Probabilistic Admissible Encoding on Elliptic Curves - Towards PACE with Generalized Integrated Mapping[⋆]

Łukasz Krzywiecki, Przemysław Kubiak[⋆⋆], and Mirosław Kutyłowski

Wrocław University of Technology, Institute of Mathematics and Computer Science

Abstract. We consider *admissible encodings on an elliptic curve*, that is, the hash functions that map bitstrings to points of the curve. We extend the framework of admissible encodings, known from CRYPTO 2010 paper, to some class of non-deterministic mapping algorithms. Using Siguna Müller's probabilistic square root algorithm we show a mapping that works efficiently for *any* finite field $\mathbb{F}_q$ of characteristic greater than 3, and that is *immune to timing attacks*. Thereby we remove limitations of the mappings analyzed in the CRYPTO 2010 paper. Consequently, we remove limitations of a so called PACE Integrated Mapping protocol, which has recently been standardized by ICAO, and is used to protect contactless identity documents against unauthorized access.

Keywords: indifferentiability, admissible encoding, non-deterministic square root algorithm, finite field, elliptic curve.

1 Introduction

Many cryptographic protocols use efficient algebraic structures, such as elliptic curves, but at the same time operate on ordinary objects such as binary strings. In such a case we frequently need a *mapping* that converts the binary strings to points of the algebraic structure. This problem has profound practical implications, since an inefficient mapping may outweigh computational advantages of using the target algebraic structure.

The above issue is very important for smart card protocols, and in particular for electronic identity documents (e-ID), which are issued on a large scale. In fact, the problem concerned in this paper emerged while constructing authentication protocols for e-ID. Note that efficiency and security issues are critical for the issuer of e-IDs – less efficient protocols require more expensive smart cards on which the e-ID could be implemented, while a security flaw might cause exchange of identity documents with enormous organizational costs.

For the rest of the paper we focus on one fundamental issue for contactless e-IDs, namely preventing activation of an e-ID via the contactless interface without consent of the document owner. This problem is solved by cryptographic protocols based on a

[⋆] The paper has been supported by the Polish Ministry of Science and Higher Education: during the initial stage (i.e., numerical experiments) by project O R00 0015 07, later by project N N206 369 839; the third author is supported by the Foundation for Polish Science, "Mistrz" Programme.

[⋆⋆] Corresponding author: `przemyslaw.kubiak@pwr.wroc.pl` secret (password) shared by the document owner and the e-ID card. There are many password-based authentication protocols, however we focus on a standardized solution called PACE (Password Authenticated Connection Establishment) introduced by the German federal authority BSI ([7]). The aim of PACE is to establish a secure channel between a contactless smart card and a reader based on a password memorized by the owner or engraved on a surface of the card. Due to PACE, an active but non-invasive adversary can only guess the password and try it in an interaction with the card. A PACE execution for a password π consists of three stages:

1. So called domain parameters and the result of encryption of a nonce s are sent from the smart card to the reader. The domain parameters consist of a definition of an elliptic curve over a finite field $\mathbb{F}_q$, point G of prime order belonging to the curve, and the co-factor ℓ, i.e., the integer such that $(\text{ord}G) \cdot \ell$ equals the number of points on the curve. The encryption of the nonce is performed with a key derived from the password π stored inside the card. On the other hand, the reader learns the password π via another channel (e.g., the owner of the card enters it manually) and decrypts the message s.
2. Secondly, with help of the nonce s (and some other data exchanged between the reader and the card) a point $\hat{G}$ is calculated locally both on the side of the reader and on the side of the card.
3. $\hat{G}$ is used as a base point in the Elliptic Curve Diffie-Hellman key agreement protocol (ECDH). The resulting key is used to generate session keys needed to establish a secure channel between the reader and the card.

The second step is called *mapping*, since it maps the nonce s to the elliptic curve indicated in the domain parameters of the smart card. The mapping implemented on German e-ID documents is called the Generic Mapping. The Generic Mapping includes a separate execution of ECDH, therefore two ECDH executions are included in this variant of PACE.

In [11], [9] another mapping for PACE is proposed – the so called Integrated Mapping is more efficient than the Generic Mapping both in terms of communication and computational costs. The resulting variant of PACE is denoted as PACE IM.

The main building block of the Integrated Mapping is an algorithm that encodes a bit string as a point of the curve. Two encoding algorithms were proposed: the simplified Shallue-Woestijne-Ulas (SWU) algorithm or Icart's encoding (see [5]). Both encoding algorithms work in time $O((\log_2 q)^3)$, where $\mathbb{F}_q$ is the field over which the elliptic curve has been defined.[1]

The SWU simplified algorithm works for any field $\mathbb{F}_q$ with $\text{char}(\mathbb{F}_q) > 3$, however, it is used by PACE IM only for $q \equiv 3 \bmod 4$. The condition $q \equiv 3 \bmod 4$ arises from focusing on deterministic square root algorithms in finite fields. Icart's encoding method is also deterministic and works for any field $\mathbb{F}_q$, where $\text{char}(\mathbb{F}_q) > 3$ and $q \equiv 2 \bmod 3$.

The choice of deterministic encoding algorithms is a consequence of utilizing the notion of *indifferentiability* in the proof of quality of the encoding (cf. [6] and the full version [5]). Proving the security of hash based cryptosystems in the Random Oracle

[1] PACE uses only prime fields $\mathbb{F}_q$. but encoding algorithms work also for extension fields.

Model (ROM) requires that a hash function is modeled as an ideal functionality accessible by all parties. However real designs impose that hashes do not have a *monolithic* construction, but are rather iteratively built from some smaller blocks. Indifferentiability property for deterministic algorithms guarantees that such a compound construction can replace the monolithic ROM model in cryptosystems with security scenarios related to a single, stateful adversary (for multi-stage protocols and resource-restriction models see [15], [10]). However, indifferentiability assumes that the algorithm implementing function f is deterministic. Therefore the security of PACE IM was analyzed only for deterministic encoding functions.

Our Contribution. We extend the framework of indifferentiable hashing given in [6], [5] to some class of non-deterministic algorithms. The extended framework justifies the use of certain probabilistic algorithms in the simplified SWU method. The resulting encoding protocol works for any field $\mathbb{F}_q$ with char$(\mathbb{F}_q) > 3$, has expected running time $O((\log_2 q)^3)$ (like the deterministic simplified SWU method and Icart's encoding) and enjoys small variance. In particular, our generalization allows to use the NIST P-224 curve in the probabilistic variant of PACE IM (the standardization report [11] excludes this curve).

Note that removing constraints imposed by the mapping algorithm on the fields $\mathbb{F}_q$ of characteristic greater than 3 gives more flexibility in other applications (an example could be the signature scheme [4]) and leaves more room for future designs. Moreover, having replaced specialized Integrated Mappings with a single general standard we could reduce deployment costs.

Timing attacks. Apart from the high quality randomness ensured by the mapping calculated in the second step of PACE, the protocol must protect confidentiality of the password π. In particular, execution time of the protocol must not depend on the *value* of π. This condition is satisfied by both versions of PACE and our extension meets this condition as well.

Notation. The following notation is used in the rest of the paper.

Definition 1 (parity operator $\text{parity}_{\mathcal{V}}() : \mathbb{F}_q \to \{0,1\}$**).** *Let $\mathbb{F}_q$ be a finite field of characteristic $p > 2$. Let $\mathcal{V} = \{v_0, v_1, \ldots, v_{d-1}\}$ be a base of the vector space $\mathbb{F}_q$ defined over the field $\mathbb{F}_p$. Accordingly, d is the extension degree $[\mathbb{F}_q : \mathbb{F}_p]$. Let $a \in \mathbb{F}_q$, and let $a = \sum_{i=0}^{d-1} a_i v_i$ be the representation of a in base $\mathcal{V}$ with $a_i \in \mathbb{F}_p$ for $i = 0, \ldots, d-1$. Define* $\text{parity}_{\mathcal{V}}(a)$ *as the parity of* a_0.

Definition 2 (mapping $\tilde{m}_{\mathcal{V}} : \mathbb{Z}_q \to \mathbb{F}_q$**).** *Let $\mathbb{F}_q$ be a finite field of characteristic $p > 2$, let $\mathcal{V}$ be a base from Def. 1. We define the mapping $\tilde{m}_{\mathcal{V}}$ as follows: for any $t \in \mathbb{Z}_q$ represent t as a nonnegative integer written in base p, that is $t = (t_{d-1} \ldots t_1 t_0)_p$, where each t_i belongs to the set $\{0, 1, \ldots, p-1\}$. Then $\tilde{m}_{\mathcal{V}}(t) = \sum_{i=0}^{d-1} t_i v_i$.*

It is easy to see that $\tilde{m}_{\mathcal{V}}()$ is a one-to-one mapping.

For $a \in \mathbb{F}_q$, let $\eta(a)$ denote the quadratic character of a, i.e., $\eta(a) = 0$ for $a = 0$, $\eta(a) = 1$ if $a = b^2$ for some $b \in \mathbb{F}_q^*$, and $\eta(a) = -1$ otherwise. Recall that $\eta(a)$ can be calculated in time $O\left((\log q)^2\right)$ (cf. [2]). By $\nu_p(n)$ we denote the greatest exponent α such that p^α divides n. By $\oplus$ we denote *exclusive or*. The expression $x \leftarrow_\$ X$ denotes the random choice of an element x from the set X with uniform probability distribution. $[k]G$ denotes the multiplication of an element G of some additive group by the scalar k.

2 Siguna Müller's Square Root Algorithm

We recall the square root algorithm from [13] constructed for finite fields $\mathbb{F}_q$ with odd characteristic and based on Lucas sequences (cf. [1, Annex I.1.3]). Its expected running time is $O((\log_2 q)^3)$. An alternative to [13] may be Peralta's algorithm (see [2, Sect. 3]) with the same expected running time. However, Peralta's algorithm works only for $q \equiv 1 \bmod 4$.

The square root algorithm from [13] uses elements of the Lucas sequence $V_k(\tilde{P}, 1)$, where $V_0(\tilde{P}, \tilde{Q}) = 2, V_1(\tilde{P}, \tilde{Q}) = \tilde{P}, V_{2n}(\tilde{P}, \tilde{Q}) = (V_n(\tilde{P}, \tilde{Q}))^2 - 2\tilde{Q}^n, V_{2n+1}(\tilde{P}, \tilde{Q}) = V_{n+1}(\tilde{P}, \tilde{Q}) \cdot V_n(\tilde{P}, \tilde{Q}) - \tilde{P} \cdot \tilde{Q}^n$. These formulas constitute a base for a left-to-right binary algorithm [14] and allow to compute $V_k(\tilde{P}, 1)$ at a cost of at most $2 \cdot \lfloor \log_2 k \rfloor$ multiplications and squarings in $\mathbb{F}_q$.

The base $\mathcal{V}$ being the input of Algorithm 1 is usually defined as a polynomial or as a normal basis. Recall that $\mathcal{V}$ and the modular polynomial defining the field $\mathbb{F}_q$ both determine arithmetic in the field.

Algorithm 1. `SM_sqrt` – Siguna Müller's square root algorithm from [13]

```
Input: definition of $\mathbb{F}_q$ and basis $\mathcal{V}$, value $Q$ being a square in $\mathbb{F}_q^*$, $parity\_bit \in \{0,1\}$
Output: $\tilde{a} \in \mathbb{F}_q^*$ such that $\tilde{a}^2 = Q$ in $\mathbb{F}_q$ and that $\mathrm{parity}_{\mathcal{V}}(\tilde{a}) = parity\_bit$,
 1: $\delta \leftarrow -1$ for $q \equiv 1 \bmod 4$, and $\delta \leftarrow 1$ for $q \equiv 3 \bmod 4$
 2: repeat
 3:     $r \leftarrow_\$ \mathbb{F}_q^*$
 4:     $P \leftarrow Q \cdot r^2 - 4$
 5: until $\eta(P) = \delta$
 6: $P \leftarrow P + 2$
 7: $a[0] \leftarrow V_{(q+\delta)/4}(P, 1)$
 8: $a[0] \leftarrow a[0] \cdot r^{-1}$   {now we have $(a[0])^2 = Q$}
 9: $a[1] \leftarrow -a[0]$   {only one of the elements $a[0], a[1] \in \mathbb{F}_q^*$ has the required $parity\_bit$}
10: return $a[\mathrm{parity}_{\mathcal{V}}(a[0]) \oplus parity\_bit]$
```

With help of the reasoning from Sect. 2 of [3] one may show that both for $q \equiv 1 \bmod 4$ and $q \equiv 3 \bmod 4$, the probability that for a single choice of $r \in \mathbb{F}_q^*$ (i.e., for a single iteration of the loop) the condition in line 5 of Algorithm 1 is not satisfied is equal to $\frac{1}{2} + \frac{1}{q-1}$.

Note that randomization by r completely equalizes the chances of all squares Q to reach line 6 of the algorithm in some given number of iterations of the **repeat...until** loop. Timing attacks at this place are therefore mitigated.

Since complexity of the computation of the quadratic character $\eta(P)$ may be bounded by $O\left((\log_2 q)^2\right)$, it is easy to see that with an overwhelming probability the algorithm will return the required square root in time $O\left((\log_2 q)^3\right)$. Thus we have some level of uncertainty that the main loop of Algorithm 1 will take no more than $\log_2 q$ iterations. However, the probability that the loop takes more iterations than $\log_2 q$ equals $(\frac{1}{2} + \frac{1}{q-1})^{\log_2 q}$. Note that some small level of uncertainty turned out to be acceptable in the case of the Miller-Rabin primality test deployed on smart cards.

To summarize, the number of iterations of the **repeat...until** loop of Algorithm 1 is described by a random variable X_{loop} having geometric distribution with parameter $\hat{p} = \frac{1}{2} - \frac{1}{q-1}$. Hence $E[X_{loop}] = 1/\hat{p}$ and $Var[X_{loop}] = (1-\hat{p})/(\hat{p})^2$.

3 Probabilistic Variant of the Simplified SWU Method

Let n be a non-square in the field $\mathbb{F}_q$. If the mapping is to be utilized for establishing a secure channel between two parties, then both parties should use the same n. The simplest way to achieve this is to define the basepoint G included in the set of domain parameters in such a way that the x-coordinate of G is a non-square in $\mathbb{F}_q$.

Henceforth we assume that $\mathbb{F}_q$ is a field of characteristic greater than 3. Let $\mathcal{V}$ be a base from Def. 1, and let $\tilde{m}_{\mathcal{V}}()$ be the mapping defined by Def. 2 . By Algorithm 2 – a probabilistic variant of the simplified SWU algorithm — we define a mapping $f : \mathbb{Z}_{2q} \to E_{a,b}(\mathbb{F}_q)$, where $E_{a,b}(\mathbb{F}_q)$ is an elliptic curve defined by equation $y^2 = x^3 + ax + b$ for $a, b, x, y \in \mathbb{F}_q$.

Algorithm 2. Encoding $f : \mathbb{Z}_{2q} \to E_{a,b}(\mathbb{F}_q)$

```
Input: $a, b, n, \mathbb{F}_q, \mathcal{V}$, and argument $t \in \mathbb{Z}_{2q}$ to be mapped on $E_{a,b}(\mathbb{F}_q)$
Output: deterministic result of the mapping in non-deterministic time
 1: $parity \leftarrow t \bmod 2$
 2: $t' \leftarrow t \bmod q$        {from the Chinese Remainder Theorem (CRT) the pair $(parity, t')$
                                   uniquely represents $t \in \mathbb{Z}_{2q}$}
 3: $S \leftarrow n \cdot (\tilde{m}_{\mathcal{V}}(t'))^2$
 4: $X_2 \leftarrow \frac{-b}{a}(1 + \frac{1}{S^2+S})$
 5: $X_3 \leftarrow S \cdot X_2$
 6: $rndbit \leftarrow_{\$} \{0,1\}$        {we shall save some computations from time to time}
 7: $h \leftarrow \left((X_{2+rndbit})^2 + a\right) \cdot X_{2+rndbit} + b$
 8: if $\eta(h) = 1$ then
 9:     return $(X_{2+rndbit}, \mathtt{SM\_sqrt}(\mathbb{F}_q, \mathcal{V}, h, parity))$
10: else
11:     $h \leftarrow \left((X_{3-rndbit})^2 + a\right) \cdot X_{3-rndbit} + b$
12:     return $(X_{3-rndbit}, \mathtt{SM\_sqrt}(\mathbb{F}_q, \mathcal{V}, h, parity))$
13: end if
```

Note that the random choice made in line 6 of Algorithm 2 is independent of the input. Moreover, the execution time of the main loop of Algorithm 1 is independent from that choice (recall the randomization by r in the main loop of Algorithm 1). Consequently, for fixed parameters $a, b, n, \mathbb{F}_q, \mathcal{V}$ the execution time of the whole Algorithm 2 is described by a random variable $Y_1 + Y_2$ that is independent of Algorithm's 2 input argument t: the random variable Y_1 describes the execution time of the code included in the listing of Algorithm 2, excluding `SM_sqrt` subroutine, whereas the random variable Y_2 describes the execution time of the `SM_sqrt` subroutine. Y_1 is independent from Y_2.

Tests. Our tests confirm that Algorithm 2 is pretty efficient. E.g., for the NIST P-224 curve, during 10^5 calls of Algorithm 2, we measured that for the number of multiplications, squaring and Jacobi symbol computations: a sample mean equals $\approx$135.503, $\approx$226.503, $\approx$3.002 correspondingly, square root of a sample variance equals $\approx$1.506, $\approx$1.506, $\approx$1.424 respectively, The worst case measured during these 10^5 calls equals 155, 246, 22 correspondingly. There are exactly 3 inversions computed during each call of the Algorithm 2.

4 Indifferentiability for a Non-deterministic Case

The "indifferentiability" notion was introduced in [12] and used also in [8]. In both papers only deterministic algorithms are concerned.

To formally justify Algorithm 2 we introduce the following notion of a *Time-oblivious Turing machine*:

Definition 3 (Time-oblivious Turing machine). *Let C be a non-deterministic Turing machine that computes a function. For each input Q, belonging to the domain $\mathcal{D}_C$ of allowable arguments, the time when C delivers the result is unknown a priori, but is described by the random variable X_Q. Let F_{X_Q} denote the probability distribution of X_Q.*

The machine C is called a time-oblivious *if for any $Q, Q' \in \mathcal{D}_C$ the probability distributions F_{X_Q}, $F_{X_{Q'}}$ are exactly the same, (thus we have a single distribution F_X), and the expected value $E[X_Q]$ is finite for each Q.*

Note that X_Q takes only non-negative values, so $E[X_Q] < \infty$ implies that the probability of the event that C will not deliver the result in finite time for an argument from $\mathcal{D}_C$ equals 0. We denote the event by $C = \perp$.

An example of a time-oblivious Turing machine is a Turing machine implementing directly the simplified SWU method. More generally, we may consider a Turing machine C' implementing a function, and possessing a single, distinguished state $\mathcal{S}$ such that starting in $\mathcal{S}$ the machine might take one of two sequences of actions: the first sequence is deterministic and terminates the execution, the second one is also deterministic but terminates in the state $\mathcal{S}$ (hence we have a loop). The choice of the sequence is made by tossing a coin that might be asymmetric but the same coin is used for all arguments and all iterations of the loop. We assume that state $\mathcal{S}$ is the only possibility allowing the machine C' to enter the infinite loop. However, we do not exclude the possibility that before the state $\mathcal{S}$ is reached for the first time machine C' makes some other random choices independent of the input. However, since C' implements a function none of the random choices (including those made at state $\mathcal{S}$) influences the final output.

Now we recall the definition of a *random oracle* and *indifferentiability* from Sect. 2.2 of [5]. According to [5], an *ideal primitive* is an algorithmic entity which receives inputs from one of the parties and delivers its output *immediately* to the querrying party.

A *random oracle* into a finite set S is an ideal primitive which provides a random output in S for each new querry; identical input queries are given the same answer.

Let $\mathcal{D}^{R_1,R_2}$ denote a Turing machine $\mathcal{D}$ that uses two different oracles R_1 and R_2. During its execution $\mathcal{D}^{R_1,R_2}$ can state queries to R_1 and R_2 and get immediately an answer.

Definition 4 (indifferentiability [12], [8], [5]). *A Turing machine C with oracle access to an ideal primitive h is said to be $(t_D, t_S, q_D, \varepsilon)$-indifferentiable from an ideal primitive H if there exists a simulator S with oracle access to H and running in time at most t_S, such that for any distinguisher $\mathcal{D}$ running in time at most t_D and making at most q_D queries, it holds that:*

$$\left|\Pr\left[\mathcal{D}^{C^h,h} = 1\right] - \Pr\left[\mathcal{D}^{H,S^H} = 1\right]\right| < \varepsilon.$$

C^h is said to be indifferentiable from H, if ε is a negligible function of the security parameter k, for polynomially bounded q_D, t_D and t_S.

Let us explain the above definition. One can try to emulate the primitive H with a Turing machine C using another primitive h as a subroutine. Intuitively, the machine C transforms the results of primitive h so that the final results mimic the primitive H. If we try to distinguish the results of H and C^h, then we not only have to show that the results of C^h are as good as the results of H (i.e., might have been obtained by H). A more difficult part is to convince an observer that a result given by H might have been obtained by C^h. For this purpose we need a simulator S: it provides "an output of h" that would be used by C to provide the same result as obtained by H. The machine $\mathcal{D}^{R_1,R_2}$ is supposed to output 1 if $(R_1, R_2) = (C^h, h)$ and 0 if $(R_1, R_2) = (H, S^H)$, hence Def. 4 means that $\mathcal{D}$ is unable to distinguish between these two cases.

Note that if the construction C^h is indifferentiable from an ideal primitive H, then C^h can replace H in a cryptosystem with security scenarios related to a single stateful adversary, and the resulting cryptosystem is almost as secure as before.

Recall that the *ideal primitive* is an algorithmic entity which receives inputs from one of the parties and delivers its output *immediately* to the querying party. Moreover, the simulator S might include some probabilistic algorithm, like e.g., the sampling algorithm from the proof of Theorem 3 in [5], yielding non-constant execution time. These remarks suggest that time does not matter for the distinguisher $\mathcal{D}$, otherwise $\mathcal{D}$ could distinguish S^H from h simply by time measurements. Indeed, it is not necessary that an observer cannot say whether he is *observing* H or C^h (in the real world we will always observe C^h). The point is that the *set of results* should have essentially the same properties no matter where it comes from and that the execution time for C^h must not leak any additional side channel information. However, the last condition is satisfied automatically if C^h is a deterministic or a time-oblivious Turing machine. Therefore we generalize Def. 4 in the following way:

Definition 5 (indifferentiability for a non-deterministic case). *A non-deterministic Turing machine $\tilde{C}$ with oracle access to an ideal primitive h is $(t_{\tilde{C}}, t_D, t_S, q_D, \varepsilon, \varepsilon_{\tilde{C}})$-indifferentiable from an ideal primitive H, if there exists a simulator S with oracle access to H and running in time at most t_S, such that for any distinguisher $\mathcal{D}$ running in time at most t_D and making at most q_D queries, the following conditions hold:*

- *$\tilde{C}^h$ is a time-oblivious Turing machine,*
- *the probability that in time $t_{\tilde{C}}$ the machine $\tilde{C}^h$ will not return an answer to the query is less than $\varepsilon_{\tilde{C}}$,*
- *for all events $\tilde{C}^h \neq \perp$*

$$\left|\Pr\left[\mathcal{D}^{\tilde{C}^h,h} = 1\right] - \Pr\left[\mathcal{D}^{H,S^H} = 1\right]\right| < \varepsilon \ ,$$

$\tilde{C}^h$ is said to be indifferentiable from H, if $\tilde{C}^h$ is $(t_{\tilde{C}}, t_D, t_S, q_D, \varepsilon, \varepsilon_{\tilde{C}})$-indifferentiable from an ideal primitive H, for $\varepsilon, \varepsilon_{\tilde{C}}$ being negligible functions of the security parameter k, and polynomially bounded $t_{\tilde{C}}, t_D, t_S, q_D$.

Note that the constraints on t_S, t_D are exactly the same in Def. 4 and 5 . As we shall see imposing additional constraints on t_S is unnecessary when extending the notion of admissible encoding to the case we investigate (in fact, almost the same simulator as the one defined in [5] could be used – the only difference lies in inversion of the mapping f, which must now take $parity$ into account). On the other hand, imposing additional constraints on t_D would weaken the property defined by Def. 5 .

5 Probabilistic Admissible and Weak Encodings

Below we extend the notions of *admissible encoding* and *weak encoding* to some non-deterministic cases. Both notions are known from [6], [5] and we recall them below. But first we recall definition of statistically indistinguishable distributions.

Definition 6 (statistically indistinguishable distributions). *Let X and Y be two random variables over a set S. The distributions of X and Y are ε-statistically indistinguishable if: $\sum_{s\in S} |\Pr[X = s] - \Pr[Y = s]| \leq \varepsilon$. The distributions X and Y are statistically indistinguishable, if ε is a negligible function of the security parameter.*

Definition 7 (admissible encoding from [5]). *A function $F : S \to R$ between finite sets is an ε-admissible encoding if it satisfies the following properties:*

1. *Computable: F is computable in deterministic polynomial time.*
2. *Regular: for s uniformly distributed in S, the distribution of $F(s)$ is ε-statistically indistinguishable from the uniform distribution in R.*
3. *samplable: There is an efficient randomized algorithm $\mathcal{I}$ such that for any $r \in R$, $\mathcal{I}(r)$ induces a distribution that is ε-statistically indistinguishable from the uniform distribution in $F^{-1}(r)$.*

F is an admissible encoding if ε is a negligible function of the security parameter.

Definition 8 (admissible probabilistic encoding). *A function $F : S \to R$ between finite sets is an $(\varepsilon, t_F, \varepsilon_F)$-admissible probabilistic encoding, if*

1. *F can be implemented on some time-oblivious Turing machine C such that probability that C will not return the result in time t_F is smaller than ε_F;*
2. *properties 2. and 3. from Def. 7 hold for F.*

F is an admissible encoding if ε, ε_F are negligible functions of the security parameter k, for polynomially bounded t_F.

By analogy to [6], [5], we define the function $\tilde{H} : \{0,1\}^* \to R$ as $\tilde{H}(m) := F(h(m))$, where $F : S \to R$ is an admissible probabilistic encoding, and $h : \{0,1\}^* \to S$ is *a function whose output is seen as an output of a random oracle*. To justify the latter condition consider the following thought experiment: we have two black-boxes, each containing the same implementation of hash function h. That is, each box include the same code for h and is built from the same hardware components. For each argument m each black-box actually calculates the value $h(m)$, but in one of the black-boxes at the very end of the computations the value $h(m)$ is replaced by a value returned by some random oracle assigned to the given black-box. If after a series of queries an external observer not knowing the code for h cannot indicate the black-box returning the values of h, then we say that the output of h might be seen as the output of a random oracle.

Theorem 1 (Theorem 1 from [5]). *Let $F : S \to R$ be an ε-admissible encoding. The construction $H(m) := F(h(m))$ is $(t_D, t_S, q_D, \varepsilon')$-indifferentiable from a random oracle, in the random oracle model for $h : \{0,1\}^* \to S$, with $\varepsilon' = 4q_D \cdot \varepsilon$ and $t_S = 2q_D \cdot t_I$, where t_I is the maximum running time of F's sampling algorithm.*

Theorem 2 (analogous to Theorem 1 from [5]). *Let $F : S \to R$ be an $(\varepsilon, t_F, \varepsilon_F)$-admissible probabilistic encoding, let maximum running time of $h : \{0,1\}^* \to S$ be t_h, where t_h is bounded by a polynomial in the security parameter. The construction $\tilde{H}(m) := F(h(m))$ is $(t_F + t_h, t_D, t_S, q_D, \varepsilon', \varepsilon_F)$-indifferentiable from a random oracle, in the random oracle model for the output of $h : \{0,1\}^* \to S$, with $\varepsilon' = 4q_D \cdot \varepsilon$ and $t_S = 2q_D \cdot t_I$, where t_I is the maximum running time of F's sampling algorithm.*

Proof (Sketch). Since h is implemented on some deterministic Turing machine $\tilde{h}$ running in time t_h which is not affected even if $\tilde{h}$ includes a call to some random oracle, then this Turing machine may be incorporated into time-oblivious Turing machine C implementing function F. In this way the time-oblivious Turing machine $\tilde{C}^h$ from Def. 5 is constructed, and from assumptions on the function F we get that the probability that in time $t_F + t_h$ the machine $\tilde{C}^h$ will not return an answer to the query is less than ε_F (note that $\tilde{h}$ is called once and output of $\tilde{h}$ constitutes input to F). The rest of the proof follows the proof of Theorem 1 from [5]. □

5.1 Weak Probabilistic Encoding

Definition 9 (weak encoding from [5]). *A function $f : S \to R$ between finite sets is said to be an α-weak encoding if it satisfies the following properties:*

1. *Computable: f is computable in deterministic polynomial time.*
2. *α-bounded: for s uniformly distributed in S, the distribution of $f(s)$ is α-bounded in R, i.e., the inequality $\Pr_s[f(s) = r] \le \alpha/|R|$ holds for any $r \in R$.*
3. *Samplable: there is an efficient randomized algorithm $\mathcal{I}_f$ such that $\mathcal{I}_f(r)$ induces the uniform distribution in $f^{-1}(r)$ for any $r \in R$. Additionally $\mathcal{I}_f(r)$ returns $N_r = |f^{-1}\{r\}|$ for $r \in R$.*

f is a weak encoding *if α is a polynomial function of the security parameter.*

Definition 10 (weak probabilistic encoding). *A function $f : S \to R$ between finite sets is said to be $(\alpha, t_f, \varepsilon_f)$-weak probabilistic encoding if*

1. *f can be implemented on some time-oblivious Turing machine C such that the probability that C will not return the result in time t_f is smaller than ε_f.*
2. *Properties 2. and 3. from Def. 9 hold for f.*

The function f is a weak probabilistic encoding *if α and t_f are polynomial functions of the security parameter, and ε_f is a negligible function of the security parameter.*

Note that properties 2 and 3 enumerated in Def. 10 concern conditions *for the function f*. Therefore they do not refer to the event $C = \perp$ concerning an *implementation of f* (recall that the event $C = \perp$ is *independent* from the input of f). The event $C = \perp$ is implicitly served by the first property enumerated in Def. 10.

Proposition 1. *f is an $(\alpha, c \cdot (\log_2 q)^3, (\frac{1}{2} + \frac{1}{q-1})^{\log_2 q})$-weak probabilistic encoding with some constant c and $\alpha = 8N/(2q)$, where N is the elliptic curve order.*

Proof (Sketch). It is easy to see that the encoding f defined by Algorithm 2 satisfies Lemma 6 [5]. Namely, the y-coordinate of the resulting point uniquely determines $t \bmod 2$, and the x-coordinate has preimage size at most 8 elements $\tilde{m}(t')$ from $\mathbb{F}_q$ and all the elements can be found in polynomial time. The proof for the x-coordinate follows exactly the proof of Lemma 6 [5]. Note that inversion of the mapping $\tilde{m}()$ from Def. 2 is easily computable, hence given an element $\tilde{t}$ from $\mathbb{F}_q$ it is easy to find $t \in \mathbb{Z}_q$ such that $\tilde{m}(t) = \tilde{t}$. All in all, from the CRT we conclude that for the mapping f the pre-image of any point $P \in E_{a,b}(\mathbb{F}_q)$ contains at most 8 elements from $\mathbb{Z}_{2q}$, and the pre-image can be computed in polynomial time. We assume that the constant c is chosen so that probability that the time-oblivious Turing machine C from Def. 10 will not return the result in time $c \cdot (\log_2 q)^3$ is *strictly smaller* than $(\frac{1}{2} + \frac{1}{q-1})^{\log_2 q}$, so the first condition from Def. 10 is also satisfied. □

Note that by Hasse Theorem $\alpha \leq \frac{8(\sqrt{q}+1)^2}{2q}$. Hence $\alpha < 8$ for $q \geq 7$. The same value $\alpha = 8N/(2q)$ applies, if we multiply the result (x, y) of the encoding f by a co-factor ℓ: let the order N of the group $E_{a,b}(\mathbb{F}_q)$ satisfy the condition $N = \ell \cdot N'$, where N' is prime and $\gcd(\ell, N') = 1$. The integer ℓ is called the co-factor (of N'). Hence $E_{a,b}(\mathbb{F}_q)$ has only one cyclic subgroup of order N'. Let G be a generator of this subgroup. If we take the result (x, y) of the encoding f and multiply by the co-factor ℓ, then we obtain an element of group the $\langle G \rangle$. Consequently, we have a map $f' = [\ell] \circ f$ such that $f' : \mathbb{Z}_{2q} \to \langle G \rangle$. Now we will find all pre-images of a given $P \in \langle G \rangle$ with respect to the map f'. For each element $P \in \langle G \rangle$ it is easy to obtain the unique element $P' \in \langle G \rangle$ being the inverse of P with respect to the scalar multiplication with $[\ell]$ in $\langle G \rangle$. Namely, to obtain P' it suffices to multiply P by the scalar $\ell^{-1} \bmod \operatorname{ord} G$. Since $\gcd(\ell, N') = 1$, we have $E_{a,b}(\mathbb{F}_q) = \langle G \rangle \times H$ where H is the subgroup of $E_{a,b}(\mathbb{F}_q)$ of order ℓ. To obtain all candidates in $E_{a,b}(\mathbb{F}_q)$ for the pre-image of P with respect to the scalar multiplication with $[\ell]$, we must collect all results of point addition $P' + P''$, where $P'' \in H$. There are ℓ such results. For each of them there are at most 8 elements in $\mathbb{Z}_{2q}$ being its preimage with respect to f. Altogether, for each of N' elements of $\langle G \rangle$ we have preimage of size at most $8 \cdot \ell$, hence $\alpha = (8 \cdot \ell) \cdot N'/(2q) = 8N/(2q)$.

5.2 The Resulting Admissible Probabilistic Encoding

Theorem 3 (Theorem 3 from [5], weak $\rightarrow$ admissible encoding). *Let $\mathbb{G}$ be cyclic additive group of order N, and let G be a generator of $\mathbb{G}$. Let $f : S \rightarrow \mathbb{G}$ be an α-weak encoding. Then the function $F : S \times \mathbb{Z}_N \rightarrow \mathbb{G}$ with $F(s,x) := f(s) + [x]G$ is an ε-admissible encoding into $\mathbb{G}$, with $\varepsilon = (1 - 1/\alpha)^t$ for any t being a polynomial in the security parameter k, and with $\varepsilon = 2^{-k}$ for $t = \alpha \cdot k$.*

The above theorem can be generalized as follows:

Theorem 4 (weak probabilistic $\rightarrow$ admissible probabilistic encoding). *Let $\mathbb{G}$ be an additive cyclic group of order N, and let G be a generator of $\mathbb{G}$. Let $f : S \rightarrow \mathbb{G}$ be an $(\alpha, t_f, \varepsilon_f)$-weak probabilistic encoding. Then the function $F : S \times \mathbb{Z}_N \rightarrow \mathbb{G}$ with $F(s,x) := f(s) + [x]G$ is an $(\varepsilon, t_f + t_x, \varepsilon_f)$-admissible probabilistic encoding into $\mathbb{G}$, where t_x is the maximum running time of the scalar multiplication $[x]G$ together with addition of resulting elements (i.e. elements $f(s), [x]G$), and $\varepsilon = (1 - 1/\alpha)^t$ for any t polynomially bounded in the security parameter k, and with $\varepsilon = 2^{-k}$ for $t = \alpha \cdot k$.*

Proof (Sketch). The *deterministic* Turing machine implementing the scalar multiplication $[x]G$ and the addition of the resulting elements may be incorporated into the time-oblivious Turing machine C implementing f. In this way execution time of the time-oblivious Turing machine grows at most by t_x (if $\mathbb{G}$ is a subgroup of some elliptic curve defined over a field $\mathbb{F}_q$, then time t_x of scalar multiplication $[x]G$ together with two points addition $f(s) + [x]G$ can be bounded by $c' \cdot (\log_2 q)^3$ for some constant c'). The rest of the proof follows exactly the proof of Theorem 3 from [5]. □

Consequently, in order to obtain an Admissible Probabilistic Encoding we should apply Theorem 4 to the encoding $f' = [\ell] \circ f$, that is, to the composition of the encoding f defined by Algorithm 2 and the scalar multiplication by the co-factor ℓ.

We also obtain an extension of the result from [5]: Let $h_1 : \{0,1\}^* \rightarrow \mathbb{Z}_{2q}$ and $h_2 : \{0,1\}^* \rightarrow \mathbb{Z}_N$ be two hash functions of running time bounded by polynomial in the security parameter. Then the function $H : \{0,1\}^* \rightarrow \mathbb{G}$ defined by:

$$H(m) := f'(h_1(m)) + [h_2(m)]G$$

is (according to Def. 5) indifferentiable from a random oracle in the random oracle model for outputs generated by h_1 and h_2.

6 Conclusions

Security analysis of PACE IM utilizes admissible encodings (cf. [9]). By extending the framework from [5] we have shown that the Admissible Probabilistic Encoding defined in Subsect. 5.2 preserves the level of security of its predecessor. Thus Algorithm 1 may be used in PACE IM in place of the currently used simplified SWU mapping.

Note that complexity of evaluation of $V_{(q+\delta)/4}(P, 1)$ in Algorithm 1 is comparable with the cost of the worst case exponentiation in field $\mathbb{F}_q$ via square and multiply. Moreover, the running time of the Algorithm 1 has small variation. Consequently, the difference between the running time of Algorithm 2 and the simplified SWU mapping will be negligible (or even unnoticeable) for the owner of a smart card. At the expense of a small decrease of efficiency we have gained more flexibility in choice of a field for a definition of the elliptic curve, hence we have more freedom in other applications like e.g., BLS signatures [4]. What is more, a standardized, general mapping procedure decreases deployment costs of the infrastructure supporting the protocols, especially in reference to the Common Criteria certification process.

Acknowledgements. We would like to thank Bart Preneel for many helpful comments.

References

1. Accredited Standards Committee X9, Inc., Financial Industry Standards: ANS X9.62-2005, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standard for Financial Services (2005)
2. Bach, E.: A note on square roots in finite fields. IEEE Transactions on Information Theory 36(6), 1494–1498 (1990)
3. Bernstein, D.J.: Faster square roots in annoying finite fields. Note: to be incorporated into author's High-speed cryptography book (November 2001)
4. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptology 17(4), 297–319 (2004)
5. Brier, E., Coron, J.S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. Cryptology ePrint Archive, Report 2009/340 (2009)
6. Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (2010)
7. BSI: Advanced Security Mechanisms for Machine Readable Travel Documents 2.11. Technische Richtlinie TR-03110-3 (2013)
8. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
9. Coron, J.-S., Gouget, A., Icart, T., Paillier, P.: Supplemental Access Control (PACE v2): Security analysis of PACE Integrated Mapping. In: Naccache, D. (ed.) Quisquater Festschrift. LNCS, vol. 6805, pp. 207–232. Springer, Heidelberg (2012)
10. Demay, G., Gaži, P., Hirt, M., Maurer, U.: Resource-restricted indifferentiability. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 664–683. Springer, Heidelberg (2013)
11. ISO/IEC JTC1 SC17 WG3/TF5 for the International Civil Aviation Organization: Supplemental access control for machine readable travel documents. Technical Report (2011) version 1.02 (March 2008)
12. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
13. Müller, S.: On the computation of square roots in finite fields. Des. Codes Cryptography 31(3), 301–312 (2004)
14. Postl, H.: Fast evaluation of Dickson polynomials. Contrib. to General Algebra 6, 223–225 (1988)
15. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 487–506. Springer, Heidelberg (2011)