An efficient mobile PACE implementation

Johannes Buchmann,
Alex Wiesmaier

Center for Advanced Security
Research CASED
Mornewegstraße 32
64293 Darmstadt, Germany
{buchmann,wiesmaier}
@cased.de

Johannes Braun,
Moritz Horsch,
Franziskus Kiefer

TU Darmstadt
Hochschulstraße 10
64289 Darmstadt, Germany
{jbraun,horsch|kiefer}
@cdc.informatik.tu-
darmstadt.de

Detlef Hühnlein

ecsec GmbH
Sudetenstrasse 16
96247 Michelau, Germany
detlef.huehnlein@ecsec.de

Falko Strenzke

Flexsecure GmbH
Industriestr. 12
64297 Darmstadt, Germany
strenzke@flexsecure.de

ABSTRACT

Many future electronic identity cards will be equipped with a contact-less interface. Analysts expect that a significant proportion of future mobile phones support Near Field Communication (NFC) technology. Thus, it is a reasonable approach to use the cell phone as mobile smart card terminal, which in particular supports the Password Authenticated Connection Establishment (PACE) protocol to ensure user consent and to protect the wireless interface between the mobile phone and the smart card. While there are efficient PACE implementations for smart cards, there does not seem to be an efficient and platform independent solution for mobile terminals. Therefore we provide a new implementation using the Java Micro Edition (Java ME), which is supported by almost all modern mobile phones. However, the benchmarks of our first, straightforward PACE implementation on an NFC-enabled mobile phone have shown that improvement is needed. In order to reach a user friendly performance we implemented an optimized version, which, as of now, is restricted to optimizations which can be realized using features of existing Java ME libraries.

In the work at hand we present a review of the relevant algorithms and provide benchmarks of the corresponding arithmetic functions in different Java ME libraries. We discuss the different optimization approaches, introduce our optimized PACE implementation, and provide timings for a desktop PC and a mobile phone in comparison to the straightforward version. Finally, we investigate potential side channel attacks on the optimized implementation.

Keywords

Extended Access Control (EAC), European Citizen Card (ECC), Efficient Exponentiation, eID, Mobile Devices, Near Field Communication (NFC), Password Authenticated Connection Establishment (PACE), Java 2 Mobile Edition (J2ME)

1. INTRODUCTION

Many countries around the world have engaged in the deployment of electronic identity (eID) cards [2,9,29]. The European Committee for Standardization (Comité Européen de Normalisation, CEN) is currently developing the technical standard series CEN prTS 15480 [7], which defines services, command sets, application interfaces and profiles for European Citizen Cards. [7, Part 4, Annex A] specifies card application templates and there is a profile, which specifies an "eID application with mandatory ICAO functionality and conditional digital signature functionality". This profile will be implemented for example by the forthcoming German eID card [10] and may serve as blueprint for other citizen cards, which also provide machine readable travel document functionality as specified by the International Civil Aviation Organization (ICAO) [15–17]. Such citizen cards will in particular provide a contact-less interface according to [19] and support version 2 of the Extended Access Control (EAC) protocol according to [11]. Because analysts expect [23] that a significant proportion of future mobile phones will be equipped with Near Field Communication (NFC) technology [21,22], it is worth to investigate whether NFC-enabled phones may serve as smart card terminals for mobile European Citizen Card applications.

The EAC protocol may in particular be used together with the Password Authenticated Connection Establishment (PACE) [11, Section 4.2] protocol. PACE ensures user consent and protects the wireless channel between the mobile phone and the smart card. It is specified in a way that allows implementing it using different cryptographic primitives. The work at hand focuses on PACE realized using Elliptic Curve Cryptography (ECC), especially on the version used in the forthcoming German eID card.

There are already very efficient hardware specific PACE implementations for low power devices written in C and Assembler [34]. In order to support a maximum of mobile devices without depending on special hardware or operating systems we provide a Java Micro Edition (Java ME) implementation of PACE. But while our reference Java implementation of PACE, which is straightforward and contains no optimizations, performs well on typical desktop systems (cf. Table 9), the efficient implementation on a mobile phone turns out to be a challenging task. Various optimizations are required in order to come up with a user friendly performance.

The optimizations investigated in the work at hand are restricted to those which can be realized using features existing libraries for Java ME provide. Adaptations of algorithms to the special needs of the PACE protocol and their implementation are subject to ongoing work.

The paper is structured as follows. Section 2 provides the necessary background on the PACE protocol. Section 3 gives an overview of efficient elliptic curve (EC) point multiplication algorithms including the number of involved point additions and doublings. In Section 4 we discuss different approaches to optimize the PACE implementation. Section 5 provides benchmarks for large number arithmetic in different Java ME libraries. It presents the main aspects of the optimized implementation and compares the resulting timings to those of the reference implementation. This is done separately for different Cryptographic Service Providers (CSP) as well as on desktop PC and cell phone. Section 6 addresses potential side channel vulnerabilities of the optimized implementation. Section 7 gives an outlook on future work and further improvements currently restricted by hard- or software limitations. Section 8 finally summarizes the presented work and concludes the paper.

2. BACKGROUND ON PACE

PACE was developed by the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI) and is designed to be free of patents. Security analyses of the PACE protocol can be found in [3, 34]. From this point on we use the terms and notation introduced in [11]. There, the term for a contact-less smart card is Proximity Integrated Circuit Card (PICC) and the contact-less smart card terminal is called Proximity Coupling Device (PCD).

Besides other things, PACE establishes a shared session key between the PICC and the PCD using the well known Diffie-Hellman (DH) key agreement protocol [8]. For elliptic curves this works as follows: Two parties A and B want to agree on a shared key. Each party selects a secret scalar d_{A} and d_{B} respectively. Using a public EC point Q then A computes $Q_{\mathrm{A}} = d_{\mathrm{A}} * Q$ and B computes $Q_{\mathrm{B}} = d_{\mathrm{B}} * Q$. After exchanging the results, A computes $Q_{\mathrm{AB}} = d_{\mathrm{A}} * Q_{\mathrm{B}}$ and B computes $Q_{\mathrm{AB}} = d_{\mathrm{B}} * Q_{\mathrm{A}}$. The common key is the x-coordinate of Q_{AB}.

As the DH key agreement does not support authentication of the communication partners it is vulnerable to man-in-the-middle attacks. In order to prevent this and to ensure user consent, PACE uses a password-based protocol (see [4, Section 7] for similar protocols of this type) to protect the wireless communication interface between the PCD and the PICC before the PICC is accessed. In the most common scenario the password π is a Personal Identification Number (PIN), which is permanently stored in the PICC and is entered into the PCD by the user. As the password is used during the calculation of the session key, entering a wrong password leads to different session keys on both sides which causes the connection establishment to fail.

	PICC		PCD
(a)	$K_\pi = \mathrm{KDF}_\pi(\pi)$		$K_\pi = \mathrm{KDF}_\pi(\pi)$
(b)	$z = \mathrm{E}(K_\pi, s)$	$\xrightarrow{z}$	$s = \mathrm{D}(K_\pi, z)$
(c)	$Y = y \cdot G$	$\xleftarrow{X}$ $\xrightarrow{Y}$	$X = x \cdot G$
(d)	$H = y \cdot X$		$H = x \cdot Y$
(e)	$G' = s \cdot G + H$		$G' = s \cdot G + H$
(f)	$\widetilde{PK}_{\mathrm{PICC}} = \widetilde{SK}_{\mathrm{PICC}} \cdot G'$	$\xleftarrow{\widetilde{PK}_{\mathrm{PCD}}}$ $\xrightarrow{\widetilde{PK}_{\mathrm{PICC}}}$	$\widetilde{PK}_{\mathrm{PCD}} = \widetilde{SK}_{\mathrm{PCD}} \cdot G'$
(g)	$K = \widetilde{SK}_{\mathrm{PICC}} \cdot \widetilde{PK}_{\mathrm{PCD}}$		$K = \widetilde{SK}_{\mathrm{PCD}} \cdot \widetilde{PK}_{\mathrm{PICC}}$
(h)	$K_{\mathrm{ENC}} = \mathrm{KDF}_{\mathrm{ENC}}(K)$		$K_{\mathrm{ENC}} = \mathrm{KDF}_{\mathrm{ENC}}(K)$
(i)	$K_{\mathrm{MAC}} = \mathrm{KDF}_{\mathrm{MAC}}(K)$		$K_{\mathrm{MAC}} = \mathrm{KDF}_{\mathrm{MAC}}(K)$
(j)	$T_{\mathrm{PICC}} = \mathrm{MAC}(K_{\mathrm{MAC}}, \widetilde{PK}_{\mathrm{PCD}})$	$\xleftarrow{T_{\mathrm{PCD}}}$ $\xrightarrow{T_{\mathrm{PICC}}}$	$T_{\mathrm{PCD}} = \mathrm{MAC}(K_{\mathrm{MAC}}, \widetilde{PK}_{\mathrm{PICC}})$

Figure 1: PACE [11, Chapter 4.2]

Figure 1 provides an overview of the steps of the PACE protocol. Before the protocol starts the PCD needs to read the domain parameters $\mathcal{D}$ from the PICC, which contain a common base point G and related ECC[1] parameters. The values x, y, $\widetilde{SK}_{\mathrm{PCD}}$ and $\widetilde{SK}_{\mathrm{PICC}}$ are uniformly chosen random numbers smaller than the order r of the elliptic curve.

1. As depicted in step (b), the PICC chooses a nonce s uniformly at random and encrypts it using the encryption function $\mathrm{E}(key, \cdot)$ with the key K_π derived in step (a) from the shared password π. The PCD decrypts the cipher text using the decryption function $\mathrm{D}(key, \cdot)$ and the same key K_π to obtain the nonce s.

2. Now both parties use s to generate new domain parameters $\mathcal{D}' = \mathbf{Map}(\mathcal{D}, s)$, which in particular contain a new common base point G', which is used for the subsequent Diffie-Hellman key agreement. There are two options for the implementation of the $\mathbf{Map}$-function:

    - *Generic Mapping*
      The Generic Mapping [11, A.3.4.1] shown in steps (c) – (e) already has been supported in version 1 of the PACE protocol. It is used by the German eID card, which is issued since November 2010. The new base point G' is given as $G' = s \cdot G + H$, where H is agreed upon by the two communication partners in an anonymous Diffie-Hellman key agreement.
    - *Integrated Mapping*
      In Integrated Mapping [18], which is only supported by version 2 of the PACE protocol, the new base point G' is computed as $G' = f(R(s \parallel t))$. Where t is chosen by the PCD and sent to the PICC, the pseudo-random function R is specified in [30] and the function f will most likely be based on [14] and [31]. Focusing on the German eID card which uses the *Generic Mapping* by

[1]The specification of the PACE protocol in [11] also covers multiplicative groups over finite fields. Here we present our results for groups of points on elliptic curves. The results can be applied to multiplicative groups analogously.

now, we do not investigate the *Integrated Mapping* any further in the present work.

3. As shown in steps (f) – (g), the PICC and the PCD respectively choose an ephemeral private key ($\widetilde{SK}_{\text{PICC}}$, $\widetilde{SK}_{\text{PCD}}$) uniformly at random and perform a Diffie-Hellman key agreement based on G'. Both calculate a common secret point K.

4. Steps (h) and (i) depict how the keys K_{ENC} for message encryption and K_{MAC} for message authentication are derived from the common secret K using the key derivation functions KDF_{MAC} and KDF_{ENC} [11].

5. In step (j) both parties calculate a token (T_{PICC}, T_{PCD}) with the use of a MAC-function $\text{MAC}(key, \cdot)$ which is a keyed hash computation using the key K_{MAC}. These tokens are then used to perform a mutual key confirmation.

3. OPTIMIZED POINT MULTIPLICATION

Since the point multiplication is one of the biggest run-time consumers (cf. Section 5.1) in the PACE protocol, it promises the most potential for optimization. As an efficient point multiplication algorithm is the basis for an efficient implementation, we present the most common multiplication algorithms in this chapter. Our analysis of the point multiplication algorithms is based on [12] and focuses mainly on the complexity of given algorithms. As we do not change existing algorithms, we do not go into details how the given algorithms work.

The complexity evaluation and the notation of the algorithms in this section is according to [12] and for the ECC and PACE parameters employed by the upcoming German eID card. The relevant values are: $\lambda(r) = \lambda(x) = 256$; $\lambda(s) = 128$.

We denote the length of a scalar e by $\lambda(e)$ as the number of binary digits. The cost of the group operation (addition) "$+$" is denoted by ADD (A). The cost of the scalar multiplication "$\cdot$" is denoted by MULT (M), the cost of point doubling by DBL (D) and the cost of an interleaved multiplication $(e_1 \cdot P_1 + e_2 \cdot P_2)$ by PSUM. Note that in general MULT $<$ PSUM $<$ 2 MULT holds if the involved scalars and points have similar magnitudes respectively.

3.1 Review of Basic Elliptic Curve Single Point Multiplication Techniques

A variety of algorithms can be used to implement the single point multiplication on an elliptic curve. Compared to the most basic algorithm, the left-to-right binary method, the more sophisticated algorithms employ time-memory trade offs to speed up the computation. Table 1 gives an overview of the complexity of available algorithms. The formulas used to determine the complexity are shown in Table 2.

The left-to-right binary method processes the bits of e from left to right one at a time. The internal intermediate result is initialized with the point at infinity. Each step doubles the current intermediate result and adds P each time a nonzero digit is processed. In the NAF methods e is transformed into the non-adjacent form (NAF) at first to reduce the density of nonzero digits to $^1/_3$ on average, therewith reducing the number of point additions. In the window methods multiples of P are precomputed and stored in additional memory. Then digits of e are scanned w at a time and the according multiple of P is taken from the precomputed points and added. This further reduces the number of ADDs.

algorithm	parameter choices	complexity	#precomp. points
left-to-right binary	-	$128A + 256D$	-
NAF	-	$85A + 256D$	-
w-NAF	$w = 5$	$50A + 257D$	7
Sliding Window NAF	$w = 5$	$50A + 257D$	10

Table 1: Single point multiplication. The window parameters w are chosen to be optimal for the 256-bit curves. The given complexities in DBL (D) and ADD (A) operations are approximate.

algorithm	complexity formula	#precomp. points
left-to-right binary	$\frac{\lambda(e)}{2}\text{A} + \lambda(e)\text{D}$	-
NAF	$\frac{\lambda(e)}{3}\text{A} + \lambda(e)\text{D}$	-
w-NAF	$(2^{w-2} - 1 + \frac{\lambda(e)}{w+1})\text{A} + (\lambda(e) + 1)\text{D}$	$2^{w-2} - 1$
Sliding Window NAF	$(\frac{2^w-(-1)^w}{3} - 1 + \frac{\lambda(e)}{w+v(w)})\text{A} + (\lambda(e) + 1)\text{D}$ $v(w) = \frac{4}{3} - \frac{(-1)^w}{3 \cdot 2^{w-2}}$	$\frac{2^w-(-1)^w}{3} - 1$

Table 2: Complexity evaluation formulas of unknown point multiplication algorithms [12]

The RAM demands of the methods given in Table 1 are maximal for the Sliding Window NAF method. Here 10 additional points (and the two we also have in the left-to-right method) need to be stored. This sums up to 768 bytes (computed as $12 \cdot 2 \cdot 32$). Note that the precomputations in Table 1 are dynamic precomputations, i.e. have to be done each time a scalar multiplication is conducted.

If the multiplication involves a known point, off-line precomputations can be used to spare point doublings.[2] This can be done for all algorithms listed in Table 1. For each doubling operation to be spared in the non-window algorithms one additional point has to be stored in the device's non-volatile memory (NVM). This sums up to 16.384 bytes to store all 256 additional points, which is not a problem on mobile phones. Note that these precomputed values can be considered public information and do not need to be kept secret. They need, however, integrity protection to avoid their manipulation.

Table 3 gives the complexity of basic algorithms used with off-line precomputation and the number of points to be stored in the NVM. Table 4 provides the complexity for the method based on the exponentiation algorithm due to Lim and Lee [26], including the

[2]In the future (some) precomputed points might be saved on the eID cards together with the base point, assumed that the space and bandwidth constraints will allow for that.

algorithm	parameter choices	complexity evaluation	#precomp. points
left-to-right binary	-	$128A$	256
Windowing	$w = 4$	$77A$	64
Windowing NAF	$w = 5$	$71A$	52

Table 3: Fixed point multiplication with off-line precomputation in terms of expected point additions (A). The window parameters w are chosen to be optimal for the 256-bit curves.

parameter w	parameter v	complexity evaluation	complexity incl. precomp.	#precomp. points
4	1	$64A + 63D$	$71A+255D$	15
4	2	$64A + 31D$	$78A+255D$	30
5	1	$51A + 50D$	$66A+255D$	31
5	2	$51A + 25D$	$81A+255D$	62

Table 4: Fixed point multiplication with off-line precomputation according to Lim and Lee in terms of expected point additions (A) and doublings (D). The parameters w and v are chosen to obtain a good trade off between total complexity and complexity in the evaluation phase for 256-bit curves.

total complexity if precomputation is done on-line. The formulas used for complexity evaluation are given in Table 5.

The main idea of the method by Lim and Lee is to divide e into w bit strings of the same length and to process them parallelly as different exponents comparable to multiple point multiplication with shorter exponents. An additional parameter v specifies how the different exponents are further partitioned and can also be seen as the number of lookup tables containing the precomputed points [12].

In cases with no off-line precomputation the method by Lim and Lee can nearly compete with the window NAF methods. In cases with off-line precomputation the method by Lim and Lee performs significantly better than the window NAF methods.

3.2 Review of Basic Elliptic Curve Multiple Point Multiplication Techniques

There are several methods available [12] which evaluate product sums $k \cdot P + l \cdot Q$ much more efficient than carrying out the individual multiplications sequentially and add the results. The complexity and the number of stored points for different approaches can be seen in Table 6. The respective complexity formulas are given in Table 7.

While in the first three algorithms addition and doubling operations are done simultaneously, interleaving only does the doubling simultaneously. In our case the latter has some advantages. Firstly, the precomputed points rely on the points P and Q only. This allows storing precomputed points for further use. Secondly, the method allows for exponents with different lengths. Both is useful in our PACE scenario.

4. OPTIMIZED PACE

In this section we discuss the most promising optimizations for a PACE implementation on cell phones. The stated amounts of ADD and DBL are derived from the findings from Section 3 and express the average case. The proposed optimizations all lead to very sim-

algorithm	complexity formula	#precomp. points
left-to-right binary	$\frac{\lambda(e)}{2}$A	$\lambda(e)$
Windowing	$(2^w + \lceil\frac{\lambda(e)}{w}\rceil - 3)$A	$\lceil\frac{\lambda(e)}{w}\rceil$
Windowing NAF	$(\frac{2^{w+1}}{3} + \lceil\frac{\lambda(e)+1}{w}\rceil - 2)$A	$\lceil\frac{\lambda(e)+1}{w}\rceil$
LimLee	precomp: $v(2^{w-1}-1)$A$+(\lambda(e)-\frac{\lambda(e)}{wv})$D main: $\frac{\lambda(e)}{w}$A + $(\frac{\lambda(e)}{wv} - 1)$D	$(2^w - 1) \cdot v$

Table 5: Complexity evaluation formulas of algorithms with off-line precomputation [12,13]

algorithm	parameter choices	complexity	#precomp. points
Simultaneous mult. point	$w = 2$	$128A + 256D$	15
Simultaneous slid. window	$w = 2$	$118A + 256D$	12
Simultaneous Joint Sparse Form (JSF)	$-$	$129A + 255D$	3
w-NAF Interleaving $\lambda(e_1) = \lambda(e_2) = 256$	$w_1 = w_2 = 5$	$99A + 258D$	16
w-NAF Interleaving $\lambda(e_1) = 256,\ \lambda(e_2) = 128$	$w_1 = w_2 = 5$	$78A + 258D$	16

Table 6: Multiple point multiplication in terms of expected point additions (A) and doublings (D). The window parameters w are chosen to be optimal for 256-bit curves.

ilar theoretic execution times. Which one is the best in practice seems to depend on the actual hardware and the respective environmental circumstances. As we have seen in Section 3, storage space for points is not an issue on modern mobile devices, and is therefore not considered here. Likewise the time to load these points can be assumed to be negligible.

Following the PACE specification (cf. Figure 1) the PCD conducts the EC computations shown in Equations (1) to (5). Counting the PCD's EC computations in this notation leads to a total of 1 ADD and 5 MULT.

$$\begin{aligned} X &= x \cdot G &(1)\\ H &= x \cdot Y &(2)\\ G' &= s \cdot G + H &(3)\\ \widetilde{PK}_{\mathrm{PCD}} &= \widetilde{SK}_{\mathrm{PCD}} \cdot G' &(4)\\ K &= \widetilde{SK}_{\mathrm{PCD}} \cdot \widetilde{PK}_{\mathrm{PICC}} &(5) \end{aligned}$$

4.1 Rearranging Equations

When rearranging the equations we have to regard the following facts: X must be sent to the PICC in order to receive Y, and therefore it must be computed before H, G', $\widetilde{PK}_{\mathrm{PCD}}$ and K. $\widetilde{PK}_{\mathrm{PCD}}$ must be sent to the PICC in order to receive $\widetilde{PK}_{\mathrm{PICC}}$ and there-

algorithm	**complexity formula**	**stored points**
Simultaneous mult. point	$(3 \cdot 2^{2(w-1)} - 2^{w-1} - 1 + \frac{2^{2w-1}}{2^{2w}} \left\lceil \frac{\lambda(e)}{w} \right\rceil - 1)\mathrm{A} + (2^{2(w-1)} - 2^{w-1} + (\left\lceil \frac{\lambda(e)}{w} \right\rceil - 1)w)\mathrm{D}$	$2^{2w} - 1$
Simultaneous slid. window	$(3 \cdot 2^{2(w-1)} - 2^{w-1} - 1 + \frac{\lambda(e)}{w+1/3})\mathrm{A} + (2^{2(w-1)} - 2^{w-1} + (\left\lceil \frac{\lambda(e)}{w} \right\rceil - 1)w)\mathrm{D}$	$2^{2w} - 2^{2(w-1)}$
Simultaneous Joint Sparse Form (JSF)	$(1 + \frac{\lambda(e)}{2})\mathrm{A} + (\lambda(e) - 1)\mathrm{D}$	3
w-NAF Interleaving	$(\sum_j (2^{w_j-2} - 1) + \sum_j \frac{\lambda(e_j)}{w_j+1})\mathrm{A} + (\lvert j : w_j > 2 \rvert + max_j \lambda(e_j))\mathrm{D}$	$\sum_j (2^{w_j-2})$

Table 7: Complexity evaluation formulas of multiple point multiplication algorithms [12]

fore it must be computed before K. In addition to Equation 3, G' is needed in the last PACE step (j), because it is part of the TLV-object, which serves as input for the computation of T_{PCD}. Thus, G' should be a result or be computable from the result without increasing the overall computational cost. We consider the following possibilities for merging the equations.

α: Standard Interleaving. One possible optimization is to merge Equations (2) and (3) as shown in Equation (6) to compute G' using interleaved multiplication. X, $\widetilde{PK}_{\mathrm{PCD}}$, and K are computed as shown in Equations (1), (4), and (5). This leads to an overall computation cost of 3 MULT and 1 PSUM. Using the best available algorithm and parameters, w-NAF Interleaving with $w = 5$, PSUM needs 78 ADD and 258 DBL (cf. Table 6). This is a vast improvement compared to 2 MULT evaluated as sequential single point w-NAF multiplications, which sum up to 78 ADD and 386 DBL (according to Table 2). A possible shortcoming is that s, which is only half the length of x, is forced into a scalar multiplication of the dimension of the size of x. Using single point multiplication this is clearly considered.

$$G' \quad = \quad s \cdot G + x \cdot Y \tag{6}$$

β: Short Scalar Interleaving. The shortcoming of the above Equation (6) regarding the interleaved multiplication with s and x may be countered using the following trick: We know that the size of s is half the size of x. We break the pair (x, Y) into two pairs (x_1, Y_1) and (x_2, Y_2) as is done when using the LimLee method. The scalars x_1, x_2, and s now have the same size and the pair (s, G) can be added as third summand to the pairs (x_1, Y_1) and (x_2, Y_2) resulting in a standard interleaved multiplication with three summands as shown in Equation (7). As with Equation (6), X, $\widetilde{PK}_{\mathrm{PCD}}$, and K are computed as shown in Equations (1), (4), and (5). This leads to an overall computation cost of 3 MULT and 1 PSUM where the scalars in the PSUM are half the size of those of the PSUM of Equation (6). Using w-NAF Interleaving with $w = 5$ PSUM needs 85 ADD and 131 DBL according to Table 7 and additionally 128 DBL to prepare Y_2. In total this is nearly the same as the PSUM costs resulting from Equation (6) and thus, is also a vast improvement compared to the not optimized version.

$$G' \quad = \quad s \cdot G + x_1 \cdot Y_1 + x_2 \cdot Y_2 \tag{7}$$

γ: Inversion. A third possibility is to merge Equations (2), (3), and (4) as shown in Equation (8) to compute $\widetilde{PK}_{\mathrm{PCD}}$ using interleaved multiplication. G' then has to be computed by a scalar multiplication with an inverse as shown in Equation (9). X and K are computed as shown in Equations (1) and (5). This also leads to an overall computation cost of 3 MULT and 1 PSUM. w-NAF Interleaving leads to a cost of 99 ADD and 258 DBL (see Table 6). The necessary inversion and multiplications of scalars modulo the order r of the elliptic curve are neglectable in comparison to the operations on the points. A possible shortcoming is that the optimization potential of having an s of half the size of all other involved scalars is given away by multiplying it with a full size scalar. A possible advantage of this approach is that the computation of G', which is not needed again until the last step of PACE is reached, can be delayed until then.

$$\widetilde{PK}_{\mathrm{PCD}} \quad = \quad (\widetilde{SK}_{\mathrm{PCD}} \cdot s) \cdot G + (\widetilde{SK}_{\mathrm{PCD}} \cdot x) \cdot Y \tag{8}$$
$$G' \quad = \quad (\widetilde{SK}_{\mathrm{PCD}}^{-1} \bmod r) \cdot \widetilde{PK}_{\mathrm{PCD}} \tag{9}$$

δ: Splitted Inversion. An also promising possibility to calculate $\widetilde{PK}_{\mathrm{PCD}}$ is to reuse the precalculated points of G (see also Section 4.3) and evaluate Equation (10) with the LimLee method. Another advantage of this method is the possibility to start the calculation of Equation (10) right after evaluating the first equation of the PACE algorithm (cf. Figure 1). Equation (11) and (13) are calculated with a standard w-NAF multiplication. Assumed the precomputed points or LimLee multiplication are available, the evaluation of Equation (10) leads to 51 ADD and 25 DBL (cf. Table 4). The evaluation of Equation (11) results to 50 Add and 257 DBL (cf. Table 1). In total the Equations (10) to (12) that replace the PSUM in approach γ lead to 102 ADD and 282 DBL.

$$\widetilde{PK}_1 \quad = \quad (\widetilde{SK}_{\mathrm{PCD}} \cdot s) \cdot G \tag{10}$$
$$\widetilde{PK}_2 \quad = \quad (\widetilde{SK}_{\mathrm{PCD}} \cdot x) \cdot Y \tag{11}$$
$$\widetilde{PK}_{\mathrm{PCD}} \quad = \quad \widetilde{PK}_1 + \widetilde{PK}_2 \tag{12}$$
$$G' \quad = \quad (\widetilde{SK}_{\mathrm{PCD}}^{-1} \bmod r) \cdot \widetilde{PK}_{\mathrm{PCD}} \tag{13}$$

4.2 Tweaked Implementation

As seen above significant improvements are possible by applying well chosen multiplication algorithms and by rearranging the equations to be solved. However, there are possibilities for improvement concerning the implementation itself. In the following we consider possibilities to speed up the protocol execution taking into account at which points dedicated results have to be available and by avoiding overhead due to implementation techniques.

Reorganization. Another possibility to speed up the calculations is the reorganization of their order. We present two different kinds of reorganizations.

One possibility is to put some calculations at the beginning of the protocol before the initial contact with the eID card. To do so we need the domain-parameters of the used eID card saved on our device. These parameters can be stored at the first contact and can be used henceforward. The generation of the private keys x and $\widetilde{SK}_{\text{PCD}}$ as well as the computation of the first ephemeral key X (Equation (1)) could be done before actually starting the protocol. Additionally multiples of the base-point G could be stored in the NVM and later reused within the point multiplications. However, this is currently not supported by the existing J2ME-libraries.

Another kind of reorganization regards the scheduling of the tasks and calculations during the protocol. Using threads for communication with the eID card allows us to use the time waiting for an answer from the card or the user for further calculations. The decryption of s can be threaded for instance, since the result will be required later in step (e) (cf. Figure 1).

Streamlining. To obtain even more performance improvement we avoid heavy Java objects whenever possible. Instead, most values are represented by basic data structures to avoid the expensive object creation and handling which is particularly important on resource constrained environments such as mobile phones.

4.3 Multiplication Algorithms

In the following three paragraphs we describe how to speed up the 5 necessary point multiplications from Equations (1) and (5) to (9) using the findings from Section 3.

Equation (1). In Section 4.1 we stated that Equation (1) has to be solved before all other equations. Furthermore, with the random number x and the fix base-point G that can be permanently stored after the first contact with the eID card the LimLee multiplication algorithm performs best according to Table 4. This is quite clear as the values precomputed during the LimLee algorithm can be stored permanently on the mobile phone and can be reused to speed up the calculation. Therfore we choose LimLee multiplication to evaluate Equation (1) $X = x \cdot G$. Therewith the computation can be started in a low priority thread before the PIN is entered or even before the mobile device has contact to the eID card.

Equations (4) & (6) / (4) & (7) / (8) & (9). For the calculation of G' and $\widetilde{SK}_{\text{PCD}}$ we have different possibilities according to the rearranged equations explained in Section 4.1. For the calculations of the product-sum of Equation (6) or alternatively Equation (7) we use interleaving which performs best with Y randomly chosen every session (cf. Table 6). As G' therewith is different in each session the additional point multiplication with $\widetilde{SK}_{\text{PCD}}$ in Equation (4) is done with a simple w-NAF multiplication (cf. Table 1). Alternatively, $\widetilde{PK}_{\text{PCD}}$ could be implemented using Equation (8). The evaluation of the product-sum is implemented using an interleaving multiplication algorithm again based on the performance calculations in Table 6. The necessary additional calculation of G' with Equation (9) is done with a w-NAF multiplication again. Here we implement all four variants $\alpha - \delta$ to compare them with each other as the theoretical results can not decide exactly which one is the best.

Equation (5). Since $\widetilde{PK}_{\text{PICC}}$ changes every session, off-line precomputations and permanent storage of precomputed points are not possible for the point multiplication from Equation (5). According to Table 1 the w-NAF algorithm performs best in that case and therefore is chosen for the calculation.

5. IMPLEMENTATION

We implement the PACE protocol as a Java ME application and test it on the mobile phone Nokia 6212[3]. The connection between the PICC and the PCD (mobile phone) is enabled by the NFC interface and the data transmission proceeds by using Application Protocol Data Units (APDU) specified in ISO/IEC 7816 [20] and TR-03110 [11]. For the required cryptographic functions, which are out of scope of the Java ME platform, we use external cryptographic libraries such as FlexiProvider[4] and Bouncy Castle[5]. These two currently seem to be the only available cryptographic libraries (Cryptographic Service Providers, CSP) for elliptic curve arithmetic on mobile devices. Other providers like IAIK[6] cannot perform elliptic curve cryptography in a mobile environment or are just out of date (e.g. Cryptix[7]). The CSPs are used for the key derivation, decryption of the nonce s, the map function (Generic Mapping), and the key generation. The domain parameters $\mathcal{D}$ (stored in the file EF.CardAccess on the PICC) include a set of Security Infos [11], which defines the ECC domain parameters, the common base point G, and the supported encryption algorithm. The data is encoded in Abstract Syntax Notation One (ASN.1) data structures as specified in [11]. Bouncy Castle includes an ASN.1 parser to decode the data structures, the FlexiProvider uses the CoDec[8] library developed by Fraunhofer IGD.

5.1 Reference Tests

In a first approach, we implemented the PACE protocol in a straightforward manner without any improvements or optimizations described in Section 4. The measurements of the performance tests are depicted in Table 8. Step (1) shows the values for selecting and reading the file EF.CardAccess (Select File and Read Binary command [20]), and decoding the ASN.1 data structures. Step (2) shows the initialization of PACE by using the MSE:Set AT command [11]. Steps (b) to (j) correspond to the steps depicted in Figure 1. The key derivation of the step (a), (h) and (i) is a single hash computation, therefore the cost can be disregarded and we skip these steps in Table 8 and 9. The value zero indicates that the operation takes less than one millisecond and the values of the PICC include the transmission time of the APDUs in both directions. The column "Total" also includes values, which were caused

[3]Currently (Dec. 2010) the Nokia 6212 is the only NFC-enabled cell phone available in Germany.

[4]`http://www.flexiprovider.de`

[5]`http://www.bouncycastle.org`

[6]`http://jce.iaik.tugraz.at`

[7]`http://www.cryptix.org`

[8]`http://codec.sourceforge.net`

		Bouncy Castle		FlexiProvider		
	Details	PCD	Total	PCD	Total	PICC
1		424	676	274	530	256
2		459	1266	597	1261	134
b	$\mathrm{D}(K_\pi, z)$	19	1396	253	1631	98
c	$x \cdot G$	5667	7724	968	3346	731
d	$x \cdot Y$	5582	19006	797	5053	
e	$s \cdot G$	4819	23851	244	5307	
	$(c) + (d)$	19		1		
f	$\widetilde{SK}_{\mathrm{PCD}} \cdot G'$	5600	35450	889	7644	535
i	$\widetilde{SK}_{\mathrm{PCD}} \cdot \widetilde{PK}_{\mathrm{PICC}}$	5564	41026	1666	9333	
j	T_{PCD}	42	41160	22	9458	87
	Total [ms]	28195	41160	5718	9458	1841

Table 8: Reference PACE performance test on Nokia 6212

		Bouncy Castle		FlexiProvider		
	Details	PCD	Total	PCD	Total	PICC
1	EF.CardAccess	120	381	100	390	290
2	MSE:Set AT	50	581	100	651	161
b	$\mathrm{D}(K_\pi, z)$	10	661	50	761	60
c	$x \cdot G$	171	1172	70	1181	350
d	$x \cdot Y$	20	1212	10	1211	
e	$s \cdot G$	20	1232	10	1221	
	$(c) + (d)$	0		0	0	
f	$\widetilde{SK}_{\mathrm{PCD}} \cdot G'$	30	1553	20	1512	261
i	$\widetilde{SK}_{\mathrm{PCD}} \cdot \widetilde{PK}_{\mathrm{PICC}}$	20		1562	40	
j	T_{PCD}	10	1663	0	1622	60
	Total [ms]	451	1663	400	1622	1182

Table 9: Reference PACE performance test on desktop PC

by additional operations such as creating Java objects, event handling and control flow statements. The timings were measured immediately before sending an APDU to the PICC and immediately after receiving the response from the PICC.

The measured values show that the point multiplications spend extensive resources. The FlexiProvider performs the elliptic curve computations significantly faster than the Bouncy Castle (cf. Table 8). One reason for this is that the FlexiProvider uses the more sophisticated exponentiation technique developed in [6] by default instead of the simple Square & Multiply technique (which is used by Bouncy Castle).

Additionally, we perform run-time tests on a desktop PC (Intel Core 2 Quad Q8300 2.5GHz) with an attached smart card reader (SCM SDI010 [33]). Information about the hardware configuration of the Nokia 6212 is not publicly available. The values in Table 9 for the performance tests on the desktop PC are quite similar for both CSPs and the major part of the run-time in this scenario is caused by the PICC (approximately 70%). On average the PICC needs about 1200 milliseconds in total, which can be characterized as a lower bound of the overall run-time.

Before implementing and testing the optimized PACE protocol, containing the optimizations stated in Section 4, we analyze the implementations of the large number arithmetic in both CSPs, because they determine the performance of the elliptic curve arithmetic. As shown in Table 10 the implementation of the BigInteger class (for large number arithmetic) in FlexiProvider is significantly faster for the important functions ModInverse and ModMult than the respective implementation in Bouncy Castle. The performance of the elliptic curve arithmetic is also much better in the FlexiProvider than in Bouncy Castle.

Function / Provider	FlexiProvider [ms]	BouncyCastle [ms]
Mod	0.093	0.0306
Multiply	0.1166	0.1143
Subtract	0.0261	0.0615
Add	0.0403	0.0406
ModInverse	4.7	20.77
ModPow (squared)	1.54	3.4
ModAdd	0.1591	0.077
ModMult	0.3976	2.9589
Point Mult	1051.3	7081.2
Point Addition	3.8	18.7
Point Doubling	2.3	23.8
Random	1364	7521

Table 10: Performance tests on Nokia 6212 (256 bit numbers)

Note the cost to generate a random number (cf. Table 10), which is needed in step (b) and (f) of the PACE protocol (x and $\widetilde{SK}_{PCD}$ in Figure 1). Bouncy Castle needs approximately 15 seconds and FlexiProvider 2.7 seconds in all to create these two random numbers. If we would use the same pseudo-random number generators (PRNG) in both CSPs (Bouncy Castle's PRNG in FlexiProvider) run-time for FlexiProvider would be approximately 24.5 seconds instead of 9.5 seconds in total.

5.2 Optimized Implementation

Since the performance measurements revealed that the performance of the FlexiProvider is much better than the performance of Bouncy Castle (cf. Section 5.1) we use the FlexiProvider for the implementation of the optimized PACE protocol with the optimizations described in Section 4. It has an ratio of about 12:9 (ADD:DBL).

Algorithm 1 Precomputations for the optimized PACE protocol

```
Input: $\mathcal{D}$ {Let $\mathcal{D}$ denote the domain parameters including the
    basepoint $G$}
Output: $x, \widetilde{SK}_{\mathrm{PCD}}, X$
 1: $x \leftarrow$ GenPrivKey()
 2: $\widetilde{SK}_{\mathrm{PCD}} \leftarrow$ GenPrivKey()
 3: $X \leftarrow \mathrm{Multiply\_LimLee}(x, G)$ {(1)}
 4: return $(x, \widetilde{SK}_{\mathrm{PCD}}, X)$
```

The PACE protocol is implemented as depicted in Algorithm 2 using the precalculations from Algorithm 1. Algorithm 2 shows the variant α as described in Section 4. The algorithms for the other variants are straightforward by replacing the functions in lines 5 and 6 accordingly. The function *GenPrivKey()* generates a uniformly chosen random number smaller than the order r of the elliptic curve. Algorithm 1 is started immediately at the application startup if the the domain parameters are available from a previous session. If there are no matching domain parameters available on the device, Algorithm 1 is started after the PIN entry, before Algorithm 2 is started. Equation (1) and the generation of the private keys are moved forward to Algorithm 1. The next point multiplications from Equation (6) are implemented using an interleaving algorithm (line 5). The last two multiplications (Equation (4) in line 6 and Equation (5) in line 9) are implemented using a w-NAF multiplication method.

There are two states for the protocol execution. The first state is the initial run of the protocol after the installation of the application. In this first state no saved points or domain parameters can be used in the protocol. During the first run these values are stored in the

Algorithm 2 The optimized PACE protocol

```
Input: $\pi, \mathcal{D}$ {Let $\mathcal{D}$ denote the domain parameters including the basepoint $G$}
Output: $(K_{\mathrm{MAC}}, K_{\mathrm{ENC}})$
 1: $z \leftarrow$ getZ()
 2: $s \leftarrow \mathrm{D}(K_\pi, z)$
 3: sendX($X$)
 4: $Y \leftarrow$ getY()
 5: $G' \leftarrow \mathrm{Multiply\_Interleaving}(s, x, G, Y)$ {(6)}
 6: $\widetilde{PK}_{\mathrm{PCD}} \leftarrow \mathrm{wNafMult}(\widetilde{SK}_{\mathrm{PCD}}, G')$ {(4)}
 7: sendX($\widetilde{PK}_{\mathrm{PCD}}$)
 8: $\widetilde{PK}_{\mathrm{PICC}} \leftarrow$ getPK()
 9: $K \leftarrow \mathrm{wNafMult}(\widetilde{SK}_{\mathrm{PCD}}, \widetilde{PK}_{\mathrm{PICC}})$ {(5)}
10: $T_{\mathrm{PCD}} \leftarrow \mathrm{MAC}(K_{\mathrm{MAC}}, \widetilde{PK}_{\mathrm{PICC}})$
11: sendT($T_{\mathrm{PCD}}$)
12: $T_{\mathrm{PICC}} \leftarrow$ getT()
13: if $T_{\mathrm{PICC}} == \mathrm{MAC}(K_{\mathrm{MAC}}, \widetilde{PK}_{\mathrm{PCD}})$ then
14:     return $(K_{\mathrm{MAC}}, K_{\mathrm{ENC}})$
15: else
16:     return Error during authentication
17: end if
```

Variant	State 1 [ms]	State 2 [ms]
α	7580	6280
β[9]	7330	6310
γ	7530	6360
δ	9034	7775

Table 11: Optimized PACE performance tests on Nokia 6212

NVM of the mobile device for future use. In the second state the domain parameters, the base point G and precomputed points are already available on the device and can be used during the protocol execution. Since it can be assumed that the owner of the mobile phone uses the application mostly with his own identity card, this second state should cover most of the protocol executions.

5.3 Performance

The runtimes of the efficient implementation are depicted in Table 11. In contrast to the performance measurements of the prototypical implementation (cf. Table 8) we cannot measure every single step, as they are strongly interwoven into each other through threading. Furthermore no API exists for Java ME to measure threads as known by Java Profilers. The measurements distinguish between the two states shown in the two columns. The first state can't make use of any stored values (precomputed points or domain parameters). The second one uses the stored values to precalculate the result of Equation (1). In variant δ Equation (10) also uses these stored values. The rows show the four different possibilities of rearranging the equations as presented in Section 4.1. Thus, except the variant δ the results are very similar so that the fastest variant may vary depending on the used hardware and operating system. Variant δ has some promising approaches but does not work so well on the used device.

[9]As no available mobile Java CSP supports this directly, x_1, x_2, Y_1 and Y_2 are generated in a preparation step and then passed to a standard interleaving multiplication method together with s, G, and Y.

6. POTENTIAL SIDE CHANNEL VULNERABILITIES IN THE PACE PROTOCOL

Elliptic curve operations are known to be vulnerable to side channel attacks if no appropriate countermeasures are taken. The basic problems are power analysis attacks [25] and timing attacks [24]. In this section, we briefly address the side channel security issues of the PACE protocol on the side of the PCD given Algorithms 2 and 1 are used. First, we identify potential targets of such attacks against the elliptic curve algorithms.

There are three scalar values that are used for point multiplication in one run of the PACE protocol: x, s, and $\widetilde{SK}_{\mathrm{PICC}}$. All of these values are ephemeral, thus only a single protocol run is available to an attacker to recover these values. If an attacker aims at gaining the encryption and MAC keys derived in the course of the protocol, he has to find all three scalar values. While s is only used in one scalar multiplication, the other two are used in two scalar multiplications each. Since differential analysis are based on far more recordings than this, only simple analysis can be used to recover these values.

This basically rules out timing attacks, since they demand differential analysis by nature. One exception is an attack recovering the hamming weight of the scalar against an unprotected binary method: here the running time is linearly dependent on the hamming weight of the scalar. However, the hamming weight will in general not suffice to enable actual recovery of the secret scalar values.

Concerning power analysis attacks, it should be pointed out that a scenario, where the attacker has actual access to CPU's power supply is not realistic. However, electro-magnetic radiation (EM) attacks [1] are similar in nature. Thus, it seems advisable to include power analysis countermeasures in order to prevent EM attacks. Note that in the case of a binary method for the elliptic curve scalar multiplication, even simple EM analysis, i.e. an attack based on a single recorded trace, is feasible. If the attacker can distinguish the doubling and the addition operation, he can read the bits of the secret scalar from the trace.

It is important not only to address the scalar multiplication, but also the exponent recoding operations that are necessary when applying NAF representations. Such an attack is presented in [32].

The symmetric operations involving secrets have to be secured as well. Specifically, the derivation of the cipher key K_π and the decryption in step 2 of algorithm 2 must be secure against differential analysis, since they involve the same secret values whenever a specific eID is used. The MAC and encryption operations performed using K_{MAC} and K_{ENC} after the PACE protocol also need to be secure with respect to differential analysis to a certain extend, since they will be used for a complete session.

7. FUTURE WORK

The differentiation between the two states of the optimized PACE protocol is only necessary due to the fact that the Nokia 6212 only recognizes the eID card when it is lying on the top of the cellphone. The reason for this is the location of the antenna and its field intensity. Most likely newer devices don't have these disadvantages. Thus, it would be possible for the user to enter the PIN while the eID card has contact to the mobile phone. Then the phone would be able to decide whether it knows the eID card or not and load the necessary domain parameters to run Algorithm 1. Thereby the runtimes for the two states would be equal so that the differentiation between the two states becomes obsolete.

The mobile device could, besides saving the base point and its multiples for LimLee, also save multiples of that point generated during the evaluation of Equation (1) for further use. This clearly would save computation time. However, the available mobile cryptographic libraries do not offer the possibility to use the stored points. Thus, this optimization is left for future work including respective modifications of the applied cryptographic library. Saving precomputed values for the w-NAF multiplication is also possible but does not lead to significant performance improvements since the computation for a window size of 5 only includes 3 ADD and 1 DBL, which is negligible. Furthermore, accessing the file system and reading the points is not free of charge either.

The method used for the second variant in Section 4.1 can be generalized to a method for various e_i, P_i pairs with different sizes of the e_i. We call this Interleaved-Lim-Lee-Combining (ILLC). Here, each e_i, P_i pair where the scalar exceeds the determined size is broken into e_j, P_j pairs with the correct scalar size. For each e_i, P_i pair where the scalar e_i is smaller than the determined size the scalar e_i is padded to the correct size. To our knowledge none of the available cryptographic libraries offer this method, it seems not even to exist in literature. Hence, we will investigate and implement ILLC ourselves.

There are more possible optimizations which we did not apply, as no available mobile provider supported their implementation. Examples involve using Montgomery Multiplication [28], efficient point triplication [5] and quintuplication [27]. While we limited ourselves in the work at hand to PACE optimizations realizable without modifications to existing mobile providers, the next round of PACE optimizations will include all mentioned methods which require such modifications.

Another, rather adventurous optimization deliberately violates the PACE protocol. Instead of choosing x and $\widetilde{SK}_{\text{PCD}}$ uniformly at random, the PCD could set $x = s$ and $\widetilde{SK}_{\text{PCD}} \equiv s^{-1} \bmod r$, where r is the order of the used elliptic curve. By this, computing $\widetilde{PK}_{\text{PCD}}$ is reduced to compute $G + Y$ which annihilates 2 random number generations and 2 MULT. Unfortunately, this allows an attacker eavesdropping on the communication to reconstruct the PIN. It is sufficient to once eavesdrop on a tuple $\mathcal{D}, z, X$ sent by the PCD. The attacker then can systematically try each possible PIN to decrypt z and verify it by checking whether $X = s \cdot G$ holds. Hence, this optimization is not suitable for the eID card scenario which is in the scope of this work. But it might be suitable for other scenarios, e.g. where the communication between the PICC and the PCD is secured against eavesdropping by other means.

8. CONCLUSION

This paper presented an efficient Java ME implementation of the PACE protocol for mobile devices. The review of the implementation of the relevant algorithms and Java ME CSPs showed that there are significant performance differences. The following investigation of the possible optimizations to a straightforward PACE version revealed different possibilities which all result in a similar theoretical speedup without changing existing CSPs. The presented benchmarks of the different possible optimizations were the basis for choosing concrete optimizations. The benchmarks revealed a significant speedup in comparison to the not optimized version. A discussion of potential side channel attacks on the optimized implementation rounded the investigation up. The future work discussion showed that there is more optimization potential when making changes to the existing cryptographic libraries. All in all we succeeded in providing a platform independent efficient mobile PACE implementation, but also showed where and how even more efficiency could be gained.

9. REFERENCES

[1] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The em side-channel(s). In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 29–45, London, UK, 2003. Springer-Verlag.

[2] Australian Government Information Management Office (AGIMO). Australian Government Smartcard Framework. Phase 2, Version 0.12, Standards and Model Specification Part c, March 2007, 2007.

[3] J. Bender, M. Fischlin, and D. Kügler. Security Analysis of the PACE Key-Agreement Protocol. In *Information Security Conference*, volume 5735 of *LNCS*. Springer-Verlag, September 2009.

[4] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[5] M. Ciet, M. Joye, K. Lauter, and P. Montgomery. Trading inversions for multiplications in elliptic curve cryptography. *Designs, Codes and Cryptography*, 39:189–206(18), May 2006.

[6] H. Cohen, A. Miyaji, and T. Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *LNCS*, pages 51–65. Springer, 1998.

[7] Comité Européen de Normalisation (CEN). Identification card systems - European Citizen Card - Part 1-4. (Draft of) Technical Specification, 2008.

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

[9] European Commission (IDABC). eID Interoperability for Pan-European Government Services, 2009. `http://ec.europa.eu/idabc/en/document/6484/5644`.

[10] Federal Ministry of the Interior. Introduction of the electronic Identity Card in Germany. BMI IT4-644 004/14#5, 02.07.2008, Einführung des elektronischen Personalausweises in Deutschland, Grobkonzept - Version 2.0, in German, 2008. `http://netzpolitik.org/wp-upload/bmi_epa-grobkonzept-2-0_2008-07-02.pdf`.

[11] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). Advanced Security Mechanism for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI). Technical Directive (BSI-TR-03110), Version 2.05, 2010. `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/TR-03110_v205_pdf.pdf`.

[12] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2003.

[13] B. Henhapl. On the efficience of elliptic curve cryptography, 2003.

[14] T. Icart. How to hash into elliptic curves. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, 2009.

[15] International Civil Aviation Organization (ICAO). Machine Readable Travel Documents - Part 1: Machine Readable Passport, Specifications for electronically enabled passports with biometric identification capabilities. ICAO Doc 9303, 2006.

[16] International Civil Aviation Organization (ICAO). Machine Readable Travel Documents - Part 3: Machine Readable Official Travel Documents, Specifications for electronically enabled official travel documents with biometric identification capabilities. ICAO Doc 9303, 2008.

[17] International Civil Aviation Organization (ICAO). Supplemental Access Control for Machine Readable Traval Documents. ISO/IEC JTC1 SC17 WG3/TF5 for ICAO, Version 0.8, Draft of 12.10.2009, 2009.

[18] International Civil Aviation Organization (ICAO). Supplemental Access Control for Machine Readable Travel Documents, 2009.

[19] ISO/IEC. ISO/IEC 14443-1: Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 1-4. International Standard, 2001.

[20] ISO/IEC. Identification cards – Integrated circuit cards, ISO/IEC 7816, 2004.

[21] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1), ISO/IEC 18092, 2004.

[22] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication Interface and Protocol -2 (NFCIP-2), ISO/IEC 21481, 2005.

[23] Juniper Research. 1 in 6 mobile subscribers to have NFC Mobile Phones by 2014. Press Release, 09.11.2009, 2009. `http://juniperresearch.com/shop/viewpressrelease.php?pr=163`.

[24] P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *Advances in Cryptology-CRYPTO'96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, 1109:103–113, 1996.

[25] P. Kocher, J. Jaff, and B. Jun. Differential Power Analysis. *Advances in Cryptology-CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, 1666:388–397, 1999.

[26] C. H. Lim and P. J. Lee. More Flexible Exponentiation with Precomputation. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 95–107. Springer-Verlag, 1994.

[27] P. K. Mishra and V. Dimitrov. Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication Using Multibase Number Representation. In *Information Security*, volume 4779 of *LNCS*, pages 390–406, 2007.

[28] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[29] National Institute of Standards and Technology (NIST). Personal Identity Verification (PIV) of Federal Employees and Contractors. FIPS PUB 201-1, March 2006, 2006. http://csrc.nist.gov/publications/fips/fips201-1/FIPS-201-1-chng1.pdf.

[30] C. Petit, F.-X. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based prng secure against side-channel key recovery. In *Proceedings of ASIACCS*. ACM, 2008.

[31] Sagem. MorphoMapping. Patent FR09-50189, 2009.

[32] Y. Sakai and K. Sakurai. A new attack with side channel leakage during exponent recoding computations. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 89–96. Springer Berlin / Heidelberg, 2004.

[33] SCM Microsystems. Smart Card Reader SDI010 . `http://www.scmmicro.com/security/view_product_en.php?PID=19`.

[34] M. Ullmann, D. Kügler, H. Neumann, S. Stappert, and M. Vögeler. Password authenticated key agreement for contactless smart cards. *Communications of the ACM*, 2008.