PIONEER—a Prototype for the Internet of Things based on an Extendable EPC Gen2 RFID Tag

Hannes Gross[1], Erich Wenger[1], Honorio Martín[2], and Michael Hutter[1]

[1] Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
{Hannes.Gross,Erich.Wenger,Michael.Hutter}@iaik.tugraz.at
[2] Department of Electronic Technology,
Carlos III University of Madrid, Leganés 28911, Spain
hmartin@ing.uc3m.es

Abstract. The Internet of Things (IoT) envisions an autonomous network between everyday objects to create real-life services. This enables new applications that necessarily require a high level of security and privacy. In this paper, we present PIONEER—a Prototype for the Internet of Things based on an Extendable EPC Gen2 RFID tag. It is the first prototype that integrates the Internet Protocol Security suite (IPsec) into the new EPC Gen2 Version 2 standard. Furthermore, it integrates all mandatory cryptographic primitives to support IPsec on an RFID tag, i.e., AES-128 for encryption/decryption, 192-bit Elliptic Curve Diffie Hellman (ECDH) for key agreement, and a True Random Number Generator (TRNG). To keep the flexibility high, we further integrated an 8-bit microcontroller that implements the new security features of the EPC Gen2 standard in C code. The entire design was synthesized for a 130 nm CMOS process technology. It requires about 52 kGEs including all necessary components to establish a secure IPsec tunnel between the RFID tag and a client on the Internet. The prototype is fully compliant with already existing Internet and RFID standards and allows first cost estimations for a practical realization of high-security IoT applications.

Keywords: Radio Frequency Identification, UHF Tags, Security, IPsec.

1 Introduction

The term “Internet of Things” goes back to the early days of the Auto-ID Labs—formerly Auto-ID Center—, and its co-founder Kevin Ashton [4] who was the first to use this term in context of logistics. The Auto-ID Center was founded at the Massachusetts Institute of Technology to work on the first version of the Electronic Product Code (EPC) standard. Therefore, the vision of the Internet of Things (IoT) is historically strongly related to the EPC standard. Later on, Ashton defined the idea behind the IoT more generally as to be not just ”a bar code on steroids”, but to be an autonomous network that allows computers to gather information about things in the real world. In 2000, Sarma, Brock, and Ashton [48] envisioned their ideas for a “single open architecture system for networking physical objects” to be used instead of “multiple smaller

scale alternatives". The situation today shows that the latter approach is dominating the IoT world at the moment. There are many different realizations of unrelated networks of things, e.g., used in logistics, in supermarkets, for home automation, for smart parking, et cetera. One reason for this situation might be that restricted networks are far easier to handle for companies than open networks—not only in terms of security considerations. This is comparable to the beginning of the Internet, when there were Internet service providers offering services restricted to their own network. However, the services that were open for the whole Internet were far more successful in the long term (e.g., email, messaging, or file sharing service).

EPC Gen2 Version 1 tags are not yet prepared to provide reliable, confidential, and authentic services in an open accessible network, because they lack the required cryptographic functionality. With the introduction of the EPC Gen2 Version 2 standard [16] released at the end of 2013, the first attempt towards standardized security on EPC tags has been performed. This represents an important step towards the realization of an open and secure Internet of Things.

Our contribution. In this paper, we first evaluate the integration of the Internet Protocol Security suite (IPsec) into the new EPC Gen2 Version 2 standard. As an outcome, we present a complete prototype system that supports all mandatory features to establish a secure tunnel between RFID tags and clients on the Internet. To keep the resource requirements low, we decided to implement the minimal set of necessary—but standardized—cryptographic primitives, i.e., AES-128, ECDH over the NIST P-192 curve, and a TRNG. Moreover, we integrated a very flexible 8-bit microcontroller into our design to be able to implement different Internet of Things applications. We call our prototype PIONEER because it is based on an extendable EPC Gen2 RFID-tag platform that features a reconfigurable Spartan-3 FPGA.

Within this work, we aim for presenting a fully working prototype including all necessary functionalities that are required for a secure Internet of Things. We base our investigations on standardized algorithms and protocols in order to facilitate the evaluation of existing system integration. Note that we do not claim for an efficient or yet practical implementation but rather provide first results of a prototyping system to estimate the costs for high-security IoT applications.

Our results interestingly show the feasibility of a running IPsec stack on an EPC Gen2 tag. With our prototype, we are successfully able to demonstrate the establishment of secure end-to-end connections between tags and clients in the Internet *without* needing to trust readers or nodes between the tag and the client. Due to the high area requirements of around 52 kGEs, we consider implementation of mid-to-high cost IoT applications that require particular security features.

Outline. The remainder of the work is organized as follows. At first, the scientific work related to this paper is discussed in Section 2. The cryptographic algorithms and the security features used in the EPC Gen2 Version 2 and the IPsec protocols are then considered in Section 3 and 4 respectively. Afterwards, Section 5 discusses the integration of IPsec into the EPC Gen2 standard as a cryptographic suite in detail. As a practical contribution we present our prototype design of an IPsec enabled EPC Gen2 tag in Section 6, and discuss the implementation results in Section 7. Finally, conclusions are drawn in Section 8.

2 Related Work

The EPC standard has opened a broad field of research topics covering different kinds of security aspects, privacy-preserving protocols [20, 51], hardware designs for tags [17, 31, 47], the practical realization of the IoT [8, 13, 21], and even the social and political impact of it [34]. Other works focus on increasing the tamper resistance of EPC Gen2 tags to prevent cloning or skimming, like the papers from Noman et al. [38] and Lehtonen et al. [32]. A lot of research has also been done on security and light-weight authentication schemes suitable for low-cost EPC Gen2 tags [39, 40, 56]. Most of these published light-weight protocols are considered insecure or are already broken [5, 46]. The integration of public key cryptography is also part of ongoing research for passive RFID tags. Besides the typical constraints, e.g., a very restrictive power budgets, limited computational power, and a small chip size of an RFID tag design, it was stated by Arbit et al. [3] that reader devices limit the suitability of such protocols trough inefficient implementation of the EPC standard functionality.

Many of the security aspects mentioned above are also covered in the new EPC Gen2 Version 2 standard, introducing so-called cryptographic suites. Engels et al. [15] were the first to evaluate different security aspects of the new EPC Gen2 Version 2 standard in 2013—even before the standard was ratified. Their work focuses on eavesdropping, snooping, relay, and man-in-the-middle attacks, with special regard to timing constraints. Two different cryptographic suites for the Advanced Encryption Standard were considered, and it was stated that the new EPC standard introduces new vulnerabilities against different attacks. The main weakness of the new standard comes with the enhanced complexity of the security commands which allow longer tag-response delays (cf. *In-process tag reply* [16], on page 38). Hinz et al. [22] showed their implementation of the RAMON cryptographic suite. This suite uses the Rabin-Montgomery (RAMON) public-key scheme to authenticate tags against the readers. The authentication mechanism consists of a challenge-response step, with optional verification of the Tag ID by using a public-key infrastructure. It is also stated that the scheme allows only tag authentication. Hence, further security features like reader authentication or secured communication, require additional cryptographic primitives.

Yao-Chung et al. [8] showed in 2008 how different RFID networks can be connected using IPv6 technology by introducing so-called RFIPv6 gateways as network bridges. In this scenario, the tag information is collected by the readers, and stored on servers connected via the RFIPv6 gateways to other RFID networks. In 2010, Dominikus et al. [12] suggested using Mobile IPv6 functionality to connect low-cost EPC tags with the Internet. Therefore, the tag needs to carry an IPv6 home address that uniquely identifies the tag on the Internet. When the tag is outside its home network, the associated "home agent" works as a proxy, and receives packets addressed to the tag's home address. The packet is then forwarded to the tag's current IPv6 care-of-address, which is a temporary leased address identifying the tag in its currently located subnet. Furthermore, it is suggested to use IPsec to establish a secure end-to-end communication between a tag and a "corresponding node" on the Internet. A more detailed description of the communication between the tag, the reader, and the corresponding node to establish an IPsec channel was given in [13].

Further related research is done by the IETF's 6LoWPAN working group on making IPv6 technology more suitable to use with Wireless Personal Area Networks (WPAN). WPAN nodes are part of the IoT and are energy constrained devices that often use batteries and active communication to communicate over higher distances than, e.g., passive UHF RFID tags. In order to save energy, so-called header compression mechanisms are used to reduce the communication overhead costs. Raza et al. [43] presented one approach to integrate these mechanisms into the IPsec protocol's packets. However, the 6LoWPAN extensions are not yet widely used and are therefore out of scope for this paper.

3 EPC Gen2 Version 2 and its Security Concepts

The Electronic Product Code (EPC) Generation 2 Version 2.0.0 standard [16] was ratified in November 2013, and offers the first comprehensive approach to integrate standardized security into the UHF RFID tag standard. The majority of changes target the User memory management, and the tag's security features. The standard itself defines no cryptographic operations or primitives, but it defines the command frames that encapsulate the payloads specified in so-called cryptographic suites. Cryptographic suites can either be assigned by the ISO/IEC 29167 standard, GS1, or by the tag manufacturer itself. The EPC Gen2 security commands include an 8-bit cryptographic suite identifier (CSI) that selects a cryptographic suite for the interpretation of the included payload. A cryptographic suite is usually defined over a particular cryptographic primitive (e.g., the Advanced Encryption Standard [11], PRESENT-80 [7], et cetera.), and contains the description of different security services.

In Figure 1 the different phases typically transited during a communication procedure are illustrated. On the top, the tag states defined by the EPC Gen2 standard are shown that are linked to the communication phases defined in the middle of the figure. The related security commands are listed below. As long as the tag is not killed, the tag enters the "Select" phase right after a reader field is detected. During this phase, tags are selected by a reader—according to criteria defined by the Select commands (*Select* or *Challenge*)—that will participate in the next inventory round. The *Challenge* command allows to trigger the computation of a cryptographic response, in order to determine the authenticity of a tag. Furthermore, it is ensured that only tags will participate in the subsequent inventory round, that possesses the ability to handle security commands defined in the selected cryptographic suite. A tag that supports the *Challenge* command with the given parameters computes a response, stores it into the so-called *ResponseBuffer*, and informs the reader device about the presence of a response by setting one bit (*C* flag) in the Protocol Control (PC) word. The response is read by the reader device during the "Access" phase.

The "Inventory" phase has not been extended by any commands, but additional information is transmitted at the end of this phase, like the *Challenge* command's response or the Extended PC word (XPC). If the XPC is supported by the tag, it informs the reader about special abilities of the tag. In the subsequent "Access" phase, a wide range of optional security commands are supported. The *Authenticate* command is used to identify the reader or the tag to the other communication partner by means of a

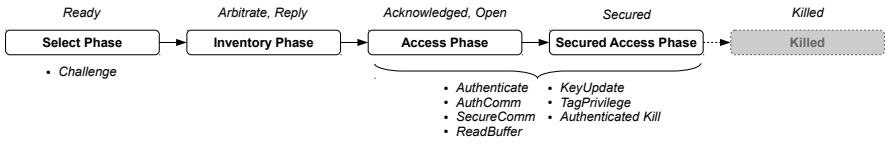

Fig. 1. Tag phases with according EPC Gen2 tag states (on top) and associated security commands

cryptographic function. It can also be applied as a secure substitution of the *Access* command, that is used in previous EPC Gen2 standards to transfer from the "Access" phase into the "Secured Access" phase, and to grant associated privileges. With the *AuthComm* command, a command is encapsulated by an integrity-protected command frame. Contrary to the *SecureComm* command (which offers confidentiality and usually also integrity) the encapsulated command is transmitted as plain text, and only the integrity of the message is ensured by, e.g., using a message authentication code (MAC). Since cryptographic operations are usually more time consuming than the strict timing of the standard allows, so-called *In-Process* tag replies have been introduced. This reply type sets the timeout value for a consecutive tag reply up to 20 milliseconds, and can be sent multiple times to give the tag enough time to finish its computations.

The rest of the security commands (*KeyUpdate*, *TagPrivilege*, *Authenticated Kill*) are of less relevance for this paper, and for the sake of brevity not discussed in the remaining work. In the next subsection, the currently available information to the ISO/IEC 29167 cryptographic suites is briefly discussed.

3.1 Cryptographic Suites of the ISO/IEC 29167 Standard

While the EPC Gen2 Version 2 protocol is already released, the standardization process of the cryptographic suites is still in progress. At the moment, the only information publicly available at the ISO web page [26] are the names of the cryptographic suites to be published in future (see Table 1).

However, in the work of Engels et al. [15], two Advanced Encryption Standard (AES [11]) based cryptographic suites are considered, using the Cipher Block Chaining mode (AES-CBC), and the Output Feedback mode (AES-OFB [14]). Furthermore, two mutual authentication schemes relying on these cryptographic suites are introduced, and a security evaluation is performed considering different passive and active attacks. Other listed symmetric-key suites use the lightweight block cipher PRESENT-80 [7], or the stream cipher Grain-128A [1] that also supports message authentication codes (MAC). The XOR suite most likely uses an ultra lightweight approach, based on one or more XOR operations as primary cryptographic primitive (cf. Vernam cipher or one-time pad in Menezes et al. [35], on page 21).

From the public-key perspective, there are currently two Elliptic Curve Cryptography (ECC) based suites listed mentioning Diffie-Hellman key agreement (ECDH [2]), and the Elliptic Curve Digital Signature Algorithm (ECDSA [28]). Another suite uses the cryptoGPS [19] algorithm either based on RSA [44] or ECC. This is a resource-

Table 1. Currently listed cryptographic suites for EPC Gen2 Version 2 (see ISO [26] for the most recent list of cryptographic suites)

ISO/IEC 29167	Cryptographic Suite	Description
... **Part 10**	AES-128	Block cipher (128-bit) with 128 bit key length
... **Part 11**	PRESENT-80	Block cipher (64-bit) with 80 bit key length
... **Part 12**	ECC-DH	ECC based Diffie-Hellman key agreement
... **Part 13**	Grain-128A	Stream cipher, 128-bit key, optional MAC
... **Part 14**	AES-OFB	Output feedback mode (OFB) for AES
... **Part 15**	XOR	Vernam cipher or one-time pad
... **Part 16**	ECDSA-ECDH	ECC-DH and digital signature algorithms
... **Part 17**	cryptoGPS	Low-cost public-key cryptography (coupons)
... **Part 19**	RAMON	Rabin-Montgomery based cryptography

aware alternative to the ECDH suite. For the key agreement, the tags do not need to calculate expensive operations like the ECC point multiplication, because they can be done in advance, and only the result—in form of coupons—is stored on the tags. Hinz, Finkenzeller, and Sysen [22] published a paper that gives some insight into the RAMON cryptographic suite (Rabin-Montgomery public-key encryption scheme [36, 41]). They show how tag authentication can be implemented conforming to the EPC Gen2 Version 2 standard, and how the keys are managed.

4 Internet Protocol Security (IPsec)

IPsec [30] is a protocol suite that generates a confidential and integrity protected connection between two Internet peers over an unsecured data channel (also known as virtual private network connection). The IPsec protocol can be subdivided into the security association (SA) negotiation phase—where the properties of the established secured channel are declared—and a subsequent working phase. The IPsec works on the Internet layer of the TCP/IP protocol stack. Thus it is fully transparent for applications that use the secured communication channel.

Phase 1: Internet Key Exchange Protocol Version 2 (IKEv2). An Internet peer can host multiple IPsec secured connections to multiple peers. Therefore, so-called security associations (SA) are used to manage the connection parameters (used cryptographic algorithms, negotiated keys, et cetera.) for different negotiated IP and Port ranges. The first SA between two peers is created by exchanging an *IKE_SA_INIT*, and a subsequent *IKE_AUTH* request and response pair using the UDP protocol (see [29]). The initial *IKE_SA_INIT* message exchange provides the negotiation of the cryptographic algorithms, and the Diffie-Hellman key agreement in plain text. Hence, the subsequent *IKE_AUTH* messages are already encrypted and integrity protected according to the terms of the mutual agreement. The identity of the hosts is then ensured either by using pre-shared keying or certificates, and the authenticity of the exchanged messages

Table 2. IKEv2 transforms and according algorithms (see IANA [25])

Algorithm Name	Transform Types			
	Encryption	PRF	Integrity	Diffie-Hellmann
Advanced Encryption Standard (AES [11])	✓	✓	✓	
Blowfish	✓			
CAST	✓			
Camellia	✓			
Data Encryption Standard (DES)	✓		✓	
Int. Data Encryption Algorithm (IDEA)	✓			
Message-Digest Algorithm 5 (MD5)		✓	✓	
Modular Exponential Groups (MODP)				✓
Elliptic Curve Groups modulo a Prime (ECP [33])				✓
Secure Hash Algorithm (SHA1, SHA2)		✓	✓	
Tiger		✓		

is proven to the opposite communication partner. Furthermore, another SA is derived (Child SA) that is associated either to the *Authentication Header* protocol or the *Encapsulation Security Payload* protocol.

4.1 Phase 2: Authentication Header and Encapsulation Security Payload

Once a secure connection has been established via IKEv2, outgoing IP packets that match the negotiated IP and Port ranges are either wrapped into an Authentication Header (AH), or an Encapsulation Security Payload (ESP). AH provides integrity and authenticity protection of the enclosed IP packets without encryption of the data. ESP also offers confidentiality protection by encrypting the encapsulated packet. On the other side of the communication channel, the packets are automatically decrypted—if the packet contains an ESP—, unwrapped, and checked before they are processed any further.

4.2 Cryptographic Algorithms used by IPsec

IPsec defines different types of cryptographic algorithms, which are also called Transforms in the context of IKEv2. Table 2 summarizes the currently available algorithms for each of the Transform types. The first category defined are encryption algorithms that are used for IKEv2 (*IKE_AUTH*) and ESP to protect the confidentiality of the exchanged messages. Pseudo-random functions (PRFs) are used to derive the initial keying material for the other Transforms, and to derive new keying material when a Child SA is created. The key seed for the key derivation is calculated from the Diffie-Hellman exchange in the *IKE_SA_INIT* step. To protect the integrity of the payloads, and in order to authenticate the involved peers, integrity algorithms are used to create irreversible message authentication codes (MAC).

5 Integrating IPsec Functionality into EPC Gen2

The prime motivation for implementing IPsec on an EPC Gen2 Class 1 tag is to enable tags—and the things they are attached to—to become secure and independent participants on the Internet of Things. The tags then no longer depend on the trustworthiness of the readers, because a secure communication channel is established between the tag and another participant on the Internet of Things. Also the service a tag provides is no longer limited to the functionality a reader provides. In the suggested system, the functionality of a reader is primarily to work as a router, and forward IP packets from and to the tags. This enables a secure connection to any device on the Internet.

5.1 IPsec as Cryptographic Suite for EPC Gen2

In order to implement IPsec with IKEv2, at least one of each Transform class listed in Table 2 needs to be implemented. When compared to Table 1, it can be seen that the only algorithms currently proposed for both the EPC Gen2 cryptographic suites and the IPsec protocol, are the Advanced Encryption Standard (AES), and the Diffie-Hellman key exchange over an Elliptic Curve (ECP/ECC-DH). However, with these two cryptographic primitives, all required Transforms can be realized. Even though not all modes of operation defined in the cryptographic suites are currently publicly available, the presence of the cryptographic primitives, e.g. as cryptographic co-processors, allows implementation of all modes needed with reasonable effort.

The communication between any active participant (initiator) on the Internet of Things, and a particular EPC Gen2 tag is illustrated in Figure 2. At first a *IKE_SA_INIT* message—encapsulated in an IP/UDP packet—is sent by an initiator and routed to the tag ①. The reader receives the packet, and sends a *Challenge* command to the tag containing the *IKE_SA_INIT* message. This message consists of a Diffie-Hellman value (ECC point), the initiator nonce, and one or more proposals for the cryptographic algorithms to be used for the next step. If one proposal is acceptable to the tag it begins to create the initial Security Association (SA). Therefore, the tag calculates its own Diffie-Hellman value, creates another nonce, and derives the key material from the Diffie-Hellman shared secret and the nonces. The key material for the initial SA consists of seven keys. The first derived key is used to create new key material for future Child SAs. Additionally, for each communication direction a key for encryption and decryption is generated, as well as a key for integrity protection, and another key for authentication purposes.

Similar to the *IKE_SA_INIT*, the tag's response contains the accepted SA proposal, the calculated Diffie-Hellman value, and the generated nonce. Once the response has been created and stored in the *ResponseBuffer*, the tag calculates the so-called *AUTH* value, which is used for authentication of the tag. The *AUTH* value is calculated by signing the whole *IKE_SA_INIT* response, the initiator nonce, and a unique identifier (e.g., the tag's IPv6 address), by applying the selected signing algorithm. The output is a signature that is transferred in the *IKE_AUTH* step. In the upcoming inventory round, the reader recognizes that the tag has finished the processing of the *Challenge* command by preselecting only tags that store a message in the *ResponseBuffer*. After the tag was successfully inventoried, the reader either received the *IKE_SA_INIT* response

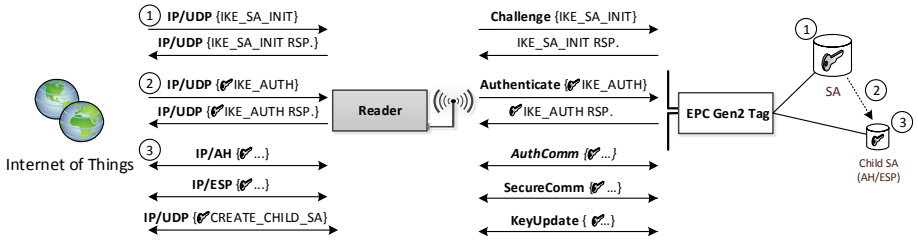

Fig. 2. Communication scenario for IPsec enabled EPC Gen2 tags

together with the tag's EPC after the tag was "acknowledged" during the inventory, or the *ResponseBuffer* is read by using the *ReadBuffer* command in the "Access" phase (see Figure 1). The reader then encapsulates the gathered response in an IP/UDP frame and sends it back to the initiator. On the initiator side, similar calculations are performed as for the tag, resulting in the same SA.

For the next step ②, the SA is used to encrypt the *IKE_AUTH* payloads, and to protect the integrity of the message by concatenating a message authentication code (MAC) calculated over the whole message. With the *IKE_AUTH* message, the initiator reveals its identity, and proves its genuineness either via a pre-shared secret (PSK), or a certificate that is used to generate a signature. This message type perfectly fits into an EPC Gen2 Version 2 *Authenticate* command, because the initiator authenticates itself against the tag, and vice versa. The rest of the message consists of a proposal for the Child SA that is created for the AH or ESP message exchange, and the Traffic Selectors. If the tag is able to verify the identity of the initiator and accepts the other parameters, it transits from the "Access" phase to the "Secured Access" phase. The response of the tag consists of the same payloads as the *IKE_AUTH* request, but carries the identity of the tag, its *AUTH* value calculated in the *Challenge* step, and confirms the proposed parameters for the Child SA and the Traffic Selectors. In step ③, the Child SA on both sides was successfully established, and the communication between initiator and tag is protected. Figure 2 shows the counterparts of the EPC Gen2 standard's *AuthComm* and *SecureComm*, which are the IP extension headers AH and ESP. On top of these extensions headers any application protocol can be implemented, without taking care of the underlying security mechanisms. Since, the lifetime of keys for a secured communication should be limited to guarantee the security of the communication channel, the IKEv2 *CREATE_CHILD_SA* message can be used for rekeying an already existing SA. The EPC Gen2 pendant for the *CREATE_CHILD_SA* message is the *KeyUpdate* that generates new key material for existing SAs.

6 Implementation of an IPsec-enabled Tag Prototype

In order to prove the feasibility of the protocol stated in Section 5, a "Prototype for the Internet of Things based on an extendable EPC Gen2 RFID tag" (*PIONEER*) has been

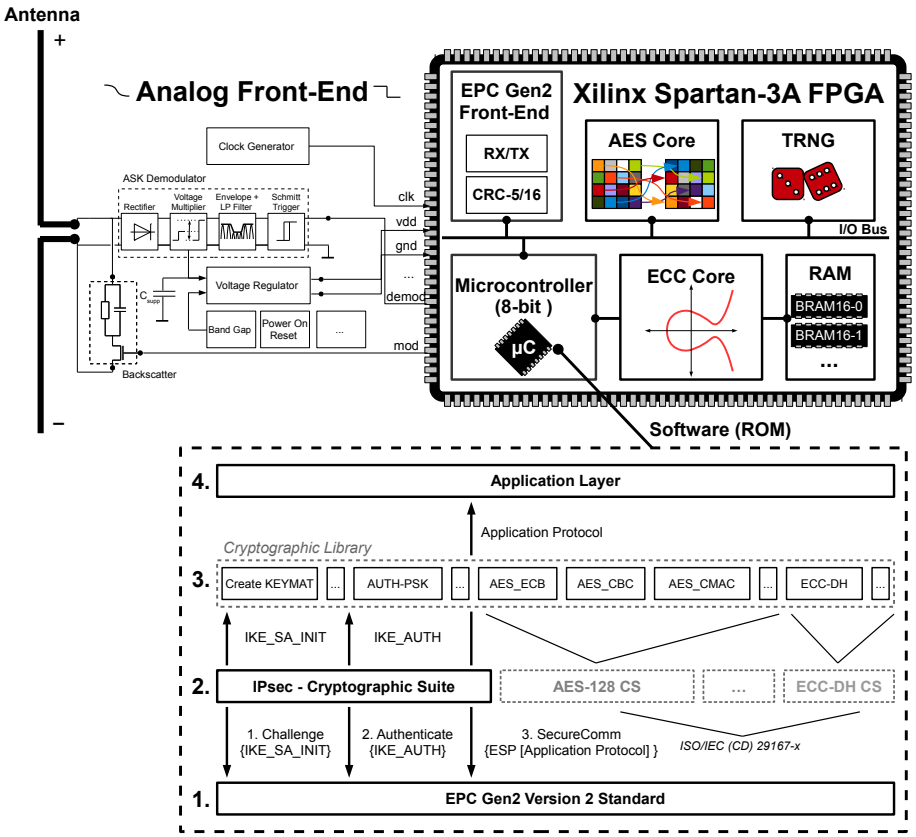

Fig. 3. System overview of the PIONEER tag containing the analog and digital components, as well as the four layers of the software architecture

implemented. The *PIONEER* tag consists of a printed circuit board, with a dipole antenna, an analog EPC Gen2 Front-end for modulation and demodulation of the electromagnetic signals, and an Xilinx Spartan-3A FPGA. All digital components are placed on the FPGA. For low-level encoding/decoding of the information signals, a dedicated hardware module is used which is connected to the I/O bus of an 8-bit microcontroller (designed by E. Wenger et al. [54]) implemented in software (C code), including the proposed IPsec cryptographic suite with its cryptographic library. All cryptographic functions are based on an already existing AES (designed by M. Feldhofer et al. [18]) and the ECC co-processors. To generate nonces that fulfill the necessary security requirements for unpredictability, a true random number generator has been implemented and connected to the microcontroller. A system overview containing the most important hardware and software modules of the tag is shown in Figure 3. In the following, the modules and their interactions are described in more detail.

6.1 The EPC Gen2 Front-end and Microcontroller

For reader-to-tag communication, the ASK (amplitude-shift keying) modulated electromagnetic reader signal is first obtained by the antenna (see Figure 3). Then the signal is filtered, amplified, and finally demodulated in the *analog font-end*. At the output of the *analog font-end* the Xilinx Spartan-3A FPGA is fed by a clean digitalized data signal. Moreover, the clock and reset signals, the power supply, and a data input signal for the tag-to-reader communication are also provided by the *analog font-end*. The next processing stage is the decoding of the PIE (pulse-interval encoding) encoded data signal in the *EPC Gen2 Front-end* module. In parallel, the CRC (cyclic redundancy checksum) is calculated, and checked for the entire reader command. The *EPC Gen2 Front-end* collects a maximum of eight bits of input data, before an interrupt is triggered that signals the *microcontroller* once data is ready to be picked up. In the interrupt service routine, the *microcontroller* fetches the received reader command by reading the data from the 8-bit I/O bus, and then processes it according to the EPC Gen2 Version 2 standard.

The back channel of the communication works similarly. At the beginning of the transmission, the *microcontroller* writes the first 8-bit chunk of the reply to the *EPC Gen2 Front-end*. The *EPC Gen2 Front-end* then signals the *microcontroller* per interrupt as soon as the next data part can be transmitted. According to the modulation settings of the last received *Query* command, the *EPC Gen2 Front-end* generates an FM0 (bi-phase space) or Miller encoded data signal. The encoded data signal is forwarded to the *analog font-end*, where this signal controls the transistor of the backscatter network that connects or disconnects a load impedance connected to the antenna.

Software Implementation. The firmware of the PIONEER tag is entirely written in C code, and can be subdivided into four layers (see Figure 3). On the first layer, all the EPC Gen2 Version 2 functionality is implemented that is marked mandatory in the standard. On top of this layer there is the cryptographic suite (CS) layer consisting of the proposed IPsec CS. The AES-128 and ECC-DH CS are shown grayed in Figure 3, because they are not actually implemented—since the standards have not yet been released—, but it is assumed here that a subset of the future CS functionality is already implemented as a side product of the IPsec CS. All IPsec related reader commands are handled on this layer. The required cryptographic functionality, e.g. derivation of the keys, hashing, encryption and decryption, et cetera., are implemented in the cryptographic library beyond. This layer also provides the software interface to the AES and ECC cryptographic co-processors, and is used in most of the other library functions. After the creation of the secure communication channel, incoming ESP packets are first handled (integrity checked, authenticated, and decrypted) in the IPsec CS. Afterwards, the payload of the ESP packet is transferred into the application layer. The inverse path through the software layers is taken for outgoing ESP packets. All IPsec functionality works completely transparently for the application layer. The implemented top-level example application of the *PIONEER* tag waits for an incoming IP/UDP packet, checks the validity of the packet, and prepends a string to the received UDP payload data before it is returned as a valid ESP packet. With this example application we have been able to create an IPsec channel between the tag and a computer in our network that uses the open source implementation of IPsec “strongSwan” [50].

6.2 The ECC Core

The requirements for the ECC module of the PIONEER tag are a small chip area, low power consumption, and that it delivers a runtime that is sufficient for interactive protocols. To reach all those goals, a rigorous hardware-software co-design approach has been taken. The timing-critical finite-field arithmetic is built into a so-called drop-in ECC module [53], while the point arithmetic is done in software. To safe area, the ECC module does not come with a dedicated memory, but instead reuses the CPU's data memory. Therefore, the drop-in module is placed between the CPU and the memory. However, the drop-in ECC module does not hinder the CPU to access the data memory, as the drop-in module has a built in light-weight arbiter that always prioritizes the CPU.

Our software and hardware is specially optimized for the NIST P-192 [37] elliptic curve. The scalar multiplication algorithm uses the Montgomery-ladder formulas as proposed by Hutter *et al.* [24], multiple point validation checks, and randomized projective coordinates [10] to counter some of the most powerful power-analysis attacks. In order to prevent timing attacks, the scalar multiplication is computed in constant time. Therefore all finite-field addition, subtraction, multiplication and inversion algorithms are also executed in constant time. The inversion algorithm is based on Itoh and Tsuji's [27] trick which minimizes the number of multiplications during inversion.

The ECC drop-in module for PIONEER is based on the drop-in module by Unterluggauer and Wenger [52]. While their drop-in module is designed to do efficient Montgomery multiplications, the drop-in module for PIONEER is optimized for the NIST P-192 prime $p = 2^{192} - 2^{64} - 1$. Additionally, to ensure a performant point multiplication, the RAM has to be modified. While the interface to the CPU is a traditional 8-bit wide bus, the data memory has a 32-bit interface (with a byte-wise write-enable feature). The drop-in ECC module is responsible for mapping the CPU's 8-bit addresses to 32-bit RAM addresses. With a 32-bit interface between the ECC module and the RAM, it was possible to keep the ECC module without dedicated memory and to achieve a practical performance.

The combined hardware-software co-design approach enables a fast runtime of a scalar multiplication in around 695 000 cycles. This runtime is especially impressive considering that a comparable AVR processor would require 15 million cycles to compute the scalar multiplication on its own (cf. [55]). Even at a clock rate of mere 4 MHz, PIONEER is able to compute a scalar multiplication within 178.63 ms.

6.3 True Random Number Generator (TRNG)

The implemented TRNG was presented by Cherkaoui et al. in [9]. This TRNG was designed following the recommendation of AIS-31 [49]. The entropy per bit can be determined by using the provided stochastic model. This TRNG consist of a Self-Timed Ring (STR) that generates multiple jittery clock signals which are sampled using the same clock. An XOR-tree is used to hash all the sampled signals in one bit. If at least one of the STR signals is sampled in the jitter zone the output will be random. The statistical test of NIST [45] has been applied to 100 traces of 10^6 bits. The TRNG output passes this test successfully. In addition, the TRNG has been evaluated using different core voltages and for different temperatures (from $35\,^{\circ}\mathrm{C}$ to $85\,^{\circ}\mathrm{C}$), obtaining good results in terms of randomness.

7 Implementation Results

The hardware implementation mainly consists of five modules—the microcontroller's program memory and RAM not included. The left part of Table 3 shows the resource utilization of these modules for the used Xilinx Spartan-3A FPGA split into the number of consumed flip-flops (FFs), the number of used look-up tables (LUTs), and the number of occupied Block RAM instances (BRAM). The microcontroller is by far the most area consuming module of the design with 389 FFs and 2 517 LUTs. For the cryptographic cores, 329 FFs and 710 LUTs for the AES core, and 229 FFs and 1 164 LUTs for the ECC core are required. The least hardware consuming modules are the EPC Gen2 Font-end with 178 FFs and 427 LUTs, and the TRNG with 90 FFs and 254 LUTs. The program memory and the microcontroller's RAM are both mapped into 16 and 4 dedicated Block RAMs respectively.

On the right side of Table 3, the post-synthesis area results for an UMC L130E Low Leakage CMOS cell library from Faraday for 4 MHz clock frequency, and 1.2 V power supply are listed. The area metric for the application-specific integrated circuit (ASIC) design flow results are the gate equivalents (GE, $1\,\mathrm{GE} \approx 5.12\,\mathrm{qm}^2$ for the 130 nm UMC cell library) of the modules. For the calculation of the RAM and ROM area requirements, dedicated UMC *Sync. High Density Single Port SRAM*, and *Via-1 ROM* macros are used. The TRNG of the FPGA design could not be synthesized for an ASIC design flow—because it uses FPGA specific functionality. Hence, the area requirement of the TRNG has been estimated on the basis of following related work. Holleman et al. [23] uses a floating-gate memory cell for their *DC-nulling* TRNG that consumes about 564 GE for a 0.35 µm process (AMS C35B4C3). Another TRNG was published by Ben-Romdhane et al. [6] which is a metastability based *open-loop delay chain* design, and needs about 300 gates in a 65 nm CMOS process from STMicroelectronics. The upper bound for the TRNG size estimation of 1 kGE comes from Ranasinghe et al. [42]. They presented a TRNG which utilizes a PUF (physically unclonable function) with challenges that create unstable (random) responses. Thus, the resulting overall area requirement for the digital part of the PIONEER tag is about 52 kGE or $0.266\,\mathrm{mm}^2$ for the UMC 130 nm process. The design was synthesized with the Cadence Encounter RTL Compiler Version v08.10-s28_1, and place and route was done using Cadence NanoRoute v08.10-s155.

In Table 4, the size of the tags program memory parts are listed according to the software layers in Figure 3. The whole implementation takes about 18.3 kilobytes of ROM. The biggest part of the program memory is consumed by the mandatory EPC Gen2 Version 2 functionality with 6.83 kilobytes. For the IPsec cryptographic suite 5.76 kilobytes ROM were needed. The cryptographic library is subdivided into three parts. The AES subpart consists of the encryption and decryption functionality, and the CBC mode (cipher block chaining). All other IPsec specific functionality like key material derivation and the PRF function (cf. AES_XCBC) are summed up in the IPsec subpart. If it is assumed that the AES and ECC subparts are already implemented as cryptographic suites, the additional implementation costs for the IPsec functionality are about 6.7 kilobytes. The smallest part of the ROM is the example application with 1.43 kilobytes only.

Table 3. Area requirement of the digital components for the Xilinx Spartan-3A FPGA, and the results for an 130 nm ASIC design flow

Hardware Modules	FPGA Resource Utiliz.			ASIC Area	
	[FFs]	[LUTs]	[BRAM]	[GEs]	[%]
AES Core	329	710	–	3 678	7.08
ECC Core	229	1 164	–	6 577	12.67
EPC Gen2 Front-end	178	427	–	2 253	4.34
Microcontroller	389	2 517	–	7 906	15.23
ROM	–	–	16	19 161	36.91
RAM	–	–	4	11 339	21.84
TRNG	90	254	–	1 000	1.93
Total	**1 215**	**5 072**	**20**	**51 914**	

In order to prove the correctness of the IPsec implementation, a laboratory setup was built that simulates the communication scenario shown in Figure 2. For the left communication partner (interrogator), a virtual machine running Linux Kernel 3.10.7 with the open source IPsec implementation strongSwan 5.1.0 is used. The network packets generated by the interrogator are then captured by a Java application that extracts the IPsec payloads of the network packets and forwards them via an UHF reader device to the PIONEER tag, and vice versa. To determine the timing of the protocol in detail, the handling of the EPC Gen2 commands encapsulating IPsec messages was also simulated using Cadence NCsim. The biggest amount of time with 410.95 ms is consumed by processing the initial Challenge (IKE_SA_INIT) command, because the Diffie-Hellman key exchange involves two point multiplications. Therefore, almost 87% of the processing time is spent on calculating the elliptic curve points. The derivation of the keying material and the calculation of the authentication and integrity check values consume another 11.94 ms (about 12%). For the handling of the Authenticate (IKE_AUTH) command 15.76 ms are required. Most of the time (94.7%) is spent on performing AES operations for decrypting the message, integrity and authenticity checking, deriving keying material for the ESP Child SA, and calculating the response. When optimum timing for the reader to tag communication is considered with 640 kbps uplink frequency and 128 kbps downlink frequency, the inventory procedures takes at least 1.2 ms. The IKE_SA_INIT exchange takes about 13.98 ms, and the IKE_AUTH message exchange consumes almost 17.38 ms communication time. Considering both communication effort and processing effort on tag side, it takes at least 459.25 ms until an IPsec channel is established, when the interrogator to reader communication is not taken into account. The results show that almost 92.9% of the time is spent on processing the messages and only 7.1% is required for communication.

After the IPsec tunnel is established, the interrogator sends an UDP datagram encapsulated into a protected ESP packet to the PIONEER tag. The tag decrypts the payload of the received packet and checks the authenticity of the sender. Eventually, the tag creates a reply and sends it back to the interrogator. Since, the ESP packet uses AES with 128-bit (16 bytes) block size, the time for sending and receiving ESP packets de-

Table 4. Size and partitioning of the program memory in bytes and percent

Software Layer	ROM Size [bytes]	[%]
1. EPC Gen2 Version 2	6 828	37.31
2. IPsec CS	5 762	31.49
3. Cryptographic Library	4 280	23.39
- AES	738	4.03
- ECC	2 608	14.25
- IPsec	934	5.11
4. Application	1 426	7.79
Total	**18 296**	

pends on the number of 16-byte blocks (n_b) that are transmitted. The up-link and down-link speed can therefore be expressed as a linear function with a fixed intercept value and an n_b dependent slope parameter. The reader to tag communication thus consumes $5.6 + 1.99 * n_b$ ms, and in the other communication direction it takes $4.2 + 1.19 * n_b$ ms to transmit one ESP packet—including sending and processing efforts.

8 Conclusions

In this paper, we presented an RFID-tag prototyping platform that integrates IPsec into the new EPC Gen2 Version 2 standard. The obtained results can help to estimate the costs for practical implementations that aim an open and secure Internet of Things where EPC tags play a central role. The cryptographic primitives suggested for the EPC Gen2 Version 2 cryptographic suites—in the ISO/IEC 29167 standard—provide strong symmetric and asymmetric cryptography. Additionally, the EPC Gen2 security functionality partially overlaps with the IPsec protocol which is used to create a secure communication channel over the Internet. We have also shown that by merging the IPsec and EPC Gen2 functionality, RFID tags can be enabled to work as independent nodes on the Internet without the need for trusted readers. Our theoretical considerations have been proven by an FPGA based tag prototype, with which we were able to establish an IPsec tunnel between the tag prototype and a computer inside our local network. We support the meaning of Sarma et al. [48], who believe in an Internet of Things based on a single and open architecture as the key to a successful system. Moreover, we believe that the potential impact is even higher, if already established protocols and standardized security is used to create the Internet of Things.

Acknowledgements. This work has been supported by the Austrian Science Fund (FWF) under the grant number TRP251-N23 (Realizing a Secure Internet of Things - ReSIT) and the FFG research program SeCoS (project number 836628).

References

1. M. Agren, M. Hell, T. Johansson, and W. Meier. Grain-128a: A New Version of Grain-128 with Optional Authentication. *Int. J. Wire. Mob. Comput.*, 5(1):48–59, Dec. 2011.
2. ANSI. *Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography*. Accredited Standards Committee X9, Incorporated, 2001.
3. A. Arbit, Y. Oren, and A. Wool. Toward Practical Public Key Anti-Counterfeiting for Low-Cost EPC Tags. In *RFID*, pages 184–191. IEEE, April 2011.
4. K. Ashton. That 'Internet of Things' Thing. `http://www.rfidjournal.com/articles/view?4986`, 2009. Accessed: 2014-02-18.
5. G. Avoine and X. Carpent. Yet Another Ultralightweight Authentication Protocol That Is Broken. In J.-H. Hoepman and I. Verbauwhede, editors, *RFIDsec*, volume 7739 of *LNCS*, pages 20–30. Springer Berlin Heidelberg, 2013.
6. M. Ben-Romdhane, T. Graba, J.-L. Danger, and Y. Mathieu. Design methodology of an ASIC TRNG based on an open-loop delay chain. In *IEEE International Workshops on New Circuits and Systems Conference (NEWCAS)*, pages 1–4, June 2013.
7. 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 *CHES*, pages 450–466, 2007.
8. Y.-C. Chang, J.-L. Chen, Y.-S. Lin, and S. M. Wang. RFIPv6 - A Novel IPv6-EPC Bridge Mechanism. In *International Conference on Consumer Electronics – ICCE*, pages 1–2, 2008.
9. A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert. A Very High Speed True Random Number Generator with Entropy Assessment. In *CHES*, pages 179–196. Springer, 2013.
10. J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *CHES 1999, Worcester, MA, USA, August 12-13*, volume 1717 of *LNCS*, pages 292–302. Springer, 1999.
11. J. Daemen and V. Rijmen. The Block Cipher Rijndael. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications*, volume 1820 of *LNCS*, pages 277–284. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67923-3.
12. S. Dominikus, M. Aigner, and S. Kraxberger. Passive RFID Technology for the Internet of Things. In *International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 1–8, 2010.
13. S. Dominikus and S. Kraxberger. Secure Communication with RFID tags in the Internet of Things. *Security and Communication Networks*, pages n/a–n/a, 2011.
14. M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. NIST, 2001.
15. D. Engels, Y. S. Kang, and J. Wang. On Security with the new Gen2 RFID Security Framework. In *RFID (RFID), 2013 IEEE International Conference on*, pages 144–151, 2013.
16. EPCglobal. EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID Specification for RFID Air Interface Protocol for Communication at 860 MHz - 960 MHz Version 2.0.0 Ratified, November 2013. Available online at `http://www.gs1.org`.
17. J. Ertl, T. Plos, M. Feldhofer, N. Felber, and L. Henzen. A Security-Enhanced UHF RFID Tag Chip. In *Euromicro Conference on Digital System Design (DSD)*, pages 705–712, 2013.
18. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES Implementation on a Grain of Sand. *IEEE Proceedings on Information Security*, 152(1):13–20, 2005.
19. M. Girault, G. Poupard, J. Stern, M. Girault, G. Poupard, and J. Stern. On the Fly Authentication and Signature Schemes based on Groups of Unknown Order. *Journal of Cryptology*, 19:463–487, 2006.

20. J. Ha, S. Moon, J. Zhou, and J. Ha. A new Formal Proof Model for RFID Location Privacy. In S. Jajodia and J. Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *LNCS*, pages 267–281. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88312-8.
21. H. Hada and J. Mitsugi. EPC based Internet of Things Architecture. In *IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, pages 527–532, 2011.
22. W. Hinz, K. Finkenzeller, and M. Seysen. Secure UHF Tags with Strong Cryptography - Development of ISO/IEC 18000-63 Compatible Secure RFID Tags and Presentation of First Results. In *SENSORNETS*, pages 5–13, 2013.
23. J. Holleman, B. Otis, S. Bridges, A. Mitros, and C. Diorio. A 2.92 uW Hardware Random Number Generator. In *European Solid-State Circuits Conference, 2006*, pages 134–137, 2006.
24. M. Hutter, M. Joye, and Y. Sierra. Memory-Constrained Implementations of Elliptic Curve Cryptography in Co-Z Coordinate Representation. In *AFRICACRYPT 2011, Dakar, Senegal, July 5-7*, volume 6737 of *LNCS*, pages 170–187. Springer, 2011.
25. IANA - Internet Assigned Numbers Authority. Referenced 2014 at `http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml`.
26. ISO - International Organization for Standardization. Referenced 2014 at `http://www.iso.org/`.
27. T. Itoh and S. Tsujii. Effective recursive algorithm for computing multiplicative inverses in $GF(2^m)$. *Electronic Letters*, 24(6):334–335, 1988.
28. D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.
29. C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996 (Proposed Standard), Sept. 2010. Updated by RFCs 5998, 6989.
30. S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), Dec. 2005. Updated by RFC 6040.
31. J.-W. Lee, N. D. Phan, D. H.-T. Vo, and V.-H. Duong. A Fully Integrated EPC Gen-2 UHF-Band Passive Tag IC Using an Efficient Power Management Technique. *IEEE Transactions on Industrial Electronics*, 61(6):2922–2932, June 2014.
32. M. Lehtonen, D. Ostojic, A. Ilic, and F. Michahelles. Securing RFID Systems by Detecting Tag Cloning. In H. Tokuda, M. Beigl, A. Friday, A. Brush, and Y. Tobe, editors, *Pervasive Computing*, volume 5538 of *LNCS*, pages 291–308. Springer Berlin Heidelberg, 2009.
33. M. Lepinski and S. Kent. Additional Diffie-Hellman Groups for Use with IETF Standards. RFC 5114 (Informational), Jan. 2008.
34. F. Mattern and C. Floerkemeier. From the Internet of Computers to the Internet of Things. In K. Sachs, I. Petrov, and P. Guerrero, editors, *From Active Data Management to Event-Based Systems and More*, volume 6462 of *LNCS*, pages 242–259. Springer Berlin Heidelberg, 2010.
35. A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.
36. P. L. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 44(170):pp. 519–521, 1985.
37. National Institute of Standards and Technology (NIST). FIPS-186-3: Digital Signature Standard (DSS), 2009. Available online at `http://www.itl.nist.gov/fipspubs/`.
38. A. Noman, M. Rahman, and C. Adams. Improving Security and Usability of Low Cost RFID Tags. In *International Conference on Privacy, Security and Trust (PST)*, pages 134–141, 2011.
39. L. Pang, L. He, Q. Pei, and Y. Wang. Secure and Efficient Mutual Authentication Protocol for RFID Conforming to the EPC C-1 G-2 Standard. In *Wireless Communications and Networking Conference (WCNC)*, pages 1870–1875, 2013.

40. P. Peris-Lopez, T.-L. Lim, and T. Li. Providing Stronger Authentication at a Low Cost to RFID Tags Operating under the EPCglobal Framework. In *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, volume 2, pages 159–166, 2008.
41. M. O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical report, Cambridge, MA, USA, 1979.
42. D. C. Ranasinghe, D. Limb, S. Devadas, B. Jamali, Z. Zhu, and P. H. Cole. An Efficient Hardware Random Number Generator.
43. S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. In *International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, 2011.
44. R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.
45. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. `http://csrc.nist.gov/rng/`, 2001.
46. M. Safkhani, N. Bagheri, P. Peris-Lopez, A. Mitrokotsa, and J. Hernandez-Castro. Weaknesses in Another Gen2-based RFID Authentication Protocol. In *RFID-Technologies and Applications (RFID-TA), 2012 IEEE International Conference on*, pages 80–84, 2012.
47. A. Sample, D. Yeager, P. Powledge, and J. Smith. Design of a Passively-Powered, Programmable Sensing Platform for UHF RFID Systems. In *IEEE International Conference on RFID*, pages 149–156, March 2007.
48. S. Sarma, D. L. Brock, and K. Ashton. White Paper: The Networked Physical World. `http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-001.pdf`, 2000. Accessed: 2014-02-18.
49. W. Schindler and W. Killmann. Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications. In *CHES 2002*, LNCS, pages 431–449, 2003.
50. strongSwan - the OpenSource IPsec-based VPN Solution. Referenced 2014 at `http://www.strongswan.org/`.
51. D.-Z. Sun and J.-D. Zhong. A Hash-Based RFID Security Protocol for Strong Privacy Protection. *Consumer Electronics, IEEE Transactions on*, 58(4):1246 –1252, 2012.
52. T. Unterluggauer and E. Wenger. Efficient Pairings and ECC for Embedded Systems. In L. Batina and M. Robshaw, editors, *CHES*, LNCS, 2014.
53. E. Wenger. Hardware Architectures for MSP430-Based Wireless Sensor Nodes Performing Elliptic Curve Cryptography. In *ACNS*, volume 7954 of *LNCS*, pages 290–306, 2013.
54. E. Wenger, T. Baier, and J. Feichtner. JAAVR: Introducing the Next Generation of Security-enabled RFID Tags. In S. Niar, editor, *Digital System Design*, pages 640–647. IEEE, 2012.
55. E. Wenger, T. Unterluggauer, and M. Werner. 8/16/32 shades of elliptic curve cryptography on embedded processors. In G. Paul and S. Vaudenay, editors, *INDOCRYPT*, volume 8250 of *LNCS*, pages 244–261. Springer, 2013.
56. X. Yi, L. Wang, D. Mao, and Y. Zhan. An Gen2 Based Security Authentication Protocol for RFID System. *Physics Procedia*, 24, Part B(0):1385 – 1391, 2012. International Conference on Applied Physics and Industrial Engineering 2012.