A Low-Resource Public-Key Identification Scheme for RFID Tags and Sensor Nodes

Yossef Oren
Computer and Network Security Lab
School of Electrical Engineering
Tel-Aviv University
P.O. Box 39040, Tel-Aviv 69978, Israel
yos@eng.tau.ac.il

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

ABSTRACT

We revisit a public key scheme presented by Shamir in [19] (and simultaneously by Naccache in [15]) and examine its applicability for general-purpose RFID tags in the supply chain. Using a combination of new and established space-saving methods, we present a full-fledged public key identification scheme, which is secure yet highly efficient. The 1024-bit scheme fits completely (including RAM) into 4682 gate equivalents and has a mean current consumption of 14.2 µA. The main novelty in our implementation is the replacement of the long pseudo-random sequence, originally stored on 260 bytes of EEPROM in [19], by a reversible stream cipher using less than 300 bits of RAM. We show how our scheme offers tag-to-reader and reader-to-tag authentication and how it can be fit into the existing RFID supply chain infrastructure.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems; B.7.7.1 [**Integrated Systems**]: Types and Design Styles—*Algorithms Implemented in Hardware*

General Terms

Algorithms, Security

Keywords

Public-Key Encryption, Rabin Encryption, Hardware Implementation, RFID Technology

1. INTRODUCTION

RFID tags will soon find their way into many items surrounding us every day. RFID tags can essentially be viewed as extremely cheap wireless computers bearing a unique identifier and coupled to a physical item such as a banknote or a medicine container. Using the wireless medium, any remote party can quickly and invisibly determine the set of tagged items carried by a person. The detrimental effect of this fact on the privacy and anonymity of consumers will be staggering unless explicit technical measures are taken to preserve them [18].

Recent results in hardware design of symmetric ciphers indicate that the often-sought 5-cent tag may have strong cryptographic abilities. For some applications symmetric crypto will be suitable for protecting tags and their users from abuse. However, it is impossible to neglect the system risks involved in storing a symmetric key on a tag. If the secret key stored on the tag is shared among many other tags (as well as the RFID reader), the actual cost of the tag grows from the 5-cent manufacturing cost to the multi-million-dollar *system cost* of having the symmetric key recovered from a tag and then used to compromise the entire RFID infrastructure consisting of many tags and readers. This is especially the case in the *supply chain* environment (EPC tags), where tags are created in very large quantities by myriad untrusted parties and their use is relatively uncontrolled. The case for public-key cryptography in tags is thus very strong – it can justifiably be argued that many RFID systems will *not* use cryptography unless it is of the public key variety. However, public key cryptography was considered out of reach for general purpose tags due to its high hardware cost.

This work shows how tag vendors can realize the advantages of public key cryptography in the supply chain (both to their users' and their businesses' advantage), while staying within the tight power and area budgets of low-cost tags. We show how a simple yet highly secure randomized public-key scheme can be used to identify tags to readers while storing only the public key on the tag. The cost of compromise in this case is minimized to the value of the tag's payload, which is generally much lower than the cost of compromising the entire supply-chain system.

In contrast to other contemporary low-resource public key schemes, our scheme can encrypt a relatively large amount of arbitrary data in every transaction, a property that makes it useful in wireless sensor nodes and other novel applications.

Our cryptographic scheme is called *WIPR,* short for *Weizmann-IAIK Public key for RFID.* After simulating both tag and reader algorithms in C++, we have implemented the tag logic on a 0.35 µm standard-cell process technology and run NanoSim power simulations on the extracted netlist. Based on our results, we believe WIPR offers a new lower bound on gate and power costs for viable public-key encryption.

The rest of this paper will be organized as follows. First, we will present the RFID authentication landscape and survey related work. We will then present our scheme in theory and in practice. We will conclude with discussion of some tag-specific concerns and list future work and open issues.

1.1 The Supply-Chain RFID Environment

The RFID environment in general consists of a *tag* T and a *reader* R, connected by a broadcast wireless medium. We wish to focus our discussion on the *identity-providing* scenario, found in supply chain environments. In this scenario the tag bears a payload ID which it wishes to provide to the reader in a privacy-preserving and properly authenticated manner. A slightly different case is the *identity-proving* scenario, in which the payload ID is a secret which is known to the reader beforehand and the tag only wishes to prove it knows ID as well.

Looking beyond the tag-reader association, there are several additional parties in any RFID supply-chain environment whose goals must also be considered. First and foremost, the *tag integrator*, or supply chain owner, should be considered. The tag integrator's security and privacy goals are the one we wish to realize, but some of these goals clash with the immediate objectives of the reader itself (for example, the tag integrator does not want a rogue reader to be able to create counterfeit tags). Another significant party is the *tag manufacturer*, whose role is to provide usable tags, preferably with minimum communication or commitment to the tag integrator. Understanding the various trust relationships between the various parties is one of the crucial elements for a successful introduction of security-enabled tags, perhaps even more than the hardware cost of the cryptographic elements themselves. One major point to consider with respect to this trust relationship is the associated logistical issue of *keying* – that is, providing the relevant cryptographic keys to the relevant parties, tracking the use of these keys and allowing revocation when the keys fall into undesirable use.

1.2 The RFID Adversarial Model

The adversary in the RFID environment can both passively observe reader and tag data exchange and actively participate in the protocol as either side. The adversary's objective in the *identity-providing* scenario is either to determine the value of ID without physically manipulating the tag, or to counterfeit a tag, that is – successfully impersonate a tag (bearing a certain desired ID or an arbitrary randomly-chosen one) in a protocol exchange. The adversary may be also interested in *tracking* a certain tag - that is, correlating a set of protocol exchanges with a specific tag, even it cannot explicitly determine this tag's ID by observing the protocol. Finally, the adversary may wish to impersonate a reader for the purpose or rewriting or disabling a tag.

As opposed to smart cards, which are reasonably hardened and designed to resist reverse engineering, RFID tags are designed for cost and have only minimal hardware countermeasures. Established results show that a tag essentially has no secrets – be they private payloads, secret encryption keys or any other data – once it falls into the hands of a moderately-funded adversary [16]. While the tag can be perfectly impersonated in this case (neglecting for the moment physical authentication measures), it is important to note the effect of such a compromise on the security of the system as a whole.

The discussion in this article does not cover side-channel attacks.

1.3 Related Work

Our scheme is based on the randomized variant of the well-known Rabin cryptosystem [17], first discussed in [8]. This scheme's applicability to low-resource smart cards was explored in [15, 19]. In the low-resource smart card world, as in the RFID world, RAM is expensive and its use needs to be minimized. However, rewritable EEPROM is cheap on smart cards and prohibitively expensive on RFID tags, due to the high power cost of the write operation.

Low resource implementations of secret-key cryptosystems, the most noteworthy of which is AES [3], have already been demonstrated on physical chips. Low-resource public key cryptosystems have yet to achieve this level of market readiness. The Rabin cryptosystem was first implemented in a low-resource setting by [7]. The low cryptographic security and high hardware cost offered by the authors' unmodified Rabin implementation (512-bit encryption in 16 700 gates) led them to declare that this cryptosystem is unsuitable for RFID tags. Other public-key RFID contenders can be found in works such as [5, 6]. These implementations generally require more gates than can fit in a low-cost 0.35 µm process tag or rely on uncommon features such as very large random sources. Of special note is the GPS scheme presented in [14]. While this scheme has a potentially low hardware cost, it is by design a zero-knowledge *identity-proving* scheme and cannot be used securely in an *identity-providing* setting.

2. THE PROTOCOL IN THEORY

2.1 A Brief Description of the Protocol

Recall that our motivation is to allow a tag to provide the value of its payload ID to an authorized reader while keeping this value secret from an adversary.

Our protocol is based on a variant of the well-known Rabin cryptosystem [17], as presented in [19]. Briefly put, the ciphertext M in such a cryptosystem is the square of the plaintext P, modulo a composite number $n = p \cdot q$ (p and q are prime). In this scheme every ciphertext has four corresponding plaintexts, requiring the addition of some inner structure to the plaintext. The plaintext P is typically generated from a shorter string (in our case ID) by padding it with random bits until it is the size of n.

The RAM efficient variant of the Rabin scheme does away with the modular reduction step, replacing it with an addition of a random multiple of the divisor. Thus, instead of $M = P^2 \pmod{n}$ the tag transmits $M = P^2 + r \cdot n$. This replacement was shown in [15, 19] and more recently in [20] to have no detrimental effect on security, as long as the size of r is properly chosen ([15] suggests a value of $|n| + 80$). It does, however, do away with the remaindering operation and drastically reduce the intermediate storage requirements of the algorithm.

To make use of this cryptographic building block to provide secure identification, we use a challenge-response construction, adding a reader-supplied random challenge to the plaintext P.

The construction makes use of the function BYTE_MIX, a simple byte-interleaving operation meant to prevent both tag and reader from dominating large consecutive segments of P.

Setup: Tag is provided with public key n and a signed unique identifier ID (see 2.3). Reader is provided with private key (p, q).

Boot: Reader generates a random bit string r_r, where $|r_r| = \alpha$. Tag generates two random bit strings $r_{t,1}$ and $r_{t,2}$, where $|r_{t,1}| = |n| - \alpha - |ID|$ and $|r_{t,2}| = |n| + \beta$. α and β are security parameters.

Challenge: Reader sends r_r to the tag.

Response: The tag generates a plaintext as follows:

$$P = BYTE\_MIX\,(r_r \# r_{t,1} \# ID)$$

, where # denotes concatenation, and then transmits the following message:

$$M = P^2 + r_{t,2} \cdot n$$

Verification: The reader uses the private key to decrypt M. There are 4 candidate decryptions (since every quadratic residue modulo $p \cdot q$ has 4 square roots), so the reader checks any of the 4 possible decryptions contains the value of the challenge r_r it sent to the reader. If such a plaintext is found, it outputs the value of ID. In all other cases, the authentication fails.

2.2 Security Benefits of the Proposed Scheme

The security benefits of the scheme are summarized in 1, followed by a more detailed exposition below. The security claims hold even if the adversary has complete knowledge of the entire system other than the private key.

Note that most of these claims stem from the use of a randomized encryption scheme based on public keys, as proven in [8, 17].

2.2.1 *Secrecy and privacy*

Because WIPR is based on the well-established Rabin encryption scheme, it provides powerful security and privacy advantages to its users. More formally, the following security properties hold:

- An adversary observing a protocol exchange cannot learn anything about the ID of an unknown tag. This property, which is shared with the underlying randomized Rabin cipher, allows the scheme to perform as a combined identification-authentication scheme, as described in the following subsection.

- An adversary cannot determine whether a tag it currently holds was a part of any past or future protocol exchange it has recorded, even if the adversary knows the secret payload ID of the tag. This property stems from the fact that the adversary does not know the values of $r_{t,1}$ and $r_{t,2}$ used in the recorded protocol exchanges. This property is very useful for articles such as banknotes, where an unscrupulous merchant may wish to track banknotes it has previously processed.

- The adversary cannot determine whether a certain public key n was used in the protocol exchange. This property shows its usefulness when there are multiple batches of tags sharing the same air space, each with a different public key. In such a case, merely discovering that a certain tag belongs to a certain batch (for example, medicine or high-denomination banknotes) poses a security risk. This *meta-privacy* property stems from the assumed intractability of the quadratic residuosity problem [8], and is enabled by the fact that the range of values for $x^2 + r \cdot n$ can be made the same for different values of n by assigning to each key n_i an appropriate maximum value for r_i such that $n_i \cdot r_i$ has the same range for all public keys.

2.2.2 *No private key stored on tag*

The only secret data stored on the tag is its payload ID. This means that if a single tag is reverse-engineered the system as a whole is not compromised (a break once-run once situation). More significantly, the tag integrator does not need to provide the tag manufacturer with its private key, drastically simplifying the keying logistics of the system.

2.2.3 *Encrypts arbitrary data*

While some low-resource public-key schemes are authentication-only, WIPR essentially encrypts an arbitrary payload with a length of up to nearly n bits. This allows WIPR to be used not only for authentication and identification but also for sensor measurements and other novel applications.

2.2.4 *No tag rewrites or coupons*

Several low-resource identification schemes ([18] and others) rely on a secret which is shared by the tag and the reader and is evolved using a hash function during the execution of the scheme . There are two implementation challenges with this approach: On the tag side, it requires memory to be rewritten every time the protocol is executed, a relatively slow and energy-consuming operation on EEPROM-based tags; on the reader side, it requires that the reader's back-end database be updated after every authentication, mandating a high-bandwidth connection between the reader's back-end databases and all reader devices, a very problematic issue for portable hand-held readers or distributed supply chain databases. The WIPR scheme is much simpler and straightforward – during the execution of the protocol the data on the tag is not rewritten, and the reader side algorithm makes no assumptions on the network connectivity of the reader. Other schemes make use of precomputed coupons stored on the tag [14], allowing only a limited amount of protocol executions (typically between 5 and 20) before the coupons run out. WIPR, in contrast, can be used indefinitely.

2.2.5 *Implicit reader authentication*

The fact that only a reader with access to the private key can decipher data coming from the tag serves as an implicit way of authenticating the reader. By making future traffic between the tag and reader depend on the data supplied by the tag, we can create a secure data channel from the reader to the tag. To explain this property, note that after the protocol execution has finished, the reader holds the plaintext values not only of ID but also of the random values $r_{t,1}$ and $r_{t,2}$. This fact allows the easy creation of a basic return channel from reader to tag using a one-time pad cipher using some function of $r_{t,1}$and $r_{t,2}$ as the cover code. An immediate application of this return channel is for secret key

Table 1: Summary of the security benefits of the scheme

Security Property	Operational Benefit
Secrecy and privacy	Tag's identity cannot be recovered (even partial information such as brand) by observing the protocol; tags cannot be tracked by correlating protocol exchanges
No private key stored on tag	System is not compromised if tag is reverse-engineered; less trust required between tag integrators and tag manufacturers
Encrypts arbitrary data	Can be used for authentication, identification, sensor measurements and other novel applications
No tag rewrites or coupons	Reader side algorithm is straightforward and efficient; tag is not limited to a fixed number of uses
Implicit reader authentication	Creates a secure backwards channel for commands from tag to reader; allows integration with secret-key encryption algorithms.

exchange – the reader can create a random secret key and send it to the tag XOR'ed with $r_{t,1}$, allowing further communications over the faster secret key channel. This form of cover coding can also be used to replace the cover coding used for the EPC Gen2 kill command [9][1].

2.3 Why WIPR Provides Authentication

If we consider the fact that the cryptography on the tag is provided by public key, the claim that WIPR can provide authentication seems counterintuitive. However, this ability can be easily achieved by embedding some internal cryptographic structure into ID. In essence, we use another public-key mechanism to encode secret information into ID, turning ID into the secret to be protected by the tag. Authentication is now based on the fact that the tag proves that it knows this secret identifier, while the cryptographic primitives prevent an adversary from recovering this secret by anything short of physical reverse-engineering of the tag.

To enable authentication the tag integrator (Walmart, for instance) generates a private/public signing key pair. It keeps the signing key secret and hands the verification key to the reader. Next, it signs every ID with its private signing key and sends a sequence of such signed IDs to the tag manufacturer. The tag's payload ID, as sent to the reader, is now the vector $(ID', S_K(ID'))$, where ID' is the domain specific payload (such as a serial number) and $S_K(ID')$ is a short public key signature of this payload. The reader, who must now possess both the public verification key and the private decryption key, can verify that a tag is valid but cannot forge a new tag.

Let us observe why this setup provides authentication:

- Since WIPR is based on the Rabin encryption scheme, which is considered very hard to crack, the only practical way to recover ID'or $SK(ID')$ would be to reverse-engineer the tag or compromise a reader.

- If an adversary attempts to compromise the system without breaking the underlying encryption algorithm or gaining knowledge of the private decryption key, his attempts are frustrated by the randomess used by the scheme. An adversary may try to impersonate a tag by recording challenge-response pairs used in successful transactions and waiting for a reader challenge to be repeated. This attack is foiled by the fact that **the reader** issues a fresh random challenge in every protocol execution, forcing the adversary to record an impractically large number of transactions before the same challenge is repeated. Another class of adversary may try to track a certain tag (known to him by a previously recorded challenge-response pair) by masquerading as a reader and monitoring the tag's response to his recorded challenge. Since **the tag** uses randomness in its response, this adversary (who, we must note again, does not posess the private decryption key) would have to engage a tag in many transactions with the same previously-encountered challenge before the tag uses the same response twice.

- A reverse-engineered tag can obviously be counterfeited, as discussed previously. However, in contrast to most secret-key schemes, where reverse-engineering a tag will allow an attacker to compromise the whole system, the only piece of secret information stored in the tag is $S_K(ID')$, which can only be used in the context of a single specific ID'. This construction will mean that even an adversary who has physically probed the tag to discover the full value of the vector ID cannot forge a new tag with a different ID, since the adversary is still without knowledge of the tag integrator's private signing key. This effectively creates a very desirable **break once-run once** situation for tags. Counterfeited merchandise will now have to bear a limited number of well-known IDs, which can be more easily tracked and revoked.

- A compromised reader cannot be used to forge new tags since it only has the verification key and not the signing key – just as in the case of a reverse-engineered tag, it can only be used to duplicate a previously-encountered tag. To further protect against tag duplication, the system can be set up such that the tag integrator's private decryption key is not stored directly on the reader. Instead, the reader is either equipped with a trusted hardware module such as a smart card or set up to communicate with an online decryption server. This trusted agent will receive the ciphertext, decrypt it and verify the signature. It will then reveal only ID' and not $S_K(ID')$ to the reader. This will allow identification and authentication while preventing even a rogue reader from duplicating tags. Storing the private keys on a smart card is also a good idea in general – using smart cards will also simplify the security

[1]Care should be taken when using this return channel for passing more predictable data, since it is easily malleable and does not resist man-in-the-middle attacks. Specifically, allowing the attacker to learn the exact value of $r_{t,2}$ completely breaks the system.

logistics on the reader side, since smart-cards are typically much easier to protect, transport and insure than cryptographically-enabled monolithic point-of-sale terminals.

One remarkable aspect of this authentication system is the low level of trust it requires among the various members of the supply chain – the private signing key never leaves the premises of the tag integrator, while neither the private private signing key nor the private decryption key are ever given to the tag manufacturer.

2.4 Reducing the Hardware Demands of the Protocol

We now show how the original scheme presented in [19] can be modified to be implementable on a very low resource RFID tag. First, let us assign some meaningful values to the security constants listed above. We choose $n = 1024$, $\alpha = 80$, $\beta = 80$ to achieve an 80-bit security level, comparable with 1024-bit RSA [12].

It is important to note at this point that our reference implementation assumes the existence of two low gate count cryptographic primitives: a true random-number generator (RNG) and a one-way function. There are established research results in both fields, as discussed further in 2.5.

The protocol is simple enough in terms of runtime – a single online multiplication ($P^2 + r \cdot n$) is all it takes. This multiplication step can readily be performed on a multiply-accumulate (MAC) register by convolution. Assuming a word size of 8 bits, a single multiply-accumulate register can carry out this multiplication in about 2^{16} steps using 25 bits of carry memory (enough to accumulate 512 8-bit multiply operations). The ciphertext can be transmitted byte by byte (LSB first) as soon as it is computed, minimizing the need for intermediate registers.

The main problem with this scheme in terms of implementability is its high memory cost. To properly compute a response the tag needs to store all 3 random strings r_r, $r_{t,1}$ and $r_{t,2}$, consuming approximately $2{\cdot}n$ bits of RAM at 6 gate equivalents per bit. In [19] this problem was solved by using a large amount of EEPROM, but EEPROM is not available on tags. Furthermore, even if sufficient EEPROM storage was available on the tag, the high power cost involved in writing a large amount of data in every protocol exchange will drastically reduce the usable range of the tag. Another high-resource element is the public key, consuming n bits of ROM at the price of approximately 1 gate equivalent per bit.

We first address the public key storage problem. To reduce the ROM cost by half, we use a well known method of generating a composite number with a predefined upper half (see for example [10]). By setting the upper half to a value easily represented in hardware (for example, the output of a counter or simply a fixed binary value), we can trim at least $n/2$ gate equivalents from the design. [2] Assuming 1024-bit keys, for any fixed value of the upper 512 bits there are more than 2^{500} possible composites, so this constraint does not weaken security. In addition, currently known factoring algorithms such as the number field sieve [11] do not gain any computational advantage against keys bearing this structure [3].

[2]Note that since the outputs of the key generation protocol presented in [10] are generated at random, it is quite feasible to run the protocol multiple times and hope to obtain a lower half whose most significant bits match the fixed bit pattern used in the upper half, thus saving another few gates of ROM. Considering that a single non-optimized run of the protocol took 2 seconds on a desktop computer, we can leave the key generation program running for a week and gain an expected savings of 18 additional bits.

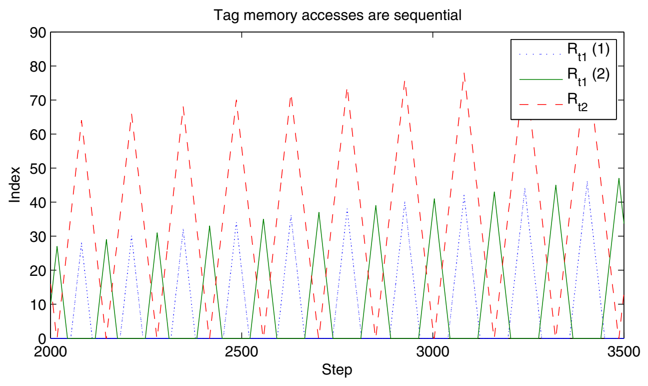

Figure 1: Tag memory accesses are sequential and follow a zig-zag pattern.

A more elaborate construction is used to reduce the RAM costs of the scheme. To do so, we make use of the fact that the three bit strings are completely random and that we only require sequential access to them, as indicated in 1.

The combination of these two facts allows us to replace the long random strings generated by the tag with pseudo-random outputs from a *reversible stream cipher*. Instead of storing the entire random string, we store short seed values (one for $r_{t,2}$ and two for each end of $r_{t,1}$), and use the stream cipher operation to evolve them in time. Due to the sequential nature of accesses to the random strings, only a single "roll left" or "roll right" operation is required for each convolution step.

There are several design alternatives for implementing reversible stream ciphers, for example linear-feedback shift registers. We chose to implement the stream cipher using a Feistel structure [13], a well-known cryptographic construct used in symmetric ciphers such as DES and TEA. The security of a Feistel structure comes from a suitably strong pseudo-random function, which is not necessarily invertible or even domain preserving. As indicated in 2 below, we can use a Feistel structure and an appropriate one-way function to evolve a random seed into an arbitrarily-long pseudo-random sequence with quick and efficient sequential access, using the following algorithm:

```
Roll Right:
    left_in  <= right_out;
    right_in <= left_out xor oneway(right_out);
Roll Left:
    right_in <= left_out;
    left_in  <= right_out xor oneway(left_out);
```

[3]A notable exception is the choice of "1000..00" as the upper half of sequence. This choice should be avoided, since it leads to composites of the form $r^e \pm s$, with small r and s. Composites of this form can be factored by the special number field sieve.

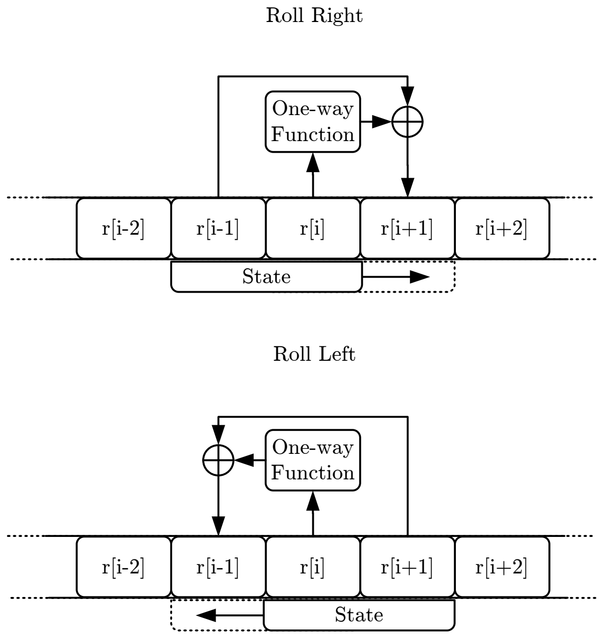

Figure 2: Creating a reversible stream cipher using a Feistel structure and an arbitrary one-way function.

Table 2: Summary of security parameters used in the scheme.

Parameter	Definition	Reference Value
n	Public-key length	1024
α	Reader challenge length	80
β	Random multiple length (when added to n)	80
δ	Feistel state size	96

The advantages of this design choice are that it creates many new pseudo-random bits per clock cycle and that there are many inventive ways to build one-way functions using constrained hardware, as discussed in the following subsection. We introduce a security parameter δ that indicates the amount of state held by the Feistel and assume the one-way function will have a domain of $\frac{\delta}{2}$. For our implementation we chose $\delta = 96$ bits (slightly longer than the 80-bit security level used by the rest of the tag, since this seed is used to generate long sequences which should not overlap). This adds an implementation cost of $96 \cdot 3 = 288$ bits of RAM and $96 \cdot 2 = 192$ random bits, as well as the one-way function and the associated multiplexing and routing logic. The rolling step can be performed several times in a row to increase the security of the cipher. Note that we do not use the entire 96 bits of state as the output of the stream cipher, instead selecting only 8 bits to match the word size of our multiply-accumulate register.

We could find no way to reduce the storage requirements for r_r, which are 80 flip-flops in our case, other than reducing the value of the security parameter α.

2.5 Choosing an Appropriate One-Way Function and Random Number Generator

Our reference implementation assumes the existence of two low gate count cryptographic primitives: a true random-number generator (TRNG) and a one-way function (OWF). Our design uses a $\frac{\delta}{2}$-bit one-way function as part of the Feistel structure. Our reference implementation, in which $\frac{\delta}{2} = 48$, uses a somewhat insecure but computationally representative boolean function, for which we allocated a total of 690 gate equivalents out of the total 4682 used for the scheme. This gate cost is consistent with the costs of similar computational blocks in low-gate-count ciphers such as PRESENT [2].

There are many inventive ways of implementing one-way functions on constrained hardware, using ideas such as substitution-permutation networks, physically unique functions, uninitialized memory and other techniques. The Feistel construction used in the reference implementation is generic enough to accept any sufficiently secure one-way function, allowing new state of the art OWF constructions to be easily integrated into our scheme. Due to the fact that the one-way function is only used by WIPR as the source of a random sequence, different WIPR tags can use different one-way functions while remaining compatible in all other aspects. Indeed, it is even possible for a tag to use a different randomly-generated one-way function for each invocation of the protocol.

The most important requirement for the TRNG is that it should be hard for an adversary to dictate the output of the generator without physically manipulating the tag. For example, a simple counter which starts incrementing its value at a constant rate when the tag is powered up is a bad candidate for such a generator, since an adversary may be able to control the supply of wireless power to the tag without physical contact. We note again that it has been shown that once a sufficiently advanced adversary has physical control of a tag, the tag can hold no secrets [16], so the security of the TRNG under such conditions is a non-issue.

3. HARDWARE IMPLEMENTATION OF THE WIPR PUBLIC-KEY SCHEME

3.1 Requirements for Hardware Design of Passive RFID Tags

Implementing cryptographic hardware for passive RFID tags is challenging due to the fierce constraints. The main objectives for the designed hardware are to minimize power consumption and to reduce the necessary chip area. The reason for the low-power constraint is the operating range. The power that is provided by the RFID reader over the air interface is reduced linearly with the operating distance for UHF tags. In order to allow cryptographic operations in the whole range of a "normal" tag, which is in the UHF frequency range up to seven meters, the power budget of approximately 20 µW must not exceeded. The second big issue is the chip area. The costs of an RFID tag linearly increase with the die size. Thus, the chip area of a cryptographic hardware module significantly influences the price of a tag. When an RFID tag today has a total chip area of 20000 gate equivalents the size of additional hardware is obviously very limited. However, the achieved gain of having a cryptographically enhanced RFID tag must also be considered.

3.2 Architecture of WIPR Scheme

During the course of our work we developed an architecture for the WIPR scheme. The datapath of the implemented module is depicted in 3. The main element of the design is a multiply-accumulate unit. Thereby, a 25-bit accumulator register is used, which provides the possibilities to reset the internal value and to shift eight positions to the right. This option is used when a byte is sent to the reader and when the next higher-order byte should be processed. A 25-bit adder is implemented without special constraints. It adds the newly processed multiplication result of the 8x8-bit multiplier with the old value in the accumulator register. The word size of 25 bits was chosen for the accumulator register and the adder to not lose any necessary bits because of too high values in the register.

The two inputs for the multiplier are selected by two four-to-one multiplexers. The remaining parts of the circuit are three Feistel states, memory for storing the challenge, a ROM for the constant n, and inputs for further constants, which are the ID of the tag and a control number. The so-called Feistel states are used to store the random values $r_{t,2}$ and $r_{t,1}$. The implementation using this Feistel structure allows getting the random values bytewise as needed for the multiplication by the two simple operations "roll left" and "roll right". In addition to the module R_t2, we need two further Feistel states. The reason for duplicating the Feistel state $r_{t,1}$ in the modules R_t1a and R_t1b is that different bytes or r_t are required in the same multiplication cycle. The datapath module R_r contains the challenge from the reader. It stores 16 times 8 bits of data, which are written to the module during reception of the challenge. The 128x8-bit ROM stores the modulus n. It is built from an unstructured mass of standard cells, which are generated during synthesis. The further inputs to the multiplexers are the tag identifier ID and a checksum value labeled as CRC.

Calculation of a tag-identification response works as follows. The tag receives the challenge r_r and stores it in the R_r module. Beginning at the least significant byte, the message $M = P^2 + r_{t,2} \cdot n$ is computed using multiplication by convolution. The variable P includes values from r_r, r_t, ID, and the checksum. When the current byte is ready, it is sent to the reader. Hence, the result does not have to be stored in the tag. Furthermore, the accumulation register is shifted eight positions to the right. This allows calculating the next higher byte in the eight least significant positions of the accumulator. The procedure is repeated until the most significant byte is transmitted.

3.3 Results of WIPR Implementation

In order to have a fair comparison to other crypto schemes for passive RFID tags, we implemented our design on a 0.35 µm standard-cell process technology from Austriamicrosystems. The circuit has been synthesized and the extracted netlist after place and route has been simulated using the power simulation tool NanoSim from Synopsys. At a supply voltage of 1.5 V and a clock frequency of 100 kHz the resulting mean current consumption is 14.2 µA. This is below the available power budget in passive RFID tags.

The synthesis results for the modules in the datapath can be seen in 3. About two thirds of the total chip area of 4682 GEs are consumed by the Feistel states and Feistel logic. Also of importance is the memory for the challenge r_r. The multiply-accumulate unit (multiplier, adder, accumulator) requires in sum only 722 GEs.

Table 3: Components and synthesis results for WIPR datapath.

	Chip area [GEs]
R_t2 Feistel state	547
R_t1a Feistel state	547
R_t1b Feistel state	547
Feistel logic + one-way function	1050
R_r memory	995
Constant n	206
Multiplexers	68
Multiplier	424
Adder	100
Accumulator	198
Total chip area	**4682 GEs**
Total power consumption	**14.2 µA**

The calculation of the whole identification procedure requires 66 048 clock cycles. Due to the fact that each result byte is transmitted to the reader, the longest computation time is required for the byte computed exactly in the middle of the whole value. This byte needs 512 cycles because two times 256 partial products have to be summed up.

3.4 Comparison with Other Hardware Implementations

A comparison of different hardware implementations with our design can been seen in Table 4. It should be noted that all presented designs have been implemented on the same target technology under equal simulation conditions. It can be seen that the presented solution in this work requires only a fourth of the hardware resources as the ECC-192 implementation of Fürbass [6] while also reducing the mean current consumption. An analysis of the best ECC implementations, in relation to our design, shows an improvement of factor two in terms of chip area.

Compared to symmetric key cryptography our design requires less chip area than the commonly used hash functions SHA-256 and SHA-1 [4] but about 1500 GEs more than AES [3]. The required number of clock cycles is commonly no big issue for passive RFID tags due to is slow data rates. Nevertheless, an appropriate integration into currently used standards is necessary.

4. DISCUSSION

4.1 Compatibility with the EPC C1G2 Air Interface

The EPC Class 1 Generation 2 (C1G2) specification is an air interface commonly used in retail applications which stand to benefit the most from the presented work. According to the C1G2 specification, tags respond to a request from the reader by sending a single packet, sized about 128 bits, containing the tag's entire payload [9]. As discussed in the previous section, our protocol requires about 600 milliseconds at 100KHz to create a 2048-bit response (twice the

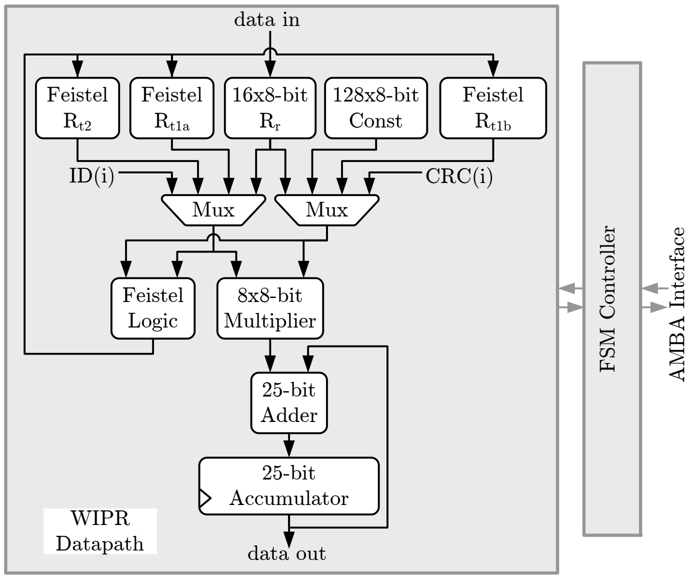

Figure 3: Datapath architecture of WIPR.

Table 4: Comparison of different cryptographic hardware implementations.

Algorithm	Security [bits]	I_{mean} [μA@100kHz]	Chip area [GE]	Clock [cycles]
SHA-256 [4]	128	5.86	10 868	1128
SHA-1 [4]	80	3.93	8120	1274
AES-128 [3]	128	3.0	3400	1032
ECC-192 [6]	96	15.7	23 656	502 000
This work	**80**	**14.2**	**4682**	**66 048**

size of the public key). Considering the fact that the EPC air interface reaches tag-to-reader data rates of 50Kbps under reasonable conditions, this seems a wasteful use of the wireless medium. This problem can be addressed by making a relatively simple adaptation to the EPC C1G2 singulation protocol.

Our idea in general is similar to the one described in [3], in which several tags send an interleaved response to the reader simultaneously. We make use of the additional fact that C1G2 are especially suited to work in a "slotted ALOHA" fashion, already having selected slots as part of the ordained response to the "Query" command.

To support this function, we propose a new reader-to-tag command called "AckRep". Similar to the "Ack" command, this command receives the tag's session-specific random handle. As a response to this command, the tag sends as many ciphertext bytes as it has prepared, LSB first. In contrast to the standard "Ack" command, the tag does not go idle after this command. Instead, it immediately starts preparing more ciphertext bytes to send. Another required command is "Challenge", in which the reader presents r_r to a selected tag. Note that all WIPR tags should respond to the standard C1G2 query command with an identical value (up to meta data such as version number), since using a unique PC+EPC value for each tag will obviously invalidate the privacy benefits of our scheme.

Figure 4 shows a modified EPC inventory process. To inventory WIPR tags, a reader should first perform the standard C1G2 population query operation, as described in [9]. After the reader obtains the session handles of all present tags, it should issue "Challenge" commands to all WIPR tags, then repeatedly call "AckRep" in a round-robin fashion until all present tags have sent their entire encrypted payloads. The exact amount of bytes which should be buffered by the tag and transmitted in a single protocol round-trip is an implementation decision – storing more bytes allows more efficient use of the air medium but comes with an added gate cost on the tag. The order in which tags are queried with this command can be chosen by the reader based on the order in which the tags replied to the "Query" command.

To allow current-generation readers to immediately support WIPR without updating their firmware, the challenge and response commands can also be implemented as memory-mapped I/O requests to predefined addresses on the tag (essentially a simplified version of the approach suggested in [1]). Thus, performing a memory write operation to a predefined address will be understood by the tag as the "Challenge" command, while a memory read operation from another address will be understood by the tag as the "AckRep" command.

This method allows multiple tags to interleave their responses, allowing more efficient use of the air medium. It also prevents the tags from having to buffer their output data indefinitely in RAM registers.

One drawback to this solution is that it forces the reader to announce with high power that it is communicating with

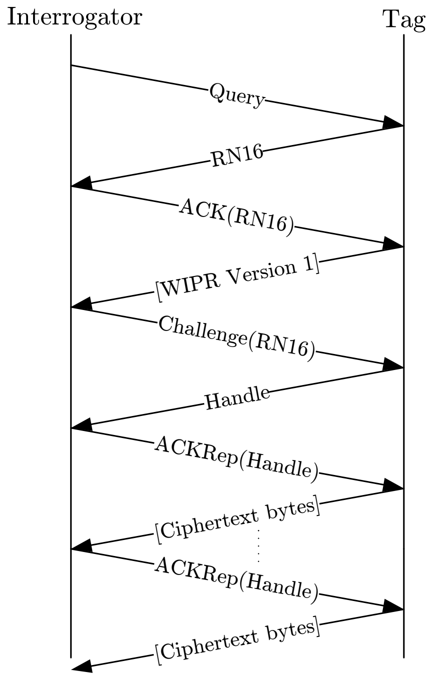

Figure 4: Introducing WIPR tags into the EPC C1G2 air interface

a WIPR tag, potentially teaching an adversary that the currently inventoried item has more value than one whose vendor chose not to implement our scheme. The authors hope that use of WIPR will be widespread enough to make this risk trivial.

4.2 Relation to the SQUASH Hashing Scheme

There are several architectural similarities between WIPR and the SQUASH hashing scheme presented in [20]: both rely on the security of modular squaring, both make use of a multiply-accumulate register and both use long pseudo-random sequences generated from a short seed. However, there is a significant difference in the purpose of the two schemes. Referring to the notation of 1.1, SQUASH, being a **hashing scheme**, offers an *identity-proving* mechanism (for tags whose ID is known beforehand to the reader), while WIPR, being an **encryption scheme**, offers an *identity-providing* mechanism for previously unknown tags. It is important to note that many of the WIPR's security claims, as listed in 1, do not hold for SQUASH. SQUASH is thus suitable for applications such as vehicle entry systems, in which the set of expected tags is small and predetermined, while WIPR is suitable for applications such as inventory management, in which the tag population is very large and potentially uncontrolled by the reader.

At the present time SQUASH has no published implementation on an ASIC-like architecture, making it difficult to perform a direct comparison of the gate cost and performance of the two schemes. It can be still noted that due to their similar architectures, SQUASH can be implemented with near trivial gate cost on a chip which already includes the WIPR hardware components.

4.3 Open Issues

The article did not compare the relative merits of different designs of one-way functions. We did not discuss low-resource methods of obtaining the prescribed amount of random bits. The parameter sizes used in the scheme need to be fine-tuned, based on the relative strengths of attacks against the scheme's various subcomponents.

It will also be interesting to find a standalone key agreement protocol suitable for tags. Using secret key encryption and a good key agreement protocol will achieve many of the security goals presented here while still relying only on secret-key cryptography.

4.4 Conclusion

We presented a public key identification scheme which is highly secure yet lightweight enough to fit on an RFID tag or a wireless sensor node. We also showed how to elegantly introduce this scheme into the current EPC air interface specification. The introduction of public key-based methods to the supply chain will offer significant security and privacy advantages both to users and to businesses.

5. ACKNOWLEDGEMENTS

We wish to thank Adi Shamir for his substantial contribution to this work.

The research described in this paper has also been supported, in part, by the European Commission through the IST Programme under contract FP6-2005-IST-034921 C@R. 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.

6. REFERENCES

[1] D. V. Bailey and A. Juels. Shoehorning security into the EPC tag standard. In R. D. Prisco and M. Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, LNCS*, volume 4116, pages 303–320. Springer-Verlag GmbH, September 2006. `http://snurl.com/wiprBJ`.

[2] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007: 9th International Workshop, LNCS*, volume 4727, pages 450–466. Springer-Verlag GmbH, 2007. `http://snurl.com/wiprBKLPPRSV`.

[3] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In J.-J. Q. Marc Joye, editor, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop, LNCS*, volume 3156, pages 357–370. Springer-Verlag GmbH, July 2004. `http://snurl.com/wiprDFW`.

[4] M. Feldhofer and C. Rechberger. A Case Against Currently Used Hash Functions in RFID Protocols. In *First International OTM Workshop on Information Security (IS'06), Montpellier, France, Oct 30 - Nov 1, 2006. Proceedings, Part I, LNCS*, volume 4277, pages 372–381, Graz, Austria, October 2006. `http://snurl.com/wiprFR`.

[5] M. Finiasz and S. Vaudenay. When stream cipher analysis meets public-key cryptography. In E. Biham and A. M.Youssef, editors, *Selected Areas in Cryptography - 13th International Workshop, SAC 2006, LNCS*, volume 4356, pages 266–284. Springer-Verlag GmbH, September 2007. `http://snurl.com/wiprFV`.

[6] J. Furbass, F.; Wolkerstorfer. ECC Processor with Low Die Size for RFID Applications. *IEEE International Symposium on Circuits and Systems, 2007*, pages 1835–1838, 27-30 May 2007. `http://snurl.com/wiprFW`.

[7] G. Gaubatz, J.-P. Kaps, E. Ozturk, and B. Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 146–150, March 2005. `http://snurl.com/wiprGKOS`.

[8] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of Computing*, pages 365–377, New York, NY, USA, 1982. ACM. `http://snurl.com/wiprGM`.

[9] E. Inc. EPC radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz – 960 MHz, version 1.0.9. Online, September 2005. `http://snurl.com/wiprEPC`.

[10] A. M. Johnston. Digitally watermarking RSA moduli. Cryptology ePrint Archive, Report 2001/013. `http://snurl.com/wiprJ`.

[11] A. K. Lenstra, J. H. W. Lenstra, M. S. Manasse, and J. M. Pollard. The number field sieve. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 564–572, New York, NY, USA, 1990. ACM. `http://snurl.com/wiprLLMP`.

[12] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001. `http://snurl.com/wiprLV`.

[13] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. `http://snurl.com/wiprLR`.

[14] M. McLoone and M. Robshaw. Public key cryptography and RFID tags. In M. Abe, editor, *Topics in Cryptology – The Cryptographers' Track at the RSA Conference 2007, LNCS*, volume 4337, pages 372–384. Springer-Verlag GmbH, February 2007. `http://snurl.com/wiprMcLR`.

[15] D. Naccache. Method, sender apparatus and receiver apparatus for modulo operation. European patent application no. 91402958.2, Filed 10/27/1992. `http://snurl.com/wiprN`.

[16] K. Nohl and H. Plötz. MIFARE – little security, despite obscurity. Technical report, 24th Chaos Communication Congress, December 2007. `http://snurl.com/wiprNP`.

[17] M. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, MIT, Cambridge, MA, USA, 1979. `http://snurl.com/wiprR`.

[18] S. E. Sarma, S. A. Weis, and D. W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *First International Conference on Security in Pervasive Computing*, 2003. `http://snurl.com/wiprSWE`.

[19] A. Shamir. Memory efficient variants of public-key schemes for smart card applications. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT '94, LNCS*, volume 950, page 445. Springer-Verlag GmbH, January 1995. `http://snurl.com/wiprS`.

[20] A. Shamir. SQUASH – a new MAC with provable security properties for highly constrained devices such as RFID tags. In A. Biryukov, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lecture Notes in Computer Science*. Springer-Verlag GmbH, To Appear.