Using Elliptic Curves on RFID Tags

Michael Braun, Erwin Hess and Bernd Meyer

Siemens AG, Corporate Technology, Munich, Germany

Summary
We present a concept for the realization of asymmetric cryptographic techniques in light-weight cryptographic devices and describe an implementation based on elliptic curve cryptography which can be used for authentication in mass applications of RFID tags. Our schemes offer advantages in large decentralized applications with many unobservable readers in the field over previous solutions. Moreover, using public key techniques cryptographic protocols that protect the privacy of the tag bearer can be easily implemented.

Key words:
RFID, Authentication, Elliptic Curve Cryptography, Privacy.

1. Introduction

RFID tags are small devices for identification purposes applicable in a wide variety of application scenarios as supply chain management, access control systems, anti counterfeiting of luxury goods, tracking of goods, inventory management, etc. It is expected that many of these devices will be deployed in the near future and hence several security and privacy threats will come up depending on the application where RFID tags used.

In the case of preventing counterfeiting of luxury goods or expensive drugs, or securing an access control, it is essential to verify the authenticity of the product or person and it is not sufficient to read only the ID of the tag. Authenticity can be achieved by a secure protocol (e.g. challenge response protocol) running between RFID tag and reader. If a unique secret information is stored on the tag and the tag can convince the reader to possess that information, the tagged product is declared to be authentic, respectively the person gains access and otherwise not. Such an authentication protocol has to be dynamic and controlled by the reader otherwise by transmitting always static information an attacker can eavesdrop on the channel and obtains by the way the information sent by the RFID tag. Afterwards, the attacker easily replays the eavesdropped information upon request of a reader and makes it believe to be the genuine tag.

On the other hand RFID tags can be read unnoticed without any intervisibility which entails privacy threats. Assuming people having tags at their body or carrying tags in their clothes or bags, they can be scanned without permission and the harvested information can be used to analyze their behavior, e.g. where people go, what people buy, etc. We distinguish between data privacy and location privacy. Protection of data privacy means that the data transmitted by the RFID tag cannot be understood by any unauthorized reader whereas protection of location privacy means that the data sent by the tag cannot be used to track this tag. Protection of location privacy includes protection of data privacy and it has the additional property that the data sent by the tag must be different at each execution of the protocol. If the data sent by the tag is static, it is possible to track an RFID tag, even if the data is not interpretable by unauthorized readers. The occurrence of the same data alone at different locations can detect the same RFID tag and hence the person carrying that tag. In order to ensure protection of the privacy an RFID protocol has to be designed such that the information sent by the tag appears to be randomly chosen to an unauthorized reader and different whenever the tag is requested. Location privacy can be achieved by semantically secure encryption of the transmitted data.

An even stronger security property is the so-called forward location privacy protection. If an attacker can reveal secret information of an RFID tag at a particular time, she or he may not be able to recognize that tag in previously recorded instances of the protocol.

It depends on the application which security properties are necessary and must be provided by the protocol. E.g., an access control or electronic passports require a high security level. On the one hand authentication is essential and on the other hand protection of the privacy of the people, i.e. location privacy protection or forward location privacy protection, has to be assured.

Many protocols for authentication and/or privacy protection were proposed in recent years: e.g. Sarma et al. [23], Weis et al. [24], Ohkubo et al. [20], Molnar et al. [19], Rhee et al. [22], and Feldhofer et al. [11]. All of them use symmetric cryptography, for example keyed hash functions or AES implementations, to meet the constraints of low-power consumption, limited chip area, and restricted computation time in order to produce low-cost RFID tags. Protocols based on asymmetric cryptography have not been considered yet because they were supposed to be too complicated. But in many application scenarios it is indispensable to obtain the high security level provided by an asymmetric approach. The use of asymmetric instead of symmetric solutions for RFID systems can radically reduce costs, maybe not necessarily on side of the RFID tags themselves, but on the part of the readers and the corresponding middle ware expenses can be saved. Proposed symmetric protocols for authentication purposes

Manuscript received February 5, 2008
Manuscript revised February 20, 2008

need to store securely secret keys in the reader itself or at least a secure connection to a back end database must be provided. Furthermore, a complex search in order to retrieve the right key in the reader or in the back end database has to be accomplished. To prevent compromising the system, at least a part of the system, the reader or the connection to the back end database must be tamper-proof which is quite cost-intensive. Especially strong decentralized applications where not necessarily trustworthy persons operate the readers or the readers are not located in a secure area, require expensive readers and corresponding expensive middle ware. These costs can be reduced using asymmetric security mechanisms since RFID systems no longer have to distribute secret keys to each reader device or readers no longer must retrieve a secret key from a database over a secure channel. For asymmetric protocols in general it is difficult to meet the constraints of low-power consumption, limited chip area, and restricted computation time of RFID tags. Recent publications on this topic investigated the applicability of elliptic curve cryptography for RFID applications and the authors proposed approaches towards a low-cost RFID tag based on elliptic curve cryptography. Kumar et al. [15] introduced a hardware implementation of elliptic curves over finite fields of even characteristic and gave a performance analysis with respect to chip size, memory, and computation time. Batina et al. [2] designed a similar elliptic curve arithmetic unit, they also gave a performance analysis and proposed a protocol for tag authentication based on Okamoto's identification protocol [21].

Our Contributions

In this paper we give a detailed description of a hardware implementation of elliptic curves over finite fields of even characteristic but also in regard to countermeasures against side channel attacks.

Many cryptographic protocols using elliptic curves also need computations in the factor ring $\mathbb{Z}/q\mathbb{Z}$ where q is the order of the base point on the elliptic curve (see also [2]) and hence additional hardware logic is necessary. In our case we completely avoid operations in $\mathbb{Z}/q\mathbb{Z}$ and design an authentication protocol that only requires elliptic curve arithmetic.

We extend our authentication protocol to an authentication protocol with protection of the location privacy of the tag bearer and also protection of the forward location privacy.

2. The Implementation

There exists a vast literature on the implementation of elliptic curve cryptography in software [14]. In this article we focus on efficient hardware implementations which enable public key cryptography for mass applications like sensors and RFID tags. For hardware implementations and corresponding applications the requirements are different and new constraints must be fulfilled. Optimization techniques developed for software implementations often rely on time-memory trade-offs and cannot be used when a device with small footprint should be designed.

The dominating cost factor is the chip size of the elliptic curve hardware. Our architecture consists of an arithmetic unit, control logic, and memory (see Figure 1). The arithmetic unit computes additions and multiplications in the finite field over which the elliptic curve is defined. The control logic implements the scalar multiplication of points on the elliptic curve and the cryptographic protocol of the application without using an additional CPU. The memory contains volatile intermediate results of the point arithmetic and non-volatile system parameters and keys.

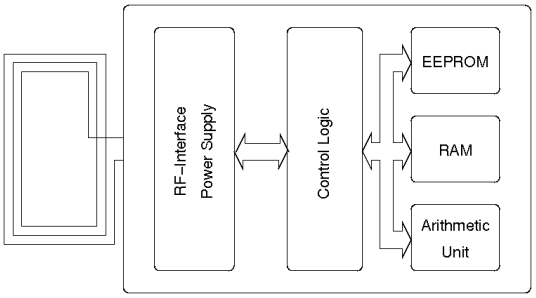

Fig. 1 The RFID Tag Architecture

Our arithmetic unit has a serial-parallel design consisting of the main component, an adder for the finite field, which can add two elements in a single clock cycle and logic for the generation of partial products for the multiplication. In this design two n bit numbers can be multiplied in n clock cycles. The serial-parallel design offers a good compromise between chip size and performance: a full parallel design could add and multiply in a single clock cycle but would increase the chip size considerably. A full serial approach could minimize the necessary hardware but would imply linear running-time for an addition and quadratic running-time for a multiplication. For fields of composite degree of extension further trade-offs are possible [8] but these fields are cryptographically weaker [12], [18].

We have chosen elliptic curves over a finite field $GF(2^n)$ since addition in this structure is very simple and consists of bitwise XOR operations. In (extension) fields of larger characteristic the adder has a long carry path which must be broken by a more complicated circuit design (carry save, carry look-ahead, redundant representation, or similar) to reduce power consumption and to increase performance.

The arithmetic unit has a highly regular design and can therefore be implemented bit-sliced in full-custom design. In Figure 2 a small example of a corresponding unit is given. The unit consists of an accumulator designed as a

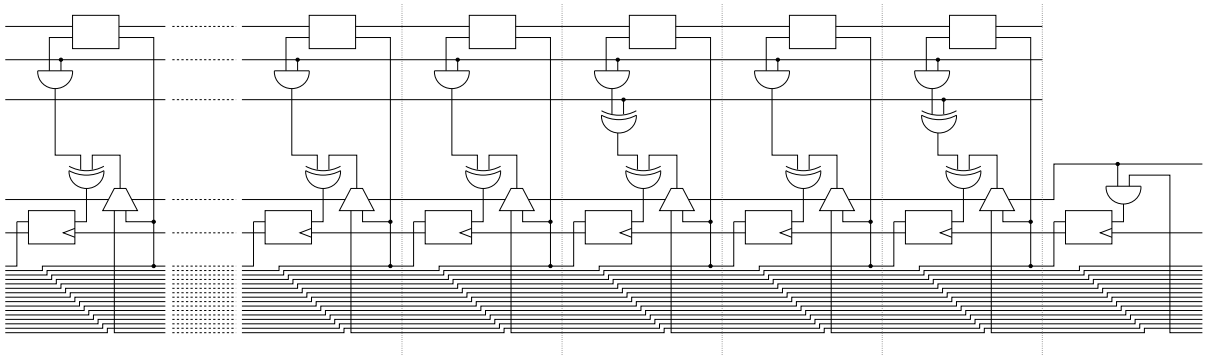

Fig. 2 The Arithmetic Unit

linear feedback shift register with irreducible feedback polynomial which is connected to a 16 bit bus, an operand register which can be loaded in parallel, AND-gates for the computation of partial products, XOR-gates for addition and modular reduction, and logic for loading values from the bus. To compute an addition, the first operand is loaded to the accumulator, copied to the operand register, the second operand is loaded to the accumulator, the operand register is added to the accumulator, and the result is stored. To compute a multiplication the first operand is loaded to the accumulator, copied to the operand register, the accumulator is cleared, and during the next n clock cycles the second operand is used bit by bit for the control of partial product generation. Finally, the result is stored. The irreducible feedback polynomial defines the polynomial basis for the representation of field elements. Despite the fact that squaring is a linear operation in $GF(2^n)$ and can be computed in one clock cycle using special hardware we decided to omit this additional hardware components and to compute squarings with the existing general multiplication hardware. Moreover, the hardware for squarings would destroy the regularity of the arithmetic unit and render a full-custom design more complicated.

Most standard cryptographic protocols using elliptic curves, for example ElGamal encryption, signature or ECDSA, need also computations in $\mathbb{Z}/q\mathbb{Z}$ where q is the prime order of the subgroup in which the cryptographic operations are done. To encounter this problem other proposals for elliptic curve hardware devices use curves over prime fields $\mathbb{Z}/q\mathbb{Z}$ and implement a more complicated arithmetic unit, design an arithmetic unit which can realize computations in $GF(2^n)$ and $\mathbb{Z}/q\mathbb{Z}$ (see [13]), or use a CPU and implement the computation modulo the order of the group of points in software [2]. In our design we use new cryptographic protocols without computations modulo the order of the group of points. This simplifies our hardware device considerably.

The arithmetic unit is not designed for fast inversions or divisions in the field $GF(2^n)$. If necessary, inversion can be done with Fermat's method using discrete exponentiation, i.e.

$$x^{-1} = x^{2^n-2}.$$

In our cryptographic protocols we use projective coordinate representation of points on the elliptic curve to avoid inversions and do not convert back results to affine representation. For example in the RFID application the conversion to affine representation can be done by the terminal which has more computing power than the tag.

Let $y^2 + xy = x^3 + ax^2 + b$ be the equation of the elliptic curve $E = E(a, b)$ where $a, b \in GF(2^n)$. Let $P = (x_P , y_P)$ be a point on the curve E and k be a scalar with binary representation $k = (k_\ell, \ldots, k_1)_2$. The scalar multiplication $Q = k \cdot P = P + \ldots + P$ (k times) is done using Montgomery's ladder [17]. Given the affine x-coordinate x_P and the binary representation of the scalar k the algorithm computes a projective representation X_1/Z_1 of the x-coordinate of $k \cdot P$. If the result is the point O at infinity then $Z_1 = 0$. We follow in the sequel the presentation of [14] (see Fig. 3) and in the following we use the term

$$(X_1, Z_1) \leftarrow \mathrm{Mul}(k, x_P)$$

whenever the algorithm is called. Montgomery's ladder algorithm also computes a projective representation X_2/Z_2 of the x-coordinate of $(k + 1) \cdot P$. If the y-coordinate of the point P is given then it is possible to reconstruct the y-coordinate of the result $k \cdot P$ from the values X_1, Z_1, X_2, Z_2, x_P and y_P (see [14]).

Input: $k = (k_\ell, \ldots, k_1)_2$ and x_P affine x-coordinate of P.
Output: (X_1, Z_1) projective representation of the x-coordinate of $k \cdot P$.

1. $X_1 \leftarrow 1, Z_1 \leftarrow 0, X_2 \leftarrow x_P, Z_2 \leftarrow 1$
2. for $i \leftarrow \ell$ downto 1 do
3. if $k_i = 1$ then
3.1. $T \leftarrow Z_1, Z_1 \leftarrow (X_1Z_2 + X_2Z_1)^2$,
3.2. $X_1 \leftarrow x_PZ_1 + X_1X_2TZ_2, T \leftarrow X_2$,
3.3. $X_2 \leftarrow X_2^4 + bZ_2^4, Z_2 \leftarrow T_2Z_2$
4. else
4.1. $T \leftarrow Z_2, Z_2 \leftarrow (X_2Z_1 + X_1Z_2)^2$,
4.2. $X_2 \leftarrow x_PZ_2 + X_2X_1TZ_1, T \leftarrow X_1$,
4.3. $X_1 \leftarrow X_1^4 + bZ_1^4, Z_1 \leftarrow T_1Z_1$
5. return (X_1, Z_1)

Fig. 3: Montgomery's Ladder

In our design we do not need to reconstruct the y-coordinate of $k \cdot P$. In the cryptographic protocols we only use the x-coordinates (see Section 5). The input x_P is an x-coordinate of a point on the elliptic curve in affine representation and the output (X_1, Z_1) an x-coordinate in projective representation.

Montgomery's ladder algorithm is advantageous for hardware implementations for several reasons:

The algorithm offers a competitive performance in comparison to non-window methods for scalar multiplication. Per bit of the scalar ten multiplications must be computed (not distinguishing between squaring and multiplication) (see [14]).

The algorithm has a highly regular structure. The only difference between the two execution paths of the if-statement is a change of (X_1, Z_1) and (X_2, Z_2). See also the countermeasures against simple power analysis and timing attacks in Section 5.1.

Since the scalar multiplication can be done with x-coordinates only the algorithm is very space-efficient. Using our arithmetic unit the implementation needs memory for five intermediate values (X_1, Z_1, X_2, Z_2 and a temporary value), the input x_P, and non-volatile memory for the curve parameter b (or the square root of b) and the scalar. The curve parameter a is not necessary for the computation and the protocols.

For security reasons it is necessary to verify that the input to the scalar multiplication is really a point on the curve (and belongs to the subgroup in which the cryptographic application should work). Otherwise, attacks to recover the scalar are possible (see [3]). These attacks can be avoided if the input to the elliptic curve scalar multiplication unit comprises a complete point (x, y) with both coordinates. Then the unit can check if the input satisfies the defining equation of the curve, but we need to communicate the y-coordinate to the device. If only the affine x-coordinate is given the device has to verify that the equation $\mathrm{Tr}(x + a + b/x^2) = 0$ holds which requires a costly computation of the inverse of x in $\mathrm{GF}(2^n)$. In addition we need to store the curve parameter a in the device in both cases.

In order to further simplify the implementation we decided not to check whether the input to the elliptic curve unit is a valid x-coordinate of a point on the elliptic curve. Instead we require that elliptic curves usable for our applications have additional security properties. This is motivated by the following observation.

For performance reasons and to assure that the elliptic curve E is cryptographically strong the curve is normally chosen such that the order of the group of points satisfies the condition $\mathrm{ord}(E) = rq$ where q is a large prime number and r is a small co-factor (in the best case $r = 2$). The base point P for cryptographic protocols is normally chosen to be a generator of the subgroup $< P >$ of order q. Then, approximately half of the elements $x \in \mathrm{GF}(2^n)$ are x-coordinates of points on E, i.e. x-coordinates of points in the group $< P >$ or of points in a different residue class of $< P >$. The remaining elements $x \in \mathrm{GF}(2^n)$ have the property that $\mathrm{Tr}(x + a + b/x^2) = 1$ and do not belong to points on E. But these elements are x-coordinates of points on a twisted curve $E_t = (a + v, b)$ where $v \in \mathrm{GF}(2^n)$ with $\mathrm{Tr}(v) = 1$. The orders of E and E_t over $\mathrm{GF}(2^n)$ are related by the equation $\mathrm{ord}(E) + \mathrm{ord}(E_t) = 2n + 1 + 2$ independent of the concrete value v. A fault tolerant strong elliptic curve E is an elliptic curve such that $\mathrm{ord}(E)$ and $\mathrm{ord}(E_t)$ are cryptographically strong, i.e., $\mathrm{ord}(E) = r_1q_1$ and $\mathrm{ord}(E_t) = r_2q_2$ such that q_1, q_2 are large prime numbers and r_1, r_2 are small co-factors (in the best case 2 and 4).

If our hardware device uses a fault tolerant strong elliptic curve then any input $x \in \mathrm{GF}(2^n)$ is either x-coordinate of a point on E and E_t. Therefore, if the prime numbers q_1, q_2 are large enough that both discrete logarithm problems are hard then given a secret scalar k an attacker would only be able to learn k mod r_1r_2 using techniques from [16]. These threats can for example be avoided by countermeasures against zero-value attacks [1] (see Section 5.1). Fault tolerant elliptic curves also offer protection against certain fault attacks (see Section 5.3).

Moreover, to protect against the attacks in [7] the elliptic curve E should be chosen in a way such that the prime numbers q_1 and q_2 dividing the orders of E and E_t are strong prime numbers, i.e., $q_1 - 1$ and $q_2 - 1$ are also divisible by large prime numbers themselves.

3. Size and Performance

We implemented our elliptic curve hardware in VHDL on Xilinx Spartan 3 FPGA. The synthesis tool counted a total of 18121 gate equivalents and computed corresponding gate counts for the different components of the unit figured in Table 1. In this table the memory includes the volatile and non-volatile parts. The VHDL code of all components of the unit (including the memory)

is completely synthesizable. If the arithmetic unit and the memory are implemented in full-custom design the resulting circuit consumes less space than the fully synthesized VHDL implementation and the corresponding equivalent gate count should be smaller. Using a system clock of 5 MHz and an elliptic curve over the field GF(2^{163}) a scalar multiplication with a 163 bit scalar can be done in 64 ms.

Table 1: Gate Count of the Elliptic Curve Unit

Component	*Gate equivalents*
Arithmetic unit	4548
Memory	11205
Control logic	2368

4. Resistance Against Side Channel Attacks

4.1. Simple Power Analysis and Timing Analysis

Montgomery's ladder is inherently protected against simple power analysis and timing attacks. The algorithm is highly regular and the overall execution of the algorithm does not depend on the value of the secret scalar. The running time depends only on the extension degree of the finite field GF(2^n) and the length of the scalar.

In addition hardware countermeasures must be taken to hide the traces of the execution of the if-statement in the power profile. It should not be visible which execution path of the if-statement has been taken. For example one could use the actual bit of the scalar to toggle between two sets of addresses for X_1, Z_1 and X_2, Z_2 of equal Hamming weight or use redundant representations of the scalar. The effectiveness of the countermeasures depends on the implementation of the control logic.

There exist only two cases such that a coordinate of a point in the representation used in Montgomery's ladder algorithm becomes zero: the point O at infinity has the representation $x \neq 0$ and $z = 0$ and the point of order two has the representation $x = 0$ and $z \neq 0$. If the input x_P to the scalar multiplication is an x-coordinate of a point P whose order is larger than the scalar then it is not possible that any intermediate result becomes zero: all projective coordinates X_1, Z_1, X_2, Z_2 belong to points of large order and are not equal to zero. This implies that all results of the products in GF(2^n) computed during the algorithm are not zero. The results of the additions are again projective coordinates of points of large order. Therefore, zero-value attacks (see [1]) can only be mounted if the input x_P belongs to a point of small order, i.e., if the order of P divides the co-factor of the curve. These attacks can be avoided if the device first computes a scalar multiplication of the input x_P with the co-factor of the curve to verify that the order of P is large. If this computation gives the result $Z_1 = 0$ then the input must be rejected.

4.2. Differential Power Analysis

Montgomery's ladder can be easily enhanced to become resistant against differential power analysis. We randomize the projective coordinate representation of points [9]. Let $r \in \mathrm{GF}(2^n)$, $r \neq 0$ be randomly chosen. The initialization step 1 in Algorithm 1 is changed as follows:

$$X_1^* \leftarrow r,\ Z_1^* \leftarrow 0,\ X_2^* \leftarrow rx_P \text{ and } Z_2^* \leftarrow r.$$

It can be easily seen that in iteration i of the for-loop the algorithm computes the values

$$X_m^* = r^{2^{2i}} X_m \text{ and } Z_m^* = r^{2^{2i}} Z_m$$

for $m \in \{1, 2\}$.

This randomization destroys the correlation in the power profile between the scalar, the input or output of the computation, and intermediate results of the scalar computation. Single order differential power attacks are no longer effective.

The generalization of the attack in [10] is a higher order power analysis. If the device must be secured against this type of attack then the irreducible polynomial which defines the finite field and which is encoded in the reduction part of he arithmetic unit must have approximately $n/2$ monomials which should be equally distributed over $\{x^0, \ldots, x^n\}$. This countermeasure renders the arithmetic unit slightly more complicated.

4.3. Fault Analysis

Most techniques against fault attacks either repeat (at least part of) the cryptographic computation (or of the inverse operation) or introduce invariants in the implementation that must hold during the computation [5], [4]. When the cryptographic operation is done, the implementation checks whether the invariant is still valid. A fault during the operation destroys the invariant with high probability. Montgomery's ladder algorithm for scalar multiplication has an implicit invariant which holds in each round of the for-loop. An efficient implementation of the test whether this invariant holds can be found in [6].

If an attacker causes a fault prior to the initialization step of Montgomery's ladder algorithm then the fault tolerant strong elliptic curve protects the device. As shown before, with high probability the scalar multiplication is computed with a point of large order on E or E_t.

In order to protect the elliptic curve unit against errors in the long term non-volatile system parameters as curve parameter and secret key the developer must implement a scheme for error detection and verify the data prior to scalar multiplication. Otherwise an attacker could change

the defining elliptic curve of the scheme and extract the secret key of the tag using a weak curve.

5. Tag Authentication

In this section we present an authentication protocol based on the elliptic curve arithmetic proposed in the previous sections. In the first version the protocol offers authentication of the tag where no privacy aspects are considered. The corresponding protocol is depicted in Figure 4.

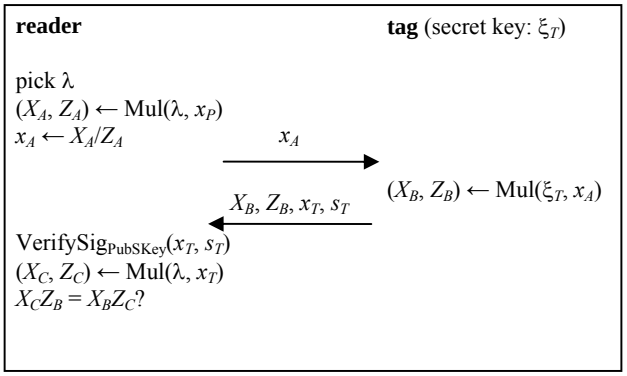

Fig. 4 Tag Authentication Protocol

5.1. Requirements

Let E be an elliptic curve over GF(2^n) satisfying the security properties described in Section 2. Let $P = (x_P, y_P)$ be a point on E with order q where q is a prime. Furthermore, let "GenSig" denote a public key signature generation algorithm and let "VerifySig" be the corresponding signature verification algorithm. A key pair consisting of a private key for the signature generation and public key for the signature verification is also given by "PrivSKey" and "PubSKey".

5.2. Tag Setup

Each tag is initialized with a randomly chosen private key $0 < \xi_T < q$ and a certificate (x_T, s_T) consisting of the corresponding public key x_T which is the affine x-coordinate of the point $T = \xi_T \cdot P$ and the signature s_T of x_T, generated with the signature generation algorithm "GenSig" using the private signature key "PrivSKey", i.e. $s_T = \text{GenSig}_{\text{PrivSKey}}(x_T)$.

5.3. Reader Setup

Each reader is initialized with the public signature key "PubSKey". No secret key has to be stored in the reader.

5.4. Interaction

The reader picks a randomly chosen value $0 < \lambda < q$ and computes the affine x-coordinate x_A of $A = \lambda \cdot P$ and sends this value to the tag. Upon reception of this challenge the tag runs the Montgomery's ladder algorithm $(X_B, Z_B) \leftarrow \text{Mul}(\xi_T, x_A)$ with the private key ξ_T as scalar and the challenge x_A as input for the affine x-coordinate. The result (X_B, Z_B) finally represents the affine x-coordinate X_B/Z_B of the point $B = \xi_T \cdot (\lambda \cdot P)$. After computation the tags sends the coordinates X_B and Z_B together with its certificate consisting of the public key x_T and the corresponding signature s_T back to the reader. Afterwards the reader verifies the certificate by calling $\text{VerfiySig}_{\text{PubSKey}}(x_T, s_T)$. If the certificate is invalid the tag will not be accepted. Otherwise the reader continues with the verification of the response. The reader calculates the projective coordinates X_C and Z_C of $C = \lambda \cdot (\xi_T \cdot P)$ and checks if the affine x-coordinates X_C/Z_C and X_B/Z_B are identical. This can be done by verifying the equation $X_C Z_B = X_B Z_C$. If the response is correct, the tag is accepted and declared to be authentic, otherwise the tag is rejected.

5.5. Security

The security of the protocol is based on the Elliptic Curve Diffie Hellman Problem (ECDHP). An attacker who wants to fake an authentic RFID tag gets the challenge $A = \lambda \cdot P$ from the reader and must return the corresponding valid response B together with an authentic public key $T = \xi_T \cdot P$ signed by the certification authority. Since the attacker only has $A = \lambda \cdot P$ and $T = \xi_T \cdot P$ without knowledge of λ and ξ_T he only can determine the correct response $B = \xi_T \cdot (\lambda \cdot P)$ if and only if he solves the ECDHP.

6. Privacy Enhanced Tag Authentication

The authentication protocol of the tag presented in the previous section can be extended to a privacy enhanced protocol, i.e. the protection of the location privacy and the forward location privacy is assured (see Figure 5).

The reader transmits the challenge together with its certificate, consisting of a public key and the corresponding signature to the RFID tag. The tag verifies the certificate, calculates the response, randomly encrypts that response and its certificate with the public key of the reader, and sends the cipher text back to the reader. Only the reader which has the corresponding private key can decrypt the message and hence can "understand" the tag. The crucial steps in that protocol are of course the certificate verification and the encryption.

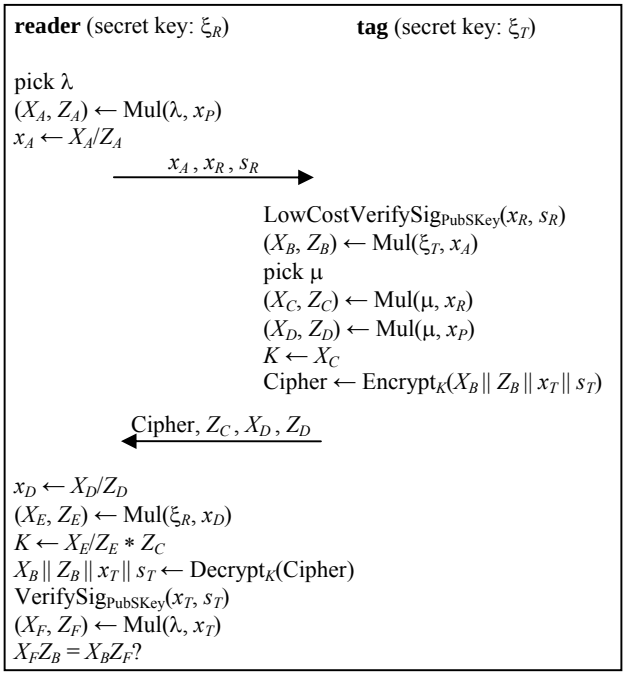

Fig. 4 Privacy Enhanced Tag Authentication Protocol

6.1. Signature Verification

The reader's certificate consists of its public key x_R and the corresponding signature s_R which may be an ECGDSA signature. The major task for the RFID tag is the verification of the signature. The signature verification can be reduced to a test whether the sum of two points resulting from scalar multiplications is equal to a point from a third scalar multiplication. This step can be implemented efficiently by evaluating a polynomial of degree 2. Following this approach we avoid expensive inversions in the field GF(2^n) and the computation of long integers modulo the order of a subgroup $< P >$. Let ξ_S be the private signature key "PrivSKey" and let x_S be the corresponding public signature key "PubSKey", i.e. x_S is the affine x-coordinate of the point S satisfying $\xi_S \cdot S = P$, where $P = (x_P, y_P)$ denotes the base point. Now the signature s_R of the reader's individual public key x_R is defined as $s_R = (\alpha, \beta)$ where the first component α is the affine x-coordinate of $k \cdot P$ for a randomly selected number k. Furthermore, the second component β satisfies the equation $\beta = \xi_S \cdot (k \cdot \alpha - x_R) \bmod q$ where q is the order of the base point.

In order to verify the certificate one has first to compute

$$U := (\beta \cdot \alpha^{-1}) \cdot S + (x_R \cdot \alpha^{-1}) \cdot P$$

and afterwards compare the affine x-coordinate of U to the signature component α. If both values are equal the signature is declared to be valid, otherwise it is invalid. This verification is equivalent to the test whether the equation

$$\alpha \cdot (k \cdot P) = \beta \cdot S + x_R \cdot P$$

holds. In order to verify this equation we evaluate a polynomial of degree 2. In the case of the signature verification we test the following equation

$$X_3^2 (X_1 Z_2 + X_2 Z_1)^2 + X_1 X_2 X_3 Z_1 Z_2 Z_3 + X_1^2 X_2^2 Z_3^2 + b Z_1^2 Z_2^2 Z_3^2 = 0$$

with corresponding input values

$$(X_1, Z_1) \leftarrow \text{Mul}(\alpha, \alpha),$$
$$(X_2, Z_2) \leftarrow \text{Mul}(x_R, x_P),$$
$$(X_3, Z_3) \leftarrow \text{Mul}(\beta, x_S).$$

6.2. Symmetric Encryption

In the privacy enhanced authentication protocol the tag computes a challenge X_D/Z_D for the reader and uses the corresponding response X_C/Z_C as a symmetric key for the encryption of the data which is transmitted to the reader. To avoid the inversion in the calculation of the affine x-coordinate $x_C = X_C/Z_C$ of the response the tag uses the value X_C as secret key and sends Z_C to the reader. Given the correct response to the challenge X_D/Z_D and the value Z_C the reader can also reconstruct the secret key chosen by the tag.

With the aid of the session key X_C the memory contents of the tag transmitted to the reader can be encrypted randomly to protect the privacy of the tag bearer. For this encryption step a symmetric encryption scheme must be implemented on the tag in addition to the hardware for the asymmetric scheme. It is possible to reuse parts of the arithmetic unit for the implementation of the symmetric encryption algorithm. For example, the linear feedback shift register of the accumulator can be extended to a stream cipher.

References

[1] T. Akishita and T. Takagi. Zero-value point attacks on elliptic curve cryptosystems. In C. Boyd and W. Mao, editors, Information Security — ISC 2003, volume 2851 of Lecture Notes in Computer Science, pages 218 – 233. Springer-Verlag, 2003.

[2] L. Batina, J. Guajardo, T. Kerins, N. Mentens, P. Tuyls and I. Verbauwhede. Public-key cryptography for RFID-tags. In Fourth IEEE International Workshop on Pervasive Computing and Communication Security — PerSec 2007, pages 217 – 222. IEEE, 2007.

[3] I. Biehl, B. Meyer and V. Müller. Differential fault

attacks on elliptic curve cryptosystems. In M. Bellare, editor, Advances in Cryptology — CRYPTO 2000, volume 1880 of Lecture Notes in Computer Science, pages 131 – 146. Springer-Verlag, 2000.

[4] J. Blömer, M. Otto and J.P. Seifert. Sign change faults on elliptic curve cryptosystems. Cryptology ePrint Archive, 2004/227.

[5] D. Boneh, R.A. De Millo and R.J. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, editor, Advances in Cryptology — EUROCRYPT 1997, volume 1233 of Lecture Notes in Computer Science, pages 37 – 51. Springer-Verlag, 1997.

[6] M. Braun, A. Kargl and B. Meyer. Efficient Fault Detection in Montgomery's Method for Scalar Multiplication. to appear.

[7] D.R.L. Brown and R.P. Gallant. The static Diffie-Hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004.

[8] M. Ciet, J.J. Quisquater and F. Sica. Hardware for collision search on elliptic curves over GF(2^m). In C. Pandu Rangan and C. Ding, editors, Progress in Cryptology — INDOCRYPT 2001, volume 2247 of Lecture Notes in Computer Science, pages 108 – 116. Springer-Verlag, 2001.

[9] J.S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In C.K. Koc and C. Paar, editors, Cryptographic Hardware and Embedded Systems — CHES 1999, volume 1717 of Lecture Notes in Computer Science, pages 292–302. Springer-Verlag, 1999.

[10] W. Dupuy and S. Kunz-Jacques. Resistance of randomized projective coordinates against power analysis. In J.R. Rao and B. Sunar, editors, Cryptographic Hardware and Embedded Systems — CHES 2005, volume 3659 of Lecture Notes in Computer Science, pages 1 – 14. Springer-Verlag, 2005.

[11] M. Feldhofer, S. Dominikus and J. Wolkerstorfer. Strong authentication for RFID systems using the AES algorithm. In M. Joye and J.J. Quisquater, editors, Workshop on Cryptographic Hardware and Embedded Systems — CHES 2004, volume 3156 of Lecture Notes in Computer Science, pages 357 – 370, Boston, Massachusetts, USA, August 2004. IACR, Springer-Verlag.

[12] P. Gaudry, F. Hess and N.P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. J. Crypt., 15(1): pages 19 – 46, 2002.

[13] J. Großschädl and E. Savas. Instruction set extensions for fast arithmetic infinite fields GF(p) and GF(2^m). In M. Joye and J.J. Quisquarter, editors, Cryptographic Hardware and Embedded Systems — CHES 2004, pages 133 – 147. Springer-Verlag, 2004.

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

[15] S. Kumar and C. Paar. Are standards compliant elliptic curve cryptosystems feasible on RFID? Printed handout of Workshop on RFID Security — RFIDSec 2006, July 2006.

[16] C.H. Lim and P.J. Lee. More flexible exponentiation with precomputation. In Y.G. Desmedt, editor, Advances in Cryptology — CRYPTO 1994, volume 839 of Lecture Notes in Computer Science, pages 95 – 107. Springer-Verlag, 1994.

[17] J. Lopez and R. Dahab. Fast multiplication on elliptic curves over GF(2^n) without precomputation. In C.K. Koc and C. Paar, editors, Cryptographic Hardware and Embedded Systems — CHES 1999, volume 1717 of Lecture Notes in Computer Science, pages 316 – 327. Springer-Verlag, 1999.

[18] A. Menezes and M. Qu. Analysis of the Weil descent attack of Gaudry, Hess and Smart. In D. Naccache, editor, Topics in Cryptology — CT-RSA 2001, volume 2020 of Lecture Notes in Computer Science, pages 308 – 318. Springer-Verlag, 2001.

[19] D. Molnar and D. Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In B. Pfitzmann and P. Liu, editors, Conference on Computer and Communications Security — ACMCCS, pages 210 – 219, Washington, DC, USA, October 2004. ACM, ACM Press.

[20] M. Ohkubo, K. Suzuki and S. Kinoshita. Cryptographic approach to "privacy-friendly" tags. In RFID Privacy Workshop, MIT, MA, USA, November 2003.

[21] T. Okamoto. Provably secure and practical identifications schemes and corresponding signature schemes. In E.F. Brickell, editor, Advances in Cryptology — CRYPTO 1992, volume 740 of Lecture Notes in Computer Science, pages 31 – 53. Springer-Verlag, 1992.

[22] K. Rhee, J. Kwak, S. Kim and D. Won. Challenge-response based RFID authentication protocol for distributed database environment. In D. Hutter and M. Ullmann, editors, International Conference on Security in Pervasive Computing — SPC 2005, volume 3450 of Lecture Notes in Computer Science, pages 70 – 84, Boppard, Germany, April 2005. Springer-Verlag.

[23] S. Sarma, S. Weis and D. Engels. RFID systems and security and privacy implications. In B. Kaliski, C.K. Koc and C. Paar, editors, Cryptographic Hardware and Embedded Systems — CHES 2002, volume 2523 of Lecture Notes in Computer Science, pages 454 – 469, Redwood Shores, CA, USA, August 2002. Springer-Verlag.

[24] S. Weis, S. Sarma, R. Rivest and D. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W.

Stephan and M. Ullmann, editors, International Conference on Security in Pervasive Computing — SPC 2003, volume 2802 of Lecture Notes in Computer Science, pages 454 – 469, Boppard, Germany, March 2003. Springer-Verlag.

Michael Braun received his Diploma degree in Mathematics in 2002 and the PhD degree in 2003 from the University of Bayreuth. Since 2004 he works at the security department of Corporate Technology at Siemens AG.

Erwin Hess studied mathematics at the University of Heidelberg and received a PhD in mathematics from the University of Heidelberg in 1978. From 1979 to 1986 he was a faculty member of the Technical University of Braunschweig. Since 1987 he has been with Siemens Corporate Technology in Munich where he is Principal Research Scientist and head of the cryptography research.

Bernd Meyer received the MS degree in Computer Science in 1992 and the PhD degree in 1996 from Universität des Saarlandes in Saarbrücken. Since 1997 he works in the department "Security" of Corporate Technology at Siemens AG.