM emanation focusing at the maximum amplitudes.

5 Further Directions

5.1 Custom RFID-Reader

In the current measurement setup it is very uncomfortable that we have to examine the reader's antenna signal at every single measurement to get the plaintext. Also it is not possible for us to choose the plaintext.

Future work will be to build a custom RFID-Reader, which can be fully controlled by the measurement equipment, and so, e.g., chosen plaintext attacks can be performed.

5.2 Analogous Filtering

Using an analogous filter it might be feasible to improve the signal to noise ratio so that side channel leakage might be detected that is currently suppressed by the clock signal.

5.3 (Semi) Invasive Analysis

As we accessed the chip in the DESFire card, it would be interesting to perform EM-measurements more localized, e.g., directly on top of the passivation of the chip after removing the package.

6 Conclusion

In this work we present first results of EM side channel analysis at a cryptographic contactless smart card. As preparation step for DEMA, the authentication procedure of the DESFire card was partially revealed. Finally, we give further directions for future work.

References

1. Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM Side-Channel(s). In B. S. Kaliski, Ç Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *LNCS*, pages 29–45. Springer-Verlag, 2003.
2. M. Aigner and E. Oswald. Power Analysis Tutorial. Available at www.iaik.tu-graz.ac.at/aboutus/people/oswald/papers/dpa_tutorial.pdf.
3. Steve Bono, Matthew Green, Adam Stubblefield, Ari Juels, Avi Rubin, and Michael Szydlo. Security Analaysis of a Cryptographically-Enabled RFID Device, available at: http://www.rfidanalysis.org/DSTbreak.pdf.
4. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstofer. Strong Authentication for RFID Systems using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editor, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume LNCS 3156, pages 357–370. Springer-Verlag, 2004.
5. K. Finkenzeller. *RFID-Handbuch.* Hanser Fachbuchverlag, Third edition, October 2002.
6. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç. K. Koç, D. Naccache and C. Paar, editor, *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2001*, volume LNCS 2162, pages 251–261, Paris, France, May 2001. Springer-Verlag.
7. P. Kocher, J. Jaffe, and B. Jun. Introduction to Differential Power Analysis and Related Attacks, 1998. Manuscript, Cryptography Research, Inc., Available at www.cryptography.com/dpa/technical.
8. Philips Semiconductors. Short Form Specification Mifare DESFire, Contactless Multi-Application IC with DES and 3DES security MF3 ICD 40, April 2004. Available at http://www.semiconductors.philips.com/acrobat/other/identification/SFS075530.pdf.
9. Sanjay E. Sarma, Stephen A. Weis, and Daniel W. Engels. RFID Systems and Security and Privacy Implications. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 454–469. Springer, 2002.

Design of Instruction Set Extensions and Functional Units for Energy-Efficient Public-Key Cryptography*

Johann Großschädl and Stefan Tillich
Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
{Johann.Groszschaedl,Stefan.Tillich}@iaik.at

Abstract

Various public-key cryptosystems require to perform arithmetic operations on very long integers or on binary polynomials of very high degree. In recent years, a number of so-called unified multipliers (i.e. multipliers that integrate both types of operands into a single datapath) have been proposed. This paper presents the design of a radix-2 and a radix-4 version of a unified (16×16)-bit multiplier with a 40-bit accumulator. The unified multiply/accumulate (MAC) unit can be used either as arithmetic core of a cryptographic co-processor or as a functional unit in an application-specific processor. A full-custom layout of both the radix-2 and the radix-4 multiplier was implemented on basis of a conventional array architecture. Simulations of netlists with extracted parasitics showed a power saving of 22% and an energy-delay advantage of 48% for the radix-4 multiplier compared to the radix-2 version. The multiplication of binary polynomials consumes about 39% less power than integer multiplication.

1. Introduction

Public-key cryptography is essential to ensure security and privacy of communication over the Internet. Virtually all modern security protocols, such as the Secure Socket Layer (SSL) protocol, rely on the concepts of public-key cryptography as introduced by Diffie and Hellman in 1976 [6]. However, the "traditional" public-key cryptosystems like RSA, DSA, or Diffie-Hellman are highly computation-intensive applications and thus difficult to implement on constrained devices like smart cards. In the past, embedded systems with poor processing capabilities used dedicated hardware (i.e. co-processors) to offload the heavy computational demands of cryptographic algorithms from the host processor. An alternative approach is to customize the processor's instruction set and micro-architecture towards the performance-critical operations carried out in cryptography [13]. Due to the increasing importance of cryptographic workloads, a number of micro-processor vendors extended their instruction set architectures by special instructions designed for efficient cryptographic processing; well-known examples are the ARM SecurCore architecture [1] and the SmartMIPS [23].

*The research described in this paper was supported by the Austrian Science Fund (FWF) under grant number P16952-N04 "Instruction Set Extensions for Public-Key Cryptography" and in part by the European Commission through the IST Programme under contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

In recent years, *elliptic curve* (EC) cryptosystems became very popular because they allow to achieve a good balance between security and performance [4]. Compared to their "traditional" counterparts like RSA, EC systems can use significantly shorter keys in order to guarantee a certain level of security. A widely accepted rule of thumb states that the security of a properly constructed 160-bit EC system is comparable to a 1024-bit RSA system. This makes EC cryptography especially attractive for mobile and wireless devices which are typically limited in terms of computational resources and/or network connectivity. During the last five years, standards for EC cryptography have been adopted by a number of standardization bodies around the world, including the ANSI, NIST, ISO, and IEEE. There is no doubt within the security community that EC cryptography is well on its way of becoming the de-facto standard for public-key services on mobile devices [17].

From a mathematical point of view, EC cryptosystems operate in a group of points on an elliptic curve defined over a *finite field* [19]. The main operation of EC cryptosystems is a so-called point multiplication, which is a computation of the form $k \cdot P$ where k is an integer and P is a point on the elliptic curve [4]. There exist a number of different algorithms for computing $k \cdot P$, but in the end they all require to carry out arithmetic operations (addition, multiplication and inversion) in the underlying finite field. Efficient implementation of the field arithmetic is therefore the key to high performance and low power consumption.

Several standards bodies, including the National Institute of Standards and Technology, recommend to use either a *prime field* GF(p) or a *binary extension field* GF(2^m) for the implementation of EC cryptography [25]. The elements of a prime field GF(p) are nothing else than the integers $0, 1, \ldots, p-1$, whereas the elements of a binary extension field GF(2^m) are usually represented by *binary polynomials* of degree up to $m-1$. Both types of operands (integers and binary polynomials) have in common that a multiplication can be accomplished by generation and addition of partial products. This has motivated the design of *unified multipliers*, i.e. multipliers that use the same datapath for both integers and binary polynomials [28, 9, 27, 12]. Having a unified multiplier instead of two separate multipliers can result in significant area savings when an application needs to deal with both types of operands.

This paper presents an analysis and comparison of two different implementations of a unified $(16 \times 16 + 40)$-bit multiply/accumulate (MAC) unit for integers and binary polynomials. Our first implementation contains a unified radix-2 multiplier, whereas the second variant is based on a radix-4 design. The unified MAC units can either be used as arithmetic core of a cryptographic co-processor [27] or as a functional unit in an application-specific processor [12]. One of the most important application domains for unified multipliers are cryptographic smart cards—devices where stringent limitations apply to silicon area as well as power consumption[1]. Therefore, we realized the unified multipliers in terms of an array architecture and produced the layout following a full-custom design methodology. Our simulation results clearly demonstrate that the unified radix-4 multiplier is superior to the radix-2 variant with respect to silicon area, delay, and power consumption. We also show that both the radix2 and the radix-4 multiplier consume less power when multiplying binary polynomials instead of integers.

The remainder of this paper is organized as follows. In the next section we summarize a number of basic facts about finite fields and review previous work on the design of unified multipliers. The Sections 3 and 4 describe several implementation details of our unified radix-2 and radix-4 MAC unit, respectively. Section 5 discusses the design flow and presents simulation results. The paper finishes with conclusions in Section 6.

[1] On the other hand, the delay of the multiplier is no major concern since smart cards have a low clock frequency (typically less than 15 MHz).

2. Background and related work

A *finite field* (or *Galois field*) is a finite set of elements on which two operations, generally called addition and multiplication, are defined such that the field axioms hold [19]. The elements of a *prime field* GF(p) are the residue classes modulo p (typically represented by the integers $0, 1, \ldots, p-1$) and the field arithmetic is nothing else than the conventional modular arithmetic, i.e. addition and multiplication modulo p. On the other hand, the elements of a *binary extension field* GF(2^m) can be represented by binary polynomials of degree up to $m-1$, i.e.

$$a(t) = \sum_{i=0}^{m-1} a_i \cdot t^i = a_{m-1} \cdot t^{m-1} + \cdots + a_1 \cdot t + a_0 \tag{1}$$

with coefficients a_i from the subfield $\mathrm{GF}(2) = \{0, 1\}$. For example, the finite field GF(2^3) has eight elements, which are the following binary polynomials.

$$\mathrm{GF}(2^3) = \{0, 1, t, t+1, t^2, t^2+1, t^2+t, t^2+t+1\} \tag{2}$$

Any element $a(t) \in \mathrm{GF}(2^m)$ can be written as a bit-string consisting of m coefficients, i.e. $a(t) = (a_{m-1}, \ldots, a_1, a_0)$. The addition of field elements is simply a logical XOR operation of the corresponding coefficients. Multiplication in GF(2^m) is performed modulo an *irreducible polynomial* $p(t)$ of degree m. This means that a multiplication in GF(2^m) requires to multiply two binary polynomials together (yielding a product-polynomial of degree up to $2m-2$), followed by a reduction of the product-polynomial modulo $p(t)$ [14].

There exist some basic similarities between prime fields and binary extension fields. Firstly, the elements of either type of field can be represented by a bit-string. Secondly, the multiplication in both GF(p) and GF(2^m) implies a modular reduction. Thirdly, the multiplication of both integers and binary polynomials can be accomplished by generation and addition of partial products. These similarities facilitate the design of a unified multiplier.

2.1. Design approaches for unified multipliers

The term *unification*, in general, can be described as the idea of identifying or creating similarities in order to make use of common structures. Prime fields and binary extension fields have structural similarities which allow to employ the same technique for the multiplication of field elements. Unification reduces the overall hardware cost when a multiplier for both integers and binary polynomials is needed. For instance, the area of the unified multiplier reported in [28] is only marginally larger than the area of a conventional integer multiplier datapath.

When classifying previous work on unified multipliers, three principal design approaches can be identified. The first approach, originally introduced in [8], uses a special *wiring methodology* in order to integrate the multiplication of binary polynomials into a tree multiplier. In the most basic case, a tree multiplier consists of full and half adders and uses a redundant representation (e.g. carry-save form) for the summation of partial products. The sum output of a full adder provides the logical XOR of its inputs, which is exactly the functionality needed for the addition of binary polynomials. This allows to design a unified multiplier by modifying the wiring of the partial product summation tree so that no carry bits are added with partial products before the final adder. In other words, the architecture presented in [8] implements two separate sub-trees for the addition of sum and carry vectors; one sub-tree consists of the adders' XOR logic, whilst the second sub-tree

comprises the other gates of the adder cells. The sum input to the final adder represents the XOR of all partial products, which is the result of the polynomial multiplication. A unified multiplier designed along these lines has the same delay as the original tree multiplier and does not increase the hardware cost (except of a few multiplexers). The authors of reference [27] employed similar ideas to design a unified multiplier for EC cryptography.

A completely different approach for combining the multiplication of integers and binary polynomials into a single datapath was proposed by Drescher *et al* [7]. They investigated the implementation of arithmetic in GF(2^m), $m \leq 8$, on a DSP datapath for signal coding, in particular BCH codes. However, unlike the approach taken in [8], Drescher *et al* did not modify the wiring within the partial product summation tree, but modified the half and full adder cells by adding some extra logic that allows to set the carry output to zero. The sum output of a full adder represents the modulo-2 sum (i.e. XOR) of the corresponding input values. Thus, suppressing all carries of the half and full adder cells enables a radix-2 integer multiplier to perform a multiplication of binary polynomials. Savaş *et al* used this concept to implement a unified multiplier for EC cryptography [28]. They presented the design of a so-called *dual-field adder* (DFA), which is nothing else than a conventional full adder with the capability to set the carry output to zero. A properly designed DFA has no higher delay than a conventional full adder and consumes significantly less power in polynomial mode than in integer mode [11]. This approach for unification is very attractive for array multipliers because it preserves the regularity of the architecture. On the other hand, DFAs increase the area compared to a standard multiplier and require to broadcast an additional control signal.

A third approach for combining integer and polynomial addition was proposed by Au and Burgess [2]. They implemented the addition of integers by using a *redundant binary representation* based on the digit set $\{0,1,2\}$, whereby the digits have the following encoding: $0 \sim (0,0)$, $1 \sim (0,1)$, and $2 \sim (1,0)$. A fourth digit, denoted 1^* and encoded as (1,1), represents the 1 in GF(2), which means that the addition of binary polynomials is performed with the digit set $\{0,1^*\}$. Taking advantage of this special encoding, Au and Burgess proposed a unified (4:2) adder for integers and binary polynomials. The (4:2) adder has a critical path of only three XOR gates and does not need extra control logic to suppress the carries for polynomial addition. However, a drawback of Au and Burgess' approach is the relatively high power consumption in polynomial mode, which is due to the use of the redundant binary representation [12].

2.2. Multiplication in finite fields of high order

The finite fields used in EC cryptography have a typical order of between 2^{160} and 2^{200}, which means that the bit-length of a field element exceeds the 16-bit word-size of our unified multiplier by an order of magnitude. Obviously, this raises the question of how a (16×16)-bit multiplier can be utilized to perform a multiplication of two 160-bit integers (or binary polynomials of such size). The mismatch between the operand length and the processor's word-size is generally resolved by representing the long integers as arrays of single-precision (e.g. 16-bit) words and using special algorithms for performing arithmetic operations on these arrays with help of the instructions provided by the processor.

Algorithm 1 illustrates the so-called product-scanning technique for *multiple-precision multiplication* of integers [14]. The operation carried out in the inner loops (line 4 and 11) is a multiply/accumulate (MAC) operation—two single-precision (w-bit) words are multiplied and the $2w$-bit product is added to a running sum. This algorithm requires a MAC unit with a "wide" accumulator (e.g. 8 guard bits) in order to prevent overflows and loss of precision. In general,

Algorithm 1 Multiple-precision multiplication

Input: Two integers $A = (A_{d-1}, \ldots, A_0)$ and $B = (B_{d-1}, \ldots, B_0)$, represented by arrays of d words, each consisting of w bits.
Output: Product $Z = A \cdot B = (Z_{2d-1}, \ldots, Z_0)$.

1: $S \leftarrow 0$
2: **for** i from 0 by 1 to $d-1$ **do**
3: **for** j from 0 by 1 to i **do**
4: $S \leftarrow S + A_j \times B_{i-j}$
5: **end for**
6: $Z_i \leftarrow S \bmod 2^w$ $\{ Z_i$ is assigned the w LSB of $S \}$
7: $S \leftarrow \lfloor S/2^w \rfloor$ $\{ S$ is shifted w bits to the right $\}$
8: **end for**
9: **for** i from d by 1 to $2d-2$ **do**
10: **for** j from $i-d+1$ by 1 to $d-1$ **do**
11: $S \leftarrow S + A_j \times B_{i-j}$
12: **end for**
13: $Z_i \leftarrow S \bmod 2^w$ $\{ Z_i$ is assigned the w LSB of $S \}$
14: $S \leftarrow \lfloor S/2^w \rfloor$ $\{ S$ is shifted w bits to the right $\}$
15: **end for**
16: $Z_{2d-1} \leftarrow S \bmod 2^w$
17: **return** $Z = (Z_{2d-1}, \ldots, Z_0)$

Algorithm 1 performs d^2 MAC operations for the multiplication of two d-word integers. We point out that the algorithm also works for binary polynomials, provided that we replace the integer MAC operation by a MAC operation on binary polynomials (see [11] for further details). This proves that the concept of unification is not only applicable to multiplier datapaths but also to arithmetic algorithms (algorithmic unification).

The availability of a unified MAC unit allows very fast software implementation of multiple-precision multiplication (for both integers and binary polynomials). Furthermore, the same concept can also be applied to the design of a cryptographic co-processor where a unified MAC unit operates as arithmetic core, such as proposed in [27].

Modular reduction: A multiplication in GF(p) or GF(2^m) is generally realized in two steps: multiplication of long integers (or binary polynomials of high degree) and reduction of the product modulo the prime p (or the irreducible polynomial $p(t)$) to obtain the final result. The most widely used generic algorithm for performing a reduction modulo a prime p is due to Montgomery [24]. Reference [15] describes a number of methods which combine multiplication and Montgomery reduction into a single operation. One of these methods, the so-called Finely Integrated Operand Scanning (FIPS) method, performs MAC operations in its inner loop, similar to Algorithm 1. It was shown in [16] that Montgomery's algorithm can also be applied to the reduction of a binary polynomial modulo an irreducible polynomial $p(t)$. As mentioned before, Montgomery's algorithm is a generic algorithm that works for any prime or any irreducible polynomial.

However, it is common practice in EC cryptography to use special primes or special irreducible polynomials for which much faster reduction techniques exist. For instance, the NIST specified a set of five prime fields defined by so-called generalized Mersenne (GM) primes and a set of five binary extension fields defined by irreducible polynomials with few non-zero coefficients, e.g. trinomials or pentanomials [25]. A typical example for a GM prime is $p = 2^{192} - 2^{64} - 1$, and a reduction modulo this prime can be accomplished with a few multiple-precision additions, which is a linear-time operation and thus much faster than Montgomery's algorithm. The reduction of a

binary polynomial modulo an irreducible trinomial such as $p(t) = t^{233} + t^{74} + 1$ requires only a few shift and XOR operations. These special reduction techniques are well documented in a number of papers and textbooks about EC cryptography, e.g. in [14].

3. Unified radix-2 multiplier

There exist a variety of implementation options for multipliers, which are typically determined by the requirements (e.g. delay, power consumption, silicon area) and/or the design methodology (e.g. standard cells versus full-custom design). Embedded systems like smart cards impose some stringent restrictions on power consumption and silicon area. In this context, the decision between an array-like and a tree-like architecture is a major concern.

Array versus tree: Array multipliers are known to provide high regularity and locality at the silicon level for the expense of a critical path delay that scales linearly with the word-size. On the other hand, tree multipliers have an irregular structure but can work at much higher frequencies due to a significantly shorter (i.e. logarithmic-length) critical path. Although the speed characteristics of array and tree multipliers are obvious and well understood, the situation is not so clear regarding power consumption.

At a first glance, one is tempted to attribute array multipliers a rather high power consumption because of the long signal paths which may cause lots of gate-output transitions to propagate through the array. It was shown in [5] and [3] that array multipliers consume significantly more power than tree multipliers when the impact of wiring is completely ignored. However, since CMOS process technology improves and transistor geometries become smaller and smaller, the parasitic capacitances of interconnect wires dominate over the transistor capacitances. Tree multipliers suffer from long interconnect wires with high capacitive load, which is due to their irregular structure. Moreover, signal paths of varying lengths lead to signal skew and an increased number of glitches or spurious transitions. Meier *et al* [21] have demonstrated that wiring has a major impact on the power consumption of tree multipliers, and that the difference between array and tree multipliers is very small when wiring effects are considered.

An array-like architecture is definitely to prefer over a tree-like architecture when one employs a full-custom design methodology instead of logic synthesis with automatic place and route. It is, of course, also possible to produce a full-custom layout of a tree multiplier, but this is a tedious task because of the irregular structure. Array architectures feature a high degree of regularity and mainly local interconnect, which makes them easy to design and lay out. Taking these aspects into account, we opted for the array architecture in combination with a full-custom design methodology.

Full-custom cell library: Bisdounis *et al* [3] compared eight circuit techniques to determine their suitability for the design of low-power adders and multipliers. Complementary static CMOS logic showed good results in this comparison and belonged to the "Top 3" logic styles with respect to power consumption. Furthermore, static CMOS logic is fast, easy to design, immune to noise, and robust against voltage scaling and transistor downsizing. All these excellent properties make complementary static CMOS the circuit technique of choice for the implementation of a low-power/small-area multiplier [31].

We developed a full-custom cell library containing all standard logic gates with two and three inputs. In addition, our cell library also includes a two-input XOR and XNOR implemented on basis of the "classical" transmission-gate XOR/XNOR circuit with a tailing inverter to increase the drive strength (see Figure 4 in [18]). Transmission gates were also used for the implementation of

an inverting 2:1 multiplexer. The XOR and XNOR gate comprise of eight transistors each, whereas the multiplexer has a transistor count of six.

3.1. Generation of partial products

A multiplication of unsigned 16-bit integers A, B can be carried out by generation and addition of partial products:

$$Z = A \cdot B = A \cdot \sum_{i=0}^{15} b_i \cdot 2^i = \sum_{i=0}^{15} A \cdot b_i \cdot 2^i \tag{3}$$

In the binary case (i.e. radix-2 representation), the generation of a partial product $A \cdot b_i$ is a simple logical AND operation between the multiplier bit b_i and the 16 bits of the multiplicand A. These partial products have to be summed up according to their weight 2^i (i.e. in the appropriate relative position) to get the correct result $Z = A \cdot B$.

The multiplication of binary polynomials can employ the same technique of generation and addition of partial products. For example, a multiplication of two binary polynomials $A(t), B(t)$ of degree 15 is performed as follows.

$$Z(t) = A(t) \cdot B(t) = A(t) \cdot \sum_{i=0}^{15} b_i \cdot t^i = \sum_{i=0}^{15} A(t) \cdot b_i \cdot t^i \tag{4}$$

All coefficients b_i of $B(t)$ are from GF(2) $= \{0,1\}$ when using a conventional (i.e. radix-t) representation, which means that the generation of a partial product $A(t) \cdot b_i$ is a logical AND operation between b_i and the coefficients of $A(t)$ [29]. The multiplication of a partial product $A(t) \cdot b_i$ by a power of the form t^i is nothing else than a left-shift by i bit positions ("Shift-and-XOR" method). A partial product generator (PPG) for a unified radix-2/radix-t multiplier consists of a row of AND gates. In other words, we can use exactly the same hardware (namely AND gates) to generate partial products for both integer and polynomial multiplication.

3.2. Addition of partial products

A conventional implementation of an array multiplier is built of rows of (3,2) counters (full adders) and (2,2) counters (half adders) to reduce the partial products to a sum and carry vector. The addition of this sum and carry vector is the final step for completing a multiplication and calls for a fast carry-propagation adder (often referred to as "final adder"). In the radix-2 case, a typical $(w \times w)$-bit array multiplier consists of w^2 AND gates, $w-1$ half adder cells, $(w-1) \cdot (w-2)$ full adder cells, and a w-bit carry-propagation adder for the redundant-to-binary conversion of the upper part of the result (see [30] for details).

As mentioned in Section 2.1, there are three basic approaches for combining integer and polynomial multiplication into a single datapath. The first approach (i.e. the special wiring technique described in [8]) is applicable to tree multipliers, but would completely destroy the regularity of an array multiplier. On the other hand, the use of a redundant binary representation, as proposed in [2], has advantages for high-speed designs, but imposes significant power consumption in polynomial mode [12]. Therefore, we opted to implement the multiplier datapath with dual field adders as this approach complies best with our requirements for regular layout and low power consumption.

Figure 1 shows the dual-field adder (DFA) design from our previous work [11]. This design implements basically a standard full adder with some extra logic to set the carry output to 0. The

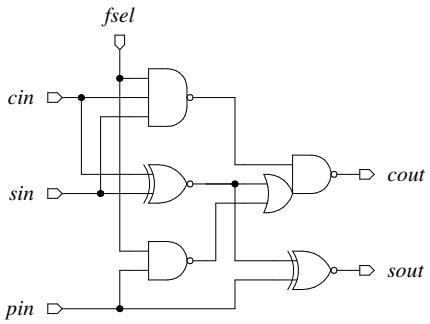


Figure 1. Dual-field adder (DFA)

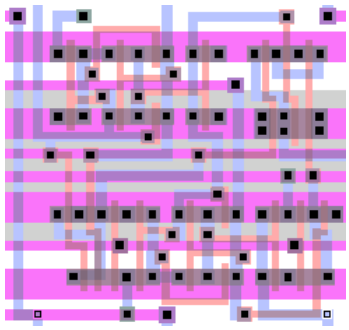


Figure 2. DFA full-custom layout

control signal *fsel* allows to switch between integer mode and polynomial mode. In integer mode ($fsel = 1$), the DFA works like a conventional full adder, i.e. both the sum and the carry output are active. On the other hand, when the DFA is operated in polynomial mode ($fsel = 0$), both NAND gates are 1, which forces the ORNAND gate (and consequently *cout*) to 0. The advantage of this design is that only the two XOR gates are active in polynomial mode; all other gates do not transition and hence they also do not consume power. Therefore, the DFA depicted in Figure 1 has a substantially lower power dissipation in polynomial mode than in integer mode. This is a major advantage over the DFA design presented in [28], which does not suppress unnecessary switchings in polynomial mode.

We implemented the DFA using the cells of our full-custom cell library, whereby the transistors of all gates have minimum size. The only exception are the PMOS transistors of the ORNAND gate, which were enlarged to ensure that the sum and carry output have similar drive strength and similar rise/fall times. Simulations of the DFA layout with extracted parasitics showed that the delay from the inputs *sin*, *cin* to the two outputs *sout* and *cout* is also almost identical. The use of minimum-size transistors in combination with delay balancing helps to reduce the impact of spurious transitions (glitches) on the total power consumption. Other advantages of the DFA are its relatively small area and a short critical path. The five gates shown in Figure 1 amount to 32 transistors altogether (on basis of our full-custom cell library), which is only four transistors more than the classical 28-transistor full adder design given in [30, p. 517]. The delay of a full adder is generally determined by the delay of the XOR gates, which means that our DFA has the same critical path delay as a conventional full adder. In summary, the DFA shown in Figure 1 is only marginally larger than a full adder and has the same propagation delay. The complete adder array of a unified (16×16)-bit multiplier has, in the radix-2 case, a critical path of 14 DFA cells and one dual-field half adder cell.

3.3. Accumulator and final adder

The adder-array reduces the partial products to a single sum and carry vector. An accumulator allows to add the result of the multiplication to a running sum. The accumulator of our unified multiplier is composed of DFAs and has a length of 40 bits (i.e. eight guard bits to prevent overflow). The outputs of the accumulator represent the result of the multiply/accumulate operation in redundant representation (carry-save form). Consequently, we have to convert the sum and carry vectors into a standard binary number to obtain the final result.

The redundant-to-binary conversion ("final addition") calls for a fast carry-propagation adder. An important aspect when designing a final adder is to consider the non-uniform signal arrival

profile of the sum and carry vector [26]. Array multipliers typically have a "staircase-like" signal arrival profile, which means that the lower half of the result arrives sequentially (bit by bit), whereas the upper part arrives simultaneously after passing through the full adder array. In order to reduce the overall delay of the MAC unit, we designed the final adder to match this special signal arrival profile. The final adder consists of a ripple-carry adder for the redundant-to-binary conversion of the sequentially-arriving bits, and a fast carry-select adder (CSA) for the upper bits of the result. We used ripple-carry adders of varying length for the sub-stages of the CSA in order to reduce the overall delay. A CSA composed of "small" ripple-carry adders features a high degree of regularity, which is clearly an advantage for full-custom design.

The multiplication of binary polynomials does not need a final adder, simply because all carry vectors are 0 in polynomial mode. Therefore, we forward the accumulator's sum output directly to the output of the multiplier when a polynomial multiplication is performed. The sum input to the final adder is set to 0 in polynomial mode to ensure that the final adder is completely inactive and does not dissipate power.

4. Unified radix-4 multiplier

High-radix multiplication schemes reduce the number of partial products compared to the conventional (i.e. radix-2 or binary) multiplication scheme, which shortens the critical path of an array multiplier since fewer partial products have to be summed up. A radix-4 multiplication of two unsigned 16-bit numbers can be performed according to the following equation.

$$Z = A \cdot B = A \cdot \sum_{k=0}^{7} b_k \cdot 4^k = \sum_{k=0}^{7} A \cdot b_k \cdot 4^k \quad \text{with} \quad b_k \in \{0,1,2,3\} \tag{5}$$

The conventional radix-4 representation of integers is based on the digit-set $\{0,1,2,3\}$. Consequently, any partial product $P_k = A \cdot b_k$ is either 0, A, $2A$, or $3A$, whereby the latter one is difficult to compile since summing $2A$ with A can generate a carry propagation. Therefore, most radix-4 multiplication schemes employ other digit sets, such as $\{-2,-1,0,1,2\}$ used in modified Booth recoding [20].

High-radix multiplication is also possible for binary polynomials and has been first described in [29, 22]. Similar to integers, the use of a high-radix representation reduces the number of partial products. The radix-t representation of binary polynomials is based on the coefficient set $\{0,1\}$, and hence it corresponds to the standard radix-2 representation of integers. On the other hand, the analogue of radix-4 integer representation is the radix-t^2 representation of binary polynomials. This representation uses the coefficient set $\{0, 1, t, t+1\}$ and allows to perform a multiplication of two binary polynomials $A(t)$, $B(t)$ of degree 15 as follows.

$$Z(t) = A(t) \cdot B(t) = A(t) \cdot \sum_{k=0}^{7} b_k(t) \cdot t^{2k} = \sum_{k=0}^{7} A(t) \cdot b_k(t) \cdot t^{2k} \quad \text{with} \quad b_k(t) \in \{0, 1, t, t+1\} \tag{6}$$

All coefficients $b_k(t)$ of a binary polynomial $B(t)$ in radix-t^2 representation are themselves binary polynomials, namely binary polynomials of degree at most one (i.e. 0, 1, t, or $t+1$). The corresponding partial product $P_k(t) = A(t) \cdot b_k(t)$ is either 0, $A(t)$, $t \cdot A(t)$, or $A(t) \oplus t \cdot A(t)$, all of which can be easily generated by 1-bit left-shifts and XOR operations. Therefore, a radix-t^2 multiplication can be efficiently performed with the standard coefficient set $\{0, 1, t, t+1\}$.

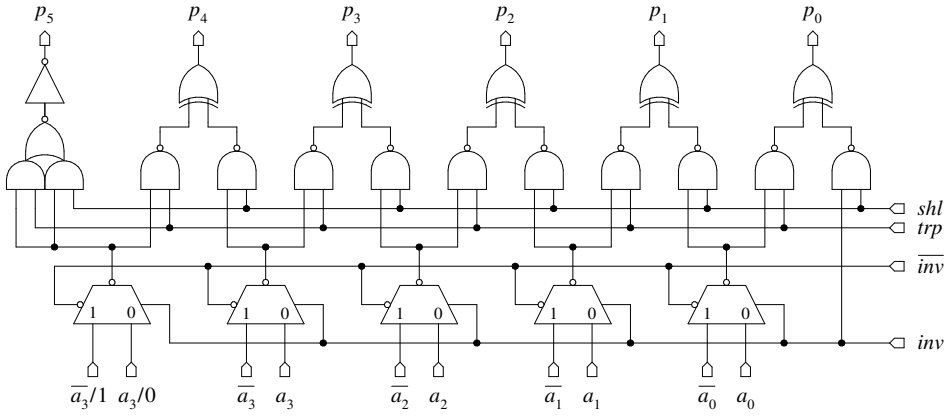


Figure 3. Unified radix-4 PPG for integers and radix-t^2 PPG for binary polynomials

4.1. Generation of partial products

Modified Booth recoding (also called Booth-MacSorley recoding [20]) is frequently used in array multipliers as it reduces the number partial products by a factor of two. Unfortunately, modified Booth recoding is not directly applicable to binary polynomials since it relies on a signed-digit representation with the digit set $\{-2,-1,0,1,2\}$. It is nonetheless possible to "unify" the high-radix multiplication of integers and binary polynomials, as we demonstrated in our previous work [10]. In the following, we summarize how a unified radix-4/radix-t^2 multiplier generates partial products in integer mode and polynomial mode.

Integer mode: Modified Booth recoding is generally performed within two steps: *Encoding* of the multiplier B and *Selection* (i.e. generation) of the partial products. The encoding step can be seen as a conversion where a radix-2 number B with digits b_i in $\{0,1\}$ is transformed into an equivalent radix-4 number $\widetilde{B}$ represented by digits $\widetilde{b}_k$ from the set $\{-2,-1,0,1,2\}$. When assuming that B is an unsigned 16-bit integer, the conversion can be carried out as follows.

$$\widetilde{B} = \sum_{k=0}^{8} \widetilde{b}_k \cdot 4^k \quad \text{with} \quad \widetilde{b}_k = -2 \cdot b_{2k+1} + b_{2k} + b_{2k-1} \text{ and } b_{17} = b_{16} = b_{-1} = 0 \tag{7}$$

The radix-4 digits $\widetilde{b}_k$ are obtained by partitioning the multiplier B into overlapping groups of three adjacent bits b_{2k+1}, b_{2k}, b_{2k-1} (for $k = 0,1,2,\ldots,8$) and calculating $-2 \cdot b_{2k+1} + b_{2k} + b_{2k-1}$ as shown in Equation (7). All digits $\widetilde{b}_k$ are available simultaneously since they can be calculated independently from each other and in parallel. A multiplication with the radix-4 number $\widetilde{B}$ instead of B reduces the number of partial products from 16 to 9 (or 8 when multiplying signed integers). The primary advantage of using the digit set $\{-2,-1,0,1,2\}$ is that the corresponding partial products $P_k \in \{-2A,-A,0,A,2A\}$ can be easily generated with shifts and bit-wise inversions. Booth multipliers generally use two's complement (TC) representation for negative numbers. Negative partial products, in TC form, are obtained by inverting the corresponding positive partial product (i.e. producing the one's complement) and adding a "1" at the least significant bit position. This "correction" is performed together with the addition of partial products in the adder array.

Figure 3 shows of a unified radix-4/radix-t^2 partial product generator (PPG) for integers and binary polynomials. This design is adapted from [10] and optimized for implementation with our full-custom cell library. The PPG is controlled by the three signals *inv* (invert), *trp* (transport), and

Multiplier bits			Integer mode ($fsel = 1$)					Polynomial mode ($fsel = 0$)				
b_{2k+1}	b_{2k}	b_{2k-1}	P_k	*inv*	*trp*	*shl*	"+1"	$P_k(t)$	*inv*	*trp*	*shl*	"+1"
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	$+A$	0	1	0	0	0	0	0	0	0
0	1	0	$+A$	0	1	0	0	$A(t)$	0	1	0	0
0	1	1	$+2A$	0	0	1	0	$A(t)$	0	1	0	0
1	0	0	$-2A$	1	0	1	1	$t \cdot A(t)$	0	0	1	0
1	0	1	$-A$	1	1	0	1	$t \cdot A(t)$	0	0	1	0
1	1	0	$-A$	1	1	0	1	$t \cdot A(t) \oplus A(t)$	0	1	1	0
1	1	1	0	1	0	0	0	$t \cdot A(t) \oplus A(t)$	0	1	1	0

Table 1. Radix-4 encoding of integers and radix-t^2 encoding of binary polynomials

shl (shift left). These signals depend on the multiplier bits b_{2k+1}, b_{2k}, and b_{2k-1} according to Equations (8)-(10), which can be easily derived from Table 1. In integer mode ($fsel = 1$), the generation of partial products is performed similar to the classical encoding scheme using *N/X1/X2* signals as specified in [30, p. 551]. Therefore, the unified PPG acts like a conventional radix-4 Booth-PPG when operated in integer mode. The logical equation for the "+1" needed in case of a negative partial product is also easily derived from Table 1.

$$inv = fsel \cdot b_{2k+1} \tag{8}$$

$$trp = fsel \cdot \left(\overline{b}_{2k} \cdot b_{2k-1} + b_{2k} \cdot \overline{b}_{2k-1}\right) + \overline{fsel} \cdot b_{2k} \tag{9}$$

$$shl = fsel \cdot \left(\overline{b}_{2k+1} \cdot b_{2k} \cdot b_{2k-1} + b_{2k+1} \cdot \overline{b}_{2k} \cdot \overline{b}_{2k-1}\right) + \overline{fsel} \cdot b_{2k+1} \tag{10}$$

The PPG shown in Figure 3 needs A (the multiplicand) and its inverse $\overline{A}$ as input, and the multiplexors select between A and $\overline{A}$, depending on the control signal *inv*. The AND/XOR combination performs a 1-bit left-shift operation when $shl = 1$, which is necessary for the generation of the partial products $2A$ and $-2A$. On the other hand, $trp = 1$ means that no left shift is performed and hence the resulting partial product is either A or $-A$.

Polynomial mode: Radix-t^2 multiplication of binary polynomials corresponds to radix-4 multiplication of integers. Given two binary polynomials $A(t)$, $B(t)$ in conventional (i.e. radix-t) representation, the radix-t^2 multiplication requires to scan two adjacent bits of $B(t)$ at a time in order to produce the corresponding partial product $P_k(t)$ as specified by Equation (6). Two adjacent bits of $B(t)$ can be interpreted as a binary polynomial of degree one, and depending on whether this polynomial is 0, 1, t, or $t+1$, the corresponding partial product $P_k(t)$ is either 0, $A(t)$, $t \cdot A(t)$, or $t \cdot A(t) \oplus A(t)$. The multiplication of $A(t)$ by t is nothing else than a 1-bit left shift of the coefficients of $A(t)$, which means that the partial-product generation for radix-t^2 multiplication is simply a matter of shift and XOR operations.

An important property of the *inv/trp/shl* scheme is that, in integer mode ($fsel = 1$), the two control signals *trp* and *shl* are never 1 at the same time (see Table 1), which allows us to use XOR gates to select between $\pm A$ and $\pm 2A$. Thanks to this property, the PPG shown in Figure 3 provides the necessary functionality to generate partial products for radix-t^2 multiplication. In polynomial mode ($fsel = 0$), the control signal *inv* is always 0 and the PPG is directly controlled by the coefficients of $B(t)$, i.e. $trp = b_{2k}$ and $shl = b_{2k+1}$. Therefore, the PPG depicted in Figure 3 is a unified radix-4 PPG for integers and binary polynomials[2].

[2]We denote the combined radix-4/radix-t^2 PPG as *"unified radix-4 PPG for integers and binary polynomials"* since the radix-t^2 multiplication of binary polynomials corresponds to the radix-4 multiplication of integers.

4.2. Addition of partial products

Employing radix-4 Booth recoding in a (16×16)-bit multiplier reduces the number of partial products from 16 to 9 (in the case of unsigned integers). On the other hand, radix-t^2 polynomial multiplication halves the number of partial products compared to the radix-t scheme (i.e. 8 partial products instead of 16). The adder array of a unified radix-4 multiplier for 16-bit operands is, in general, dimensioned to sum up 9 partial products. This means that the radix-4/radix-t^2 scheme reduces the overall number of DFAs by almost 50% in relation to the radix-2/radix-t scheme. However, the adder array of a unified radix-4 multiplier differs in the following aspects from the radix-2 version.

- The length of the partial products is 18 bits instead of 16 bits since they can be twice the multiplicand A and can have a negative value. The MSB of the partial product is its sign bit.
- The rules of two's complement arithmetic demand a sign extension, which increases both area and power consumption. Therefore, it is important to minimize the effects of sign extension.
- The PPG shown in Figure 3 performs merely an inversion when the generation of a negative partial product is required. Therefore, the adder array has to add a "1" at the least significant position of the partial product in order to get the correct two's complement representation.
- The partial products have to be summed up according to their "weight" (i.e. in the appropriate relative position) to obtain the correct result. For instance, the partial product P_k has four times the weight of P_{k-1}, which means that the offset between P_k and $P_{k-1}$is two bit positions.

Further details about the implementation of an adder array for radix-4/radix-t^2 multiplication can be found in our previous paper [12]. Although originally developed for standard-cell implementation, the architecture in [12] is also well suited for full-custom design since it features a high degree of regularity and mainly local interconnect. The adder array of a unified (16×16)-bit multiplier implemented according to [12] consists of 7 adder stages to sum up the 9 partial products. Each adder stage is composed of 18 DFAs, which amounts to 126 DFAs altogether. The first three partial products, P_0, P_1, and P_2, can be summed up by one adder stage. All remaining partial products require an additional adder stage, which results in the 7 adder stages mentioned before. The critical path of the radix-4 adder array (i.e. 7 DFA cells) is only one half of the radix-2 array.

4.3. Accumulator and final adder

The 40-bit accumulator of the unified radix-4 multiplier is identical to the one of the radix-2 version. On the other hand, the signal arrival profiles, and hence the final adders, differ slightly. The radix-4 multiplier also has a "staircase-like" profile, but the low-order bits (i.e. the 16 LSB) of the sum and carry vector arrive sequentially in two-bit blocks, instead of one bit at a time as in the radix-2 multiplier. Therefore, the final adder uses concatenated 2-bit ripple-carry adders (RCAs) for the redundant-to-binary conversion of these bits. The high-order bits of the sum and carry vector arrive simultaneously, as in the radix-2 version, and are summed up by a fast carry-select adder.

5. Results and discussion

We created a full-custom layout of both the unified radix-2 and radix-4 MAC unit using a standard 0.6 µm CMOS technology with two metal layers and one polysilicon layer. The transistor width of gates with ordinary (1x) drive strength is generally 1.5 µm; gates with 2x drive strength

Parameter	Radix-2 version	Radix-4 version
Transistor count	12,384	11,744
Delay (INT mode)	82.4 nsec	54.3 nsec
Avg. current (INT mode)	7.37 mA	5.75 mA
PDP (INT mode)	2.43 nJ	1.90 nJ
EDP (INT mode)	$200.2 \cdot 10^{-18}$ Js	$103.0 \cdot 10^{-18}$ Js
Delay (POLY mode)	66.8 nsec	38.0 nsec
Avg. current (POLY mode)	3.66 mA	3.49 mA
PDP (POLY mode)	1.21 nJ	1.15 nJ
EDP (POLY mode)	$80.7 \cdot 10^{-18}$ Js	$43.8 \cdot 10^{-18}$ Js

Table 2. Simulation results (f = 10 MHz, Vdd = 3.3 V)

are realized with PMOS transistors of width 3.0 µm and NMOS transistors of 1.5 µm, respectively. Both the radix-2 and the radix-4 multiplier have a regular structure with mainly local interconnect (i.e. short wires), which enables the use of minimum-size transistors in gates and drivers. However, some gates demand an increased drive strength in order to achieve equal output signal strength or to balance the delay and rise/fall times of signal slopes. Delay balancing guarantees synchronously arriving signals at the input of logic gates, thereby eliminating, or substantially reducing, the power dissipation caused by spurious transitions (glitches).

Design flow: We developed the layout of the multiply/accumulate unit following a hierarchical bottom-up approach. First, a *full-custom cell library* consisting of basic CMOS gates was implemented. Then, the *leaf cells* (DFA cells, PPG cells) were composed of these gates. We carefully optimized the layout of the leaf cells in order to reduce the intra-cell routing and to keep the overall silicon area as small as possible. The multiplier datapath is made up of an ensemble of (almost) identical *bit-slices*, which allows to exploit the regularity of the array architecture. This bit-slice organization has the advantage that the place-and-route problem needs to be solved only once since all slices have an (almost) uniform layout. Furthermore, the floorplanning and place-and-route of the multiplier datapath becomes very simple, especially when the bit-slices are designed to connect "by abutment". In this case, the generation of the multiplier datapath is simply a matter of replicating instances of bit-slices, whereby these instances are automatically connected. Not only the intra-slice but also the inter-slice wires are very short due to the regularity of the array structure (i.e. mainly nearest-neighbor connections). The correctness of the multiplier was verified by simulations with test vectors obtained from a functional model.

Simulation results: Table 2 summarizes the simulation results and main characteristics of the two unified $(16 \times 16 + 40)$-bit MAC units. Both the radix-2 and the radix-4 version consist of roughly 12,000 transistors, whereby the silicon area of the the radix-2 MAC is slightly larger. The radix-4 variant has a worst-case delay of 54.3 nsec (integer mode, 3.3 V supply voltage), which corresponds to a maximum clock frequency of 18.4 MHz. As expected, the radix-2 MAC is significantly slower, mainly due to the longer critical path in the adder array. Both versions are not very fast since we used a 0.6 µm CMOS process and because most devices are of minimum size. However, the performance is appropriate for typical smart card applications. Each of the two MAC units executes a polynomial multiplication much faster than an integer multiplication since a redundant-to-binary conversion of the result is not necessary in polynomial mode (i.e. the final adder is bypassed).

The average current dissipation of both the radix-2 and radix-4 implementation is also specified in Table 2. These results were obtained through simulation of netlists with extracted parasitics. In

general, the current drawn during a multiplication depends heavily on the input values. Therefore, we simulated 1,000 multiply/accumulate operations with independent, pseudo-random input patterns and measured the current consumption. The simulations were performed at a frequency of 10 MHz (i.e. new inputs were presented with a period of 100 nsec), which is a typical clock frequency for smart cards. Our results show that the radix-4 version dissipates, on average, 22% less power than the radix-2 version. This power saving is mainly due to the shorter adder array in the radix-4 multiplier, which reduces the number of glitches compared to the radix-2 version. In both cases, the multiplication of binary polynomials consumes significantly less power than integer multiplication (e.g. 39% in the radix-4 version). Polynomial multiplication needs no redundant representation and therefore causes less switching activities in the adder array (only the XOR gates of the DFAs are active) and no switchings at all in the final adder since it is bypassed (i.e. its inputs do not toggle).

Besides silicon area, delay, and power dissipation, also the *power-delay product* (PDP) and the *energy-delay product* (EDP) are generally accepted as comparison metrics for MAC units. When the leakage current is ignored, the PDP of a static CMOS multiplier can be interpreted as the average amount of *energy consumed per multiplication*. Therefore, the PDP is an important metric for battery-operated devices as it determines the battery-lifetime. Our simulations showed that the average amount of energy required to perform an integer multiplication is 2.43 nJ for the radix-2 multiplier, but only 1.90 nJ for the radix-4 variant, which represents a saving of 22%. The EDP differs by more than 48% in integer mode. Both the radix-2 and radix-4 version have much better energy characteristics in polynomial mode than in integer mode. This confirms that polynomial multiplication is more energy-efficient than integer multiplication.

6. Concluding remarks

In this paper, we analyzed and compared two implementations of a unified $(16 \times 16 + 40)$-bit MAC unit for integers and binary polynomials. The first implementation is based on a unified radix-2/radix-t multiplication scheme, while the second employs modified radix-4 Booth recoding for integers and radix-t^2 multiplication for binary polynomials. Both the radix-2 and radix-4 variant are based on a conventional array architecture and perform multiplications and multiply/accumulate operations in one clock cycle. The MAC unit can be integrated into a cryptographic co-processor or an application-specific processor to accelerate EC cryptography over GF(p) and GF(2^m).

Our simulation results show that the unified radix-4 MAC unit is superior to its radix-2 counterpart in terms of silicon area, delay, and power consumption. While the difference in silicon area is only marginal, the radix-4 version achieves a 34% improvement in delay and a power advantage of 22% compared to the radix-2 version. Moreover, the radix-4 MAC exhibits a significantly better EDP. Taking the tight energy budget of mobile devices like smart cards into account, the unified radix-4 multiplier represents a major improvement over the radix-2 designs proposed in [28, 9, 27]. We have also demonstrated that the multiplication of binary polynomials is more energy-efficient than integer multiplication since the latter generally uses a redundant representation which causes more signal transitions. This is an important result since typical EC cryptosystems require to carry out hundred-thousands, or even millions, of (16×16)-bit multiplications and MAC operations.

References

[1] ARM Limited. ARM SecurCore Solutions. Product brief, available online at `http://www.arm.com/aboutarm/4XAFLB/$File/SecurCores.pdf`, Feb. 2002.

[2] L.-S. Au and N. Burgess. A (4:2) adder for unified GF(p) and GF(2^n) Galois field multipliers. In *Conference Record of the 36th Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1619–1623. IEEE, 2002.

[3] L. Bisdounis, D. Gouvetas, and O. Koufopavlou. Circuit techniques for reducing power consumption in adders and multipliers. In *Designing CMOS Circuits for Low Power*, ch. 5, pp. 71–96. Kluwer Academic Publishers, 2002.

[4] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.

[5] T. K. Callaway and E. E. Swartzlander. Power-delay characteristics of CMOS multipliers. In *Proceedings of the 13th IEEE Symposium on Computer Arithmetic (ARITH '97)*, pp. 26–32. IEEE Computer Society Press, 1997.

[6] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, Nov. 1976.

[7] W. Drescher, K. Bachmann, and G. Fettweis. VLSI architecture for datapath integration of arithmetic over GF(2^m) on digital signal processors. In *Proceedings of the 22nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, vol. 1, pp. 631–634. IEEE, 1997.

[8] J. E. Garcia and M. J. Schulte. A combined 16-bit binary and dual Galois field multiplier. In *Proceedings of the 16th IEEE Workshop on Signal Processing Systems (SIPS 2002)*, pp. 63–68. IEEE, 2002.

[9] J. R. Goodman. *Energy Scalable Reconfigurable Cryptographic Hardware for Portable Applications*. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[10] J. Großschädl. A unified radix-4 partial product generator for integers and binary polynomials. In *Proceedings of the 35th IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, vol. III, pp. 567–570. IEEE, 2002.

[11] J. Großschädl and G.-A. Kamendje. Instruction set extension for fast elliptic curve cryptography over binary finite fields GF(2^m). In *Proceedings of the 14th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2003)*, pp. 455–468. IEEE Computer Society Press, 2003.

[12] J. Großschädl and G.-A. Kamendje. Low-power design of a functional unit for arithmetic in finite fields GF(p) and GF(2^m). In *Information Security Applications*, vol. 2908 of *Lecture Notes in Computer Science*, pp. 227–243. Springer Verlag, 2003.

[13] J. Großschädl and E. Savaş. Instruction set extensions for fast arithmetic in finite fields GF(p) and GF(2^m). In *Cryptographic Hardware and Embedded Systems — CHES 2004*, vol. 3156 of *Lecture Notes in Computer Science*, pp. 133–147. Springer Verlag, 2004.

[14] D. R. Hankerson, A. J. Menezes, and S. A. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Verlag, 2004.

[15] Ç. K. Koç, T. Acar, and B. S. Kaliski. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26–33, June 1996.

[16] Ç. K. Koç and T. Acar. Montgomery multiplication in GF(2^k). *Designs, Codes and Cryptography*, 14(1):57–69, Apr. 1998.

[17] K. Lauter. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Communications*, 11(1):62–67, Feb. 2004.

[18] H. Lee and G. E. Sobelman. New low-voltage circuits for XOR and XNOR. In *Proceedings of IEEE SouthEastCon '97*, pp. 225–229. IEEE, 1997.

[19] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, 1994.

[20] O. L. MacSorley. High-speed arithmetic in binary computers. *Proceedings of the IRE*, 49(1):67–91, Jan. 1961.

[21] P. C. Meier, R. A. Rutenbar, and L. R. Carley. Exploring multiplier architecture and layout for low power. In *Proceedings of the 18th IEEE Custom Integrated Circuits Conference (CICC '96)*, pp. 513–516. IEEE, 1996.

[22] M. C. Mekhallalati, A. S. Ashur, and M. K. Ibrahim. Novel radix finite field multiplier for GF(2^m). *Journal of VLSI Signal Processing*, 15(3):233–245, Mar. 1997.

[23] MIPS Technologies, Inc. SmartMIPS Architecture Smart Card Extensions. Product brief, available online at `http://www.mips.com/ProductCatalog/P_SmartMIPSASE/SmartMIPS.pdf`, Feb. 2001.

[24] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, Apr. 1985.

[25] National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). Federal Information Processing Standards (FIPS) Publication 186-2, Feb. 2000.

[26] V. G. Oklobdzija. Design and analysis of fast carry propagate adder under non-equal input signal arrival profile. In *Conference Record of the 28th Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1398–1401. IEEE, 1994.

[27] A. Satoh and K. Takano. A scalable dual-field elliptic curve cryptographic processor. *IEEE Transactions on Computers*, 52(4):449–460, Apr. 2003.

[28] E. Savaş, A. F. Tenca, and Ç. K. Koç. A scalable and unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$. In *Cryptographic Hardware and Embedded Systems — CHES 2000*, vol. 1965 of *Lecture Notes in Computer Science*, pp. 277–292. Springer Verlag, 2000.

[29] L. Song and K. K. Parhi. Efficient finite field serial/parallel multiplication. In *Proceedings of the 10th IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP '96)*, pp. 72–82. IEEE Computer Society Press, 1996.

[30] N. H. Weste and K. Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 1993.

[31] R. Zimmermann and W. Fichtner. Low-power logic styles: CMOS versus pass-transistor logic. *IEEE Journal of Solid-State Circuits*, 32(7):1079–1090, July 1997.

Arithmetic on Binary Genus 2 Curves Suitable for Small Devices

Tanja Lange

Faculty for Mathematics, Technical University of Denmark, Matematiktorvet 303, DK-2800 Kgs. Lyngby, Denmark, t.lange@mat.dtu.dk

Abstract. In the recent past, hyperelliptic curve cryptosystems received a lot of attention especially for restricted environments. In this article we investigate the arithmetic of some hyperelliptic curves of genus 2 defined over a binary field $\mathbb{F}_{2^d}$. At SAC 2004, Lange and Stevens provided doubling formulae in affine coordinates which are very fast – only 1 inversion, 6 squarings and 5 multiplication are needed some special choices – but still these formulae require 1 inversion per group operation which is prohibitive for small devices like smart cards and FPGAs.
Our goal is to present inversion-free formulae for the special type of curves with h of degree 1 identified in that paper. Our formulae are much faster than the general methods presented in the literature so far. As a second topic we consider compression techniques. Especially for small devices and low communication bandwidth it might be worth some effort to save half of the size. Our proposal has the advantage over previous methods that all operations are performed in the ground field.

Keywords. Hyperelliptic curves, fast arithmetic, explicit group operations, binary fields, compression.

1 Introduction

Curve based cryptography offers the advantage that so far no subexponential algorithm for solving the discrete logarithm problem is known for small genus[1]. This means that for equal security the group size can be chosen much smaller than for discrete logarithm systems based on the multiplicative group of a finite field or RSA. Therefore, curve based systems raised quite some interest for embedded systems. Elliptic curves were studied for applications during almost two decades now. For a long time, hyperelliptic curves were not considered to be competitive, until recently the group arithmetic was improved by several authors [Har00,Lan01,MDM+,Tak02,Lan05,SMCT02,KGM+02,Pel02,Wol04,GMA+05].

Avanzi [Ava04] gives a comparison via implementation for fields of odd characteristic showing genus two curves to lead to systems almost as fast as those from elliptic curves. For characteristic two fields the group law depends a lot on

[1] Here, *small* really means genus $g \leq 3$ by [Gau00,Thé03,GT04,Nag04], and even for $g = 3$ some care has to be taken.

the equation of the curve. In [CY02,BD04] and [ACD+05, Ch. 14] four different types of curves are classified out of which three are nonsupersingular, hence, useful for applications based on the DLP. Pelzl, Wollinger, and Paar [PWP04] give doubling formulae for a very special case which is faster than the most efficient case in [BD04]. In [LS05] all three nonsupersingular cases are considered and the to-date most efficient doubling formulae are given. Implementations show that for the case of $\deg(h) = 1$ scalar multiplications are faster than on elliptic curves.

In this paper we concentrate on this most efficient case from [LS05] and investigate different inversion-free systems. The result is a detailed classification of how many field operations are needed to perform doublings and additions in the respective systems such that the implementor can choose the most appropriate system depending on the number of precomputations he can store, i. e. how much more frequent doublings appear compared to additions.

As a second topic we propose a new compression algorithm which has the advantage over previous proposals [HSS01] that all computations are performed in the same field over which the divisor class is defined and that they do not require working in extension fields. The approach is similar to Stahlke's [Sta04] but he considers only odd characteristic and in fact his approach fails for even characteristic in general and in the considered special case a lot of changes are necessary.

The remainder of the paper is structured as follows: first we briefly review background on hyperelliptic curves and provide doubling formulae in projective coordinates. The new coordinates from [Lan05] are easily adapted to the new doubling formulae but are not particularly efficient. Therefore, we present a new system of weighted coordinates for which we provide both addition and doubling formulae. For applications the following comparison is important as a summary of the previous sections and because also mixed systems are discussed. Before concluding we state the compression and decompression algorithms.

2 Basic Notations and Preliminaries

We refer the interested reader to [ACD+05,FL03,Lor96,MWZ98,Sti93] for mathematical background.

Let $\mathbb{F}_q, q = 2^d$, be a finite field of characteristic 2 and let C be a hyperelliptic curve defined over $\mathbb{F}_q$. In cryptography one usually deals with curves C given by

$$\begin{gathered} C: \quad Y^2 + h(X)Y = f(X) \\ h, f \in \mathbb{F}_q[X],\ f \text{ monic},\ \deg f = 2g+1,\ \deg h \leq g \end{gathered} \tag{1}$$

for which no point $(x, y) \in C$ satisfies both partial derivative equations. For characteristic 2 one needs to have a non-zero h to achieve this quality. The integer g appearing in (1) is called the *genus of* C. We concentrate on curves of genus 2.

The group one uses for cryptographic applications is the ideal class group $\mathrm{Cl}(C/\mathbb{F}_q)$ of C over $\mathbb{F}_q$. This is the quotient of the group of fractional ideals of $\mathbb{F}_q[X,Y]/(Y^2+h(X)Y+f(X))$ by the group of principal ideals. As in the case of quadratic imaginary fields, one finds ideals generated by two polynomials $\langle u(X), v(X)+Y\rangle$ in each ideal class and there is a unique ideal for which $\deg(u)$ is minimal. Actually, each class $\bar{D}$ in $\mathrm{Cl}(C/\mathbb{F}_q)$ can be represented by an ordered pair of polynomials $D=[u(X),v(X)]$, with $u,v\in\mathbb{F}_q, \deg v<\deg u\leq g$ and u monic satisfying $u|v^2+hv+f$.

The group operation in $\mathrm{Cl}(C/\mathbb{F}_q)$ is performed by first computing the product of the representing ideals and then reducing the result modulo the principal ideals. This is the idea behind Cantor's algorithm [Can87,Kob89].

Obviously, this algorithm has to depend on the properties of the input – to derive explicit formulae one needs to study additions independently from doublings. For a complete study of all possible inputs together with formulae we refer to [Lan05].

3 Binary hyperelliptic curves with 2-rank one

In this contribution we concentrate on those binary genus 2 curves which are given by an equation of the form

$$C: Y^2+h_1XY=X^5+f_3X^3+f_2X^2+f_0. \tag{2}$$

As shown in [ACD+05, Proposition 14.37] each curve with $\deg(h)=1$ can be transformed to form (2) by isomorphic transformations and one can even place further restrictions on the coefficients. For applications in cryptography extension fields of odd degree d are interesting. In this case one can require $h_1=1$ and $f_2\in\mathbb{F}_2$ giving a unique representative per isomorphism class. In particular this implies that multiplications with h_1 and f_2 need not be counted.

These curves are not completely generic – one would expect that a random curve satisfies $\deg h=2$ and as there are only two free parameters one sees that only $O(q^2)$ instead of $O(q^3)$ different curves can be reached. From the arithmetic properties there is one particularity: the order of the ideal class group $|\mathrm{Cl}(C/\mathbb{F}_q)|$ is divisible by 2 and the 2-rank is only 1 instead of the maximal 2. It is known that supersingular curves (i. e. curves of 2-rank 0) have a DLP which is easier to solve as the Tate pairing maps the DLP from $\mathrm{Cl}(C/\mathbb{F}_q)$ to $\mathbb{F}_{q^k}^*$ with a relatively small k. So far no attack for curves of the form (2) is known.

Since on the other hand they offer big advantages [LS05] for implementations it seems worthwhile to study arithmetic in inversion-free systems on them.

4 Inversion-free arithmetic

In this section we assume our reader to be familiar with [Lan05] and [LS05]. The former reference contains a section on curves with $\deg(h)=1$ but the formulae

were derived based on the general addition and doubling formulae. In fact, except for the obvious savings due to zero coefficients there is no advantage in additions but the doubling formulae are much more efficient for curves of form (2).

We briefly state formulae for projective coordinates $\mathcal{P}$. For the additions the changes are quite obvious and are simply obtained by fixing the respective curve parameters to be zero. Hence, we only treat doublings there. Then we introduce a new system of coordinates with completely different weights of the coordinates. As there is apparently a clash in the names we refer to them as "recent" coordinates $\mathcal{R}$. For this system we also need to consider additions.

4.1 Doubling in projective coordinates

Doubling in projective coordinates		
Input	$\bar{D} = [U_1, U_0, V_1, V_0, Z]$, precomputed values h_1^2 and h_1^{-1}.	
Output	$[U_1', U_0', V_1', V_0', Z'] = [2][U_1, U_0, V_1, V_0, Z]$.	
Step	Expression	Operations
1	precomputations $Z_2 \leftarrow Z^2$, $z_0 \leftarrow U_0^2$, $t_1 \leftarrow U_1^2 + f_3 Z_2$, $w_0 \leftarrow f_0 Z_2 + V_0^2$, $w_1 \leftarrow z_0 Z_2$ $z_1 \leftarrow t_1 z_0$, $w_2 \leftarrow h_1^2 w_1$, $w_3 \leftarrow w_2 + t_1 w_0$, $w_4 \leftarrow w_0 Z$ $s_0 \leftarrow z_1 + U_1 w_4$, $w_4 \leftarrow w_4 Z$	9M + 4S
2	compute U' $U_1' \leftarrow w_1 w_2$, $U_0' \leftarrow s_0^2 + w_2 w_4$	2M + S
3	compute V' $w_5 \leftarrow w_0 w_4$, $V_1' \leftarrow h_1^{-1}\big(w_2 U_1' + \big(w_3 z_1 + (f_2 Z_2 + V_1^2) w_5\big) w_4\big)$ $w_5 \leftarrow w_5 w_4$, $V_0' \leftarrow h_1^{-1}(w_3 U_0' + z_0 w_5)$	11M + S
4	adjust $Z' \leftarrow w_5 Z_2$, $U_1' \leftarrow U_1' w_4$, $U_0' \leftarrow U_0' w_4$	3M
total		25M +6S

If h_1^{-1} is small one saves $2M$, and if $h_1 = 1$ — as one can always achieve for odd extension degrees — $22M + 6S$ are used in total.

4.2 Recent coordinates in even characteristic ($\mathcal{R}$)

We now propose the weighted coordinates $[U_1, U_0, V_1, V_0, Z, z]$ with $u_i = U_i/Z$, $v_i = V_i/Z^2$ and the precomputation $z = Z^2$. Lacking a better name we refer to them as *recent coordinates* $\mathcal{R}$. These coordinates have the advantage of allowing faster doublings while the additions are even more expensive. However, usually mixed additions are chosen for implementations. They are not too much slower, and furthermore, in windowing methods the number of additions is reduced considerably.

The results in brackets refer to the case in which the second input is in affine coordinates.

Addition in recent coordinates		
Input	$[U_{11}, U_{10}, V_{11}, V_{10}, Z_1, z_1], [U_{21}, U_{20}, V_{21}, V_{20}, Z_2, z_2]$	
Output	$[U'_1, U'_0, V'_1, V'_0, Z', z'] = [U_{11}, U_{10}, V_{11}, V_{10}, Z_1, z_1] + [U_{21}, U_{20}, V_{21}, V_{20}, Z_2, z_2]$	
Step	Expression	Operations
1	precomputation: $Z \leftarrow Z_1 Z_2$, $z \leftarrow Z^2$, $\tilde{U}_{21} \leftarrow U_{21} Z_1$, $\tilde{U}_{20} \leftarrow U_{20} Z_1$ $\tilde{V}_{21} \leftarrow V_{21} z_1$, $\tilde{V}_{20} \leftarrow V_{20} z_1$	5M +S (none)
2	compute resultant $r = \mathrm{res}(U_1, U_2)$ $y_1 \leftarrow U_{11} Z_2 + \tilde{U}_{21}$, $y_2 \leftarrow U_{10} Z_2 + \tilde{U}_{20}$ $y_3 \leftarrow U_{11} y_1 + y_2 Z_1$, $r \leftarrow y_2 y_3 + y_1^2 U_{10}$	6M + S (5M + S)
3	compute almost inverse of u_2 modulo u_1 $inv_1 \leftarrow y_1$, $inv_0 \leftarrow y_3$	
4	compute s $w_0 \leftarrow V_{10} z_2 + \tilde{V}_{20}$, $w_1 \leftarrow V_{11} z_2 + \tilde{V}_{21}$ $w_2 \leftarrow inv_0 w_0$, $w_3 \leftarrow inv_1 w_1$ $s_1 \leftarrow (inv_0 + inv_1 Z_1)(w_0 + w_1) + w_2 + w_3(Z_1 + U_{11})$ $s_0 \leftarrow w_2 + U_{10} w_3$	8M (7M)
5	precomputations $\bar{Z} \leftarrow s_1 r$, $w_4 \leftarrow rZ$, $w_5 \leftarrow w_4^2$, $S \leftarrow s_0 Z$, $Z' \leftarrow Z\bar{Z}$ $\tilde{s}_0 \leftarrow s_0 Z'$, $\bar{s}_1 \leftarrow s_1 \bar{Z}$, $\tilde{s}_1 \leftarrow \bar{s}_1 Z$	7M + S
6	compute l $L_2 \leftarrow \bar{s}_1 \tilde{U}_{21}$, $l_2 \leftarrow L_2 Z$, $l_0 \leftarrow \tilde{s}_0 \tilde{U}_{20}$ $l_1 \leftarrow (\tilde{U}_{21} + \tilde{U}_{20})(\tilde{s}_0 + \tilde{s}_1) + l_2 + l_0$, $l_2 \leftarrow L_2 + \tilde{s}_0$, $\tilde{h}_1 \leftarrow h_1 z$	5M
7	compute U' $U'_0 \leftarrow r(S^2 + y_1(s_1^2(y_1 + \tilde{U}_{21}) + Z w_5) + \tilde{h}_1 Z') + y_2 \tilde{s}_1$ $U'_1 \leftarrow y_1 \bar{s}_1 + w_4 w_5$	8M + 2S
8	precomputations $w_1 \leftarrow l_2 + U'_1$, $U'_1 \leftarrow U'_1 w_4$, $\bar{Z} \leftarrow Z'\bar{Z}$, $l_0 \leftarrow l_0 \bar{Z}$ $w_2 \leftarrow U'_1 w_1 + (U'_0 + l_1)\bar{Z}$, $\bar{Z} \leftarrow \bar{Z}^2$	5M + S
9	compute V' $V'_1 \leftarrow w_2 s_1 + (\tilde{V}_{21} + \tilde{h}_1)\bar{Z}$, $U'_0 \leftarrow U'_0 r$, $w_2 \leftarrow U'_0 w_1 + l_0$ $V'_0 \leftarrow w_2 s_1 + \tilde{V}_{20}\bar{Z}$, $Z' \leftarrow Z'^2$, $z' \leftarrow Z'^2$	6M + 2S
total		50M + 8S (43M + 7S)

If $h_1 = 1$ as we can always assume for d odd one more multiplication is saved in Step 6.

Doubling in recent coordinates		
Input	$\bar{D} = [U_1, U_0, V_1, V_0, Z, z]$, precomputed values h_1^2 and h_1^{-1}.	
Output	$[U_1', U_0', V_1', V_0', Z', z'] = [2][U_1, U_0, V_1, V_0, Z, z]$.	
Step	Expression	Operations
1	precomputations $Z_4 \leftarrow z^2$, $y_0 \leftarrow U_0^2$, $t_1 \leftarrow U_1^2 + f_3 z$, $w_0 \leftarrow Z_4 f_0 + V_0^2$ $\bar{Z} \leftarrow z w_0$, $w_1 \leftarrow y_0 Z_4$, $y_1 \leftarrow t_1 y_0 z$, $s_0 \leftarrow y_1 + U_1 w_0 Z$ $w_2 \leftarrow h_1^2 w_1$, $w_3 \leftarrow w_2 + t_1 w_0$	10M + 4S
2	compute U' $U_1' \leftarrow w_2 w_1$, $w_2 \leftarrow w_2 \bar{Z}$, $U_0' \leftarrow s_0^2 + w_2$	2M + S
3	compute V' $Z' \leftarrow \bar{Z}^2$, $V_1' \leftarrow h_1^{-1}\big(w_2 U_1' + (w_3 y_1 + f_2 Z' + (V_1 w_0)^2) Z'\big)$ $V_0' \leftarrow h_1^{-1}\big(\bar{Z}(w_3 U_0' + y_0 w_0 Z')\big)$, $z' = Z'^2$	11M + 3S
total		23M + 8S

For small h_1^{-1} we save 2M, if even $h_1 = 1$ a total of only 20M + 8S is needed.

4.3 Different sets of coordinates

We now state the number of operations for curves of form (2) in even characteristic. For the table we assume that $\mathbb{F}_q = \mathbb{F}_{2^d}$ with d odd as this is the most frequent case for the applications. We also include $\mathcal{N}$ to denote the new coordinates from [Lan02]. Note that compared to the figures given in that paper ours are computed on the basis of the much better doubling formulae from [LS05].

Doubling		Addition	
Operation	Costs	Operation	Costs
$2\mathcal{N} = \mathcal{N}$	28M + 5S	$\mathcal{R} + \mathcal{R} = \mathcal{R}$	49M + 8S
$2\mathcal{P} = \mathcal{P}$	22M + 6S	$\mathcal{P} + \mathcal{P} = \mathcal{P}$	49M + 4S
$2\mathcal{R} = \mathcal{R}$	20M + 8S	$\mathcal{N} + \mathcal{N} = \mathcal{N}$	42M + 6S
—	—	$\mathcal{A} + \mathcal{R} = \mathcal{R}$	42M + 7S
—	—	$\mathcal{A} + \mathcal{P} = \mathcal{P}$	39M + 4S
—	—	$\mathcal{A} + \mathcal{N} = \mathcal{N}$	36M + 6S
$2\mathcal{A} = \mathcal{A}$	I + 5M + 6S	$\mathcal{A} + \mathcal{A} = \mathcal{A}$	I + 21M + 3S

This table allows to read off which coordinate systems should be chosen for implementation depending on the ratio between inversions and multiplications. If inversions are affordable the affine coordinates should be used as they offer to use a very low number of operations.

If the input is in affine coordinates but one cannot perform one inversion per group operation the choice of the coordinate system depends on the window width, i. e. the ratio between doublings and additions and the ratio between squarings and multiplications. If no precomputations can be made and a NAF

of the scalar is used, one has twice as many doublings as additions in the worst case. Even then the recent coordinates are most likely to be fastest as in the mixed setting 41M + 11S are needed per bit while in $\mathcal{P}$ one needs 41.5M + 8S and 44.5M+12S in $\mathcal{N}$. In the average case where the NAF expansion has density 1/3 the figures reduce to 34M+10S, 35M+7.3S, and 40M+7S per bit respectively. For larger windows recent coordinates are clearly the systen of choice. If the input is not affine the same ranking holds showing that $\mathcal{N}$ is always least efficient and $\mathcal{R}$ gains with the window size.

5 Compression

We now study the question of how to compress divisor classes $[u, v]$ for a genus 2 curve given by (2). The compression technique proposed by Hess, Seroussi, and Smart [HSS01] requires a factorization of u over $\mathbb{F}_{2^d}$. If u splits then one can recover two points $P_1, P_2 \in C(\mathbb{F}_{2^d})$ such that the divisor class is represented by $\bar{D} = P_1 + P_2 - 2P_\infty$. Compression techniques like for elliptic curves can be applied to P_1 and P_2 separately and one stores the x-coordinates of the points together with one bit for each to determine the y-coordinate. This requires solving an equation of degree 2 for each point in the compression and decompression. If the polynomial u does not split it can be used to construct a field extension $K = \mathbb{F}_{2^d}[X]/u(X)$. In K the second polynomial $v(X)$ considered as a field element is uniquely determined by one bit; the algorithm requires solving one quadratic equation in K in the decompression step.

Obviously, the drawback of this method is that it is very hard to implement arithmetic in the extension fields of degree 2 as their defining polynomials vary. Furthermore, different routines are required to handle the two cases. This was noticed by Stahlke [Sta04] who details an alternative compression technique on genus 2 curves in odd characteristic.

We now present a corresponding method for curves of the form (2). The representation by two polynomials $[u, v]$ proposed by Mumford satisfies $u \mid v^2 + hv + f$ and $\deg(v) < \deg(u) \leq 2$ for u monic. For $\deg(u) = 1$ compression works like in [HSS01] and on elliptic curves. Hence, we concentrate on $\deg(u) = 2$ in the sequel.

Assume that v is unknown and put $(v^2 + hv + f)/u = s$ for some unknown s monic of degree 3. For $h(X) = h_1 X$ formal division leads to $s_2 = u_1$ and $s_1 = u_1^2 + u_0 + f_3$ and the solvability condition gives a system of equations

$$\begin{aligned} s_0 + u_1 s_1 + u_0 s_2 + f_2 &= v_1^2 + h_1 v_1 \\ u_1 s_0 + u_0 s_1 &= h_1 v_0 \\ u_0 s_0 + f_0 &= v_0^2 \end{aligned}$$

in the three unknowns s_0, v_1, v_0.

Insert $v_0 = h_1^{-1}(u_1 s_0 + u_0 s_1)$ in the last equation to obtain a quadratic equation in s_0

$$u_1^2 h_1^{-2} s_0^2 + u_0 s_0 + h_1^{-2} u_0^2 s_1^2 + f_0 = 0.$$

By a change of variables this is transformed to

$$z^2 + z + \frac{u_1^2(s_1^2 u_0^2 + h_1^2 f_0)}{h_1^4 u_0^2}, \text{ with } s_0 = z\frac{h_1^2 u_0}{u_1^2}.$$

To compute the respective values one proceeds as detailed in the following table. The costs reflect the interesting case of odd d and thus $h_1 = 1$; for completeness the general case is described. The costs of solving one equation of the form $T^2 + T + A = 0$ are denoted by Q. Note that by construction this equation is solvable in the ground field $\mathbb{F}_{2^d}$ and has two solutions z and $z + 1$. Thus, one bit is sufficient to determine the correct solution. If — as usual — $\mathbb{F}_{2^d}$ is represented with respect to a polynomial basis in some variable ξ one can use the coefficient of ξ^0 as encoding of the choice.

Decompression		
Input	$\bar{D} = [u_1, u_0, b_1, b_0]$, precomputed values h_1^2 and h_1^{-1}.	
Output	$[u_1, u_0, v_1, v_0]$, with v_1, v_0 as prescribed by b_1, b_0.	
Step	Expression	Operations
1	precomputations $u_0' \leftarrow h_1^2 u_0$, $U_0 \leftarrow {u'}_0^2$, $U_1 \leftarrow u_1^2$, $w_1 \leftarrow U_0 U_1$ $w_2 \leftarrow U_1 w_1$, $w_3 \leftarrow U_0 w_1$	$1\mathrm{I} + 4\mathrm{M} + 2\mathrm{S}$
2	compute s_1 $s_1' \leftarrow U_1 + f_3$, $s_1 \leftarrow s_1' + u_0$	
3	precomputations $w_4 \leftarrow u_0 s_1$, $w_5 \leftarrow w_4^2$, $w_6 \leftarrow u_1 s_1'$	$2\mathrm{M} + 1\mathrm{S}$
4	compute s_0 $A \leftarrow (h_1^2 f_0 + w_5) U_1 w_1$ solve $T^2 + T + A$ and determine solution z by b_0 $s_0 \leftarrow u_0' z w_2$	$3\mathrm{M} + 1\mathrm{Q}$
5	compute v_0 $v_0 \leftarrow h_1^{-1}(u_1 s_0 + w_4)$	$1\mathrm{M}$
6	compute v_1 $A \leftarrow s_0 + w_6 + f_2$ solve $T^2 + h_1 T + A$ and determine solution v_1 by b_1	$1\mathrm{Q}$
total		$1\mathrm{I} + 10\mathrm{M} + 3\mathrm{S} + 2\mathrm{Q}$

If $h_1 \neq 1$ the values h_1^{-1} and h_1^2 are precomputed and $1\mathrm{I} + 15\mathrm{M} + 4\mathrm{S} + 2\mathrm{Q}$ are needed in total.

To compress one computes $s_0 \leftarrow \frac{v_0^2 + f_0}{u_0}$ and $z \leftarrow \frac{u_1}{h_1^2 u_0} s_0$ in $1\mathrm{I} + 3\mathrm{M} + 1\mathrm{S}$ and puts b_0 the coefficient of ξ^0 in z and b_1 that of v_1.

Remark 1. Note that the more obvious order of first computing v_0 and then s_0 needs 1I+12M+1S+2Q which is more expensive assuming that squarings are less expensive than multiplications. On the other hand, in this case the compression is faster as v_0 and v_1 are both known.

For comparison, the method in [HSS01] starts by computing $A \leftarrow \frac{u_0}{u_1^2}$ and the attempt to solve $T^2 + T + A$. If solutions exist they are determined and give rise to x_1, x_2, the x-coordinates of P_1, P_2. Computing $v(x_1)$ and $v(x_2)$ by 2M reveals the y-coordinates and hence b_1, b_0. Otherwise the coefficient of ξ^0 in v_0 fixes b_0 and b_1 is not used. Thus $1\mathrm{I} + 5\mathrm{M} + 1\mathrm{S} + 1\mathrm{Q}$ are needed for compression in the worst case.

To decompress in the first case, the recipient determines x_1, x_2 by the same procedure as above and computes y_i by solving $T^2 + h_1 T + f(x_i)$. Finally, $v = \frac{y_1+y_2}{x_1+x_2} x + \frac{y_1 x_2 + y_2 x_1}{x_1+x_2}$. In total, $2\mathrm{I} + 11\mathrm{M} + 2\mathrm{S} + 3\mathrm{Q}$ are needed.
If u is irreducible one solves $T^2 + h_1 T + f(X)$ in $K = \mathbb{F}_{2^d}[X]/u(X)$, i. e. first f is reduced modulo u in $4\mathrm{M} + 1\mathrm{S}$ and then the quadratic equation is solved. Solving the equation in the extension field corresponds to at least 3Q and the even more complicated problem of working in the extension field on the fly. As the costs for solving quadratic equations are dominant, this case is also more expensive than in our proposal.

6 Conclusion

We presented inversion-free systems for binary genus 2 curves of a special form which leads to particularly fast doublings. The advantages of this special case described in [LS05] for affine coordinates carry over to these coordinate systems: the proposed formulae require far less operations than the general ones considered in [Lan05]. Our work is ready to implement and adopts to the special requirements of the device by offering a choice of coordinate systems.

A new and more efficient compression algorithm in the spirit of Stahlke completes the study. We would like to point out that a similar algorithm seems to be hard to find for the case $\deg(h) = 2$.

The present version does not include any timing results as the effects are less visible in software. We hope to present some actual timings on the RFID workshop.

References

[ACD+05] R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC, 2005. to appear.

[Ava04] R. M. Avanzi. Aspects of hyperelliptic curves over large prime fields in software implementations. In *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Comput. Sci.*, pages 148–162. Springer-Verlag, 2004.

[BD04] B. Byramjee and S. Duquesne. Classification of genus 2 curves over $\mathbb{F}_{2^n}$ and optimization of their arithmetic. preprint, 2004.

[Can87] D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48:95–101, 1987.

[CY02] Y. Choie and D. Yun. Isomorphism classes of hyperelliptic curves of genus 2 over $\mathbb{F}_q$. In *Australasian Conference on Information Security and Privacy – ACISP 2002*, volume 2384 of *Lecture Notes in Comput. Sci.*, pages 190–202. Springer-Verlag, Berlin, 2002.

[FL03] G. Frey and T. Lange. Mathematical background of public key cryptography. Technical Report 10, IEM Essen, 2003. To appear in Séminaires et Congrès.

[Gau00] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807, pages 19–34. Springer-Verlag, Berlin, 2000.

[GMA+05] M. Gonda, K. Matsuo, K. Aoki, J. Chao, and S. Tsuji. Improvements of addition algorithm on genus 3 hyperelliptic curves and their implementations. *IEICE Trans. Fundamentals*, E88-A(1):89–96, 2005.

[GT04] P. Gaudry and E. Thomé. A double large prime variation for small genus hyperelliptic index calculus. Cryptology ePrint Archive, Report 2004/153, 2004.

[Har00] R. Harley. Fast arithmetic on genus 2 curves.
available at `http://cristal.inria.fr/~harley/hyper`, 2000.

[HSS01] F. Hess, G. Seroussi, and N. P. Smart. Two topics in hyperelliptic cryptography. In *Selected Areas in Cryptography – SAC 2000*, volume 2259 of *Lecture Notes in Comput. Sci.*, pages 181–189. Springer-Verlag, Berlin, 2001.

[KGM+02] J. Kuroki, M. Gonda, K. Matsuo, J. Chao, and S. Tsuji. Fast Genus Three Hyperelliptic Curve Cryptosystems. In *Proc. of SCIS2002, IEICE Japan*, pages 503–507, 2002.

[Kob89] N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1:139–150, 1989.

[Lan01] T. Lange. *Efficient arithmetic on hyperelliptic curves*. PhD thesis, Universität-Gesamthochschule Essen, 2001.

[Lan02] T. Lange. Weighted coordinates on genus 2 hyperelliptic curves. preprint, 2002.

[Lan05] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Appl. Algebra Engrg. Comm. Comput.*, 15(5):295–328, 2005.

[Lor96] D. Lorenzini. *An invitation to arithmetic geometry*, volume 9 of *Graduate studies in mathematics*. AMS, 1996.

[LS05] T. Lange and M. Stevens. Efficient doubling for genus two curves over binary fields. In *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Comput. Sci.*, pages 170–181. Springer-Verlag, Berlin, 2005.

[MDM+] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao, and S. Tsuji. A fast addition algorithm of genus two hyperelliptic curve. In *Symposium on Cryptography and Information Security – SCIS 2002*, pages 497–502. In Japanese.

[MWZ98] A. J. Menezes, Y.-H. Wu, and R. Zuccherato. An Elementary Introduction to Hyperelliptic Curves. In N. Koblitz, editor, *Algebraic Aspects of Cryptography*, pages 155–178. Springer, 1998.

[Nag04] K. Nagao. Improvement of Thériault Algorithm of Index Calculus for Jacobian of Hyperelliptic Curves of Small Genus. Cryptology ePrint Archive, Report 2004/161, 2004.

[Pel02] J. Pelzl. Fast hyperelliptic curve cryptosystems for embedded processors. Master's thesis, Ruhr-University of Bochum, 2002.

[PWP04] J. Pelzl, T. Wollinger, and C. Paar. Special hyperelliptic curve cryptosystems of genus two: Efficient arithmetic and fast implementation. In *Embedded Cryptographic Hardware: Design and Security*. Nova Science Publishers, 2004.

[SMCT02] H. Sugizaki, K. Matsuo, J. Chao, and S. Tsujii. An Extension of Harley algorithm addition algorithm for hyperelliptic curves over finite fields of characteritic two. Technical Report ISEC2002-9(2002-5), IEICE, 2002.

[Sta04] C. Stahlke. Point compression on Jacobians of hyperelliptic curves over $\mathbb{F}_q$. preprint, 2004.

[Sti93] H. Stichtenoth. *Algebraic Function Fields and Codes*. Springer, 1993.

[Tak02] M. Takahashi. Improving Harley Algorithms for Jacobians of genus 2 Hyperelliptic Curves. In *Proc. of SCIS2002, IEICE Japan*, 2002. in Japanese.

[Thé03] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In *Advances in Cryptology – Asiacrypt 2003*, volume 2894 of *Lecture Notes in Comput. Sci.*, pages 75–92. Springer-Verlag, Berlin, 2003.

[Wol04] T. Wollinger. *Software and Hardware Implementation of Hyperelliptic Curve Cryptosystems*. PhD thesis, Ruhr-University of Bochum, 2004.

Is Elliptic-Curve Cryptography Suitable to Secure RFID Tags?

Johannes Wolkerstorfer

Institute for Applied Information Processing and Communications,
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria.
Johannes.Wolkerstorfer@iaik.at, http://www.iaik.at/

Abstract. This article scrutinizes the suitability of Elliptic-Curve Cryptography (ECC) as a technology for solving security concerns of RFID systems. The article focuses on the technical feasibility of realization: Can ECC be implemented on RFID tags without increasing the price of tags too much due to high circuit complexity? Does ECC consume so little power that the operative distance of passively powered RFID tags is not shortened? The article summarizes requirements which have to be met and analyzes how hardware implementations of ECC can be scaled down to a minimum. The architecture of an ECC-enabled tag is presented. A core element of the tag is an arithmetic unit for computing ECC that is tailored for constricted environments where low-area and low-power optimizations are crucial. The results obtained with the new ECC architecture show that the implementation of ECC on RFID tags seems to be viable when 180-*nm* CMOS process technology is used for manufacturing RFID tags.

Keywords. Elliptic-Curve Cryptography, RFID Technology, Dual-Field Arithmetic, Montgomery Multiplication, Low-Power Optimization.

1 Introduction

Radio Frequency Identification (RFID) produced much interest during the last couple of months. The interest was mainly twofold. On one hand, RFID tags—small wireless chips with an little antenna—can be used to label all kind of goods. The long-term purpose of RFID tags is to replace printed barcodes by RFID tags which allows making logistics processes more efficient. On the other hand, the pervasive use of RFID technology entails security concerns which have to be handled. The data protection working party of the European Commission addressed this topic [1]. They analyzed that many data protection and privacy implications of RFID technology must be sorted out to raise the acceptance of the new technology. Measures to achieve these goals comprehend legal matters, organizational measures, and technical solutions. In this article we investigate how elliptic-curve cryptography helps to raise information security of RFID tags. In particular, the feasibility of implementing ECC on RFID tags is scrutinized.

Some months ago, first steps were made to implement strong cryptography on RFID tags. Martin Feldhofer et al. reported the first implementation of the AES algorithm that fulfills the stringent area and power requirements of RFID tags [2]. On a mature 0.35 μm CMOS process technology, on which most RFID tags are manufactured today, they reported an AES implementation with a complexity of 3,500 gates. Its power consumption is only 0.045 mW/MHz which is fairly below the requirements of RFID tags.

Symmetric encryption like the AES algorithm will surely help securing RFID tags. But symmetric encryption cannot solve all security concerns of RFID technology. A major hindrance is the key distribution issue. Most applications of RFID are open systems where not all parties are trusted. Key distribution must ensure that only trusted parties can obtain keys. Proving the genuineness of RFID-labeled products is an application where symmetric cryptography succumbs asymmetric techniques: Everyone who wants to prove the genuineness of a labeled good has to possess the according symmetric key but this allows ... Another application example where asymmetric cryptography excels symmetric cryptography is privacy-enhancing behavior to prevent tracking. When RFID tags should only respond to authenticated readers, no tracking of goods (and customers) is possible by unauthorized readers. Symmetric cryptography demands the tag to store the keys of all authorized readers. Contrary, asymmetric cryptography can achieve this by storing a single certificate.

Elliptic-curve cryptography is a candidate technology to secure RFID systems. Moreover, ECC is the only technology that can cope with heavily constricted resources. RFID tags have very limited silicon area due to economical reasons. The silicon area of a tag determines its cost. The power consumption of RFID tags is very limited too because they are powered over the air interface by the reader. To save power, the clock frequency of RFID tags is often below 1 *MHz*. This affects also the throughput and the latency of cryptographic operations.

This article explores the feasibility of integrating elliptic-curve cryptography in RFID tags. To our knowledge, this is the first article covering this topic. We investigate an ECC-enabled tag and estimate how much resources the implementation of strong asymmetric cryptography will cost in terms of silicon area and power consumption. A central element is a very compact dual-field arithmetic unit. This novel arithmetic unit is capable to calculate addition, multiplication, and also inversion in the finite fields GF(2^m) and in GF(p). The article also gives reliable estimates how much resources a small ECC processor needs and which CMOS technology is suitable to manufacture secure RFID tags.

The discussion about ECC-enabled RFID tags starts with writing down limitations for implementing RFID tags in §2. §3 analyzes possibilities to scale hardware implementations of ECC to a minimum in order to save area and power. §4 presents the architecture of an ECC-anabled RFID tag. The core component of the tag, a novel dual-field arithmetic unit, is presented in closer detail in §5. §6 gives a summary of results achieved by the new approach. §7 concludes the article and gives an outlook on further developments and possible improvements.

2 Technical and Economical Restrictions of RFID Tags

ECC seems to be the most appropriate technology for implementing strong asymmetric cryptography on RFID tags. ECC can be implemented using least resources compared to other technologies. Asymmetric cryptography can be implemented by many different algorithms. RSA signatures and Diffie-Hellman key establishment are widely used algorithms which are not feasible for RFID system due to their large operand sizes of 1024 bits and more. Other asymmetric algorithms like XTR and NTRU are often mentioned when resource-optimized implementations are the primary objective. These algorithms are prone to security weaknesses which demanded repeatedly increasing key lengths in the past. This opposes resource-efficient implementations.

Today, ECC is favored in resource-constrained environments like smartcards where silicon area is sparse and electrical power is limited. This is reasoned by the fact that ECC is contented with much smaller cryptographic key sizes. At present, ECC keys range from 163 bits to 283 bits. Hardware implementations of ECC benefit from short key sizes because ECC processors use less silicon area and consume less power. Moreover, short wordsizes of ECC needs less memory for parameter storage and they save bandwidth during communication.

In this article, we constrict the analysis of RFID systems to passively powered tags operating at a frequency of 13.56 MHz that conform to the ISO 18000-3 standard [4]. This constriction does not take 125 kHz and UHF systems into account which play too a significant role in the RFID market too. Nevertheless, 13.56 MHz systems do represent the majority of RFID tags being used in supply-chain applications. Security concerns of supply-chain applications are the most urgent challenge because they influence the public opinion most.

The silicon area of RFID tags ranges typically between 0.1 mm^2 and 1 mm^2. The size of RFID tags is an important issue for the economical success of the product. The price of a tag grows with the silicon area. Smart tags—like secure tags, which offer additional functionality—may have higher prices due to their added value. Thus, the area requirements for ECC-enabled tags are somewhat difficult to pin down. An upper limit of 1 mm^2 seems to be realistic.

The requirements for the power consumption can be defined more precisely. Excessive power consumption reduces the operative range of RFID tags. ISO-18000 tags have an operative range of roughly 1 m. If the power consumption of the tag exceeds the maximum, the supply voltage of the tag drops below a threshold. This will trigger the hardware reset of the tag and render all previous communication and computation useless. A consequence of high power consumption is a shortened operative range because the power supply by the reader's magnetical field drops with the third power of distance. In supply-chain applications, a short operative range will be a hindrance for the proliferation of RFID tags. Thus, low power consumption is even more strict than silicon area. Power consumption up to 30 μW has no impact on operative range of RFID tags.

3 Scaling ECC to its limits

Although ECC is an economical technology, its computational effort is too high to make software implementations on 8-bit or 16-bit platforms possible. To offer reasonable performance, dedicated hardware is necessary. General purpose processors implementing ECC are not useful for RFID tags. The most obvious argument is the size of the processor. Computing ECC in acceptable time requires using of 32-bit processors. Such processor cores have a complexity of 50,000 gates and more. This complexity does not include memories like RAM and ROM. Besides being too large for integration on RFID tags, circuits of this size consume considerable power. RFID technology demands dedicated ECC hardware that is tailored to the specific needs and limitations.

There are several options for implementing ECC on hardware. One elementary choice is the underlying finite field. Most hardware implementations opt for the finite field GF(2^m) using polynomial basis representation. Unfortunately, ECC cannot be implemented using exclusively binary-field arithmetic. EC cryptographic primitives also require conventional integer arithmetic. For instance, the elliptic-curve digital-signature algorithm ECDSA needs conventional modular arithmetic too for computing the signature pair (r, s): $r = R_x \bmod n$, $s = k^{-1}(e + dr)$ [6]. The vast majority of finite-field operations is needed to compute the scalar multiplication $(R_x, R_y) = k \cdot (G_x, G_y)$, which is the central operation of ECC. The security of all EC cryptographic primitives relies on this operation. The computation of $R = k \cdot G$ is based on repeated point doubling and point addition: $R = \sum_{i=0}^{m-1} k_i 2^i \cdot G$, $k_i \in \{0, 1\}$. Formulas for point doubling and point addition are computed by arithmetic operations in the finite field underlying the elliptic curve. A point operation comprehends roughly 10 finite-field multiplications, several additions, and other simple operations. Most often, EC points are represented in projective coordinates to avoid the most costly finite-field operation: inversion.

The scalar k has roughly 200 bits. Thus, the computation of an EC primitives comprises approximately 300 point operations which in turn require roughly 3000 finite-field multiplications. The efficiency of finite-field multiplication determines significantly the overall efficiency of an ECC implementation. The silicon size and the computing speed are determined widely by finite-field multiplication. Optimizing finite-field multiplication is the key for efficient ECC implementations.

Another very basic design decision is the question whether the 200-bit finite-field elements should be processed at full precision or at smaller wordsizes—let's say 32 or 64 bits. Processing operands at full precision allows fast data transfer between computing elements and memories. This approach is also very efficient from the computational point of view. It lowers the computing complexity of ECC from $\mathcal{O}(m^3)$ to $\mathcal{O}(m^2)$ when m ist the wordsize [10]. The reason for this is that full-precision multiplication has a complexity of $\mathcal{O}(m)$. When operating on smaller wordsizes it has a complexity of $\mathcal{O}(m^2)$.

The modular multiplication is the crucial operation of ECC. Thus, it is most efficient to improve the multiplier. For instance, increasing the multiplier's degree of parallelism improves the operational speed more than any other measure

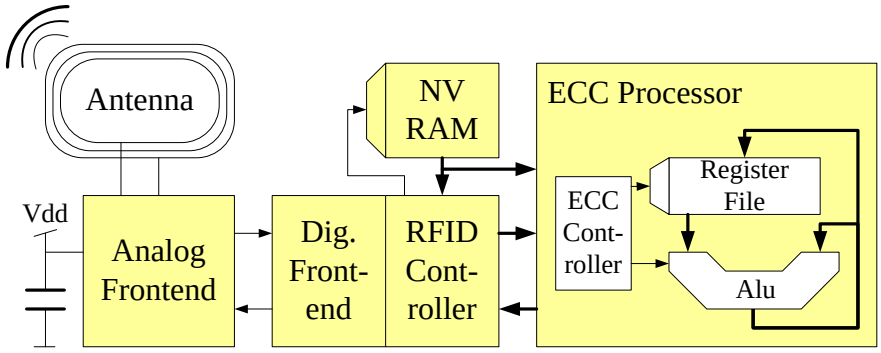


Fig. 1. Architecture of an ECC-enabled RFID tag.

having the same cost. For most RFID applications, a bit-serial multiplier will suffice.

Similar considerations can be made about other operations. Accelerating operations like inversion, which are used only infrequently, does not pay off. Providing extra hardware for inversion will not improve the area-delay product of the circuit. It is more advisable to skillfully reuse existing circuitry. For instance, exponentiation can compute inversion by iterating multiplications.

3.1 Related Work

In literature, only a few hardware implementations of ECC try to minimize silicon area and to consume low power [3, 8]. None of these ECC implementations were designed with RFID as intended target application. Nevertheless, they have similar optimization goals. The elliptic-curve digital-signature chip of R. Schroeppel et al. is able to calculate a complete EC signature [8]. It includes a hash module and a random number generator. It is not able to produce standard-conform signatures because it uses a modified signature scheme to avoid inversion in GF(p) and operates in the tower field GF(2^{89^2}) that is not recommended by any standard.

J. Goodman et al. presented a VLSI implementation of an ECC processor [3]. Their so-called domain-specific reconfigurable cryptographic processor can calculate all operations required for elliptic-curve cryptography including inversion. The datapath has limited possibilities for reconfiguration and allows to adapt the hardware to shorter finite fields. A 1024-bit silicon implementation of the processor has a silicon area of 8.4 mm^2 on a 0.25 μm CMOS process. No performance figures are given for EC operations. The processor is optimized for low-power operation.

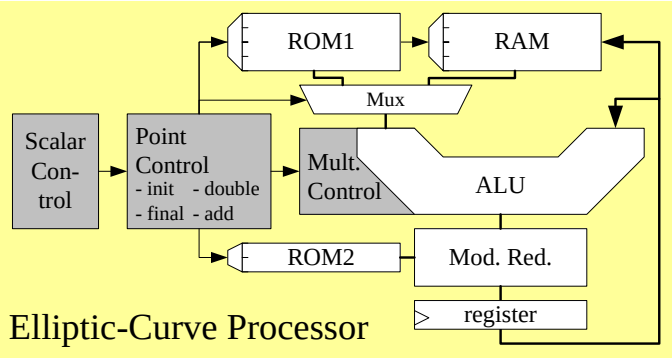


Fig. 2. Architecture of the ECC processor.

4 Architecture of an ECC-Enabled RFID Tag

Our proposal for computing ECC on RFID tags is an ECC processor tailored for the RFID specific requirements. The whole tag including the ECC processor is depicted in 1. The ECC processor is a dedicated hardware implementation. Its main component is a sophisticated dual-field arithmetic unit, which is presented in detail in §5.

The ECC-enhanced RFID tag has typical components like any other RFID tag has: An analog frontend supplies the chip with power by rectifying the 13.56 *MHz* carrier received from the reader. The analog frontend also extracts a clock signal from the carrier, demodulates communication from the reader to the tag and does the load modulation for sending responses to the reader. The digital frontend and the RFID controller handle the ISO-18000 protocol. The non-volatile RAM stores the unique ID of the tag, which is used for identification and anti collision. The proposed ECC-enabled tag stores also EC data in this memory. EC data comprehends the private key and possibly the public key and other certificate-related data.

The ECC processor is a stand-alone processor which needs no external interaction for computing EC operations. It consists mainly of an arithmetic unit, which actually does all the computations. Further components are a register file for storing EC parameters and intermediate results during the computation. A control unit sequences the operations of the arithmetic unit and addresses the register file.

The ECC processor computes all arithmetic operations in the Montgomery domain. It uses an arithmetic unit that operates on the full wordsize. The sequence of arithmetic operations for computing ECC must assure that intermediate results cannot grow larger than the hardware size of the arithmetic unit. Therefore, the hardware is some bits larger than the wordsize of the EC parameters.

The ECC processor is depicted in Figure 2. It is optimized for use in heavily constrained systems and for applications where EC parameters do not change during the product's lifetime. This is definitely the case for RFID tags. A whole batch of RFID tags will be produced sharing the same public EC parameters. Thus, it is economical to optimize the ECC processor for these parameters and to produce an own set of reticles for production to save silicon area. A considerable amount of silicon area can be saved when EC parameters are fixed. On one hand, they do not have to be stored in the costly non-volatile memory. And on the other hand, the ECC processor has not to provide a large register file where they are loaded to during processing. Figure 2 shows that EC parameters can be stored in two small ROM blocks. ROM2 has only two entries: the modulus and the order n of the base-point G. ROM1 has only four entries: the remaining curve parameters. ROMs of this size can be easily realized by combinational logic rather than true ROM circuits. The RAM block needs eight entries to store all intermediate results during computation. RAMs of this size can be implemented as a register files made up of flip-flops or latches. Clock gating is an appropriate technique to lower the power consumption of register files.

Most of the ECC processor's area is occupied by the full-precision datapath consisting of the arithmetic unit and the memories. Control units, which steer modular multiplication and sequence finite-field operations to compute EC point operations, are much smaller in comparison. Nevertheless, controlling ECC operations is not a trivial task. In order to reduce the complexity of the control tasks the functionality is split into three different levels as shown in Figure 2. One unit controls finite-field arithmetic, another unit controls EC-point operations including the conversion of points into affine coordinates. The third unit controls the scalar multiplication. The control unit for finite-field operations, is constricted to count the cycles of modular multiplication which is the only instruction of the arithmetic unit which takes more than one cycle. For the remaining control tasks, an highly optimized 4-bit RISC controller is used. The RISC controller executes one operation per clock cycle and has a proprietary instruction set that is tailored to control hardware datapaths efficiently. It does not compute any algorithms—it just steers the datapath. This task demands only a few instructions. Most of it are move instructions and bit-oriented instructions. Besides controlling EC operations, the controller can be reused to do simple IO tasks. We proved on FPGA prototypes that processing the ISO-18000 protocol can easily done by the RISC processor. Software for the controller is written in a macro-assembler language. The syntax of macros conforms to Java allowing to write fairly complex programs that are still concise. The compiled program ist stored in a ROM of 12-bit wordsize. Roughly 500 ROM entries are necessary for implementing ECC over GF(p), roughly 300 for ECC over GF(2^m). A RAM size of 8 bytes—organized in 4-bit nibbles—is sufficient for both ECC implementations.

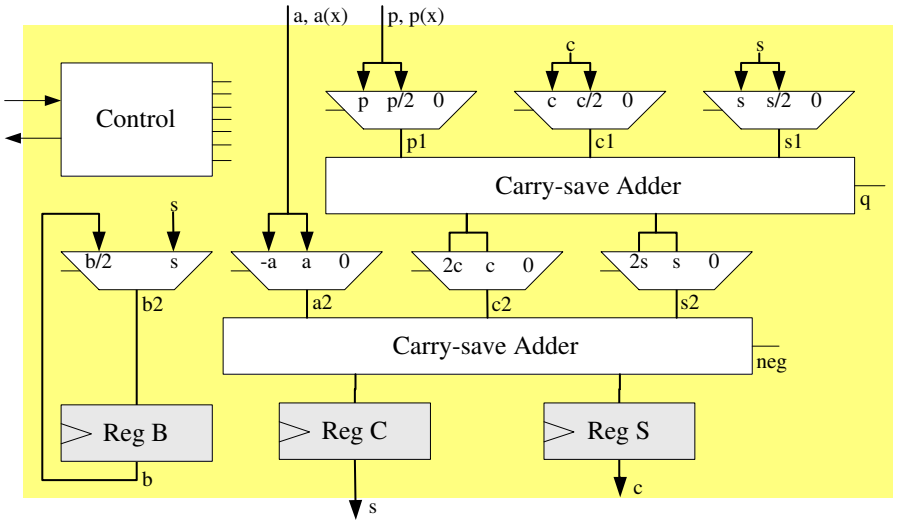


Fig. 3. Architecture of the dual-field arithmetic unit.

5 A Very Compact Arithmetic Unit for ECC

The arithmetic unit of the ECC processor is the most important part. It is a very compact arithmetic unit that combines operations in GF(2^m) and in GF(p) in a novel way. The compactness of the circuit is achieved by rigorously reusing hardware for implementing various finite-field operations. The dual-field arithmetic unit has the smallest footprint reported in literature. It supports operations like addition, subtraction, squaring, multiplication, and inversion in both fields. The complete set of instructions is needed for generation and verification of signatures over GF(2^m), which demands operations in GF(p) too.

The proposed dual-field arithmetic unit processes operands at full precision. It has a short critical path to prevent undesired power consumption caused by glitches. The critical path remains even short when the arithmetic unit is scaled for future key sizes. Low silicon area is guaranteed by using a bit-serial version of Montgomery's algorithm for multiplication. The GF(p)-multiplier hardware resources are rigorously reused for GF(2^m)-multiplication and for other operations. The bunch of operations offered by the arithmetic unit allows calculating the multiplicative inverse in both fields. The most stunning feature is its compactness in terms of silicon area: all the offered functionality costs only slightly more than a mere multiplier for operation in GF(p).

The dual-field arithmetic unit depicted in Figure 3 computes modular multiplication using the Montgomery method. Montgomery multiplication circumvents laborious remainder calculation [5]. It must neither calculate a computational-intensive division nor use quotient estimating techniques. Its preeminence is reasoned by the simplicity of its interleaved modular reduction step. Montgomery multiplication works also in GF(2^m). Both algorithms are quiet similar.

Table 1. Operations offered by the Montgomery dual-field arithmetic unit.

Operation		Operation	
Name	Function	Name	Function
clear	$(s, c) = (0, 0)$		
hold	$(s, c) = (s', c') \rightarrow (s'', 0)$	load	$(s, c) = (a, 0)$
add	$(s, c) = (s + a, c)$	sub	$(s, c) = (s - a, c)$
shftl	$(s, c) = (2s, 2c)$	shftr	$(s, c) = ((s + p \cdot q)/2, c/2)$
$\text{mul}_{i=0}$	$(s, c) = (a \cdot b_0, 0)$	$\text{mul}_{i>0}$	$(s, c) = ((s + p \cdot q)/2 + a \cdot b_i, c/2)$

This helps to save hardware resources in a dual-field approach. Both algorithms can use the same result register, the same partial-product generator, the same control logic, and the same shifter for b_i because the multiplier b is processed in both cases from LSB to MSB.

Montgomery multiplication calculates not directly a modular multiplication but $\text{MonMul}(a, b, p) = a \cdot b/R \bmod p$ where $R = 2^m \bmod p$. The constant m reflects the hardware size. For using Montgomery multiplication efficiently, input data has to be converted into the so-called Montgomery domain before calculation. A conversion back from the Montgomery domain is needed after all calculations have finished. Both conversions can be calculated by Montgomery multiplication and require no additional resources. These considerations are essentially the same for the finite field $\text{GF}(2^m)$.

When computing ECC, all curve parameters and input data have to be converted beforehand. After computing the EC scalar multiplication, the result has to be converted back from the Montgomery domain. If all EC parameters are pre-computed, the conversion requires only two Montgomery multiplications to obtain the result—a negligible overhead compared to the savings achieved by Montgomery multiplication.

The arithmetic unit is able to handle negative numbers which might result from subtractions in $\text{GF}(p)$. Although, they are converted immediately into their least non-negative residue to avoid sign-bit testing in subsequent multiplications. Another interesting feature of the arithmetic unit is the use of a redundant number representation for storing intermediate $\text{GF}(p)$ results. This allows scaling the arithmetic unit for arbitrary precision without affecting the maximum clock frequency. Results of $\text{GF}(p)$ operations have to be converted from redundant representation to binary representation before output. This saves memory, bus bandwidth, multiplexing logic, and eases communication.

The dual-field arithmetic unit offers many finite-field operations in $\text{GF}(2^m)$ and in $\text{GF}(p)$. Different operations reuse existing hardware resources skillfully. Table 1 shows a list of offered operations. The table does not list $\text{GF}(2^m)$-operations and $\text{GF}(p)$-operations individually because most operations are meaningful in both fields. The *clear* and the *load* operation are simple operation which are obviously useful. The *add* and *sub* operations take one operand from the register file and the other one from the result register. The partial product gen-

erator is able to pass a for addition and $\overline{a}$ for subtraction. The two's complement $-a = \overline{a} + 1$ is calculated by adding 1 at the lower carry-save adder.

The dual-field capability of the arithmetic unit is achieved by using different adders for GF(p) than for GF(2^m). Addition in GF(p) conforms to conventional integer addition whereas addition in GF(2^m) is a bit-wise XOR function. Addition in GF(p) makes use of full-adders by calculating a carry-save addition. Carries c_i are saved instead of being propagated to the next full adder. Addition in GF(2^m) is a bit-wise XOR function. The desired XOR function is sub-function of CSAs when the third input is fixed to 0. Thus, supporting operations in GF(2^m) causes nearly no overhead.

The *hold* operation maintains the results in the registers R and S. It is not a trivial operation as it seems—at least not for operation in GF(p). The hold operation can convert GF(p) integers from redundant representation into binary representation. During each hold cycle the two CSAs propagate the carry information until it vanishes. Issuing a hold operation for four cycles will convert nearly all integers smaller than 256 bits from their redundant representation into their binary representation because the longest carry chain is on average $\log_2 m$ bits. If GF(p) results are negative the modulus is added until the result is positive.

The most important operation offered the dual-field arithmetic unit is modular multiplication. As already mentioned, a bit-serial version of the Montgomery algorithm is used. To be more precisely, a modified version of Montgomery's original algorithm is used. Holger Orup presented variants of Montgomery's algorithm which are more convenient for hardware implementation [7]. Orup simplified the quotient determination at the cost of an additional iteration. The benefit for a hardware implementation is that the quotient q can be registered before it is used. This inhibits glitching activity which is crucial for low-power optimization. While multiplying, partial products $a \cdot b_i$ are accumulated. Multiplier bits b_i are processed from LSB to MSB.

Inversion in the Montgomery domain calculates $\mathrm{MonInv}(a') = a^{-1} \cdot R = a'^{-1} \cdot R^2$. Our implementation utilizes the extended Euclidean algorithm to calculate the division.

Remaining operations of the dual-field arithmetic unit are shift operations: *shiftl* and *shiftr*. The *shiftl* operation does not involve a modular reduction step. The *shiftr* operation is especially useful because it is required for calculating modular inverses by the extended Euclidean algorithm. When *shiftr* is executed, the modulus is added in case the value is odd. This operation is essentially the same as the aligning of intermediate result during multiplication. Shift operations without modular reduction are useful for serial input and output.

The architecture of the presented dual-field arithmetic unit corresponds widely to a pure GF(p) multiplier. This in turn, accounts for the low amount of resources used for hardware realization although many operations are offered. The arithmetic unit shown in Figure 3 mainly consists of two carry-save adders, some multiplexers, and three registers. Each of these components handles full-precision operands. The control unit generates control signals for the datapath by

Table 2. Cycle count of operations for different hardware sizes.

GF(p)				GF(2^m)			
Size	Mult.	Inv.	EC	Size	Mult.	Inv.	EC
192-bit	197	11,200	677,500	191-bit	197	6,200	426,300
224-bit	229	14,400	904,900	233-bit	241	7,500	635,100
256-bit	261	17,700	1,175,500	283-bit	289	8,800	920,600

decoding input operations. It contains a counter for controlling multiplications. Registers S and C store intermediate results (s, c). Register B stores the multiplier b which is processed bit-by-bit during multiplication. The upper carry-save adder is used for interleaved modular reduction. It adds an appropriate multiple of the modulus p to previous intermediate results (s, c). Carry-save adders (CSA) are implemented by conventional full-adder cells. The lower CSA accumulates partial products. Partial products are generated by a multiplexer: either a or 0 is selected.

In literature, J. Wolkerstorfer presented a dual-field arithmetic using a similar approach as we do [9]. His approach is based on a bit-serial dual-field multiplier. Contrary to our approach, conventional modular multiplication is used instead of using Montgomery's algorithm. His approach consumes 0.69 mm^2 on a 0.35 μm CMOS process. This is 50% larger than our approach.

6 Results Obtained

The results obtained by the new ECC processor are analyzed regarding performance in terms of cycle count and maximum clock frequency. Resource efficiency is measured in terms of silicon area and power consumption.

Most of the operations offered by the arithmetic unit are are single cycle operations. Only Montgomery multiplication is a multi-cycle operation. The actual number of cycles needed for multiplication depends on the hardware size and is shown in Table 2. Inversion is a compound operation. Using the extended Euclidean algorithm which exploits the *shiftr* operation, inversion takes on average 70 times longer than multiplication in GF(p) and 35 times longer than in GF(2^m) (see Table 2). These timings advise to use projective coordinates for implementing elliptic-curve cryptography to avoid inversions. Table 2 also gives cycle counts for running an EC scalar multiplication. The numbers include back-conversion of results from projective coordinates to affine coordinates. A Montgomery ladder was used for calculating the scalar multiplication in both GF(p) and in GF(2^m). The cycle count includes overhead for control.

The presented ECC processor was realized using a VHDL description. The VHDL model was synthesized on a 0.35 μm CMOS process. The amount of required hardware resources grows linearly with the wordsize parameter of the arithmetic unit. This is not surprising because the fraction of hardware resources required for control is small ($< 13\%$). The largest amount of control resources

Table 3. Silicon area of the ECC processor on a 0.35 μm CMOS process.

EC size $[bit]$	HW size $[bit]$	Area: arith+ram+risc+rom$_{risc}$ $[mm^2]$	Gate count $[GE]$	f_{max} $[MHz]$
192	196	0.449 + 0.662 + 0.079 + 0.12 = 1.31	23,800	68.5
224	228	0.544 + 0.776 + 0.076 + 0.12 = 1.51	27,500	68.5
256	260	0.615 + 0.888 + 0.077 + 0.12 = 1.70	31,000	68.5

requires the program memory which (0.12 mm^2). The standard-cell circuitry of the RISC controller uses only 0.08 mm^2. Table 3 details the area demands. A 196-bit arithmetic unit has an area of 0.45 mm^2 which equals 8,200 gate equivalents (36 %). EC parameters are stored in two small ROMs, which are realized as combinational logic. These ROMs do not contribute to the total area ($< 0.1\%$). In contrast, the register file, which is realized by flip-flops, contributes with 51 % to the total area.

The performance of the ECC processor depends on its maximum clock frequency. The clock frequency is determined by the critical path which is inside the RISC processor. On the 0.35 μm CMOS process, a maximum clock frequency of 68.5 MHz is possible. This allows to compute nearly 100 EC operations over GF(p_{192}) per second. GF(2^{191}) performance is even higher: 150 EC operations per second. The length of the critical path does not depend on the hardware size. The short critical path can be exploited for low-power operation: running the arithmetic unit not at maximum clock frequency allows decreasing the supply voltage. This can effect substantial power savings. Spice-level accurate simulation of the circuit using the NanoSim simulator from Synopsys indicate an average current of 80.0 μA at 1 MHz, 3.3 V for the 196-bit datapath. The complete ECC processor has an estimated total power consumption of 500 $\mu W/MHz$. Operation at much lower supply voltages is possible. At 1.5 V, the power consumption will be approximately 125 $\mu W/MHz$.

The proposed ECC processor is very competitive with related work regarding silicon area and power consumption. The elliptic-curve digital-signature chip of R. Schroeppel et al. has a complexity of 191,000 gate equivalents [8]. It takes 4.4 ms for a signature when operated at 20 MHz. No power figures are given.

J. Goodman et al. reconfigurable cryptographic processor has a bit-slice architecture [3]. If we shorten their 1024-bit architecture to 196 bits and move from 0.25 μm to 0.35 μm CMOS, it will require roughly 3.15 mm^2. This is 140 % more than our approach. The clock frequency on a 0.25 μm CMOS process is stated to be 50 MHz. No performance figures are given for ECC but they must be similar to ours when using same EC algorithms. The power consumption is normally 1.5 mW/MHz.

The silicon area of the 192-bit ECC processor is 1.31 mm^2. This is nearly acceptable for security enhanced RFID tags. An interesting question is how this size decreases when moving to more advanced process technologies. Table 4 summarizes the estimates for moving to 180-nm and 90-nm technologies. 180-

Table 4. Scaling the ECC processor to 180-nm and 90-nm CMOS technologies.

CMOS Technology			ECC Processor					
CMOS l_{gate}	VDD $[V]$	Power/gate $[nW/MHz/gate]$	Area $[mm^2]$	Power ECC $[\mu W/MHz]$	Budget $[\mu W]$	f_{ECC} $[kHz]$	GF(p_{192}) $[s]$	GF(2^{191}) $[s]$
0.35 μm	3.3	45	1.31	500	30	60	11.3	7.1
180 nm	1.8	15	0.35	170	30	175	3.9	2.5
90 nm	1.0	5	0.09	55	30	545	1.3	0.8

nm CMOS is a familiar technology for manufacturing RFID tags. 90-nm CMOS might be used in some years. Manufacturing the ECC processor on a 180-nm technology shrinks the circuit size to 0.35 mm^2—this well below the required 1 mm^2. At 90-nm, the size of the ECC processor is only 0.09 mm^2.

Analyzing the power consumption of the ECC processor on different CMOS technologies shows a less convincing picture. On the 0.35-μm technology, the assumed power budget of RFID tags of 30 μW is exhausted, when the processor is clocked with 60 kHz only. It is assumed that the power budget does not change when the tag is manufactured using a different technology. Moreover, we assume that only 50% of the ECC processor is active because 90% of the register file, which takes 51% of the total area, has surely no activity due to clock gating. To meet the power requirements, the clock frequency can be lowered. Clocking the ECC processor at very low frequencies causes long computation times: 11.3 seconds for a EC operation over GF(p_{192}). Operation over GF(2^m) is somewhat faster. Moving to more advanced technologies allows to clock the circuit at 175 kHz and at 545 kHz, respectively, which lowers the computation times to 3.9 s and 1.3 s. The computation time of EC operations is rather long. This has to be considered by the protocol layer. By interleaving requests and responses, a large number of tags can perform security functions without deteriorating the throughput.

7 Conclusion

In this article we presented an ECC processor that fits the requirements of RFID tags. A central element of the ECC processor is a novel arithmetic unit. The dual-field arithmetic unit is able to calculate all arithmetic operations required for ECC computation: addition, multiplication, and inversion in the finite fields GF(p) and GF(2^m). Bit-serial multiplication is calculated by an improved version of Montgomery's algorithm. A 196-bit implementation of the ECC processor on a 0.35 μm CMOS process has an area of 1.31 mm^2 and can be clocked with 68.5 MHz. On a 180 nm CMOS process, the area will shrink to 0.35 mm^2 which is acceptable for RFID tags.

Although ECC is small enough to fit RFID tags, the power consumption of ECC operations is still a problem that has to be solved. Our approach to meet the power requirements of RFID tags by lowering the operational frequency of the

ECC processor might inhibit applications where latency is of importance. The estimated computation time of 0.8 s up 3.9 s might jeopardize the acceptance. In future, we have to look out for even more power efficient implementations and utilize possibilities on all layers. For instance, the circuit could be made more power efficient by replacing the RISC controller by hardwired control that demands no program ROM. The register file has also a scope for improvement by moving to a latch-based design. Beyond the ECC processor are opportunities too: Cryptographic functionality could only be made available when the tag is close to the reader where the power supply is much better. Another possibility is adopting the clock frequency to the available power.

References

1. European Commission - ARTICLE 29 Data Protection Working Party. Working document on data protection issues related to RFID technology, January 2005. Available online at http://www.europa.eu.int/comm/privacy.
2. Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2004.
3. James Goodman and Anantha Chandrakasan. An Energy Efficient Reconfigurable Public-Key Cryptography Processor Architecture. *IEEE Journal of Solid-State Circuits*, 36(11):1808–1820, November 2001.
4. International Organization for Standardization (ISO). ISO/IEC 18000-3: Information Technology AIDC Techniques — RFID for Item Management, March 2003.
5. Peter Montgomery. Modular Multiplication without Trial Division. *Mathematics of Computation*, 44:519–521, 1985.
6. National Institute of Standards and Technology (NIST). FIPS-186-2: Digital Signature Standard (DSS), January 2000. Available online at http://www.itl.nist.gov/fipspubs/.
7. Holger Orup. Simplifying Quotient Determination in High-Radix Modular Multiplication. In Simon Knowles and William H. McAllister, editors, *12th Symposium on Computer Arithmetic*, pages 193–199. IEEE Computer Society Press, 1995.
8. Richard Schroeppel, Cheryl Beaver, Rita Gonzales, Russell Miller, and Tim Draelos. A Low-Power Design for an Elliptic Curve Digital Signature Chip. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 2003.
9. Johannes Wolkerstorfer. Dual-Field Arithmetic Unit for GF(p) and GF(2^m). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems—CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2003.
10. Johannes Wolkerstorfer. *Hardware Aspects of Elliptic Curve Cryptography*. PhD thesis, Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, August 2004.

Noisy cryptographic protocols for low cost RFID tags *

Hervé Chabanne[1] Guillaume Fumaroli[2]

[1] SAGEM S.A – Av. du Gros Chêne – 95610 Eragny-sur-Oise (France)
herve.chabanne@sagem.com
[2] AXALTO S.A – 50 av. Jean Jaurès BP 620-12 – 92545 Montrouge (France)
gfumaroli@axalto.com

Abstract

We here prove the feasibility of exchanging some secret data between an RFID tag and its reader by public discussion. For this, the inherent noise on their communication link is exploited and classical protocols are adapted to these small devices. More precisely, we first present the canvas of our study and discuss the advantage distillation phase. Then, we show how Brassard and Sailvail's Cascade protocol can be modified in order to reduce the hardware implementation cost while still maintaining adequate correction rate and tolerable leaked information during the reconciliation phase. Finally, as for the privacy amplification phase, we point out Kaan Yüksel's work on low-cost universal hash functions, achieving to allege that public discussion under noisy environment might be an interesting possibility for low-cost RFID tags.

1 Introduction

An RFID tag is a small device which consists of an integrated circuit attached to an antenna capable of transmitting wirelessly a sole identifier at several meters to a reading device in response to a query.
Securing the RFID systems transmissions is of great concern [8]. Its difficulty comes mainly from two major problems. First, the distribution of keys to billions of products. Second, the inability for such low cost devices to handle classical arithmetic based solutions. Many solutions have been proposed so far [11].
Here we go through a completely different track and use the channel noise as it is done in [1], so as to suggest a protocol whose hardware implementation is simple and which ensures that the exchanges in an RFID system be confidential when a passive eavesdropper is present.
After an initialization step in which a damaged version of a bit string sent by a reader is received by a tag and possibly an eavesdropper, the protocol consists essentially in three phases:

*The work presented in this paper has been exclusively supported by SAGEM S.A.

- Advantage distillation – The legitimate parties turn the situation to their advantage if necessary,
- Information reconciliation – They apply correction techniques to come to a common string about which the adversary only has partial information,
- Privacy amplification – By applying a universal hash function, they obtain another string about which the adversary almost has no information.

First in section 2, we show the benefit of increasing the initial given advantage between the RFID tag and its reader against an eavesdropper in order to straighten out the subsequent computations. As for the information reconciliation phase, a slightly modified version of Cascade optimized for low-cost hardware implementation is described and analyzed in section 3. Then in section 4, the choice of a universal class of hash functions in the privacy amplification phase is motivated. Yet, a complete scenario for our proposed design is summarized in section 5. Section 6 concludes.

2 Gaining the advantage

The so-called satellite scenario can be described as follows: A bit string S sent by a satellite is received by three entities **R**, **T** and **E** as S_R, S_T and S_E with different noise patterns characterized respectively by p_R, p_T and p_E. **R** and **T** can subsequently communicate over an error free channel while **E** is eavesdropping their communication.
The probability that a given bit r from S is received as x is given by:

$$\begin{aligned} p_{S_R|S=r}(x) &= (1-p_R)^{n-d_H(x,r)} p_R^{d_H(x,r)} \\ p_{S_T|S=r}(x) &= (1-p_T)^{n-d_H(x,r)} p_T^{d_H(x,r)} \\ p_{S_E|S=r}(x) &= (1-p_E)^{n-d_H(x,r)} p_E^{d_H(x,r)} \end{aligned}$$

where d_H denote the Hamming distance.
The worst case is achieved when both p_R and p_T are greater than p_E. Should that be the case, **R** and **T** have to perform an "advantage distillation" phase to gain the advantage over **E** i.e. to eventually get less errors than **E**.
The Bit Pair Iteration Protocol introduced in [4] turns out to be a quite efficient one implementing the advantage distillation phase. **R** and **T** group their bits by pair and then tell each other the parity of each pair. If both parities do not match, then **R** and **T** get rid of the pair. Otherwise, they undertake to keep the information associated with the involved pair while giving **E** as little information as possible. Namely, they keep only the first bit of the pair since **E** globally got one bit of information about the pair from its parity. The retained bit might still differ, but it can be shown that **R** and **T**'s bits agree more and more each time the process is repeated [4].
For our purpose, let **R** be the reader, **T** be the tag and **E** be some passive eavesdropper. Then either **R** or **T** has to send the initial string S in the satellite's place. If **R** is the initial sender (see Figure 1), then $p_R = 0$ and $p_T > 0$.
Let the communication channels $R \to T$ and $R \to E$ be considered independant and **T**'s version of the string taken as reference. The previous scenario is then equivalent to **T** having sent the string and **R** and **E** having received it with

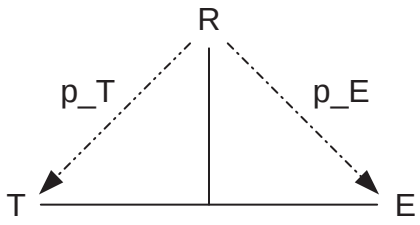


Figure 1: Actual scenario

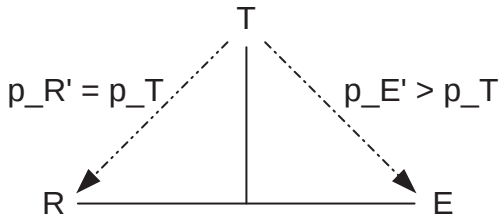


Figure 2: Equivalent scenario

independant noise patterns respectively characterized by $p'_R = p_T$ and $p'_E = p_E + p_T - 2p_E p_T$ (see Figure 2).
Hence $p'_R < p'_E$ even if $p_E < p_T$ at the outset.
Let

$$h : x \mapsto -\sum_{x \in \mathcal{X}} x \log_2 x$$

denote the Shannon bit entropy function for some random variable x on a set $\mathcal{X}$.
A string received with probability $x \in \mathcal{X}$ provides $I(x) = 1 - h(x)$ information bit. Let $I_R = I(p'_R)$ (resp. $I_E = I(p'_E)$) be the information rate learnt by R (resp. by E). Since h is strictly decreasing on $[0, 1/2]$, we get $I_E < I_R$ in terms of Shannon information.
Under these conditions, **R** and **T** always have an advantage over **E**, a fact already known to Wyner in [9].
Note that the advantage distillation phase is not necessary anymore whatever low $p_E > 0$ may be. Practically however, implementing the Bit Pair Iteration Protocol in the first stage provides one with an effective way of increasing both the reliability of **R** and **T**'s string as well as the eavesdropper's disadvantage $I_R - I_E$.

3 A low cost reconciliation protocol

Some errors in **R**'s string may remain. During the information reconciliation phase, **R** and **T** exchange some information to correct these errors. Cascade, introduced in [2], is built so that **R** and **T** efficiently correct their errors while maintaining the information leaked to **E** relatively low (see [2] for a detailed description of Cascade). Cascade's performance is actually very close to the

Shannon bound in terms of amount of leaked information. However, Cascade would be too complex to fit into simple low cost tags.
Practically, when the error rate is sufficiently low – which can be easily achieved by performing enough Bit Pair Iteration protocol passes – most errors are corrected during the first pass of Cascade. From this observation, we propose introducing two major changes in Cascade.

- First, the same block size is set for every pass. The block size should also divide the string size, so that only fixed length blocks have to be analyzed.
- Second, a permutation σ is set once and for all and cabled inside the tag. It is hence straightforward to apply it to the string. On the contrary, choosing the permutation at random and sending it through the communication channel at the beginning of each pass as required in Cascade would have been infeasible in low cost tags.

Much less efficient than Cascade but also much easier to implement, our protocol still converges in the stated context.

The proposed reconciliation protocol

Let

- n be the length of the strings to be reconciled,
- k be the block size with $k|n$,
- σ be a pre-cabled permutation in the set of all bijections of $\{0, \ldots, n-1\}$.

The protocol is composed of several identical passes. Let x_0 and y_0 respectively denote **R** and **T**'s string at the beginning of the protocol. The i-th pass of our protocol is described as follows:

1. **R** and **T** respectively compute $x_i = \sigma(x_{i-1})$ and $y_i = \sigma(y_{i-1})$.
2. **R** and **T** divide x_i and y_i in n/k blocks.
3. For j from 1 to n/k,
   (a) Let $x_i(j)$ and $y_i(j)$ denote the j-th block of R and T string respectively. If the parity of $x_i(j)$ and $y_i(j)$ are the same, **R** and **T** continue with the next block (or the next pass if all blocks in the current pass have already been checked out). Otherwise, they perform a dichotomic search which returns a position l such that $x_i(j)[l] \neq y_i(j)[l]$.
   (b) **R** invert $x_i(j)[l]$ to correct the error.

Protocol analysis

Estimating the amount of leaked information during the reconciliation phase is needed by the subsequent phases and achieved through the following proposition.

Proposition 1. *Let k be the block size, $e^{(i)}$ (resp. $d^{(i)}$) be the bit error rate (resp. the bit leak rate) after i passes of the reconciliation protocol. We have:*

1. $$\forall i > 0 \quad e^{(i)} = e^{(i-1)} - \frac{1 - \left(1 - 2e^{(i-1)}\right)^k}{2k}$$
   where $e^{(0)}$ denotes the bit error rate at the beginning of the reconciliation protocol

2. $$\forall i \geq 0 \quad d^{(i)} = \frac{i}{k} + \left(e^{(0)} - e^{(i)}\right) \lceil \log k \rceil$$

Proof.

1. Let X be a random variable representing the number of errors in a given block of size k when the string's bit error probability is e. If the errors are uniformely distributed at the beginning of the protocol and the permutation is chosen at random among all permutations of $\{0, \ldots, n-1\}$(or has adequate properties, see the following section), it is legitimate to consider that these errors remain uniformely distributed within the string at the beginning of each pass. Thus, X can be approximated by a binomial law with parameters (k, e). Let α_1 be the probability that X is odd, we have
   $$\alpha_1(k, e) = \sum_{l=1}^{\lceil k/2 \rceil} \binom{k}{2l-1} e^{2l-1}(1-e)^{k-2l+1} = \frac{1 - (1-2e)^k}{2} \ .$$
   Since one error per odd parity block is corrected, we have
   $$\forall i > 0 \quad e^{(i)} = e^{(i-1)} - \frac{\alpha_1\left(k, e^{(i-1)}\right)}{k} \ .$$

2. Let us consider the j-th pass of the protocol with $j \in \{1, \ldots, i\}$. For each block, at least one bit is revealed for parity testing. If the block's parity is odd, then $\lceil \log k \rceil$ more bits are revealed to locate the error. Thus, the bit leak rate during the j-th pass is given by
   $$\frac{1}{k}\left(1 + \alpha_1(k, e^{(j-1)}) \lceil \log k \rceil\right) \ .$$
   Hence,
   $$\forall i > 0 \quad d^{(i)} = \sum_{j=1}^{i} \frac{1}{k}\left(1 + \alpha_1(k, e^{(j-1)}) \lceil \log k \rceil\right) = \frac{i}{k} + \left(e^{(0)} - e^{(i)}\right) \lceil \log k \rceil \ .$$
   The formula also holds for $i = 0$ corresponding to the trivial case $d^{(0)} = 0$.

□

Smaller k obviously lead to cheapest hardware implementation of the protocol and faster bit error rate decrease (see Figure 3). However, the parameter k cannot be chosen too small because it also leads to higher bit leak rates (see Figure 4). This analysis shows that there is a trade-off between error correction rate and leaked information rate according to the initial bit error rate and available gate count.

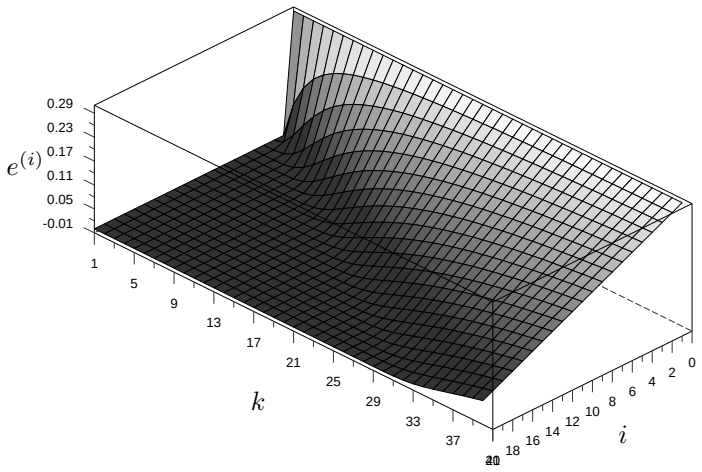


Figure 3: Bit error rate $e^{(i)}$ in function of the block size k and the number of passes i

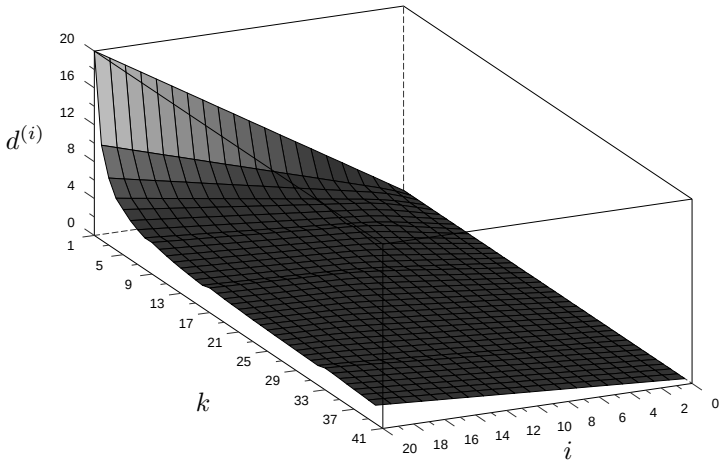


Figure 4: Bit leak rate $d^{(i)}$ in function of the block size k and the number of passes i

Choice of a permutation

The estimate of remaining errors pass after pass is based on the hypothesis that the permutation is chosen at random. However, the choice of a permutation with adequate properties proves sufficient practically.
An adequate permutation for our reconciliation protocol should map distinct positions in a given block to distinct blocks. This would guarantee the composition of the blocks be very different from pass to pass.
The rest of the section formally describes a quality measure for such a candidate permutation.
Let $X_{j,\ j\in\{1,\ldots,n/k\}}$ denote the set containing the positions in the j-th block:

$$X_j = \{x \mid (j-1)k \leq x < jk\}$$

Let $M_j(\sigma)$ denote the cardinality of the set of all elements $X \in \mathcal{P}(X_j, l)$ satisfying

$$\forall x, y \in X, x \neq y \implies (\forall j' \in \{1, \ldots, n/k\}, \sigma(x) \in X_{j'} \implies \sigma(y) \notin X_{j'})$$

where $\mathcal{P}(\Omega, \lambda)$ denote the set of all subsets of Ω of size λ.
Eventually, an adequate measure M according to the stated reconciliation problem for a permutation σ might be defined as

$$M(\sigma) = \sum_{j=1}^{n/k} M_j(\sigma)$$

Suppose $l \leq n/k$. An ideal permutation σ^* according to M and l is such that

$$M(\sigma^*) = \frac{n}{k}\binom{k}{l}$$

The most accurate M is reached with $l = \min(k, n/k)$.

4 Towards privacy amplification

At the end of the reconciliation protocol, **R** and **T** agreed on a string with very high probability. In this last phase, they publicly pick a compression function G which, applied to this partially secret string, allows them to derive a shorter – but almost perfectly secure – key K. Thereby, K can be chosen has a secret key during the subsequent exchanges in, for example, the so-called one-time pad encryption scheme.
The compression function is actually chosen from a universal class of hash functions we introduce in the following definition.

Definition (Universal class of functions)**.** *A class $\mathcal{F}$ of functions from A to B is* universal *if, for all pairs (x_1, x_2) of distinct elements in A, the probability that the event $f(x_1) = f(x_2)$ occurs is at most $1/|B|$ when f is chosen randomly uniformely in $\mathcal{F}$.*

Some universal class of functions are quite easy to implement. Actually, in [10] such a class is exhibited wich requires less than 460 logic gates.

Discussion on the achievable key length

The following theorems allow us to derive the length of K.

Theorem 1 (Bennett, Brassard, Crépeau, Maurer [1])**.** *Let X be a random variable with values in the set $\mathcal{X}$, and G be another random variable corresponding to the choice of an element in a universal class of hash functions $\mathcal{X} \to \{0,1\}^r$ according to a uniform distribution. Then,*

$$H(G(X)|G) \geq H_C(G(X)|G) \geq r - \log\left(1 + 2^{r-H_C(X)}\right) \geq r - \frac{2^{r-H_C(X)}}{\ln 2} \ .$$

Theorem 2 (Cachin, Maurer [3])**.** *Let X and U be random variables with alphabets $\mathcal{X}$ and $\mathcal{U}$ respectively, and let $s > 0$ be an arbitrary security parameter. With probability at least $1 - 2^{-s}$, U takes on a value u for which*

$$H_C(X) - H_C(X|U = u) \leq 2\log|\mathcal{U}| + 2s \ .$$

Proposition 2 (Achievable secret key length)**.**
Let

- *n be the length of the strings at the beginning of the reconciliation phase,*
- *p be $\mathbf{T}$ and $\mathbf{E}$'s relative bit error rate,*
- *D be the number of bits revealed during the reconciliation,*
- *s et s' be two security parameters.*

Then the final secret string length is $nh_C(p) - 2D - 2s - s'$ about which $\mathbf{E}$ only learns $2^{-s'}/\ln(2)$ information bits with probability at least $1 - 2^{-s}$. $h_C(x) = -log(x^2 + (1-x^2))$ here denotes the bit collision entropy.

Proof. See [3]. □

5 Summary

The hardware implementation of these protocols is easily scalable. It can be optimized to reach a compromise with communication efficiency and gate count. Suppose that $\mathbf{R}$ broadcasts a m-bit string that is received by $\mathbf{T}$ and $\mathbf{E}$ with a bit error rate respectively p_T and p_E.
After this initialization phase, considering $\mathbf{T}$ owns the reference version, $\mathbf{R}$ and $\mathbf{E}$'s version are the image of $\mathbf{T}$'s string received through a binary symmetric channel with a bit error rate respectively $p'_R = p_T$ and $p'_E = p_E + p_T - 2\,p_E\,p_T > p'_R$.
Although $\mathbf{E}$'s bit error rate is higher than $\mathbf{R}$'s, several passes of the Bit Pair Iteration protocol are performed. In so doing, the advantage is increased while $\mathbf{R}$'s bit errors compared to $\mathbf{T}$'s are decreased so that less information bits are needed during the reconciliation phase. The Bit Pair Iteration Protocol leads to a reduction rate (see [4] for the details) thus we get a new shorter string of length n.
With regard to $\mathbf{T}$, $\mathbf{R}$'s bit error probability is $e^{(0)}$ while $\mathbf{E}$'s is p. $\mathbf{E}$'s collision entropy at this time is hence estimated at $h_C(p)$.

As regards the reconciliation phase, our reconciliation protocol is implemented with an ad-hoc block length. N passes of our protocol are performed where N is chosen such that **R** and **T**'s shared errors evaluates to $ne^{(N)} \ll 1$ while the number of revealed bits $D = nd^{(N)}$ is not too high. Both $e^{(N)}$ and $d^{(N)}$ are computed using the proposition 1. **R** and **T** share a partially secret string of length n.
In a last phase, **R** and **T** apply a universal hash function to this partially secret string. An advisable choice is the universal class of hash functions proposed in [10] especially well suited for our context. The security parameter being set to s and s', $nh_C(p) - 2D - 2s - s'$ highly secret information bits can actually be distilled from the n only partially secure bits. More precisely, with probability at least $1 - 2^{-s}$ which can be very close to 1 provided s is big enough, **E** learns at most $2^{-s'}/\ln(2)$ bit about this highly secret string.

6 Conclusion

We here show how to exploit the noisy environment of RFID tags to circumvent low-end eavesdroppers. Our solution requires some bandwith and few gates. Moreover, no key management is needed. Though not tested against physical experimentation, the feasibility of our scenario is very likely provided low signal to noise ratio during the initialization phase. The decorrelated part of the noise should also be sufficient for the independance condition to practically hold. Eventually, our approach seems pragmatic for this difficult problem.

References

[1] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli Maurer, *Generalized privacy amplification*, IEEE Transaction on Information Theory **41** (1995), no. 6, 1915–1923.

[2] Gilles Brassard and Louis Salvail, *Secret-key reconciliation by public discussion*, EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology, Springer-Verlag New York, Inc., 1994, pp. 410–423.

[3] Christian Cachin and Ueli Maurer, *Linking information reconciliation and privacy amplification*, Journal of Cryptology **10** (1997), no. 2, 97–110.

[4] Martin Gander and Ueli Maurer, *On the secret-key rate of binary random variables*, Proc. 1994 IEEE International Symposium on Information Theory (Abstracts), 1994, p. 351.

[5] Ueli Maurer, *Secret key agreement by public discussion from common information*, IEEE Transaction on Information Theory **39** (1993), no. 3, 733–742.

[6] A. Rényi, *On measures of entropy and information*, Proceedings of 4th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, 1961, pp. 547–561.

[7] Claude E. Shannon, *Communication theory of secrecy systems*, Bell Sys. Tech. Journal **28** (1949), 656–715.

[8] Stephen August Weis, *Security and privacy in radio-frequency identification devices*, Master's thesis, Massachusetts Institute of Technology, May 2003, pp. 51–55.

[9] A.D. Wyner, *The wire-tap channel*, Bell System Technical Journal **54** (1975), 1355–1387.

[10] Kaan Yüksel, *Universal hashing for ultra-low-power cryptographic hardware applications*, Master's thesis, Worcester Polytechnic Institute, 4 2004.

[11] *Security and privacy in RFID systems*, 2003–2005, Web-based bibliography referenced at `http://lasecwww.epfl.ch/~gavoine/rfid/`.

Lightweight Key Exchange and Stream Cipher based solely on Tree Parity Machines

Markus Volkmer and Sebastian Wallner

Hamburg University of Technology
Department of Computer Engineering VI
D-21073 Hamburg, Germany
{markus.volkmer,wallner}@tu-harburg.de

Abstract. Alternative security solutions are considered in science and industry, motivated by the strong restrictions as they are often present in embedded security scenarios – especially in a RFID setting. We investigate a low hardware-complexity cryptosystem for lightweight symmetric key exchange and stream cipher based on Tree Parity Machines. The speed of a key exchange is basically only limited by the channel capacity as is the stream cipher throughput. This work significantly improves and extends previously published results on TPMRAs. Again, characteristics of standard-cell ASIC design realizations as IP-core in 0.18μ-CMOS technology are evaluated.

Keywords: Embedded Security, Lightweight Symmetric Key Exchange, Lightweight Stream Cipher, Tree Parity Machine

1 Key Exchange, Stream Ciphers and RFID

The investigation of alternative security primitives and technologies is stimulated by the strong restrictions present in resource-limited devices. In sensor networks, RFID-systems or Near Field Communication (NFC), the devices in use (as nodes of a network) can impose severe size limitations and power consumption constraints. The available size for additional cryptographic hardware components is often limited if not available at all [1, 2, 3]. The RFID-industry should have a particular interest in security, because the commercial prosperity of their products is directly linked to the secrecy of data via customer acceptance [2, 4]. To optimize a cost-performance-ratio regarding chip-area, channel bandwidth, power consumption and code-size with respect to a given platform (Microcontroller, FPGA, ASIC) represents a challenge in general [5].

Secure key exchange is considered most critical and complex in this context and of major importance with regard to security. Regarding applications in embedded systems, asymmetric (public-key) group-based cryptosystems based on Elliptic Curve Cryptography (ECC), the generalization to Hyper-Elliptic Curves (see e.g. [6]) and hardware-specific extensions for efficient arithmetic [7] are state-of-the-art. Without a reduction of the security, these representations allow to reduce the size of the numbers to calculate with. Yet, more complex expressions

need to be calculated. Also, the ring-based asymmetric cryptosystem NTRU [8, 9] calculates on rather small numbers.

According to Paar [5] implementations of ECC on 16-bit microprocessors (clock-frequency ≤ 50 *MHz*) are feasible, while RSA and Diffie-Hellman are still hard. On an 8-bit microprocessor (clock-frequency ≤ 10 *MHz*) only symmetric algorithms are considered applicable given low data rates. Asymmetric algorithms here require an additional crypt-coprocessor. As ECC requires more than 10000 gates and DES alone already demands a few 1000 gates, only lightweight stream ciphers are considered applicable for the extreme case of an RFID-tag with around 1000 gates and no microprocessor available (cf. [5]). Symmetric algorithms for this class are sought and stream ciphers are again considered for such niche applications [10]. Stream ciphers are regarded competitive with block ciphers when a small footprint in hardware implementations is required. Though the security aspects of RFID have not been standardized so far, the use of stream ciphers here seems forseeable due to the present constraints. Next to higher bandwidth, second generation RFID tags are planned to have improved security (encryption, password functions, authentication) and read/write capability. Key exchange requires read/write RFID devices for bidirectional communication and first tags with challenge and response authentication are developed [11, 12].

After all, a key exchange still remains of prohibitive cost and only stream ciphers seem to be applicable for encryption in strongly restricted domains. Seeking and investigating alternative approaches beyond efficient implementations of established primitives thus remains a challenge for research. In practice, a necessary tradeoff between the level of security and the available resources or computation time often has to be faced.

We suggest to discuss a hardware solution for lightweight symmetric key exchange and stream cipher based on so-called *Tree Parity Machines* [13]. We present a fully serial architecture-variant based on [14] using this key exchange concept and a trajectory mode, that allow for fast successive key generation and exchange, as well as for a synchronous stream chipher. It enables short key lifetimes through the achievable speed of a key exchange and consequently fast resynchronisation for the stream cipher. We focus on a low hardware-complexity IP-Core solution for resource-limited devices. Feasible frequent rekeying and variable key lengths allow for flexible security levels especially in environments with moderate security concerns.

2 Key Exchange by Tree Parity Machines

The fast synchronization of two interacting identically structured Tree Parity Machines (TPMs) is proposed by Kinzel and Kanter [13] as a method for symmetric key exchange. It does not involve large numbers and principles from number theory and is related to secret key agreement based on interaction over a public insecure channel as it is discussed under information theoretic aspects by Maurer and others [15, 16, 17, 18]. The exchange protocol is realized by an interactive adaptation (error-correction) process between the two interacting parties A and B. The TPM (see Figure 1a) consists of K independent summation units

$(1 \leq k \leq K)$ with non-overlapping inputs in a tree structure and a single parity unit at the output. Each summation unit receives different N inputs $(1 \leq j \leq N)$,

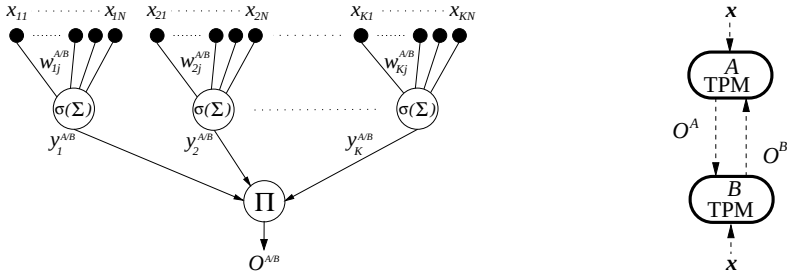


Fig. 1. (a) The Tree Parity Machine. A single output is calculated from the parity of the outputs of the summation units. (b) Outputs on commonly given inputs are exchanged between parties A and B for adaptation of their preliminary key.

leading to an input field of size $K \cdot N$. The vector-components are random variables with zero mean and unit variance. The output $O^{A/B}(t) \in \{-1, 1\}$ (A/B denotes equivalent operations for A and B), given bounded coefficients (weights) $w_{kj}^{A/B}(t) \in [-L, L] \subseteq \mathbb{Z}$ (from input unit j to summation unit k) and common random inputs $x_{kj}(t) \in \{-1, 1\}$, is calculated by a parity function of the signs of summations:

$$O^{A/B}(t) = \prod_{k=1}^{K} y_k^{A/B}(t) = \prod_{k=1}^{K} \sigma\left(\sum_{j=1}^{N} w_{kj}^{A/B}(t)\ x_{kj}(t)\right) . \tag{1}$$

$\sigma(\cdot)$ denotes the sign-function. The so-called *bit package* variant (cf. [13]) reduces transmissions of outputs by an order of magnitude down to a few packages. Parties A and B start with an individual randomly generated secret initial vector $w_{kj}^{A/B}(t_0)$. These initially uncorrelated random variables become correlated (identical) over time through the influence of the common inputs and the interactive adaptation as follows. After a set of $b > 1$ presented inputs, where b denotes the size of the bit package, the corresponding b TPM outputs (bits) $O^{A/B}(t)$ are exchanged over the public channel in one package (see Figure 1b). The b sequences of signs of the summation units $y_k^{A/B}(t) \in \{-1, 1\}$ are stored for the subsequent adaptation process. A hebbian learning rule adapts the coefficients (the preliminary key), using the b outputs and b sequences of signs. They are changed only on equal output bits $O^A(t) = O^B(t)$ at both parties. Furthermore, only coefficients of those summation units are changed, that agree with this output:

$$O^{A/B}(t) = y_k^{A/B}(t): \quad w_{kj}^{A/B}(t) := w_{kj}^{A/B}(t-1) + O^{A/B}(t)\ x_{kj}(t) . \tag{2}$$

Coefficients are always bound to remain in the maximum range $[-L, L] \subseteq \mathbb{Z}$ by reflection onto the boundary values. Iterating the above procedure in as an interactive protocol, each component of the preliminary key performs a random walk with reflecting boundaries. The resulting key space is of size $(2L+1)^{KN}$. Two corresponding components in $w^A_{kj}(t)$ and $w^B_{kj}(t)$ receive the same random component of the common input vector $x_{kj}(t)$. After each bounding operation, the distance between the two components is successively reduced to zero. When both parties adapted to produce each others outputs, they remain synchronous without further communication (see Equation 2) and continue to produce the same outputs on every commonly given input. Common coefficients are now present in both TPMs in each of the following iterations. This preliminary key can be used to derive a common time-dependent final key by privacy amplification [15, 16] or can be used directly. Furthermore, synchrony is achieved only for *common* inputs. Thus, keeping the common inputs secret between A and B can be used to have an (entity) authenticated key exchange. There are $2^{KN} - 1$ possible inputs in each iteration, yielding as many possible initializations for a pseudo random number generator.

2.1 Trajectory Mode, Security and Attacks

Again consider that when the two parties are synchronous they also have the same outputs in each iteration. Communication can thus be stopped and each party then simply applies the adaptation (Equation 2) with its own output in order to have a next key from the trajectory in key-space. Using the *Trajectory Mode* this way avoids the stated security weakness in [19], which assumes an ongoing communication. As soon as a new key is present, it is used for encryption. Each integer component of the the key is again XORred with an appropriate length concatenation of L input bits to further decorrelate subsequent keys. It can then be used block-wise or be used on a per-packet basis, depending on the concrete application. In any case, the key is only used to encrypt a certain small subset of the plaintext. This especially allows to realize short key lifetimes.

For the key exchange protocol *without* entity authentication, eavesdropping attacks have concurrently been proposed by Shamir et al. [20] and Kanter, Kinzel et al. [21, 22, 23]. But the prevalent definition of a successful attack is having a 98 percent average overlap $|w^E(t) \cdot w^{A/B}(t)|$ (averaged over all summation units) with the coefficients of one party, when parties A and B are already synchronous and thus successfully finished the key exchange and the communication. The authors chose this definition, because of the strong fluctuations in the success probability using a strict definition. The attacks in [20, 21, 22, 23] can all be made arbitrarily costly and thus can practically be defeated by simply increasing the parameter L. The security increases proportional to L^2 while the probability of a successful attack decreases exponentially with L [21]. The approach is thus regarded computationally secure with respect to these attacks for sufficiently large L [24, 23].

The latest attack, which does not seem to be affected by an increase of L (but still by an increase of K) uses a hundred coordinated and communicating

TPMs [23]. A successful attack according to the definition given above could be achieved with a probability of 0.5. The success probability of achieving a 99 percent average overlap drops down to 0.25. However, an attacker here does not know either, which of the $K \cdot N$ components of the coefficients (the key) are correct. In currently used symmetric encryption algorithms, the flipping of a single bit only already leads to a complete failure in decryption. Due to the only partial knowledge of an attacker on the final key, an added or included privacy amplification through hashing can further significantly decrease this knowledge and increase the secrecy of the final key (compare [17, 18]) and also the security of the trajectory mode. It increases the entropy of the keys and destroys partial knowledge an attacker might have gained on the key from the known attacks.

2.2 Key Exchange between Multiple Parties

Multiple parties can exchange a common key again based on TPM interaction and the synchronisation property. Once two parties p_1 and p_2 have synchronized and thus exchanged a common key, they have identical internal states w^{p_1/p_2} and can be considered a single TPM $p_{1,2}$. The exchange of a common key between $G > 2$ parties can thus be achieved by two basic strategies: parallel interaction processes and sequential interaction processes. Without loss of generality an appropriate numbering (and renumbering) of parties can be performed.

Using parallel interaction processes, an even number of parties G is initially divided into k groups of interacting pairs $(p_i, p_j)_k$ with $k = 1, \cdots, G/2$, $i, j = 1, \cdots, G$ and $i \neq j$, performing a pairwise (independant) key exchange in parallel as explained before. After each group has a common key, pairs of synchronous groups now interact again (in a divide-and-conquer strategy) to exchange a common key. This is done until two remaining groups synchronize the final common key in a final interaction process:

$$(p_1, p_2)_1, (p_3, p_4)_2, \cdots, (p_{G-1}, p_G)_{G/2} \tag{3}$$

$$\rightsquigarrow (p_{1,2}, p_{3,4})_1, (p_{5,6}, p_{7,8})_2, \cdots, (p_{(G/2)-1}, p_{G/2})_{G/4} \rightsquigarrow \quad \cdots \quad \rightsquigarrow p_{1,\cdots,G} \tag{4}$$

If G is odd, the remaining party waits until all other $G-1$ groups have exchanged a common key and then performs one last interaction process with the synchronized group. The complexity of this multi-party key-exchange scales logarithmic with the number of parties, i.e. $O(log\ G)$. Note that the parallel variant requires either independent parallel or multiplexed communication channels. Also note that in practice only two TPMs in each group have to actively send and receive output bits, whereas the others in the group only receive.

In a sequential interaction processes, two parties p_1 and p_2 exchange a common key as described before. Having a common key they become a group $p_{1,2}$ that now interacts with a third party p_3, and so on. This way, a linear chain

$$(\cdots((p_1, p_2), p_3), \cdots), p_G) \rightsquigarrow (\cdots((p_{1,2}, p_3), p_4)\cdots), p_G) \rightsquigarrow \cdots \rightsquigarrow p_{1,\cdots,G} \tag{5}$$

of interaction processes is performed. Note that for the group only one sequence of outputs has to be communicated, as it is identical to all parties (TPMs)

in the group. Again, in practice only one TPMs in the group has to actively send and receive output bits, whereas the others in the group only receive. The complexity of this multi-party key-exchange scales linear with the number of parties, i.e. $O(G)$.

As each key exchange process (parallel or sequential) can independantly be attacked, the security in the presented multi-party scenario scales inversly proportional to the number of parties.

3 Stream Cipher by Tree Parity Machines

A TPM stream cipher can be constructed as follows. Remember that once two parties are synchronous and successfully exchanged a key, they remain synchronous in each further iteration (trajectory mode) and produce equal outputs. The synchronous TPM stream cipher is based on the iteration of K coupled non-linear dynamic functions $y_k(t)$. The keystream generator can so be viewed as being composed of K dynamic filter generators and a final (static) combiner stage that acts similar to a threshold generator (see Figure 2). The initial state of the keystream generator depends on the key $w_{kj}^{A/B}(t_0)$ and the initialization vector $x_{kj}(t_0)$. Each dynamic filter generator consists of an N-bit LFSR (counter variables $x_{kj}(t)$) and of N L-bit up/down counters (U/D-CTRs) with a non-linear dynamic filtering stage. The filter depends on the key or current state (state variables $w_{kj}(t)$). The pseudo-random states of the LFSRs are expanded and mixed with the key state, pseudo-randomly modifiying signs of the key state. The subsequent integer addition (Equation 1) and reduction σ to a single sign (bit) $y_k(t)$ extracts the output of the dynamic filter generator. The final keystream output is an Exclusive-Or of K sign bits σ: $O(t) = \sigma_1 \oplus \sigma_2 \oplus \cdots \oplus \sigma_K$. The Exclusive-Or is also used as the statically balanced combiner to generate the ciphertext $c(t)$ from the keystream and the plaintext, i.e. $c(t) = O(t) \oplus p(t)$. The number of cycles to calculate one output bit (with the serial TPMRA) is $t_O = (K \cdot N + K) + 3$.

The next-state function

$$\phi_t : \mathbb{B} \times \mathbb{L} \times \mathbb{B} \times \mathbb{B} \mapsto \mathbb{L}, \quad \mathbb{B} = \{-1, 1\},\ \mathbb{L} = [-L, L] \subseteq \mathbb{Z} \tag{6}$$

$$\phi_t(O, w_{kj}, x_{kj}, y_k) \mapsto w'_{kj}\ , \tag{7}$$

defined via Equation 2, adapts and bounds the filter coefficients (the state) and represents a nonlinear state update, i.e. the keystream depends on a non-linear state-machine. As explained in Section 2, the state variables $w_{kj}(t)$ perform a random walk with reflecting boundaries in a state space of size $(2L+1)^{KN}$. The TPM stream cipher has $(2^{KN} - 1) \cdot (2L+1)^{KN}$ possible internal states divided into $K \cdot N$ state variables $w_{kj}(t) \in [-L, L]$ and the same number of counter variables $x_{kj}(t) \in \{-1, 1\}$.

Unlike other stream ciphers in output feedback mode (OFB), the keystream $O(t)$ is fed back to the next-state function and not to the LFSR (see Figure 2). As an alternative, the ciphertext bits can be fed back (CFB) instead of the

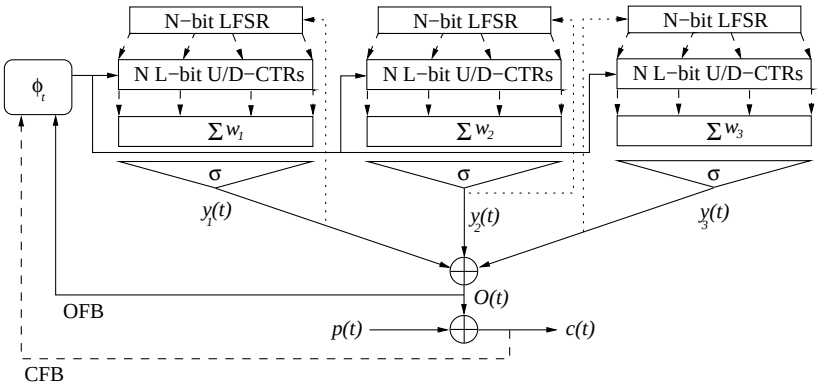


Fig. 2. The synchronous TPM stream cipher for $K = 3$. OFB and CFB modes are indicated. Alternatively, the LFSRs can also have an internal feedback from the summation and thresholding units (dotted lines) resulting in the socalled *Confused Tree Parity Machine* [25] with a simple shift register (with non-linear feedback) per hidden unit replacing the LFSR.

output bits (see Figure 2), making the internal state change also depend on the plaintext and yielding $\phi_t(c, w_{kj}, x_{kj}, y_k)$ in Expression 7. An integrity mechansim is present in the TPM stream cipher in CFB mode. Due to the feedback to the next-state function, a manipulation of the ciphertext leads to a change of the state update at the receiving side. A manipulated keystream thus leads to a decryption failure.

Often, in a real application of a stream cipher, it is required to use a single key many times but with a different initialization vector (IV). Using public initial values of the LFSRs, $2^{KN} - 1$ IVs can be chosen. Yet, the TPMRA allows for fast resynchronsation as a new key can efficiently be exchanged. Preliminary statistical analysis yield the keystream to be indistinguishable from random. Attacks on the stream cipher still have to be investigated.

4 ASIC-Implementation and Results

The *Tree Parity Machine Rekeying Architectures* (TPMRAs) [14] can be functionally separated into two main structures. One structure comprises the Handshake/Key Controller as well as the Bitpackage Unit and the Watchdog, the other structure contains the Tree Parity Machine Unit for calculating the basic TPM functions (Section 2). Figure 3a gives an overview of the hardware structure. The Handshake/Key Controller Unit handles the key transmission (eventually after privacy amplification) with an encryption unit and the bit package exchange process with the other party by using a simple request and acknowledge handshake protocol. It approves the handling of different synchronization cycles between two key exchange parties in order to permit a regulated key- and

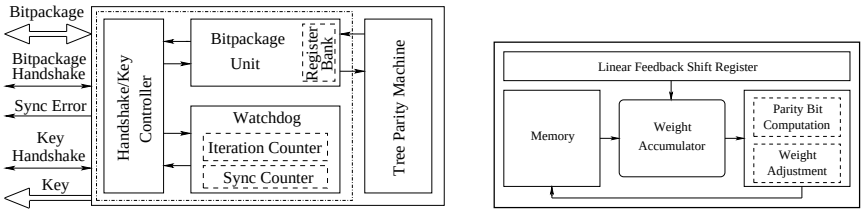


Fig. 3. (a) Basic diagram of the Handshake/Key Controller with the Watchdog and the Bitpackage Unit. (b) The serial Tree Parity Machine Unit. The TPM controller state machine is omitted for clarity.

bit package exchange process. A key is handed over when the synchronization process is finished, indicated by an acknowledge signal.

As described in Section 2, we implemented the bit package generalization of the protocol [13]. It reduces communication down to a few packages. In both architectures, the Bitpackage Unit partitions the parity bits (Equation 1) from the TPM Unit in tighter bit slices. In addition, it serializes the incoming bit packages from other TPM for the adaptation (Equation 2). The Bitpackage Unit handles bit package lengths up to n bits (depending on the key length).

The Watchdog supervises the synchronization between the two parties, which is determined by the chosen parameters and the random initial values of the parties. The Iteration Counter in the Watchdog counts the number of exchanged parity bits. It generates a synchronization error (Sync Error), if there is no synchronization within a specific number of iterations. In this case, the synchronization process is triggered again. The Sync Counter is needed to determine the synchronization of the TPMs by comparing and counting equal output bit packages. It is increased when a sent bit package and the corresponding received bit package is identical and otherwise cleared. A synchronization is recognized when a specified number of equal bit packages is reached. Both Sync- and Iteration-Counter are programmable for variable average synchronization times subject to the chosen TPM structure.

4.1 Serial TPM Unit

Different from the realization in [14], the serially realized TPM Unit calculates a parity bit serially in time and is a fully parameterizable hardware structure. The parameters K, N and L as well as the bit package length can be set arbitrarily in order to adopt this architecture variant for different system environments. The serial TPM Unit consists of a TPM control state machine, a LFSR, a Weight Accumulator, a Parity Bit Computation and Weight Adjustment Unit and a

memory (Figure 3b). The TPM controller is realized as simple finite state machine. It handles the initialization of the TPM, the adaptation with the parity bits of the bit package from the other party and controls the parity calculation and weight adjustment. The LFSR generates the pseudo random bits for the inputs $x_{kj}(t)$ of the TPM. The Parity Bit Computation computes the output parity (Equation 1) and the Weight Adjustment Unit accomplishes the adaptation (Equation 2). The Weight Accumulator computes each sum of the summation units. Each partial result must be temporarily stored in the memory, due to the serial processing of the summation units. The memory, implemented as a simple register bank, stores the weights and the output bits from the summation units in order to process the bit packaging. It could also be implemented as a register file composed of several flip-flops. The memory size depends on the length of the key, which is equal to $K \cdot N \cdot L$. The number of cycles for calculating a n-bit package is $t_{BP} = (2n-1) \cdot (K \cdot N + K) + 3$.

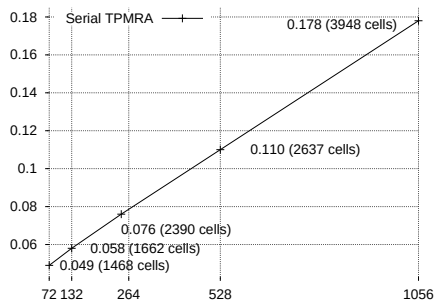
4.2 Results

Parameterizable serial TPMRAs were designed and simulated by using VHDL (compare [14]). While a FPGA-realization was used for easy prototyping, standard cell ASIC-realization prototypes were build to verify the suitability as an embedded system component. The underlying process is a 0.18μ six-layer CMOS process with $1.8V$ supply voltage based on the UMC library [26]. The linear complexity of the key exchange protocol scales with the size $K \cdot N$ of the TPM structure, which defines the size $K \cdot N \cdot L$ of the key. We chose $K = 3$, a maximal $N = 88$ and $L = 4$ for the serial architecture. This leads to a key size of up to 1056 bit.

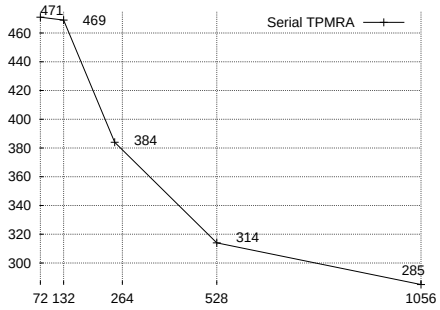
The cell-area (Figure 4a) of the serial TPMRA scales approximately linear due to the linear increase in required memory and ranges around 0.11 square-millimeter for the investigated key sizes. The number in braces denotes used standard cells. Note, that most of the area is consumed by the memory, because of the necessary storage of the partial results. The achievable clock-frequency (Figure 4b) ranges between 285 and 471 *MHz* for the investigated key lengths.

Additionally, we established the throughput for key exchange (i.e. keys per second) subject to the average synchronization time of 400 iterations for different key lengths in Figure 4c. A practically finite channel capacity is neglected here. We assumed the maximally achievable clock frequency with regard to each key length, which can be achieved by Digital Phase Lock Loop (DPLL), regardless of the systems clock frequency. The serial TPMRA achieves a maximal theoretical throughput in the *kHz*-range. After the initial synchronization, the trajectory mode allows to increase the throughput by two orders of magnitude due to the reduced number of cycles for one bit package and the missing communication (interaction) overhead. This mode is identical to the stream cipher mode and we also appoint the theoretical throughput (bit-rate) of the TPM stream cipher which is in the *MHz*-range.

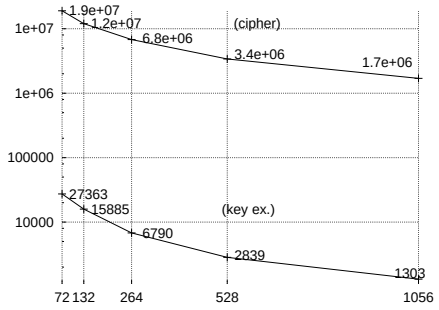
Figure 4d shows throughput regarding key exchange and stream cipher subject to a NFC and a RFID communication channel and their bandwidths. In key



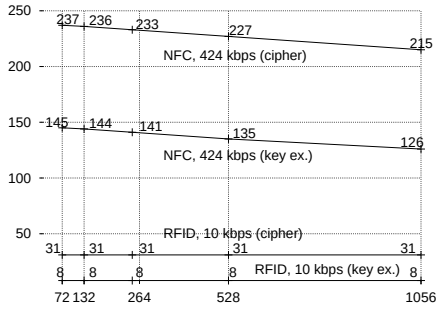
(a) Area $[mm^2]$ vs. key length $[bit]$



(b) Speed $[MHz]$ vs. key length $[bit]$



(c) Average key exchange rate and stream cipher bit-rate $[Hz]$ (log-scaled) vs. key length $[bit]$ (idealized infinite channel bandwidth)



(d) Average key exchange rate and stream cipher bit-rate $[Hz]$ vs. key length $[bit]$ (NFC and RFID channel/protocol)

Fig. 4. Serial TPMRA post-synthesis area-optimized results (key exchange and stream cipher) vs. key length (UMC 0.18μ six-layer CMOS standard cell process).

exchange mode, for every protocol the minimum available packet length was used due to the necessary interaction through our bit packages of 32 bit: NFC (ECMA Intl. NFC IP-1) 136 bit and RIFD (TI TagIt-Protocol), 94 bit. For the RFID channel we appointed a 10 *kbps* channel for simplicity. The capacities here vary with regard to Reader-to-Transponder (5-11 *kbps*) and Transponder-to-Reader (26 *kbps*) communication. In stream cipher mode, for every protocol the maximum available packet length can be exploited (NFC 359 byte with 255 byte payload, RFID 317 bit with 256 bit payload). A comparison among the different communication channels indicates different slopes of the calculated throughput characteristics (Figure 4d). They denote the rising influence of the output bit (bit

packaging) calculations at smaller key lengths for channels of higher bandwidth. Thus, the slope of the NFC throughput characteristic is slightly higher than for RFID. As expected, the influence of the channel bandwidth significantly determines the performance of the key exchange protocol and of the stream cipher. Obviously, the bottleneck is the underlying communication-bus.

5 Conclusions

We suggest to discuss Tree Parity Machine Rekeying Architectures (TPMRAs) for low hardware-complexity lightweight authenticated symmetric key exchange and stream cipher. Efficient frequent rekeying (equivalent to a resynchronisation of the stream cipher) and short key lifetimes can be implemented. Next to using sophisticated encryption algorithms like Rijndal (AES), for example, performed with the exchanged key, the TPMRA itself allows for a lightweight stream cipher with feasible frequent rekeying. The stream cipher allows for a high throughput and is basically limited by the communication channel.

We regard the TPMRAs as IP-cores in embedded system environments with a particular focus on transponder-based applications such as RFID-systems, but also on devices in ad-hoc and sensor networks, in which a small area for cryptographic components is often mandatory. They are especially suited for devices of limited resources with no or only very limited microcontrollers available – even more in moderate security scenarios.

Acknowledgments

The authors would like to thank Sebastian Staiger (Hamburg University of Technology) for his contribution to the project, as well as University of Hamburg for allowing us to use their industry-standard synthesis-toolchain the ASIC design.

References

[1] Stajano, F.: Security in pervasive computing. In: Proc. of the 1st International Conference on Security in Pervasive Computing (SPC 2003). Volume 2802 of LNCS., Springer Verlag (2003) 1

[2] Stanford, V.: Pervasive computing goes the last hundred feet with RFID systems. Pervasive Computing, IEEE Computer Science (2003) 9–14

[3] Sarma, S.E., Weis, S.A., Engels, D.W.: RFID systems and security and privacy implications. In Kaliski, B., ed.: Proc. of the Workshop on Cryptographic Hardware and Embedded Systems 2002, CHES 2002. Volume 2523 of LNCS., Springer-Verlag (2003) 454–469

[4] Weis, S.A., Sarma, S.E., Rivest, R.L., Engels, D.W.: Security and privacy aspects of low-cost radio frequency identification systems. In Hutter, D., ed.: Proc. of the 1st International Conference on Security in Pervasive Computing, SPC 2003. Volume 2802 of LNCS., Springer-Verlag (2004) 201–212

[5] Paar, C.: Past and future of cryptographic engineering. Tutorial at HOT CHIPS 2003, Stanford University, USA (2003)

[6] Pelzl, J., Wollinger, T., Paar, C.: Low cost security: Explicit formulae for genus-4 hyperelliptic curves. In: 10th Annual Workshop on Selected Areas in Cryptography (SAC 2003), Springer Verlag (2003)
[7] Bailey, D., Paar, C.: Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. Journal of Cryptology **14** (2001)
[8] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In Buhler, J., ed.: Proc. of Algorithmic Number Theory (ANTS III), Portland, Oregon. Lecture Notes in Computer Science, Springer-Verlag, Berlin (1998) 267–288
[9] Hoffstein, J., Silverman, J.: Optimizations for NTRU. In: Proc. of Public-Key Cryptography and Computational Number Theory, Warsaw. (2000)
[10] Shamir, A.: Stream ciphers: Dead or alive? In: Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security – ASIACRYPT 2004. Volume 3329 of LNCS. (2004) 78 (see also Proceedings of SASC – The State of the Art of Stream Ciphers, Brugge, Belgium, 2004).
[11] Atmel Corporation: e5561 Read/Write transponder IC for contactless RF identification for highly sophisticated security applications. (2003) Datasheeet.
[12] Atmel Corporation: TK5561A-PP Read/Write Crypto Transponder for Short Cycle Time. (2005) Datasheet revision B.
[13] Kanter, I., Kinzel, W., Kanter, E.: Secure exchange of information by synchronization of neural networks. Europhysics Letters **57** (2002) 141–147
[14] Volkmer, M., Wallner, S.: Tree parity machine rekeying architectures. IEEE Transactions on Computers **54** (2005) 421–427
[15] Maurer, U.: Protocols for secret key agreement by public discussion based on common information. In: Advances in Cryptology – CRYPTO '92. Volume 740 of LNCS., Springer Verlag (1993) 461–470
[16] Maurer, U.: Secret key agreement by public discussion. IEEE Transactions on Information Theory **39** (1993) 733–742
[17] Brassard, G., Savail, L.: Secret-key reconciliation by public discussion. In: Advances in Cryptology – EUROCRYPT 1993. Volume 765 of LNCS., Springer-Verlag (1994) 410–423
[18] Renner, R., Wolf, S.: Unconditional authenticity and privacy from an arbitrarily weak secret. In: Advances in Cryptology – CRYPTO 2003. Volume 2729 of LNCS., Springer-Verlag (2003) 78–95
[19] Kinzel, W., Kanter, I.: Interacting neural networks and cryptography. In Kramer, B., ed.: Advances in Solid State Physics. Volume 42. Springer Verlag (2002)
[20] Klimov, A., Mityagin, A., Shamir, A.: Analysis of neural cryptography. In: Proc. of AsiaCrypt 2002. Volume 2501 of LNCS., Queenstown, New Zealand, Springer Verlag (2002) 288–298
[21] Mislovaty, R., Perchenok, Y., Kanter, I., Kinzel, W.: Secure key-exchange protocol with an absence of injective functions. Phys. Rev. E **66** (2002)
[22] Kanter, I., Kinzel, W.: Neural cryptography. In: Proc. of the 9th International Conference on Neural Information Processing, Singapore (2002)
[23] Kanter, I., Kinzel, W., Shacham, L., Klein, E., Mislovaty, R.: Cooperating attackers in neural cryptography. Phys Rev. E **69** (2004)
[24] Rosen-Zvi, M., Klein, E., Kanter, I., Kinzel, W.: Mutual learning in a tree parity machine and its application to cryptography. Phys. Rev. E. **66** (2002)
[25] Ruttor, A., Kinzel, W., Shacham, L., Kanter, I.: Neural cryptography with feedback. Physical Review E **69** (2004) 7
[26] UMC: High Performance Standard Cells Design Kit Rev 2.2. (2001)

Grain - A Stream Cipher for Constrained Environments

Martin Hell[1], Thomas Johansson[1] and Willi Meier[2]

[1] Dept. of Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
{martin,thomas}@it.lth.se
[2] FH Aargau, CH-5210 Windisch, Switzerland
meierw@fh-aargau.ch

Abstract. A new stream cipher, Grain, is proposed. The design targets hardware environments where gate count, power consumption and memory is very limited. It is based on two shift registers and a nonlinear filter function. The cipher has the additional feature that the speed can be increased at the expense of extra hardware. The key size is 80 bits and no attack faster than exhaustive key search has been identified. The hardware complexity and throughput compares favourably to other hardware oriented stream ciphers like E0 and A5/1.

1 Motivation

When designing a cryptographic primitive there are many different properties that have to be addressed. These include e.g. speed, security and simplicity. Comparing several ciphers, it is likely that one is faster on a 32 bit processor, another is faster on an 8 bit processor and yet another one is faster in hardware. The simplicity of the design is another factor that has to be taken into account, but while the software implementation can be very simple, the hardware implementation might be quite complex.

There is a need for cryptographic primitives that have very low hardware complexity. An RFID tag is a typical example of a product where the amount of memory and power is very limited. These are microchips capable of transmitting an identifying sequence upon a request from a reader. Forging an RFID tag can have devastating consequences if the tag is used e.g. in electronic payments and hence, there is a need for cryptographic primitives implemented in these tags. Today, a hardware implementation of e.g. AES on an RFID tag is not feasible due to the large number of gates needed. Grain is a stream cipher primitive that is designed to be very easy and small to implement in hardware.

Many stream ciphers are based on linear feedback shift registers (LFSR), not only for the good statistical properties of the sequences they produce, but also for the simplicity and speed of their hardware implementation. Several recent LFSR based stream cipher proposals, see e.g. [5, 6] and their predecessors, are based on word oriented LFSRs. This allows them to be efficiently implemented in software

but it also allows them to increase the throughput since words instead of bits are output. In hardware, a word oriented cipher is likely to be more complex than a bit oriented one. We have addressed this issue by basing our design on bit oriented shift registers with the extra feature of allowing an increase in speed at the expense of more hardware. The user can decide the speed of the cipher depending on the amount of hardware available.

The proposed primitive is a bit oriented synchronous stream cipher. In a synchronous stream cipher the keystream is generated independently from the plaintext. The design is based on two shift registers, one with linear feedback (LFSR) and one with nonlinear feedback (NFSR). The LFSR guarantees a minimum period for the keystream and it also provides balancedness in the output. The NFSR, together with a nonlinear filter introduces nonlinearity to the cipher. The input to the NFSR is masked with the output of the LFSR so that the state of the NFSR is balanced. Hence, we use the notation NFSR even though this is actually a filter. What is known about cycle structures of nonlinear feedback shift registers cannot immediately be applied here. Both shift registers are 80 bits in size. The key size is 80 bits and the IV size is specified to be 64 bits. The cipher is designed such that no attack faster than exhaustive key search should be possible, hence the best attack should require a computational complexity not significantly lower than 2^{80}.

Grain provides a higher security than several other well known ciphers intended to be used in hardware applications. Well known examples of such ciphers are E0 used in Bluetooth and A5/1 used in GSM. These ciphers, while also having a very small hardware implementation, have been proven to be very insecure. Compared to E0 and A5/1, Grain provides higher security while maintaining a small hardware complexity.

The paper is organized as follows. Section 2 provides a detailed description of the design. Section 3 gives the design criterias and the design choices and the strengths and limitations of the design are presented in Section 4. In Section 5 we consider the hardware implementation of the cipher and in Section 6 we give the results of our security analysis. Section 7 concludes the paper.

2 Design Specification

This section specifies the details of the design. An overview of the different blocks used in the cipher can be found in Fig. 1 and the specification will refer to this figure. The cipher consists of three main building blocks, namely an LFSR, an NFSR and a filter function. The content of the LFSR is denoted by $s_i, s_{i+1}, \ldots, s_{i+79}$ and the content of the NFSR is denoted by $b_i, b_{i+1}, \ldots, b_{i+79}$. The feedback polynomial of the LFSR, $f(x)$ is a primitive polynomial of degree 80. It is defined as

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}.$$

To remove any possible ambiguity we also define the update function of the LFSR as

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i.$$

The feedback polynomial of the NFSR, $g(x)$, is defined as

$$\begin{aligned} g(x) = 1 &+ x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + \\ &+ x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + \\ &+ x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + \\ &+ x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}. \end{aligned}$$

Again, to remove any possible ambiguity we also write the update function of the NFSR. Note that the bit s_i which is masked with the input is included in the update function below.

$$\begin{aligned} b_{i+80} = s_i &+ b_{i+63} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + \\ &+ b_{i+15} + b_{i+9} + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} + \\ &+ b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + \\ &+ b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} + \\ &+ b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + \\ &+ b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}. \end{aligned}$$

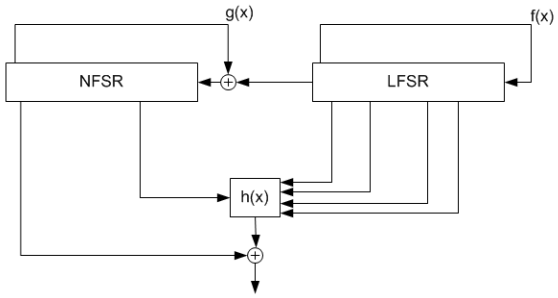


Fig. 1. The cipher.

The contents of the two shift registers represent the state of the cipher. From this state, 5 variables are taken as input to a boolean function, $h(x)$. This filter function is chosen to be balanced, correlation immune of the first order and has algebraic degree 3. The nonlinearity is the highest possible for these functions, namely 12. The input is taken both from the LFSR and from the NFSR. The function is defined as

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

where the variables x_0, x_1, x_2, x_3 and x_4 corresponds to the tap positions s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64} and b_{i+63} respectively. The output of the filter function is masked with the bit b_i from the NFSR to produce the keystream.

2.1 Key Initialization

Before any keystream is generated the cipher must be initialized with the key and the IV. Let the bits of the key, k, be denoted $k_i,\ 0 \leq i \leq 79$ and the bits of the IV be denoted $IV_i,\ 0 \leq i \leq 63$. The initialization of the key is done as follows. First load the NFSR with the key bits, $b_i = k_i,\ 0 \leq i \leq 79$, then load the first 64 bits of the LFSR with the IV, $s_i = IV_i,\ 0 \leq i \leq 63$. The remaining bits of the LFSR are filled with ones, $s_i = 1,\ 64 \leq i \leq 79$. Because of this the LFSR cannot be initialized to the all zero state. Then the cipher is clocked 160 times without producing any running key. Instead the output of the filter function, $h(x)$, is fed back and xored with the input, both to the LFSR and to the NFSR, see Fig. 2.

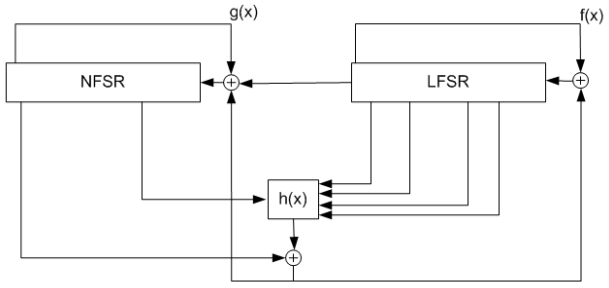


Fig. 2. The key initialisation.

3 Design Criteria

The design of the cipher is chosen to be as simple as possible for a hardware implementation. The security requirements correspond to a computational complexity of 2^{80}, equivalent to an exhaustive key search. To meet this requirement it is necessary to build the cipher with a memory of 160 bits. Implementing 160 memory bits in hardware can be seen a lower bound for the complexity. To develop a small hardware design we have to focus on minimizing the functions that are used together with this memory. The functions used need to be small in order to save gates but still large enough to provide high security. It is well known that an LFSR with primitive feedback polynomial of degree d produces an output with period $2^d - 1$. The LFSR in the cipher is of size 80 and since the feedback polynomial is primitive it guarantees that the period is at least $2^{80} - 1$. Because of the NFSR and the fact that the input to this is masked with the output of the LFSR the exact period will depend on the key and the IV used. The input to the NFSR is masked with the output of the LFSR in order to make

sure that the NFSR state is balanced. The nonlinear feedback is also balanced since the term x^{80} only appears linearily.

The filter function is quite small, only 5 variables and nonlinearity 12. However, this is compensated by the fact that one of the inputs is from the NFSR. The input bit from the NFSR will depend nonlinearily on other bits in the state, both from the LFSR and from the NFSR.

In the key initialization phase the goal is to scramble the contents of the shift registers before the running key is generated. The number of clockings is a tradeoff between security and speed. If the cipher is to be reinitialized often with a new IV, then the efficiency of the initialization is a possible bottleneck. Before initialization the LFSR contains the IV and 16 ones. For initialization with two different IVs, differing by only one bit, the probability that a shift register bit is the same for both initializations should be close to 0.5. Simulations show that this is achieved after 160 clockings. See section 6.4 for further discussion about this. Finally, no hidden weaknesses have been inserted by the designers.

3.1 Throughput Rate

Both shift registers are regularly clocked so the cipher will output 1 bit/clock. It is possible to increase the speed of the cipher at the expense of more hardware. This can very easily be done by just implementing the feedback functions, $f(x)$ and $g(x)$ and the filter function, $h(x)$ several times. In order to simplify this implementation, the last 15 bits of the shift registers, $s_i,\ 65 \leq i \leq 79$ and $b_i,\ 65 \leq i \leq 79$ are not used in the feedback functions or in the input to the filter function. This allows the speed to be easily multipled by up to 16 if a sufficient amount of hardware is available. An example of the implementation when the speed is doubled can be seen in Fig. 3. Naturally, the shift registers also need to be implemented such that each bit is shifted t steps instead of one when the speed is increased by a factor t. By increasing the speed by a factor 16, the cipher outputs 16 bits/clock. Since, in the key initialization, the cipher is clocked 160 times, the possibilities to increase the speed is limited to factors ≤ 16 that are divisible by 160. The number of clockings used in the key initialization is then $160/t$. Since the filter and feedback functions are small, it is quite feasible to increase the throughput in this way.

4 Strengths and Limitations

The design of a cipher needs to be focused on some specific properties. It is not possible to have a design that is perfect for all purposes i.e., processors of all word lengths, all hardware applications, all memory constraints etc. Grain is designed to be very small in hardware, using as few gates as possible while maintaining high security. The cipher is intended to be used in environments where gate count, power consumption and memory needs to be very small. While Grain is still possible to use in general application software, there are several ciphers that are designed with software efficiency in mind and thus are more appropriate

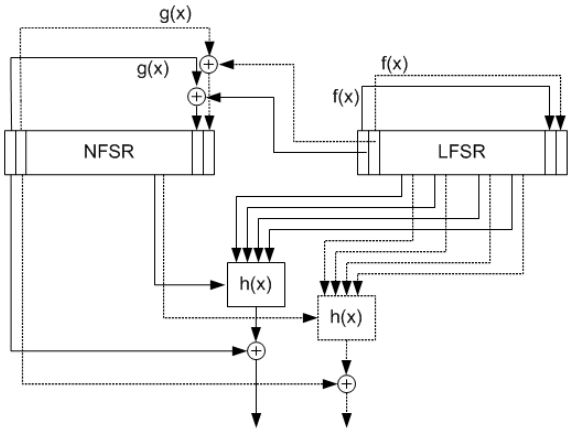


Fig. 3. The cipher when the speed is doubled.

when high speed in software is required. Because of this it does not make sense to compare the software performance of Grain to other ciphers. To emphasize the focus on hardware, no software speed measurements have been conducted.

The basic implementation has rate 1 bit/clock cycle. The speed of a word oriented cipher is typically higher since the rate is then 1 word/clock. Grain is bit oriented due to the high focus on small hardware complexity and this has been compensated by the possibility to increase the speed at the cost of more hardware. This allows a vendor to choose how fast the cipher should be according to the amount of hardware available in the product produced.

5 Hardware Complexity

To get some practical indications on complexity and other important features of a possible hardware implementation of the stream cipher, we performed a design based on standard FPGA architectures.

Starting with Fig. 1 (normal operating mode) and Fig. 2 (key initialisation) we added a third mode (loading key bits into NFSR and IV into LFSR, as described in Sect. 2.1). This whole circuit was described in VHDL (about 300 lines of code) depending on the parameter t as defined in Sect. 3.1. The ALTERA MAX 3000A family was choosen since we have most experience with the associated design equipment and it is seen as adequate for this purpose. MAX 3000A is a low end product using flash Memory as storage for the programming data; i.e. these data are persistent, and no loading procedure is necessary as with RAM-based FPGAs. The EPM3256 is the smallest chip of this family which will meet our requirements (more than 160 flipflops and some combinatorial logic). Using the ALTERA Quartus design tool, we carried out logical synthesis, place/route

and post-layout timing analysis. We found that $t \leq 4$ fits into the EPM3256, leading to a usage of about 90% of the 256 available macrocells. The maximum clock frequency is in the range of 35–50 MHz, depending on the operating mode and the output interface. Also $t = 8$ fits into this chip, but the maximum clock frequency was limited to 30 MHz. The number of output bits per second is t times clock frequency.

In order to make a fair comparison between different ciphers, the implementations has to be tested on the same FPGA. To highlight the performance difference between different FPGAs we have simulated our design on two additional FPGA families, namely the ALTERA MAX II and ALTERA Cyclone. These two allowed the cipher to be clocked at higher speed and it also allowed an implementation of the cipher when the speed was increased 16 times the original speed, i.e. when $t = 16$. It should be mentioned that there are other manufacturers of FPGAs (e.g. Actel, Xilinx), which may offer devices that will meet all requirements too at lower prices. Some products are including security mechanisms, prohibiting reverse engineering of a programmed chip.

The gate count for a function varies depending on the complexity and functionality. The numbers are no natural constants and will depend on the implementation in an actual chip. We have chosen a gate count of 8 for a flip flop. This figure ensures enough functionality for our application. Table 1 lists the factors chosen in our implementation.

Table 1. The gate count used for different functions.

Function	*Gate Count*
D flip flop	8
NAND2	1
NAND3	1.5
NAND4	2
NAND5	2.5
NAND6	3
XOR2	2.5
MUX3	5

In our design we have calculated the gate count for $t = 1, 2, 4, 8$ and 16. Table 2 shows the gate count and the corresponding throughput for the 3 different FPGA/CPLDs. More details regarding the figures in the hardware implementation can be found in Appendix A.

This gate count can be compared to other hardware oriented stream ciphers, e.g. E0 used in Bluetooth and A5/1 used in GSM. Using figures taken from [2], the gate count for E0 is about the same as for Grain. A5/1 has a gate count of approximately half. In all 3 ciphers, most of the gates are used for memory implementation. Grain, E0 and A5/1 use 160, 128 and 64 bits memory respectively. Moreover, the throughput of Grain also compares favourably to E0 and

Table 2. The gate count and throughput of Grain for $t = 1, 2, 4, 8$ and 16.

t	*Gate Count*	*Throughput*		
		MAX 3000A	MAX II	Cyclone
1	1435	49 Mbit/s	200 Mbit/s	282 Mbit/s
2	1607	98.4 Mbit/s	422 Mbit/s	576 Mbit/s
4	1950	196 Mbit/s	632 Mbit/s	872 Mbit/s
8	2636	240 Mbit/s	1184 Mbit/s	1736 Mbit/s
16	4008	—	2128 Mbit/s	3136 Mbit/s

A5/1, mostly due to the fact that it can be increased efficiently with just a small increase in gate count. Both E0 and A5/1 have been proven to be very insecure. In [8], an attack against E0 using 2^{35} frames and computational complexity 2^{40} was shown. This attack is on the borderline of being practical. Also, several attacks against A5/1 have been shown, see e.g. [1, 4, 9]. Grain has been designed to provide much better security than both E0 and A5/1 while maintaining a low gate count.

6 Cryptanalysis

In this section we consider some general attacks on stream ciphers and investigate to which extent they can be applied to Grain. Resistance against all known cryptanalytic attacks is the most important property of a new cipher. There should be no attack faster than exhaustive key search. Initial cryptanalytic attempts against the cipher show the following.

6.1 Correlations

Due to the statistical properties of maximum-length LFSR sequences, the bits in the LFSR are (almost) exactly balanced. This may not be the case for a NFSR when it is driven autonomously. However, as the feedback $g(x)$ is xored with a LFSR-state, the bits in the NFSR are balanced. Moreover, recall that g is a balanced function. Therefore, the bits in the NFSR may be assumed to be uncorrelated to the LFSR bits.

The function h is chosen to be correlation immune of first order. This does not preclude that there are correlations of the output of $h(x)$ to sums of inputs. As one input comes from the NFSR and as $h(x)$ is xored with a state bit of the NFSR, correlations of the output of the generator to sums of LFSR-bits will be so small that they will not be exploitable by (fast) correlation attacks.

6.2 Algebraic Attack

A filter generator alone with output function $h(x)$ of degree only three would be very vulnerable to algebraic attacks. On the other hand, algebraic attacks

will not work for solving for the initial 160-bit state of the full generator, as the update function of the NFSR is nonlinear, and the later state bits of the NFSR as a function of the inital state bits will have varying but large algebraic degree. Using key initialization, it may be possible to express the output of the generator as a function of state bits of the LFSR alone. As the filter function $h(x)$ has one input coming from the NFSR, and $h(x)$ is xored with a NFSR-state bit, the algebraic degrees of the output bits when expressed as a function of LFSR-bits, are large in general, and varying in time. This will defeat algebraic attacks.

6.3 Time/Memory/Data Tradeoff Attack

The cost of time/memory/data tradeoff attacks on stream ciphers is $O(2^{n/2})$, where n is the number of inner states of the stream cipher, [3]. To obey the margins set by this attack, $n = 160$ has been chosen. It is known that stream ciphers with low sampling resistance have tradeoff attacks with fewer table lookups and a wider choice of parameters, [3]. The sampling resistance of $h(x)$ is reasonable: This function does not become linear in the remaining variables by fixing less than 3 of its 5 variables. Similarly, the variables occuring in monomials of $g(x)$ are sufficiently disjoint. Hence the resulting sampling resistance is large, and thus time/memory/data tradeoff attacks are expected to have complexity not lower than $O(2^{80})$.

6.4 Chosen-IV Attack

A necessary condition for defeating differential-like or statistical chosen-IV attacks is that the initial states for any two chosen IV's (or sets of IV's) are algebraically and statistically unrelated. The number of cycles in key initialization has been chosen so that the Hamming weight of the differences in the full initial 160-bit state for two IV's after initialization is close to random. This should prevent chosen-IV attacks.

It may be tempting to improve the efficiency of the key initialization by just decreasing the number of initial clockings. Indeed, after only 80 clocks, all bits in the state will depend on both the key and the IV. However, in a chosen-IV attack it is possible to reinitialize the cipher with the same key but with an IV that differs in only one position from the previous IV. Consider the case when the number of initial clockings is 80 and the last bit of the IV is flipped i.e., s_{63} is flipped. This is the event that occurs if the IV is chosen as a sequence number. Looking at the difference of the states after initialization it is clear that several positions will be predictable. The bit s_{63} is not used in the feedback or in the filter function, hence, the first register update will be the same in both cases. Consequently, the bit s_0 will be the same in both initializations. In the next update, the flipped bit will be in position s_{62}. This position is used in the linear feedback of the LFSR, and consequently the bit s_1 will always be different for the two initializations. Similar arguments can be used to show that the difference in the state will be deterministic in more than half of the 160 state bits. This deterministic difference in the state can be exploited

in a distinguishing attack. Let $\underline{x} = x_0, x_1, x_2, x_3, x_4$ be the input variables to $h(x)$ after the first initialization and let $\underline{x}_\Delta = x_0, x_1, x_2, x_3, x_4$ be the input variables to $h(x)$ after the second initialization. Now, compute the distribution of $P(\underline{x}, \underline{x}_\Delta)$. If this distribution is biased, it is likely that the distribution of the difference in the first output bit,

$$P(h(\underline{x}) \oplus h(\underline{x}_\Delta)),$$

is biased. Assume that

$$P(h(\underline{x}) \oplus h(\underline{x}_\Delta) = 0) = 1/2 + \epsilon,$$

then the number of initializations we need will be in the order of $1/\epsilon^2$. This attack can be optimized by calculating which output bit will give the highest bias since it is not necessarily the bits in the registers corresponding to the input bits of $h(x)$ that have deterministic difference after the initializations. This attack shows that it is preferred that the probability that any state bit is the same after initialization with two different IVs should be close to 0.5. As with the case of 80 initialization clocks, it is easy to show that after 96, 112 and 128 there are also state bits that will always be the same or that will always differ.

6.5 Fault Attack

Amongst the strongest attacks conceivable on any cipher, are fault attacks. Fault attacks against stream ciphers have been initiated in [7], and have shown to be efficient against many known constructions of stream ciphers. This suggests that it is hard to completely defeat fault attacks on stream ciphers. In the scenario in [7] it is assumed that the attacker can apply some bit flipping faults to one of the two feedback registers at his will. However he has only partial control over their number, location, and exact timing, and similarly on what concerns his knowledge. A stronger assumption one can make, is that he is able to flip a single bit (at a time instance, and thus at a location, he does not know exactly). In addition, he can reset the device to its original state and then apply another randomly chosen fault to the device. We adapt the methods in [7] to the present cipher. Thereby, we make the strongest possible assumption (which may not be realistic) that an attacker can induce a single bit fault in the LFSR, and that he is somehow able to determine the exact position of the fault. The aim is to study input-output properties for $h(x)$, and to derive information on the 5 inputs, out of known input-output pairs (similar as for S-boxes in differential cryptanalysis of DES). As long as the difference induced by the fault in the LFSR does not propagate to position b_{i+63}, the difference observed in the output of the cipher is coming from inputs of $h(x)$ from the LFSR alone. If an attacker is able to reset the device and to induce a single bit fault many times and at different positions that he can correctly guess from the output difference, we cannot preclude that he will get information about a subset of the state bits in the LFSR. Such an attack seems more difficult under the (more realistic) assumption that the fault induced affects several state bits at (partially) unknown positions, since

in this case it is more difficult to determine the induced difference from output differences.

Likewise, one can consider faults induced in the NFSR alone. These faults do not influence the contents of the LFSR. However, faults in the NFSR propopagate nonlinearly and their evolution will be harder to predict. Thus, a fault attack on the NFSR seems more difficult.

7 Conclusion

A new stream cipher, Grain, has been introduced. It is designed with small hardware implementation in mind. A complete description of the algorithm as well as a security analysis based on known attacks have been given. The construction is based on two shift registers, one with linear feedback and one with nonlinear feedback, and a nonlinear filter function. The key size is 80 bits and no attack with complexity better than exhaustive key search has been identified.

Grain is a bit oriented stream cipher producing 1 bit/clock in its simplest implementation. However, as an important feature, it is very easy to increase the rate up to 16 bits/clock if some additional hardware is used.

Acknowledgement

We are indebted to Werner Witz and Peter Steigmeier for carrying out an implementation of Grain in hardware and extracting the data as given in Section 5.

References

1. E. Barkan, E. Biham and N. Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. Crypto 2003, Springer Verlag, LNCS 2729, Oct 2003, Pages 600–616
2. L. Batina, J. Lano, N. Mentens, S. B. Örs, B. Preneel and I. Verbauwhede. Energy, Performance, Area Versus Security Trade-offs for Stream Ciphers. In *The State of the Art of Stream Ciphers: Workshop Record, Brugge, Belgium, October 2004*, pages 302–310, 2004.
3. A. Biryukov, A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. Asiacrypt 2000, Springer Verlag, LNCS 1976, pp. 1–13.
4. A. Biryukov, A. Shamir and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC. FSE 2000, Springer Verlag, LNCS 1978, Jan 2001, pp. 1–13.
5. P. Ekdahl and T. Johansson. A New Version of the Stream Cipher SNOW. Selected Areas in Cryptography, SAC 2002, Springer Verlag, LNCS 2595, pp. 47–61, 2002.
6. P. Hawkes and G. Rose. Primitive Specification for SOBER-128. IACR ePrint Archive, `http://eprint.iacr.org/2003/81/`, 2003.
7. J.J. Hoch, A. Shamir. Fault Analysis of Stream Ciphers. CHES 2004, Springer Verlag, LNCS 3156, pp. 240–253.
8. Y. Lu and S. Vaudenay. Cryptanalysis of Bluetooth Keystream Generator Two-Level E0. Asiacrypt 2004, Springer Verlag, LNCS 3329, Nov 2004, pp 483–499.

9. A. Maximov, T. Johansson and S. Babbage. An Improved Correlation Attack on A5/1. Selected Areas in Cryptography, SAC 2004, Springer Verlag, LNCS 3357, Jan 2004, Pages 1 - 18.

A Hardware Figures

In Table 3 we summarize hardware figures when the implementation was simulated on three different FPGAs. A fair comparison between different implementations and different ciphers requires that the same FPGA family is targeted.

Table 3. Hardware related figures for Grain.

	Supplier	ALTERA	ALTERA	ALTERA
	FPGA/CPLD	MAX 3000A	MAX II	Cyclone
	Type	EPM3256ATC144-7	EPM570T100C3	EP1C3T100C6
	Total LABs [LE]	[256]	57	291
t=1	Gate Count	1435	1435	1435
	Max. Clock	49.2 MHz	200 MHz	282 MHz
	%LAB [LE] usage	[68%]	38%	8%
	Power drain	700 mW	365 mW	835 mW
	Throughput	49 Mbit/s	200 Mbit/s	282 Mbit/s
	Efficiency	70 Mbit/Joule	0.55 Gbit/Joule	0.38 Gbit/Joule
t=2	Gate Count	1607	1607	1607
	Max. Clock	49.2 MHz	211 MHz	288 MHz
	%LAB [LE] usage	[75%]	42%	10%
	Power drain	750 mW	395 mW	855 mW
	Throughput	98.4 Mbit/s	422 Mbit/s	576 Mbit/s
	Efficiency	131 Mbit/Joule	1.07 Gbit/Joule	0.67 Gbit/Joule
t=4	Gate Count	1950	1950	1950
	Max. Clock	49 MHz	158 MHz	218 MHz
	%LAB [LE] usage	[89%]	49%	12%
	Power drain	835 mW	330 mW	660 mW
	Throughput	196 Mbit/s	632 Mbit/s	872 Mbit/s
	Efficiency	235 Mbit/Joule	1.95 Gbit/Joule	1.32 Gbit/Joule
t=8	Gate Count	2636	2636	2636
	Max. Clock	30 MHz	148 MHz	217 MHz
	%LAB [LE] usage	[98%]	61%	11%
	Power drain	775 mW	350 mW	665 mW
	Throughput	240 Mbit/s	1184 Mbit/s	1736 Mbit/s
	Efficiency	310 Mbit/Joule	3.38 Gbit/Joule	2.6 Gbit/Joule
t=16	Gate Count		4008	4008
	Max. Clock		133 MHz	196 MHz
	%LAB [LE] usage		85%	19%
	Power drain		420 mW	625 mW
	Throughput		2128 Mbit/s	3136 Mbit/s
	Efficiency		5.06 Gbit/Joule	5.18 Gbit/Joule

0 The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT, as well as by Gebert Rüf Stiftung, and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Small Scale Variants of the Secure Hash Standard

Marco Macchetti[1] and Philippe Rivard[2]

[1] Politecnico di Milano, Milan, Italy
[2] ALaRI-USI, Lugano, Switzerland
macchett@elet.polimi.it
philippe.rivard@alari.ch

Abstract. In this paper we present effective small scale formulations of the Secure Hash Standard; we focus on the SHA-2 family of algorithms, introducing new compact instances baptized SHA-16, SHA-32, and SHA-64. These may be useful for computing hashes and Message Authentication Codes (MACs) on small platforms where only 8-bit processors are available, such as in the case of Radio Frequency Identification (RFID) devices and embedded systems. To prove the soundness of our scaling approach, we analyze the cryptographic properties of the proposed constructions in terms of adherence to the Strict Avalanche Criterion (SAC) and of robustness to birthday attacks, by also comparing the results with the expected values from random functions. As an additional contribution, we complete the theoretical results for the balance property of random functions, thereby also calculating the expected robustness of the original SHA-2 family versus birthday attacks.
Keywords: hash functions, balance, SAC, small scale, RFID.

1 Introduction

The ever-growing availability of mobile and embedded devices poses new challenges with regards to performance, quality of service and, most of all, security and privacy. Research on ad-hoc networks and especially sensor networks has increased significantly in the last few years, proposing interesting solutions but also introducing new questions. The reader is referred to [1] for a good survey on the topic.

There is a clear need for strong authentication primitives to be introduced in sensor and RFID devices. In fact, the former may be used in large quantities to monitor environmental changes, or dangerous events such as landslides, tsunamis, and earthquakes; in this case an adversarial forgery of messages can lead to false alarms, which eventually may decrease the level of trust in these services and may constitute an indirect, but effective, terroristic act. RFID tags can be conveniently used to monitor goods or can be embedded into personal documents (e.g. passports) to strengthen authentication procedures; it is clear that in this context strong authentication means increased difficulty in forging false documents or in counterfeiting goods.

The flow of information to and from these embedded devices can be secured with known techniques, such as by introducing MACs. These can be built starting from available primitives like block ciphers, e.g. with the CBC-MAC [2], or cryptographic hash functions, e.g. with the HMAC [3] or the UMAC [4] constructions. There are several related proposals for lightweight security layers, among those is TinySec [5]. The designers of TinySec correctly point out [6] that MACs are necessary, and that small sized ones are probably sufficient in the foreseen use-cases, as they make up for a good compromise between security and other desirable properties (e.g. battery duration, latency overhead).

While the panorama of block ciphers is relatively quiet, with the Advanced Encryption Standard (AES) [7] being an efficient and secure solution, the hash functions world has been recently shaken by the publishing of successful attacks against the MD5 [8] and SHA-1 [9] algorithms. It seems today that the SHA-2 family [10] is an appealing replacement, as it benefits from a strengthened structure with regards to SHA-1. It is also probable that the research community will introduce new, open and secure hash functions in the near future.

The SHA-2 family of algorithms was designed specifically for 32-bit and 64-bit processors, and the hash lengths are 256, 384 or 512 bits. These algorithms are too cumbersome to manage on 8-bit embedded platforms. It is always possible to compute a 256-bit MAC and truncate it to the desired length, but this is not an efficient approach.

We show that a small scale design approach can be adopted and conveniently applied to the SHA-2 algorithm; small scale versions of cryptographic algorithms can be useful to test the quality of the design methodologies and also, in our case, to define and validate reasonably-sized hash functions that can be conveniently used in RFID and sensor platforms. A similar approach was used in [11] to test the expected robustness of the AES versus algebraic attacks.

In Section 2, we present our small scale SHA variants, discussing the parts of the original algorithm that scale quite easily and the parts that instead need to be customized; since no details are known about the original design strategy, we make, and explain why we think this is the case, the assumption that the robustness properties are inherited from the older siblings in the family. We examine the possible solutions to the scaling problem, giving guidelines that may also shed some light on the original design rationale (not published by NIST). These small scale variants are named SHA-X, where X is the bit size of the final hash, following the notation used in the original specification.

In Section 3, we tackle the problem of how to validate the robustness of the proposed constructions; we also improve on the current results for the balance metric [12], deriving a simple formula that can be used to estimate the average and minimum expected balance values for an arbitrarily-sized random function. As a side result, we prove that for functions with the input/output size of SHA-256 the difference between random and regular functions vanishes, and quantify the minimum expected effort to produce collisions for the SHA-256 function.

In Section 4, we present the results of our experiments, which validate the quality of our design approach. Section 5 concludes the paper.

2 Small Scale Formulations

The concept of small scale variants of cryptographic algorithms is not new. They have been used on block ciphers as learning tools to understand the functioning of such algorithms, and as tools in the evaluation of techniques which could be used to break them [11]. To our knowledge, no small scale variants of hashing constructs have been built for this purpose.

2.1 The Scaling Strategy

The first useful observation that can be made about SHA-256 is that it seems to scale up quite easily, as NIST has also presented SHA-512, an adaptation of SHA-256 which produces a 512-bit hash [10]. The two algorithms are virtually identical, with the exception that SHA-256 uses 32-bit operands, while SHA-512 uses 64-bit operands. However, it is important to note that the choice of the so-called *sigma* functions is different, and that NIST has not provided any explanation as to their choices in the design of either of these algorithms.

It is clear that the algorithm can also be scaled down, as the primitive operations it bases its security on can also scale easily. Hence, by simply reducing the size of the registers used in the function, most of the algorithm will scale along, as modulo addition, and the choice and majority bitwise non-linear functions can be defined over any operand size. Our small scale variants of SHA-256 can keep the exact same structure with smaller register sizes, as long as we change the sigma functions to reflect the new operand sizes. Also, the constants that are used in the main loop and the initial values of the chaining variable can be adapted by simply truncating the originally specified values to the desired bit sizes.

For very small register sizes, the number of unique operations which can be combined to form the sigma functions is quite limited. In this case, we can simply explore the full design space in order to find the optimal combination of these functions and set these as the sigma functions of our small scale variant. However, as the word size increases, such an approach becomes infeasible. We now turn our attention to the known properties of the sigma functions of SHA-256 and SHA-512, in an effort to minimize this design space exploration.

The constants chosen for the sigma functions in SHA-256 and SHA-512 make each sigma function invertible. Hence, we can first choose our constants in such a way that the resulting function is invertible. Small scale variants of SHA-256 have relatively small operand sizes, and thus, such functions can easily be tested with a brute force approach. It is important to note that there exist more formal methods of testing for invertibility.

It has been reported that the choice of the constants in the sigma functions results in a faster convergence of the hash function towards the strict avalanche criterion [13]. To our knowledge, there exists no formal methodology to obtain a shape for these functions which can guarantee the speedup of this process. However, the adherence of a hash function to the strict avalanche criterion can easily be measured. We can compare the adherence of a potential small scale

variant to the SAC, with that of a subset of the bits of the output of SHA-256. We can also compare this result with the expected value of a purely random function, obtained by using simple discrete mathematics, as can be seen in Section 3.

It is interesting to note that individual subsections of SHA-256 behave in a very similar way in the SAC test to the results expected from a random function. We can thus say that SHA-256 behaves in a pseudo-random manner from this point of view. A second interesting observation that can be made is that by replacing the sigma functions in the full version of SHA-256 with an identity operation, its new outputs begin to diverge from that of a true random function in the SAC test.

Hence, it is clear that close adherence of a potential small scale variant of SHA-256 to the expected results of a random function is a very desirable property. It will be shown that small scale variants also exhibit a divergence from the random case when their sigma functions are replaced by identity operations.

The sigma functions in SHA-256 and in SHA-512 also have some particular properties with respect to their number of fixed points. In the case of SHA-256, σ_0 and σ_1 each have two fixed points, while Σ_0 has two fixed points and Σ_1 has eight. In the case of SHA-512, σ_0 and σ_1 have a unique fixed point corresponding to the null input, while Σ_0 and Σ_1 have two.

2.2 Choice of Parameters

The small scale variants obtained by following the few design tips available in Section 2.1 are denoted as SHA-X, where X is the number of output bits of the hash function, and must be a multiple of 8, as we are primarily decreasing the word length of the operands in the original algorithm. For practical purposes, we will limit ourselves to 3 different word sizes (2, 4, and 8), which will yield small scale variants named SHA-16, SHA-32 and SHA-64. The sigma parameters selected for each of these variants are shown in Table 1, where SH_R^X (RT_R^X) denotes right shift (rotate) by X bit positions.

Table 1. Sigma functions of selected small scale variants

Sig.	SHA-16	SHA-32	SHA-64
$\Sigma_0(x)$	$\{x_2, x_1 \oplus x_2\}$	$RT_R^3(x) \oplus SH_R^1(x) \oplus SH_R^2(x)$	$RT_R^1(x) \oplus RT_R^3(x) \oplus RT_R^4(x)$
$\Sigma_1(x)$	$\{x_1, x_1 \oplus x_2\}$	$x \oplus RT_R^1(x) \oplus SH_R^3(x)$	$RT_R^2(x) \oplus RT_R^5(x) \oplus RT_R^6(x)$
$\sigma_0(x)$	$\{x_1 \oplus x_2, x_2\}$	$x \oplus RT_R^1(x) \oplus SH_R^1(x)$	$RT_R^1(x) \oplus RT_R^6(x) \oplus SH_R^4(x)$
$\sigma_1(x)$	$\{x_1 \oplus x_2, x_1\}$	$x \oplus RT_R^1(x) \oplus RT_R^3(x)$	$RT_R^2(x) \oplus RT_R^7(x) \oplus SH_R^5(x)$

The design options for the sigma functions in the case of SHA-16 are quite limited, as the small word size restricts the number of functions which can be considered. Furthermore, it is important to note that the sigma functions of SHA-16 exhibit symmetric behaviour, which could potentially decrease the overall strength of the construction. The small size of the output would make such

a construction highly ineffective for most cryptographic applications, and this formulation is therefore considered to be a purely academic exercise.

However, the number of possible sigma function choices increases drastically as we move to larger word sizes. It is thus possible to attempt to mimic the behaviour of the original sigma functions in the small scale versions with larger operand sizes. The sigma function choices for SHA-32 and SHA-64 reflect this approach.

2.3 A Note About SHA-64

SHA-64 is probably the most interesting construction from a practical point of view; a calculation of a 64-bit hash can be computed very efficiently on 8-bit microcontrollers, such as the widely used ATMEL devices, with an effort roughly 1/4 of that required to run a full SHA-256 computation.

Even if a 64-bit hash does not protect from collision attacks, in some applications the real objective is the calculation of a relatively short MAC. A possible method to obtain a secure 64-bit MAC is as follows. First, a 128-bit key k is chosen (possibly the result of a key distribution undertaken previously); the length of the key is chosen in order to rule out the risk of exhaustive searches onto the key-space. The key is used to replace the initial constants in the SHA-64 algorithm, following the NMAC construction [3]:

$$\mathrm{MAC}_{64} = \text{SHA-64}_{k_l}(\text{SHA-64}_{k_h}(m)) \tag{1}$$

where m is the message and $k = (k_h||k_l)$.

To obtain a fully usable specification, there are some padding issues to be solved with regards to SHA-64. We propose to use the same padding strategy used for the full SHA-256, the only difference being that the last 4 message bytes are reserved for the message length. This allows for messages of length up to 2^{32} bits to be hashed, corresponding to a maximum length of 512 Mbytes. Considering the typical usage on embedded platforms, this should be high enough.

In some applications, e.g. sensor devices, the limit may be safely decreased, by reserving less space for padding (2 bytes allowing the computation of MACs on messages with length up to 8 Kbytes). It is important to note that such customizations always preserve the relative security of the full construction, the parameters being only scaled down to the desired amount.

3 Benchmarking Techniques

Even if there are good reasons to believe that the cryptographic robustness is inherited from the original algorithms, we need to test the cryptographic properties of the proposed small-scale variants of SHA-256. This will confirm our assumptions and possibly shed some light on the role that the different components play in the global construction.

We will not investigate the robustness to differential [14] and linear [15] cryptanalysis techniques. The first reason is that, due to space and time constraints,

we prefer to focus on basic properties, such as the adherence to the SAC, and the collision resistance. A second reason is that, since the structure of the original algorithm is essentially preserved, attacks of this kind that prove successful on small-scale variants are probably also applicable, with some scaling effort, to SHA-256, and vice versa. These results would exceed the goals of this paper.

3.1 Adherence to SAC

The Strict Avalanche Criterion is a basic, but important tool that can be used to test the quality of cryptographic constructions. The approach is to collect statistics for the number of bit-flips in the hash values, when single bits are flipped in the input messages. The results are then compared with what is expected if the function is randomly picked from the set of all functions with the same domain and range sizes; if the gain of the adversary in distinguishing the two cases is small, i.e. if the data series are in good agreement, we can say that the hash function behaves like a good pseudo-random function (PRF), at least with regard to the SAC. This is only a necessary robustness condition for cryptographic functions.

For a randomly-picked function $f : \{0,1\}^m \Rightarrow \{0,1\}^n$, and a set of pair-wise distinct input values $D_s = \{d_i, 1 \leq i \leq 2^s\}$, considering the corresponding set of output values $R_s = \{r_i, 1 \leq i \leq 2^s\}$ we have that:

$$\text{Prob}[w(r_i \oplus r_{i+1}) = k] = \frac{2^{-n} n!}{k!(n-k)!} \tag{2}$$

where $w(x)$ denotes the Hamming weight of x. This is the expected distribution because for the process we assume that every single output bit has exactly a 50% probability of being flipped; the expected number of bit flips is $\frac{n}{2}$. The validity of (2) extends to the case when

$$D_s = \{d_i : w(d_i \oplus d_{i+1}) = 1\} \tag{3}$$

thereby offering a comparison ground for the SAC adherence of real functions.

3.2 The Balance Metric

The balance property is, roughly speaking, an estimate of the regularity of a vectorial Boolean function f; it is a real number lying in the $(0, 1)$ interval, equal to 1 if all pre-image classes of f have the same cardinality, and equal to 0 if f is a constant function. The exact formulation is given in [12], along with the proof that the complexity of birthday attacks on a given hash function is directly linked to its balance value.

It is still an open problem to derive an expected range of balance values for random functions with domain and range sizes comparable to those of SHA-256. In the following, we will obtain a simple expression for the latter, which can be used to test our small-scale variants and also to calculate the collision resistance of functions like the full SHA-256.

A first thing to clear out is that, in general, there is no link between the balance value of a function and its randomness, since it is easy to see that there exist highly balanced functions that are not at all random, and any random function has a finite probability of having any balance value between 0 and 1. Thus the two properties seem to be orthogonal, at least in general.

However, we are justified to say that a function randomly picked from the set of all functions with domain size $|D| = 2^m$ and range size $|R| = 2^n$ is expected to have a definite balance value. This is nothing but the expected value (mean) of the distribution of balance values for all functions in the set. If the variance of this distribution is very small, the claim has a strong basis, and if we have good reasons to believe that a function is behaving randomly, then we can say that its balance is likely to be very near to the mean value of the balance for the set of functions with the same domain and range size.

If f is randomly chosen, we can model its behaviour with the following process. There is a basket which contains 2^n balls, each one indexed by a unique number (for instance in the range $0 \ldots 2^n - 1$). The index of each ball corresponds to the output of f. We randomly pick one ball from the basket, we read the index and we increment a corresponding counter; the ball is then put back into the basket. We repeat the extraction 2^m times and in the end we calculate the expected balance from the values of all the counters.

The problem can be reduced to calculating the expected values of the counters. In the above experiment, the chance that a ball with a given index is extracted k times is P_k; also, $z = m - n \geq 0$ and we may assume that $m, n \geq 8$.

$$P_k = \binom{2^m}{k} \left(\frac{1}{2^n}\right)^k \left(\frac{2^n - 1}{2^n}\right)^{2^m - k} = \tag{4}$$

$$= \frac{2^m!}{k!(2^m - k)!} \frac{1}{2^{nk}} \left(\frac{2^n - 1}{2^n}\right)^{2^m} \left(\frac{2^n - 1}{2^n}\right)^{-k} =$$

$$= \frac{\prod_{i=0}^{k-1}(2^m - i)}{k!} \frac{1}{(2^n - 1)^k} \left(\left(\frac{2^n - 1}{2^n}\right)^{2^n}\right)^{2^z} \simeq$$

$$\simeq \frac{2^{mk}}{k!} \frac{1}{2^{nk}} \left(\frac{1}{e}\right)^{2^z} = \frac{(2^z)^k}{k!} \left(\frac{1}{e}\right)^{2^z} \tag{5}$$

As expected, the distribution approaches a Poisson distribution [16], and thus the probabilities depend only on the difference between m and n. The above distribution of probability is true for each counter value. From this, one can calculate the balance with the usual formula, taking into account that the mean value and the variance of distribution (5) are:

$$\mu_p = E[P_k] = 2^z \tag{6}$$

$$\sigma_p^2 = E[(P_k - \mu_p)^2] = 2^z \tag{7}$$

We can write the balance formula considering that every counter value c_i will be distributed according to (6) and (7), and indicating the distance of each counter

value from the mean value with d_i:

$$\begin{aligned} B &= \log_{2^n}\left(\frac{2^{2m}}{\sum_{i=1}^{2^n} c_i^2}\right) = \\ &= \log_{2^n}\left(\frac{2^{2m}}{\sum_{i=1}^{2^n} (\mu_p + d_i)^2}\right) = \\ &= \log_{2^n}\left(\frac{2^{2m}}{\sum_{i=1}^{2^n} (\mu_p^2 + d_i^2 + 2\mu_p d_i)}\right) \end{aligned} \tag{8}$$

In any realization of the above process, one has to remember that the values of the counters are not independent, since it must hold that $\sum_{i=1}^{2^n} c_i = 2^m$. This implies that the third term in the sum of (8) can be ignored, since:

$$\begin{aligned} \sum_{i=1}^{2^n} 2\mu_p d_i &= \sum_{i=1}^{2^n} 2\mu_p (c_i - \mu_p) = \\ &= 2\mu_p \sum_{i=1} 2^n c_i - 22^n \mu_p^2 = \\ &= 22^{m-n} 2^m - 22^n 2^{2m-2n} = 0 \end{aligned} \tag{9}$$

Moreover, if we want to get an expected value of the balance, we can use the expected value of d_i^2 which is exactly σ_p^2 (this is very easy to see). Elaborating on (8) we obtain:

$$B_{exp} = \log_{2^n}\left(\frac{2^{2m}}{\sum_{i=1}^{2^n} (\mu_p^2 + \sigma_p^2)}\right) = \tag{10}$$

$$\begin{aligned} &= \log_{2^n}\left(\frac{2^{2m}}{2^n(2^{2z} + 2^z)}\right) = \\ &= \log_{2^n}\left(\frac{2^{2m}}{2^{2m-n} + 2^m)}\right) = \\ &= -\log_{2^n}\left(\frac{1}{2^m} + \frac{1}{2^n}\right) \end{aligned} \tag{11}$$

which is an elegant and simple formulation of the expected balance of a random function with the given domain and range sizes.

For interesting values of m, n the value of B_{exp} will be very near to 1, which is always an upper bound on the balance value and is reached by regular functions. What is an acceptable tight lower bound for B for a random function? We can easily answer starting from (10) and using the Chebyshev inequality. For each counter value, we know that:

$$\text{Prob}[(\mu_p - c_i)^2 \geq \phi^2] \leq \frac{\sigma_p^2}{\phi^2} \tag{12}$$

$$\text{Prob}[d_i^2 \geq 2^{z+1}] \leq \frac{1}{2} \tag{13}$$

The probability that *all* the d_i^2 are greater or equal than 2^{z+1} is thus negligible, although this may happen and there is no contradiction with the fact that $\sum_{i=1}^{2^n} c_i = 2^m$.

$$\text{Prob}[\forall i, d_i^2 = 2^{z+1}] = \frac{1}{2^{2^n}} \tag{14}$$

When this happens, the value of the balance is equal to:

$$B_{min} = \log_{2^n}\left(\frac{2^{2m}}{2^n(2^{2z} + 2^{z+1})}\right) = -\log_{2^n}\left(\frac{2}{2^m} + \frac{1}{2^n}\right) \tag{15}$$

We now have a definite range for the balance of a random function; in our experiments the balance should be near to B_{exp} and should never fall below B_{min}.

Using a program such as Mathematica, it is possible to tabulate the expected and minimum balance values for all interesting input and output sizes. Comparing the results from this paper with the ones from the experiments in [12] we always have a very good agreement, an indication that the few approximations we have made are reasonable.

The above formulas can be used to estimate the expected resistance to birthday attacks of a random function with the domain and range size of SHA-256. The minimum balance value can be calculated and the minimum collision-finding effort C_{min} can be obtained using the formula in [12]. If SHA-256 has good random properties the probability that less than 2 to the power of:

$$127.99999999999999999999999999999999999999999999999999\ldots$$
$$\ldots 99999999999999999999999999987540642453284040182 1$$

trials are needed to obtain a collision is negligible.

We can clearly say that for such domain and range sizes, the difference (with regard to birthday attacks) between regular and random functions vanishes.

4 Experimental Results

Two experiments were conducted on the resulting small scale formulations, in order to ensure that they behaved similarly to the full SHA-256. The first experiment was testing the adherence of each small scale version to the strict avalanche criterion, which is a common way to evaluate the fitness of a secure hash construct. The second was to obtain a measure of the balance of the new constructs.

4.1 Strict Avalanche Criterion

The adherence of the three small scale formulations to the strict avalanche criterion is shown in Figure 1; with SHA-256-X we denote the function obtained by taking the X most significant bits of the output of SHA-256. One can see that all

three formulations behave identically to their parent. However, it is important to note that there is a noticeable difference in the behaviour of the formulations with that of their parent (and the random case) when the sigma functions are replaced by identity transformations (NS stands for no sigma).

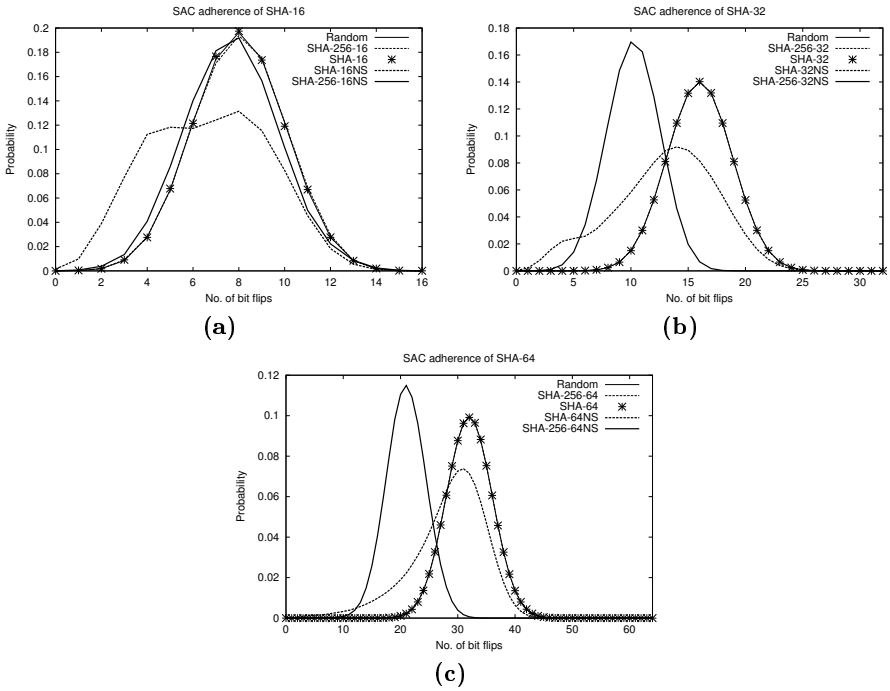


Fig. 1. SAC adherence of SHA-16, SHA-32 and SHA-64

Thus, we can see that careful selection of the sigma functions in each small scale variant is critical. It must be noted that there are many possible design choices resulting in constructions which behave well with respect to this criterion. Since NIST has not made their selection criteria publicly available, and adherence to the SAC alone is not enough to confirm the fitness of the introduced variants, we should consider further fitness tests.

4.2 Balance

The balance of the small scale formulations of SHA-256 was also measured using the method described earlier. The results of this experiment in the case of SHA-16 can be seen in Figure 2. It is interesting to note that also from the balance point of view, SHA-16 behaves exactly like a random function. Furthermore, the version of SHA-16 where all sigma functions are the simple identity

operation does *not* behave randomly. It is clear that any hashing construction which does not meet the minimum bound on balance is not a suitable candidate. The balance of each 16-bit subset of SHA-32 and SHA-64 has also been obtained

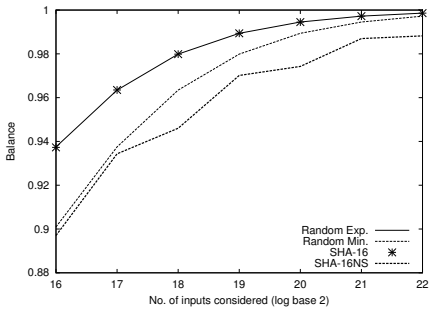


Fig. 2. Balance of SHA-16

experimentally. The size of the segments considered was chosen due to the extensive storage space and execution time required to test larger 8-bit increments thoroughly. These results are shown in Table 2.

Table 2. Balance of 16-bit portions of small scale SHA-256 formulations

Inp.	**S-16**	**S-32**$_{0..15}$	**S-32**$_{16..31}$	**S-64**$_{0..15}$	**S-64**$_{16..31}$	**S-64**$_{32..47}$	**S-64**$_{48..63}$
2^{16}	0.937288	0.937455	0.937589	0.937324	0.937457	0.937544	0.937816
2^{17}	0.963500	0.963503	0.963366	0.963071	0.963195	0.963497	0.963363
2^{18}	0.979860	0.979892	0.979911	0.979846	0.979667	0.979784	0.980001
2^{19}	0.989392	0.989422	0.989307	0.989413	0.989381	0.989332	0.989437
2^{20}	0.994510	0.994542	0.994503	0.994516	0.994498	0.994543	0.994547
2^{21}	0.997213	0.997236	0.997208	0.997223	0.997232	0.997221	0.997221
2^{22}	0.998599	0.998607	0.998592	0.998607	0.998606	0.998593	0.998600

One can see that all 16-bit subsets of SHA-32 and SHA-64 behave in a very similar manner to SHA-16, which was previously shown to behave like a random function with respect to this test. Hence, we now have small scale formulations which behave identically to SHA-256 under two different tests. Further assessments on the quality of these constructions are beyond the scope of this paper. However, one can see from the results of these experiments that it is possible to obtain seemingly well-behaved small scale versions of SHA-256 by using a careful design methodology.

It is also important to note that all experiments were performed on the full 64 rounds of SHA-256. Considering the impact of round reduction on the small scale variants compared to the full SHA-256 would also be interesting.

5 Conclusions and Future Work

In this paper, we have presented scaled versions of the Secure Hash Algorithm that can be used to implement security features on embedded devices; the most appealing variant, named SHA-64, can be used to efficiently calculate MACs on 8-bit microcontrollers.

We have studied the security of the proposed constructions by considering the adherence to the SAC and the balance metrics, also improving on the known results regarding the latter. In particular we calculate the expected minimum effort needed to find collisions for the full SHA-256 hash function.

Further work includes publishing a full specification and reference code of the small scale variants, analysing the effect of reducing the number of rounds and assessing the quality of the scaling method using different randomness tests.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A Survey on Sensor Networks. IEEE Communications Magazine, Vol. 40(8), 102–114, 2002.
2. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. Proceedings of CRYPTO 1994, 341–358, 1994.
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. Proceedings of CRYPTO 1996, 1–15, 1996.
4. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: Fast and Secure Message Authentication. Proceedings of CRYPTO 1999, 216–233, 1999.
5. TinySec Webpage, http://www.cs.berkeley.edu/ nks/tinysec/
6. Karlof, C., Sastry, N., Wagner, D.: TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004), 2004.
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES-The Advanced Encryption Standard. Springer-Verlag, 2002.
8. Klima, V.: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications. 3rd International Conference Security and Protection of Information, 2005.
9. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA1. To appear in CRYPTO 2005 proceedings.
10. NIST: Announcing the Secure Hash Standard. Federal Information Processing Standards Publication 180-2, 2002.
11. Cid, C., Murphy, S., Robshaw, M.: Small Scale Variants of the AES. Proceedings of Fast Software Encryption 2005.
12. Bellare, M., Kohno, T.: Hash Function Balance and Its Impact on Birthday Attacks. Proceedings of EUROCRYPT 2004, 401–418, 2004.
13. Gilbert, H., Handschuh, H.: Security Analysis of SHA-256 and Sisters. Proceedings of SAC 2003, 175–193, 2003.
14. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology, 4(1):3–72, 1991.
15. Matsui, M.: Linear Cryptanalysis method for DES cipher. Proceedings of EUROCRYPT 1993, 386–397, 1994.
16. Preneel, B.: Analysis and Design of Cryptographic Hash Functions. Doctoral Dissertation, Katholieke Universiteit Leuven, 1993.

SEA
a Scalable Encryption Algorithm for Small Embedded Applications

François-Xavier Standaert[1,2], Gilles Piret[1],
Neil Gershenfeld[2], Jean-Jacques Quisquater[1]

[1]UCL Crypto Group
Laboratoire de Microélectronique
Université Catholique de Louvain
Place du Levant, 3, B-1348 Louvain-La-Neuve, Belgium

[2]Center for Bits and Atoms
Massachusetts Institute of Technology
20 Ames Street, Cambridge, MA 02139, USA

{standaert,piret,quisquater}@dice.ucl.ac.be,
neilg@cba.mit.edu

Abstract. Most present symmetric encryption algorithms result from a tradeoff between implementation cost and resulting performances. In addition, they generally aim to be implemented efficiently on a large variety of platforms. In this paper, we take an opposite approach and consider a context where we have very limited processing resources and throughput requirements. For this purpose, we propose low-cost encryption routines (*i.e.* with small code size and memory) targeted for processors with a limited instruction set (*i.e.* AND, OR, XOR gates, word rotation and modular addition). The proposed design is parametric in the text, key and processor size, provably secure against linear/differential cryptanalysis, allows efficient combination of encryption/decryption and "on-the-fly" key derivation. Target applications for such routines include any context requiring low-cost encryption and/or authentication.

1 Introduction

Resource constrained encryption does not have a long history in symmetric cryptography. Noticeable examples of such ciphers are the Tiny Encryption Algorithm TEA [32] or Yuval's proposal [33]. However, both of them are relatively old and do not provide provable security against attacks such as linear and differential cryptanalysis. Present block ciphers, like the Advanced Encryption Standard Rijndael [17, 18] rather focus on finding a good tradeoff between cost, security and performances. While this approach is generally the most convenient, there exist contexts where more specialized ciphers are useful. As a motivating example, `ICEBERG` [30] is targeted for hardware implementations and shows significant efficiency improvements on these platforms compared to other algorithms.

Embedded applications such as building infrastructures present a significant opportunity and challenge for such new cryptosystems. Introducing programmability into the configuration of lights and switches, thermostats and air handlers, promises to improve the cost of construction, flexibility in occupancy, and energy

efficiency of buildings. But meeting this demand on a scale compatible with the economics of the trillion-dollar construction industry is going to require secure lightweight implementations of peer-to-peer networks in resource-constrained systems. The Internet-0 approach to end-to-end modulation for interdevice internetworking is typically appropriate in this limit [20]. $\text{SEA}_{n,b}$ constitutes a suitable solution for low-cost encryption/authentication within such networks. RFID's or any power/space-limited applications are similarly targeted.

In this paper, we consequently consider a general context where we have very limited processing resources (*e.g.* a small processor) and throughput requirements. It yields design criteria such as: low memory requirements, small code size, limited instruction set. In addition, we propose the flexibility as another unusual design principle. $\text{SEA}_{n,b}$ is parametric in the text, key and processor size. Such an approach was motivated by the fact that many algorithms behave differently on different platforms (*e.g.* 8-bit or 32-bit processors). In opposition, $\text{SEA}_{n,b}$ allows to obtain a small encryption routine targeted to any given processor, the security of the cipher being adapted in function of its key size.

Beyond these general guidelines, alternative features were wanted, including the efficient combination of encryption and decryption or the ability to derive keys "on the fly". Both of them result in an improved efficiency and are particularly relevant in contexts where the same constrained device has to perform encryption and decryption operations (*e.g.* authentication). As a final bonus, the simplicity of $\text{SEA}_{n,b}$ makes its implementation in assembly code straightforward.

2 Specifications

2.1 Parameters and definitions

$\text{SEA}_{n,b}$ operates on various text, key and word sizes. It is based on a Feistel structure with a variable number of rounds, and is defined with respect to the following parameters:

- n: plaintext size, key size.
- b: processor (or word) size.
- $n_b = \frac{n}{2b}$: number of words per Feistel branch.
- n_r: number of block cipher rounds.

As only constraint, it is required that n is a multiple of $6b$. For example, using an 8-bit processor, we can derive 48, 96, 144, ...-bit block ciphers, respectively denoted as $\text{SEA}_{48,8}$, $\text{SEA}_{96,8}$, $\text{SEA}_{144,8}$, ...

Let x be a $\frac{n}{2}$-bit vector. In the following, we will consider two representations:

- Bit representation: $x_b = x(\frac{n}{2}-1)\ x(\frac{n}{2}-2)\ \ldots,\ x(2)\ x(1)\ x(0)$.
- Word representation: $x_W = x_{n_b-1}\ x_{n_b-2}\ \ldots\ x_2\ x_1\ x_0$.

2.2 Basic operations

Due to its simplicity constraints, $\mathrm{SEA}_{n,b}$ is based on a limited number of elementary operations (selected for their availability in any processing device) denoted as follows: (1) bitwise XOR $\oplus$, (2) substitution box S, (3) word (left) rotation R and inverse word rotation R^{-1}, (4) bit rotation r, (5) addition mod 2^b $\boxplus$. These operations are formally defined as follows:

1. Bitwise XOR $\oplus$: The bitwise XOR is defined on $\frac{n}{2}$-bit vectors:

$$\oplus : \mathbb{Z}_2^{\frac{n}{2}} \times \mathbb{Z}_2^{\frac{n}{2}} \to \mathbb{Z}_2^{\frac{n}{2}} : x, y \to z = x \oplus y \;\Leftrightarrow\; z(i) = x(i) \oplus y(i), \quad 0 \le i \le \frac{n}{2} - 1$$

2. Substitution box S: $\mathrm{SEA}_{n,b}$ uses the following 3-bit substitution table:

$$S_T := \{0, 5, 6, 7, 4, 3, 1, 2\},$$

in C-like notation. For efficiency purposes, it is applied bitwise to any set of three words of data using the following recursive definition:

$$S : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to x = S(x) \Leftrightarrow$$

$$\begin{aligned} &x_{3i} = (x_{3i+2} \wedge x_{3i+1}) \oplus x_{3i}, \\ &x_{3i+1} = (x_{3i+2} \wedge x_{3i}) \oplus x_{3i+1}, \\ &x_{3i+2} = (x_{3i} \vee x_{3i+1}) \oplus x_{3i+2}, \quad 0 \le i \le \tfrac{n_b}{3} - 1, \end{aligned}$$

where $\wedge$ and $\vee$ respectively represent the bitwise AND and OR.

3. Word rotation R: The word rotation is defined on n_b-word vectors:

$$R : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = R(x) \Leftrightarrow \begin{aligned} &y_{i+1} = x_i, \quad 0 \le i \le n_b - 2, \\ &y_0 = x_{n_b - 1} \end{aligned}$$

4. Bit rotation r: The bit rotation is defined on n_b-word vectors:

$$r : \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x \to y = r(x) \Leftrightarrow \begin{aligned} &y_{3i} = x_{3i} \ggg 1, \\ &y_{3i+1} = x_{3i+1}, \\ &y_{3i+2} = x_{3i+2} \lll 1, \quad 0 \le i \le \tfrac{n_b}{3} - 1, \end{aligned}$$

where $\ggg$ and $\lll$ represent the cyclic right and left shifts inside a word.

5. Addition mod2^b $\boxplus$: The mod 2^b addition is defined on n_b-word vectors:

$$\boxplus : \mathbb{Z}_{2^b}^{n_b} \times \mathbb{Z}_{2^b}^{n_b} \to \mathbb{Z}_{2^b}^{n_b} : x, y \to z = x \boxplus y \;\Leftrightarrow\; z_i = x_i \boxplus y_i, \quad 0 \le i \le n_b - 1$$

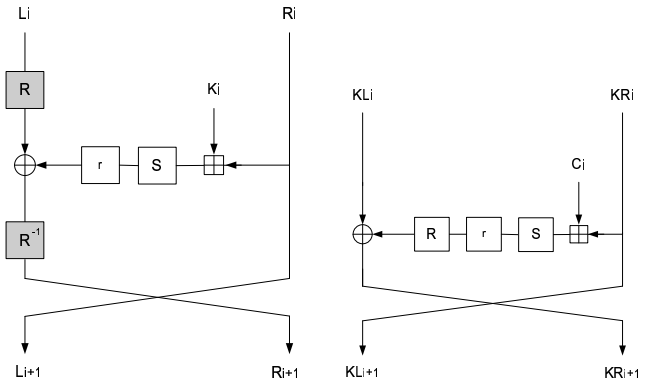


Fig. 1. Encrypt/decrypt round and key round.

2.3 The round and key round

Based on the previous definitions, the encrypt round F_E, decrypt round F_D and key round F_K are pictured in Figure 1 and defined as the functions $F : \mathbb{Z}_{2^{n/2}}^2 \times \mathbb{Z}_{2^{n/2}} \to \mathbb{Z}_{2^{n/2}}^2$ such that:

$$
\begin{aligned}
[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) \quad &\Leftrightarrow \quad R_{i+1} = R(L_i) \oplus r\big(S(R_i \boxplus K_i)\big) \\
&\qquad\quad L_{i+1} = R_i \\
[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) \quad &\Leftrightarrow \quad R_{i+1} = R^{-1}\Big(L_i \oplus r\big(S(R_i \boxplus K_i)\big)\Big) \\
&\qquad\quad L_{i+1} = R_i \\
[KL_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) \quad &\Leftrightarrow \quad KR_{i+1} = KL_i \oplus R\Big(r\big(S(KR_i \boxplus C_i)\big)\Big) \\
&\qquad\quad KL_{i+1} = KR_i
\end{aligned}
$$

2.4 The complete cipher

The cipher iterates an odd number n_r of rounds. The following pseudo-C code encrypts a plaintext P under a key K and produces a ciphertext C. P, C and K have a parametric bit size n. The operations within the cipher are performed considering parametric b-bit words.

C=**SEA**$_{n,b}(P, K)$
{
  % initialization:
    $L_0 \& R_0 = P$;
    $KL_0 \& KR_0 = K$;
  % key scheduling:
    for i **in** 1 **to** $\lfloor \frac{n_r}{2} \rfloor$
      $[KL_i, KR_i] = F_K(KL_{i-1}, KR_{i-1}, C(i))$;
    switch $KL_{\lfloor \frac{n_r}{2} \rfloor}$, $KR_{\lfloor \frac{n_r}{2} \rfloor}$;
    for i **in** $\lceil \frac{n_r}{2} \rceil$ **to** $n_r - 1$
      $[KL_i, KR_i] = F_K(KL_{i-1}, KR_{i-1}, C(r - i))$;

% encryption:
for i **in** 1 **to** $\lceil \frac{n_r}{2} \rceil$
$[L_i, R_i] = F_E(L_{i-1}, R_{i-1}, KR_{i-1})$;
for i **in** $\lceil \frac{n_r}{2} \rceil + 1$ **to** n_r
$[L_i, R_i] = F_E(L_{i-1}, R_{i-1}, KL_{i-1})$;
% final:
$C = R_{n_r} \& L_{n_r}$;
switch KL_{n_r-1}, KR_{n_r-1};
},

where & is the concatenation operator, $KR_{\lfloor \frac{n_r}{2} \rfloor}$ is taken before the switch and $C(i)$ is a n_b-word vector of which all the words have value 0 excepted the LSW that equals i. Decryption is exactly the same, using the decrypt round F_D.

3 Security analysis

3.1 Design properties of the components

Substitution box S: The substitution box was searched exhaustively in order to meet the following security and efficiency criteria:

- λ-parameter[1]: 1/2.
- δ-parameter[2]: 1/4.
- Maximum nonlinear order, namely 2.
- Recursive definition.
- Minimum number of instructions.

Remark that, if 3-operand instructions are available, the recursive definition allows to perform the substitution box in 2 operations per word of data. As a comparison, the 3×3 bitwise substitution box used in 3-WAY [15] requires 3. The counterpart of this efficiency is the presence of two fixed points in the table.

Bit and word rotations r and R: The cyclic rotations were defined in order to provide predictable low-cost diffusion within the cipher, when combined with the bitslice substitution box. It is illustrated in Figure 2 for a single substitution box scheme with parameters $n = 48$, $b = 8$, $n_b = 3$.

Looking at the figure, it can be seen that $\text{SEA}_{n,b}$ divides its data in $\frac{2n_b}{3}$ blocks of 3 words. The substitution box is applied in parallel to these blocks. Therefore, the diffusion process (starting with one single active bit in the left branch) is divided into two steps[3]:

[1] We define the *bias* of a linear approximation that holds with probability p as $\epsilon = |p - 1/2|$. The λ-*parameter* of a substitution box is equal to 2 times the bias of its best linear approximation.

[2] The δ-parameter equals the probability of the best differential approximation.

[3] For simplicity purposes, we don't consider the additional diffusion provided by the carry propagation in the $\mod 2^b$ key addition in this discussion.

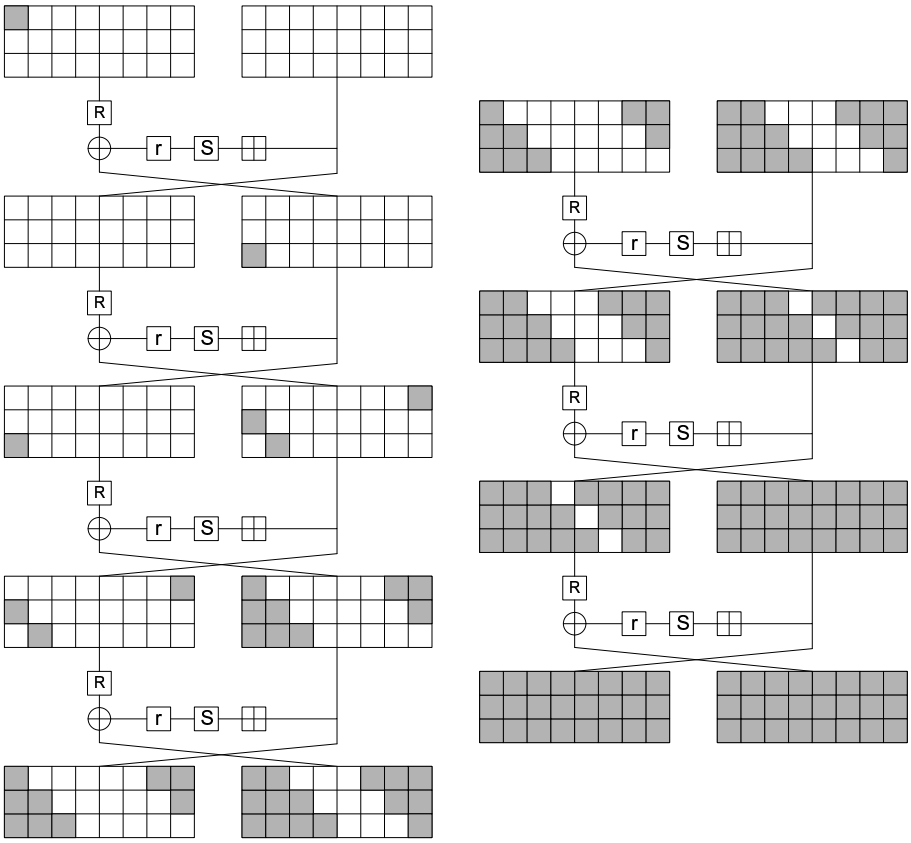


Fig. 2. Diffusion process: grey boxes represent active bits.

- During an initialization step, the single active bit has to be propagated to all the words of the cipher (*e.g.* to our six words in Figure 2).
- During the second step, the diffusion has to be completed within each block.

The first phase is obtained by the combination of the word rotation R (which is the only transform to provide inter-word diffusion) with the substitution box. It requires at most n_b rounds to be completed (in our example, $n_b = 3$ which yields 3 rounds). Once every word has at least one active bit, the combination of r and S yields six more active bits per block in each round. Therefore, finishing the diffusion of all the blocks requires at most $\lfloor b/2 \rfloor$ rounds. Combining these observations, the diffusion is complete after $n_b + \lfloor b/2 \rfloor$ rounds.

Addition mod 2^b $\boxplus$: Using a mod 2^b key addition in place of a bitwise XOR was motivated by different reasons: (1) improvement of the diffusion process, (2) improvement of the non-linearity, (3) same cost/speed as the bitwise XOR in most processors, (4) necessity to avoid structural attacks (see next section).

Overall structure: The overall structure of the cipher follows the Feistel strategy. However, a few points are specific to $\text{SEA}_{n,b}$, namely the key schedule and the position of R, R^{-1} in the encrypt/decrypt rounds.

The key schedule is designed such that the master key is encrypted during half the rounds and decrypted during the other half. It allows to obtain a particular structure of the sequence of round keys such that the key expansion is exactly the same in encryption and decryption. Namely, we have:

$$K_0, K_1, K_2, \ldots, K_{\lfloor \frac{r}{2} \rfloor}, K_{\lfloor \frac{r}{2} \rfloor - 1}, \ldots, K_2, K_1, K_0$$

As a consequence of this structure, the encryption/decryption rounds cannot keep the traditional Feistel structure: it would result in having identical encryption and decryption functions. This is the reason of moving the word rotation to the left branch of the Feistel round.

3.2 Resistance against known attacks

Linear and differential cryptanalysis. From the properties of the substitution box, we can compute bounds for the best linear and differential characteristics through the cipher. We use the following lemma [29]:

Lemma 1. Let f be the bijective nonlinear function of a 3-round Feistel cipher. Assuming that the linear parameter of f is smaller than λ and its differential parameter is smaller than δ, then the linear, differential parameters of the 3-round cipher Δ, Λ are respectively smaller than λ^2, δ^2.

For a n-bit block cipher, it is required that $\Delta < 2^{-n}$ in order to have resistance against differential cryptanalysis [4]. As our nonlinear function S has parameter $\delta = 2^{-2}$, it is required that:

$$\left(2^{-2}\right)^{2n_r/3} < 2^{-n}$$

Similarly, for resistance against linear cryptanalysis [28], it is required that $\Lambda < 2^{-\frac{n}{2}}$. As our nonlinear function S has parameter $\lambda = 2^{-1}$, it yields:

$$\left(2^{-1}\right)^{2n_r/3} < 2^{-\frac{n}{2}}$$

In both cases, the required number of rounds is: $n_r \geq 3n/4$.

Extensions of linear and differential cryptanalysis. Classical extensions of linear and differential cryptanalysis are non-linear approximations of outer rounds [26], bi-linear cryptanalysis [14], differential-linear cryptanalysis [27], multiple linear cryptanalysis [22, 10], boomerang [31] and rectangle [8] attacks,... However these extensions usually imply only a small improvement compared to the basic attacks. As a matter of fact, non-linear approximations of outer rounds allow to improve the bias of one or two rounds only. Regarding bi-linear cryptanalysis, we quote the author of [14]: *For ciphers similar to DES, based on small substitution boxes, we claim that bi-linear cryptanalysis is very closely related to LC, and we do not expect to find a bi-linear attack much faster than by LC.* It is difficult to evaluate the efficiency of multiple linear cryptanalysis, but it seems more promising for big substitution boxes (as mentioned in [22]). Moreover the improvement on classical cryptanalysis obtained in [10] for the case of DES (which shares with $\mathrm{SEA}_{n,b}$ a Feistel structure and a poor diffusion) is limited. Finally, the complexity of differential-linear cryptanalysis and of the boomerang attack and its variants is inherently greater than the one of the basic attacks. As an example, the boomerang (or rectangle) attack allows us to use two short differentials instead of a long one, but using a long differential with probability pq is in general highly preferable to applying a boomerang attack with two short differentials of probability p and q. Therefore although these attacks can perform slightly better in specific cases, the expected improvement is never outstanding. The conclusion is that these extensions actually deserve to be considered in the estimation of the number of rounds necessary to achieve security, but that a reasonable multiplicative factor should be enough to take them into account.

A dedicated attack against a modified version. For $x \in \mathbb{Z}_{2^b}^{n_b}$, we denote by $x \lll a$ the left rotation by a bits of each of the n_b words of x. The non-linear and diffusion layers have the following properties:

- $S(x \lll a) = S(x) \lll a$
- $r(x \lll a) = r(x) \lll a$
- $R(x \lll a) = R(x) \lll a$

Consider a modified version of our cipher where key addition is performed using $\oplus$ rather than modular addition. We denote it by $\oplus$-$\mathrm{SEA}_{n,b}$. As a consequence of the previous observations, the modified round F'_E has the following property:

Property 1. Let $[L_1, R_1]$ and $[L_2, R_2]$ be such that

$$\begin{aligned} [L_1, R_1] \oplus [L_2, R_2] &= \Delta_1 \\ \text{and} \quad F'_E([L_1, R_1], K) \oplus F'_E([L_2, R_2], K) &= \Delta_2, \end{aligned}$$

for a given round key K. Then if we define $[L_1^*, R_1^*] := [L_1, R_1] \lll a$ and $[L_2^*, R_2^*] := [L_2, R_2] \lll a$, for a given a, we have:

$$\begin{aligned} [L_1^*, R_1^*] \oplus [L_2^*, R_2^*] &= \Delta_1 \lll a \\ \text{and} \quad F'_E([L_1^*, R_1^*], K) \oplus F'_E([L_2^*, R_2^*], K) &= \Delta_2 \lll a \end{aligned}$$

This property is iterative, in the sense that it also holds for the composition of several rounds. It is immediate to deduce from it a distinguisher on the modified cipher, which requires 4 chosen encryption queries.

In $\mathrm{SEA}_{n,b}$, the key addition is performed word-wise mod 2^b. As the property $(\Delta \lll a) \boxplus K = (\Delta \boxplus K) \lll a$ is prevented by certain carry propagations, it only holds with a probability p, depending on the word size b. In the worst case, $b = 1$ and we have $p = 1$ (*i.e.* $\oplus$ and $\boxplus$ are equivalent). For larger b's, we have:

b	1	2	3	4	5	6	7	8
p	1	0.625	0.4375	0.3047	0.2129	0.1489	0.1042	0.0730

Of course, these value are averaged for all possible keys and certain keys (*e.g.* "*all zeroes*") yield no carry propagation at all. However, the design properties of the key schedule prevent $\mathrm{SEA}_{n,b}$ from having such weak keys. It avoids this structural distinguisher to be propagated through more than a few rounds.

Square attacks. We explored square attacks [16] on $\mathrm{SEA}_{48,8}$. More precisely, we considered all possible sets of inputs to one branch of the Feistel structure, where the input to some of the substitution boxes is active (*i.e.* takes all possible input values the same number of times), and the input to the other substitution boxes is constant. The other branch is also constant. Therefore the number of plaintexts considered goes from 2^3 (when the input to only one substitution box is active) to 2^{21} (when the input to 7 substitution boxes is active). Our experiments showed that square attacks do not allow to pass through more rounds than the diffusion pattern illustrated in Figure 2. It is expected that it remains the same when different parameters n and b are considered, which implies that $n_b + \lfloor b/2 \rfloor$ rounds are enough to prevent square attacks. Note that although our observations also hold for $\oplus$-$\mathrm{SEA}_{n,b}$, the use of addition mod 2^b provides better resistance against square attacks.

Truncated and impossible differentials. As for square attacks, the diffusion analysis illustrated in Figure 2 provides an estimation of the number of rounds required to prevent truncated differential attacks [25]. Impossible differentials [7] are usually built by concatenating two incompatible truncated differentials. As a consequence, we estimate the number of rounds necessary to prevent the construction of an impossible differential distinguisher as $2 \cdot (n_b + \lfloor b/2 \rfloor)$.

Interpolation attacks. The interpolation attack [21] is possible when the whole cipher can be written as a relatively simple algebraic expression. It requires the substitution box to have a compact expression, and the diffusion layer to permit the composition of these expressions. In the case of $\mathrm{SEA}_{n,b}$, there is a priori no such expression, and the bitwise diffusion would make the combination of algebraic expressions difficult anyway.

Slide attacks. The sequence of round keys of $\text{SEA}_{n,b}$ is the same as the one of `ICEBERG`. Therefore the analysis done in [30] is still valid. Namely, the non-periodicity of the sequence should make slide attacks [11, 12] irrelevant. The particular structure of this sequence also has some similarities with the one of GOST, of which the vulnerability against slide attacks is examined in [12]. None of the attacks presented in [12] seems to be applicable to our cipher.

Related-key attacks. The first related-key attack has been described in [5]. It is the related-key counterpart of the slide attack. Such an attack is applicable when a round key K_i is computed from the previous round key K_{i-1} using a function f which is always the same: $K_i = f(K_{i-1})$. However in the case of $\text{SEA}_{n,b}$, a round constant that changes for each key round is used, which prevents this attack. Another type of related-key attack is the differential related-key attack [23, 24]. The non-linearity of the $\text{SEA}_{n,b}$ key schedule should prevent it. Moreover, note that the improvement of the differential related-key attack over classical differential cryptanalysis usually results from the fact that choosing a given round key difference allows to "counter" the effect of the diffusion layer on the differential characteristic; a typical example is the attack on 3-WAY [24]. As the security of $\text{SEA}_{n,b}$ against differential cryptanalysis results from its large number of rounds rather than from its diffusion, this effect is not relevant here.

Complementation properties. The DES has the following complementation property: if $P \xrightarrow{K} C$ denotes the fact that encryption of P under key K gives ciphertext C, then: $P \xrightarrow{K} C \iff \overline{P} \xrightarrow{\overline{K}} \overline{C}$. The non-linear key scheduling and the presence of carry propagations in the actual $\text{SEA}_{n,b}$ algorithm prevents this property. We are not aware of any other similar structural feature in the design.

Algebraic attacks. Algebraic attacks intend to exploit the simple algebraic structure of a block cipher. For example, certain block ciphers can be written as an overdefined system of quadratic equations. Reference [13] argues that a method called XSL might provide a way to effectively solve this type of equations and recover the key from a few plaintext-ciphertext pairs.

Clearly, $\text{SEA}_{n,b}$ has a simple algebraic structure, as it is based on a 3-bit substitution box. Therefore, if such an attack practically applies to a cipher like Serpent [1], it is likely applicable to one of the versions of our routines. As the complexity of XSL is supposedly polynomial in the plaintext size and number of rounds, it is specially true when those values increase. However, as the criteria for these techniques to be successful are presently speculated [9], we did not consider them in our design.

3.3 Suggested number of rounds

From the previous descriptions, the minimum required number of rounds to provide security against known attacks would be $\frac{3n}{4} + 2 \cdot (n_b + \lfloor b/2 \rfloor)$. This roughly corresponds to the number of rounds to resist linear/differential attacks plus twice the number of rounds to obtain complete diffusion (to prevent both structural attacks and outer rounds improvements of statistical attacks). A more conservative approach (applied in most present block ciphers) would be to take a large security margin, *e.g.* by doubling this number of rounds[4]. n_r has to be odd: we add one if it is even. We also assume a minimum word size $b \geq 8$ bits.

4 Performance analysis

$\text{SEA}_{n,b}$ is targeted for being implemented on low-cost processors, with little code size and a small instruction set. However, $\text{SEA}_{n,b}$'s simple structure makes it easy to implement on any processor. In appendix, we propose a pseudo-assembly code of an encryption/decryption design with "on the fly" key scheduling. The implementation objectives were, in decreasing order of importance: (1) low RAM and registers usage, (2) low code size and (3) speed. It is based on the following (very) reduced instruction set (assuming 2-operand instructions only):

- Arithmetic and logic operators: $\vee, \wedge, \oplus, \boxplus, \ggg, \lll$.
- Branch instructions: goto, subroutine call and return.
- Comparison, load RAM in register, store register in RAM.

According to the code in appendix, the performances can be roughly estimated as follows. First, the combined number of RAM words and registers equals $5n_b + 3$. Then, the code size and implementation time (both in expressed in ops.) is evaluated by summing the values given in appendix. For the code size, it directly yields $31n_b + 36$ ops. For the implementation time, the round and key round respectively require $12n_b + 11$ ops. and $10n_b + 11$ ops. It yields a total of $(n_r - 1) \times (12n_b + 11 + 10n_b + 11 + 7) + (12n_b + 11) + 8n_b + 7$. These values are summarized in Table 1. Remark that, due to the particular structure of the

	# ram	# regs.	code size (ops.)	implementation time (ops.)
$\text{SEA}_{n,b}$	$4n_b$	$n_b + 3$	$31n_b$+36	$(n_r - 1) \times (22n_b + 29) + 20n_b + 18$

Table 1. Performance evaluation of $\text{SEA}_{n,b}$ (encryption + decryption).

key scheduling, we do not need to keep the master key in memory as, at the end of an encryption/decryption, we have $K_{n_r-1} = K_0$. Remark also that this implementation uses a low number of registers, namely $n_b + 3$. However, if more registers are available, they can be traded for RAM words, which will result in lower code size and faster implementation.

[4] Note that the additional non-linearity provided by the modular addition also provides a security margin, under-estimated in our predictions.

For illustration purposes, we implemented $\text{SEA}_{n,b}$ on Atmel AVR ATtiny [3] and ARM [2] microprocessors. The Atmel ATtiny represents a typical target for such a low-cost encryption routine. We chose the ARM platform in order to provide rough comparisons between $\text{SEA}_{n,b}$ and the AES Rijndael.

Algorithm	E/D	Device	# ram	# regs.	code size	# clock cycles	# cycles × code size
$\text{SEA}_{96,8}$	yes	Atmel ATtiny	1	32	386	17 745	6849.10^3
$\text{SEA}_{192,32}$	yes	ARM (risc-32)	6	12	420	27 059	$11\ 364.10^3$
Rijndael [19]	no	ARM (risc-32)	16	12	1404	2889	4056.10^3
$\text{SEA}_{128,32}$	yes	ARM (risc-32)	6	12	280	18 039	5050.10^3

Table 2. Comparisons: the code size is expressed in bytes. The results of $\text{SEA}_{128,32}$ where obtained by multiplying the code size and number of cycles of $\text{SEA}_{192,32}$ by 2/3, since 128 is not a multiple of 6.

While direct comparisons are made difficult by their high dependencies on the target devices, the following general comments can be made:

- $\text{SEA}_{n,b}$ designs combine encryption and decryption more efficiently than most other encryption algorithms. In particular, key agility in decryption is usually not possible (*e.g.* for the AES Rijndael).
- The combined number of RAM words and registers of $\text{SEA}_{n,b}$ implementations (*i.e.* $5n_b + 3$) is generally lower than for other block ciphers.
- The code size of $\text{SEA}_{n,b}$ is generally lower than for other block ciphers implemented on similar platforms.

The flexibility of $\text{SEA}_{n,b}$ also makes it less sensitive to the choice of a processor than fixed-sized algorithms, although it is obvious that large buses improve efficiency. The drawback of these limited resources is in the number of cycles required for the encryption (*i.e.* $\text{SEA}_{n,b}$ trades space for time, which may be relevant due to present processors speeds). Looking at the code size - cycles product, the efficiency of $\text{SEA}_{n,b}$ remains similar to the one of Rijndael (encryption only) that is well known for its efficient smart cards implementations.

5 Conclusion

$\text{SEA}_{n,b}$ is a scalable encryption algorithm targeted for small embedded applications. The plaintext size n, key size n and processor (or word) size b are parameters of the design. The structure of $\text{SEA}_{n,b}$ allows provable security against linear/differential attacks and a fast evaluation of the cipher efficiency on any RISC machine. The typical performances of $\text{SEA}_{n,b}$ (encryption + decryption) for present key sizes and processors (*e.g.* 128-bit key, 1 Mhz 8-bit RISC) are in the range of an encryption/decryption in a few milliseconds, using a few hundreds bytes of ROM. One additional advantage of the design is its extreme simplicity. Based on the pseudo code provided in this paper, it is expected that the implementation of the cipher in assembly can be done within a few hours.

Acknowledgements: The authors would like to thank François Koeune for his help and comments about ARM assembly tools and the NSF grant CCR-0122419, Center for Bits and Atoms.

References

1. R. Anderson, E. Biham, L. Knudsen, *Serpent: A Flexible Block Cipher With Maximum Assurance*, in the proceedings of The First Advanced Encryption Standard Candidate Conference, Ventura, California, USA, August 1998.
2. ARM, *32-bit RISC microprocessors*, http://www.arm.com/products/CPUs/
3. Atmel, *AVR 8-Bit RISC*, http://www.atmel.com/products/AVR/
4. E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, 1993, Springer Verlag.
5. E. Biham, *New types of cryptanalytic attacks using related keys*, Journal of Cryptology, vol 7, num 4, pp 229-246, Fall 1994, Springer Verlag.
6. E. Biham, A. Biryukov, A. Shamir, *Miss-in-the-Middle Attacks on IDEA, Khufu, and Khafre*, in the proceedings of FSE 1999, Lecture Notes in Computer Sciences, vol 1636, pp 124-138, Rome, Italy, March 1999, Springer-Verlag.
7. E. Biham, A. Biryukov, A. Shamir, *Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials*, in the proceedings of Eurocrypt 1999, Lecture Notes in Computer Sciences, vol 1592, pp 12-23, Prague, Czech Republic, May 1999, Springer Verlag.
8. E. Biham, O. Dunkelman, N. Keller, *The Rectangle Attack, Rectangling the Serpent*, in the proceedings of Eurocrypt 2001, Lecture Notes in Computer Science, vol 2045, pp 340-357, Innsbruck, Austria, May, 2001 Springer-Verlag.
9. A. Biryukov, C. De Cannière, *Block Ciphers and Systems of Quadratic Equations*, in the proceedings of FSE 2003, Lecture Notes in Computer Science, vol 2887, pp 274-289, Lund, Sweden, February 2003, Springer-Verlag.
10. A. Biryukov, C. De Cannière, M. Quisquater, *On Multiple Linear Approximations*, in the proceedings of Crypto 2004, Lecture Notes in Computer Science, vol 3152, pp 1-22, Santa Barbara, USA, August 2004, Springer-Verlag.
11. A. Biryukov, D. Wagner, *Slide attacks*, in the proceedings of FSE 1999, Lecture Notes in Computer Sciences, vol 1636, pp 245-259, Rome, Italy, March 1999, Springer-Verlag.
12. A. Biryukov, D. Wagner, *Advanced Slide Attacks*, in the proceedings of Eurocrypt 2000, Lecture Notes in Computer Science, vol 1807, pp 589-606, Bruges, Belgium, May 2000, Springer-Verlag.
13. N. Courtois, J. Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, in the proceedings of Asiacrypt 2002, Lecture Notes in Computer Science, vol 2501 , pp 267-287, Queenstown, New Zealand, December 2002, Springer-Verlag.
14. N. Courtois, *Feistel Schemes and Bi-linear Cryptanalysis*, in the proceedings of Crypto 2004, Lecture Notes in Computer Science, vol 3152, pp 23-40, Santa Barbara, USA, August 2004, Springer-Verlag.
15. J. Daemen, R. Govaerts, J. Vandewalle, *A New Approach Towards Block Cipher Design*, in the proceedings of FSE 1993, Lecture Notes in Computer Science, vol 809, pp 18-32, Cambridge, UK, December 1993, Springer-Verlag.
16. J. Daemen, L. Knudsen, V. Rijmen, *The Block Cipher SQUARE*, in the proceedings of FSE 1997, Lecture Notes in Computer Science, vol 1267, pp 149-165, Haifa, Isreal, January 1997, Springer-Verlag.

17. J. Daemen, V. Rijmen, *The Design of Rijndael,* Springer-Verlag, 2001.
18. FIPS 197, *"Advanced Encryption Standard,"* Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26, 2001.
19. G. Hachez, F. Koeune, J.-J. Quisquater, *cAESar Results: Implementation of Four AES Candidates on Two Smart Cards*, in the proceedings of the Second Advanced Encryption Standard Candidate Conference, pp 95-108, Rome, Italy, March 1999.
20. N. Gershenfeld, R. Krikorian, D. Cohen, *The Internet of Things*, Scientific American, Octobre 2004, pp 76-81.
21. T. Jakobsen, L.R. Knudsen, *The Interpolation Attack on Block Ciphers*, in the proceedings of FSE 1997, Lecture Notes in Computer Science, vol 1267, pp 28-40, Haifa, Israel, January 1997, Springer-Verlag.
22. B.S. Kaliski, M.J.B. Robshaw, *Linear Cryptanalysis using Multiple Approximations*, in the proceedings of Crypto 1994, Lecture Notes in Computer Science, vol 839, pp 26-39, Santa Barbara, California, USA, August 1994, Springer-Verlag.
23. J. Kelsey, B. Schneier, D. Wagner. *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, in the proceedings of Crypto 1996, Lecture Notes in Computer Science, vol 1109, pp 237-251, Santa Barbara, California, USA, August 1996, Springer-Verlag.
24. J. Kelsey, B. Schneier, D. Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*, in the proceedings of ICICS 1997, Lecture Notes in Computer Sciences, vol 1334, pp 233-246, Bejing, China, November 1997, Springer-Verlag.
25. L.R. Knudsen, *Truncated and Higher Order Differentials*, in the proceedings of FSE 1995, Lecture Notes in Computer Sciences, vol 1008, pp 196-211, Leuven, Belgium, 1995, Springer-Verlag.
26. L.R. Knudsen and M.J.B. Robshaw, *Non-Linear Approximations in Linear Cryptanalysis*, in the proceedings of Eurocrypt 1996, Lecture Notes in Computer Science, vol 1070, pp 224-236, Saragossa, Spain, May 1996, Springer-Verlag.
27. S. Langford, M. Hellman, *Differential-Linear Cryptanalysis*, in the proceedings of Crypto 1994, Lecture Notes in Computer Science, vol 839, pp 17-25, Santa Barbara, California, USA, August 1994, Springer-Verlag.
28. M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, in the proceedings of Eurocrypt 1993, Lecture Notes in Computer Science, vol 765, pp 386-397, Lofthus, Norway, May 1993, Springer-Verlag.
29. M. Matsui, *Supporting Document of MISTY1*, Submission to the NESSIE project, available from http://www.cosic.esat.kuleuven.ac.be/nessie/
30. F.-X. Standaert, G. Piret, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, *ICEBERG : an Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware*, in the proceedings of FSE 2004, Lecture Notes in Computer Science, vol 3017, pp 279-299, New Delhi, India, February 2004, Springer-Verlag.
31. D. Wagner, *The Boomerang Attack*, in the proceedings of FSE 1999, Lecture Notes in Computer Sciences, vol 1636, pp 156-170, Rome, Italy, March 1999, Springer-Verlag.
32. D.J. Wheeler, R. Needham, *TEA, a Tiny Encryption Algorithm*, in the proceedings of FSE 1994, Lecture Notes in Computer Science, vol 1008, pp 363-366, Leuven, Belgium, December 1994, Springer-Verlag.
33. G. Yuval, *Reinventing the Travois: Encryption/MAC in 30 ROM Bytes*, in the proceedings of FSE 1997, Lecture Notes in Computer Science, vol 1267, pp 205-209, Haifa, Israel, January 1997, Springer-Verlag.

Pseudo-assembly code:	**# ram**	**# regs.**	**# ops.**
% Init			
L_0, R_0, KL_0, KR_0 stored in RAM;	$4n_b$		
Set $i = 1$;		1	
Set E/D;		1	
% Subroutines (including return):			
S: $reg \leftarrow S(reg)$;		$n_b + 1$	$3n_b$+1
r: $reg \leftarrow r(reg)$;		n_b	$n_b + 1$
sw: switch KL_i, KR_i;		2	$4n_b + 1$
Round:			
$reg \leftarrow R_i$;		n_b	n_b
if $i \leq \lceil n_r/2 \rceil$			1
goto a:			1
$reg \leftarrow reg \boxplus KL_i$;		$n_b + 1$	$2n_b$
goto b:			1
a: $reg \leftarrow reg \boxplus KR_i$;		$n_b + 1$	$2n_b$
b: call S;			1
call r;			1
if E/D=1;			1
goto c:			1
$reg \leftarrow reg \oplus L_i$;		$n_b + 1$	$2n_b$
goto d:			1
c: $reg \leftarrow reg \oplus R(L_i)$;		$n_b + 1$	$2n_b$
d: $L_{i+1} \leftarrow R_i$;		1	$2n_b$
if E/D=1;			1
goto e:			1
$R_{i+1} \leftarrow R^{-1}(reg)$;		n_b	n_b
goto f:			1
e: $R_{i+1} \leftarrow reg$;		n_b	n_b
f: return;			1
Key round:			
$reg \leftarrow KR_i$;		n_b	n_b
if $i < \lceil n_r/2 \rceil$			1
goto g:			1
$temp \leftarrow n_r - i$;		1	2
$reg \leftarrow reg \boxplus temp$;		$n_b + 1$	1
goto h:			1
g: $reg \leftarrow reg \boxplus i$;		n_b	1
h: call S;			1
call r;			1
$reg \leftarrow R(reg) \oplus KL_i$;		$n_b + 1$	$2n_b + 1$
$KL_{i+1} \leftarrow KR_i$;		1	$2n_b$
$KR_{i+1} \leftarrow reg$;		n_b	n_b
return;			1
% Total:			
j: call round;			1
if $i \neq \lceil n_r/2 \rceil$			1
goto k:			1
call sw;			1
k: if $i = n_r$			1
goto end:			1
call key round;			1
$i = i + 1$;			1
goto j:			1
end: call sw;			1
switch L_i, R_i;		2	$4n_b$