Lenient/Strict Batch Verification in Several Groups

Fumitaka Hoshino, Masayuki Abe, and Tetsutaro Kobayashi

NTT Information Sharing Platform Laboratories, NTT Corporation
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
{fhoshino,abe,kotetsu}@isl.ntt.co.jp

Abstract. Batch verification is a useful tool in verifying a large number of cryptographic items all at one time. It is especially effective in verifying predicates based on modular exponentiation. In some cases, however, the items can be incorrect although they pass batch verification together. Such leniency can be eliminated by checking the domain of each item in advance. With this in mind, we investigate if the strict batch verification can remain more effective than separate verification. In this paper, we estimate the efficiency of such strict batch verification in several types of groups, a prime subgroup of $\mathbb{Z}_p$ with special/random prime p and prime subgroups defined on elliptic curves over $\mathbb{F}_p$, $\mathbb{F}_{2^m}$ and $\mathbb{F}_{p^m}$, which are often used in DL-based cryptographic primitives. Our analysis concludes that the efficiency differs greatly depending on the choice of the group and parameters determined by the verifying predicate. Furthermore, we even show that there are some cases where batch verification, regardless of strictness, loses its computational advantage.

1 Introduction

In cryptographic protocols verification of items such as signatures, zero-knowledge proofs, and ciphertexts plays an important role to maintain robustness. In large-scale applications, such verification could dominate efficiency since these items are typically verified by a central server while generated by each player. A typical example would be a bank serving in an off-line e-cash system, which verifies all electronic coins, i.e., the bank's signatures, spent each day [1]. Another good example would be electronic voting schemes where millions of voters cast encrypted ballots and a group of centralized servers shuffles and opens them in a verifiable way [2,3,4,5]. When real-time response is not required, batch verifying items would reduce the cost. In [2,6,7,8], it was shown that a type of verification predicates based on modular exponentiation can be efficiently batch verified.

It was shown in [9], however, that a set of items that is surely rejected in individual verification can be accepted in the batch mode with high probability depending on the group that defines the items. Intuitively, it happens when some elements of the items, which are supposed to be in a group, are in fact out of the group in such a way that the deviating parts eliminate each other in combination so that they are accepted in batch mode. As observed in [9], such erroneous items might not be a critical problem when the items are signatures since it is only the legitimate signer who can generate the erroneous signatures. *However, when the verifying items are zero-knowledge proofs, it could be a serious threat since the mistakenly accepted items may be used in further processing.*

In this paper, we consider *strict* batch verification that provides virtually the same confidence level as that provided by separately verifying each item. We estimate the efficiency of such strict batch verification in several types of groups such as a prime subgroup of $\mathbb{Z}_p$ with special/random prime p and prime subgroups defined on elliptic curves over $\mathbb{F}_p$, $\mathbb{F}_{2^m}$ and $\mathbb{F}_{p^m}$, which are often used in DL-based cryptography. We show that the efficiency level differs greatly depending on the choice of the group and the parameter determined by the verifying predicate. Furthermore, we even show that there are some cases where batch verification, regardless of the strictness, loses its computational advantage. This result would be an independent interest as batch verification is supposed to be faster than separate verification.

The rest of the paper is organized as follows. In Section 2, we define the items to check, and review some verification methods including separate verification and lenient/strict batch verification. In Section 3, our results are summarized. Section 4 introduces a criterion for the advantage of batch verification Based on this criterion we estimate the cost of batch verification in Section 5.

2 Preliminaries

2.1 Separate Verification of DL-Based Items

Let g be a generator of a cyclic group of order q. In this section, we use multiplicative notation for the group operation and all arithmetic operations are done in the group unless otherwise noted. Let $\mathcal{I}_n^k$ be a collection of n items for verification. Let I_i^k be the i-th item in $\mathcal{I}_n^k$ defined as

$$I_i^k = (e_{i1}, \ldots, e_{ik}, h_{i0}, h_{i1}, \ldots, h_{ik}) \in \mathbb{Z}_q^k \times \langle g \rangle^{k+1}$$

for some $k \geq 1$. Separate verification, $V_{\mathrm{sep}}(\mathcal{I}_n^k)$, consists of two phases

$$\text{Domain test:} \quad h_{ij} \overset{?}{\in} \langle g \rangle \quad \text{for } 1 \leq j \leq k,\ 1 \leq i \leq n, \text{ and} \tag{1}$$

$$\text{Separate relation test:} \quad h_{i0} \overset{?}{=} \prod_{j=1}^{k} h_{ij}^{e_{ij}} \quad \text{for } 1 \leq i \leq n. \tag{2}$$

We denote $V_{\mathrm{sep}}(\mathcal{I}_n^k) = \mathsf{true}$ if $\mathcal{I}_n^k$ passes the above tests. The items in $\mathcal{I}_n^k$ are defined to be correct if $V_{\mathrm{sep}}(\mathcal{I}_n^k) = \mathsf{true}$. It is stressed that the domain test does not apply to h_{i0} because if an item satisfies the predicate h_{i0} must also pass the domain test.

2.2 Lenient/Strict Batch Verification

Without loss of generality, we assume that $h_{ik_x}, \ldots, h_{ik}$ are common to all i and the rest of the bases are different in each item. To simplify the notation, we omit suffix i if it's unnecessary, thus $h_{ik_x}, \ldots, h_{ik}$ are denoted as $h_{k_x}, \ldots, h_k$. Lenient batch verification $V_{\mathrm{bat}}(\mathcal{I}_n^k)$ consists of

$$\begin{aligned} &\text{Lenient domain test:} && \text{(the same as in (1))} \\ &\text{Batch relation test:} && 1 \overset{?}{=} \left(\prod_{i=1}^{n} \prod_{j=0}^{k_x-1} h_{ij}^{r_i e_{ij}} \right) \left(\prod_{j=k_x}^{k} h_j^{\hat{e}_j} \right), \end{aligned} \tag{3}$$

where $e_{i0} = -1$ and $\hat{e}_j = \sum_{i=1}^{n} r_i e_{ij} \in \mathbb{Z}_q$ for $r_i \in_U R \subseteq \mathbb{Z}_q$. We denote $V_{\mathrm{bat}}(\mathcal{I}_n^k) = \mathsf{true}$ if $\mathcal{I}_n^k$ passes the above tests. From the results described in [9], it is shown that if all items are taken from the correct domain and $V_{\mathrm{bat}}(\mathcal{I}_n^k) = \mathsf{true}$, then $V_{\mathrm{sep}}(\mathcal{I}_n^k) = \mathsf{true}$ with probability $1 - 1/|R|$.

In the above, note that h_{i0} is still outside the domain testing. We define *strict batch verification* $V_{\mathrm{str}}(\mathcal{I}_n^k)$ as a combination of

$$\begin{aligned} &\text{Strict domain test:} && h_{ij} \overset{?}{\in} \langle g \rangle \quad \text{for } 0 \le j \le k,\ 1 \le i \le n, \text{ and} \\ &\text{Batch relation test:} && \text{(the same as in (3))} \end{aligned} \tag{4}$$

Note that h_{i0} is now included in the domain test. We denote $V_{\mathrm{str}}(\mathcal{I}_n^k) = \mathsf{true}$ if $\mathcal{I}_n^k$ passes these tests. When $V_{\mathrm{str}}(\mathcal{I}_n^k) = \mathsf{true}$, this implies that $V_{\mathrm{sep}}(\mathcal{I}_n^k) = \mathsf{true}$ with probability $1 - 1/|R|$.

To see the implications of the strict domain test, we review the results of [9], which state that batch verification without a strict domain test does not provide the same security as does separate verification. The example abuses the implicit assumption that h_{i0} is taken from the correct domain. Here, we assume that $\langle g \rangle$ is a multiplicative subgroup in $\mathbb{Z}_p$ where $ord(g) = q$, $p = 2q + 1$, and q is prime. Let $\mathcal{I}_2^k = \{(e_{i1}, \ldots, e_{ik}, h_{i0}, h_{i1} \ldots, h_{ik})\}_{i=1,2}$ be correct items with regard to the separate verification predicate. Note here that -1 is not in $\langle g \rangle$. Then, let $\tilde{\mathcal{I}}_2^k = \{(e_{i1}, \ldots, e_{ik}, (-1)h_{i0}, h_{i1}, \ldots, h_{ik})\}_{i=1,2}$. Note here that $\tilde{\mathcal{I}}_2^k$ fails in the separate verification, but it passes the batch verification with probability $1/2$. This happens when r_1 and r_2 are both even or odd.

2.3 Overlap Factor

In this section, we introduce the overlap factor, which is an essential parameter in the evaluation of verification cost in later sections.

The advantage of batch verification is based on the fact that

- computing $(n \times k)$-base exponentiation is faster than computing k-base exponentiation n times, and
- exponents for overlapped bases can be wrapped up.

Accordingly, batch relation Predicate (3) can be computed faster than repeatedly computing separate Predicate (2) n times. We refer to Algorithm 14.88 in [10] for an exponentiation algorithm with multiple bases. When many of the bases are common to all the items, the advantage clearly grows. We define *overlap factor* k_o as $k - (k_x + 1)$ such that it represents the number of bases commonly included in all the items.

$k_o = 0$	No base is common among the items. This occurs, for instance, in the predicate of zero-knowledge proofs that proves equality of the discrete logarithms, $\log_{g_1} y_1 = \log_{g_2} y_2, \ldots$
$k_o = 1$	Only one base is common among the items. Such a case would be observed in applications where all items are generated based on a single generator, say g.
$k_o \geq 2$	More than two bases are common. Such a case would be observed when the items are based on the representation problem such as the Okamoto identification scheme [11] or Brand's electronic cash scheme [1].

The advantage of batch verification is maximized when $k_o = k - 1$ where all but one base are common and minimized when $k_o = 0$ where all bases are different.

3 Summary of Our Results

We analyze the advantage of batch verification by comparing it to the separate verification. The advantage is estimated w.r.t. the computational cost. We say that it has advantage if it saves more than one exponentiation per item. Our results are summarized in Table 1. See Section 5 for the classification of groups and specific schemes of strict batch verification. The symbols **a**,**d** and **D** in Table 1 respectively denote

a: Strict batch verification is unconditionally advantageous,
d: Strict batch verification is disadvantageous, and
D: Lenient batch verification is unconditionally disadvantageous.

The lower case symbols (**a**,**d**,**q**, and **e**) also have an additional meaning,

"Lenient batch verification is unconditionally advantageous,"

and the upper case symbols (**E** and **D**) represent

"Strict batch verification is unconditionally disadvantageous."

What is interesting in Table 1 is

I-1 Almost all lower case areas are occupied by symbol **a**.
I-2 Symbols **d** and **D** have completely different reasons why the strict batch verification is disadvantageous.

Item I-1 means that strict batch verification can be performed efficiently on almost all practical groups on which lenient batch verification can be performed efficiently. Item I-2 implies the following.

- Symbol **d** implies that no effective domain test is found. If an effective test is found, strict batch verification becomes advantageous. The symbol **q** is a special case of **d**.
- Symbol **D** implies that batch verification cannot be faster than separate verification although an effective domain test is available. This means that the separate verification is sufficiently fast by itself. Symbols **E** and **e** are special cases of **D**.

Note that **D** never means that the binary field is unsuited to large-scale applications.

Table 1. Effectiveness of Batch Verification.

Section	Group	$k_o = 0$	$k_o = 1$	$k_o = 2$	$k_o \geq 3$
5.1	Subgroup of $\mathbb{Z}_p^*$ (random prime)	**d**	**d**	**q**	**a**
5.2	Subgroup of $\mathbb{Z}_p^*$ (special prime)	**a**	**a**	**a**	**a**
5.3	Prime Field	**a**	**a**	**a**	**a**
	OEF (random curve)	**a**	**a**	**a**	**a**
5.4	Binary Field (random curve)	**a**	**a**	**a**	**a**
5.5	OEF (special curve)	**E**	**e**	**a**	**a**
	Binary Field (special curve)	**D**	**D**	**a**	**a**

Parameter k_o is the number of common bases among all verification items.
a : Strict batch verification is unconditionally **a**dvantageous.
d : Strict batch verification is **d**isadvantageous if no effective test is found.
q : Strict batch verification is advantageous if the q-th power is fast,
e : Strict batch verification is advantageous if the **e**xtension degree is small.
E : Lenient batch verification is advantageous, if the **E**xtension degree is small.
D : Lenient batch verification is unconditionally **D**isadvantageous.

4 Basic Concepts for Analysis

4.1 Gain and Loss

In this section, we introduce the concept of gain and loss, which provides a simple criterion for the advantage of batch verification. In the rest of this paper, we estimate computation costs by the number of group operations in $\langle g \rangle$ and assume that all other arithmetic operations, equality tests, and random number generations are done for free.

The gain, c_{gain}, is defined as the difference in the cost per verification item between separate relation test (2) and batch relation test (3). That is, let C_{sep} and

C_{bat} be the computational cost of Predicate (2) and Predicate (3) respectively, then

$$c_{\text{gain}} = (C_{\text{sep}} - C_{\text{bat}})/n \tag{5}$$

and the loss, c_{loss}, is defined as the cost to ensure $h_0 \in \langle g \rangle$ for strict batch verification. That is the difference in the cost per verification item between lenient domain test (1) and strict domain test (4).

The concept of gain and loss provides a simple criterion for the advantage of batch verification: If the loss is less than the gain, then the batch verification is meaningful, otherwise it does not make any sense. The advantage of the batch verification is given by

$$c_{\text{loss}} < c_{\text{gain}} \tag{6}$$

For discussion on lenient batch verification, set $c_{\text{loss}} = 0$ since the domain test is the same as that in separate verification, while for strict batch verification, c_{loss} must be estimated.

4.2 Generic Evaluation of the Gain

This section estimates c_{gain} more precisely without specifying an underlying group. For this purpose, we focus on an abstract exponentiation scheme called the *'optimal'* exponentiation scheme, and we give an estimation of c_{gain} for such optimal exponentiation scheme. The concrete scheme may be the binary method, the signed binary method, or something window method [10]: however, this is not specified because it varies with the choice of the underlying group. We will discuss such variations in later sections in this paper.

Let G be the group we focus on where c_{m} is the computational cost of a multiplication on G and c_{s} is the cost of a square on G. Let N_{m} be the average number of multiplications used in the optimal exponentiation scheme and N_{s} be the average number of squares required by the scheme. The average cost of the exponentiation scheme on G is $N_{\mathsf{m}}c_{\mathsf{m}} + N_{\mathsf{s}}c_{\mathsf{s}}$. By extending the scheme [8,10], k-base exponentiation can be computed with costs $kN_{\mathsf{m}}c_{\mathsf{m}} + N_{\mathsf{s}}c_{\mathsf{s}}$. Therefore,

$$C_{\text{sep}} = n \times (kN_{\mathsf{m}}c_{\mathsf{m}} + N_{\mathsf{s}}c_{\mathsf{s}})$$

The dominant task of batch relation test (3) is the product of $nk_x + k_o$ exponentiations which costs

$$C_{\text{bat}} = (nk_x + k_o)N_{\mathsf{m}}c_{\mathsf{m}} + N_{\mathsf{s}}c_{\mathsf{s}}$$

In general, $V_{\text{sep}}(\mathcal{I}_n^k)$ and $V_{\text{bat}}(\mathcal{I}_n^k)$ can have different values of $N_{\mathsf{m}}, N_{\mathsf{s}}, c_{\mathsf{m}}$ and c_{s}. We ignore this effect, since N_{s} has only a slight effect on $V_{\text{bat}}(\mathcal{I}_n^k)$, and no significant difference is expected with respect to N_{m}. When G is a subgroup of $\mathbb{Z}_p^*$, c_{m} and c_{s} can be fixed. Even for elliptic curves, there is no significant difference between $V_{\text{sep}}(\mathcal{I}_n^k)$ and $V_{\text{bat}}(\mathcal{I}_n^k)$[12]. Then from Definition (5), we have

$$c_{\text{gain}} = \left((k_o - 1) - \frac{k_o}{n}\right) N_{\mathsf{m}}c_{\mathsf{m}} + \left(1 - \frac{1}{n}\right) N_{\mathsf{s}}c_{\mathsf{s}}$$

Since we are only concerned with a large number of items, we assume that $k_o \ll n$ (and $1 \ll n$). Then we have

$$c_{\text{gain}} \approx (k_o - 1)N_{\mathsf{m}}c_{\mathsf{m}} + N_{\mathsf{s}}c_{\mathsf{s}} \tag{7}$$

Through the rest of this paper, we estimate the efficiency averaged among the verification items, since the above approximation is independent of n. According to Approximation (7), the criterion $c_{\text{loss}} < c_{\text{gain}}$ is written by

$$c_{\text{loss}} < (k_o - 1)N_{\mathsf{m}}c_{\mathsf{m}} + N_{\mathsf{s}}c_{\mathsf{s}} \tag{8}$$

We call this inequality *'Criterion'* (8). Let q be the order of G: for almost all practical exponentiation schemes[1] , we observe

$$N_{\mathsf{m}} = \frac{1}{\omega}\lfloor \log_2 q \rfloor \ , \ \ N_{\mathsf{s}} = \lfloor \log_2 q \rfloor \tag{9}$$

where $2 \le \omega$. We call factor ω the *'abstract window size'*, which depends on the exponentiation scheme (and q). For example, $\omega = 2$ for the binary method, $\omega = 8/3$ for the signed binary method, and will be greater for a more sophisticated method [10]. In general, $\omega \ll \log_2 q$, and if $q \sim 2^{160}$ then we observe $\omega \lesssim 5$. Suppose that $c_{\mathsf{m}} \approx c_{\mathsf{s}}$, then from Inequality (8) we have

$$c_{\text{loss}}/c_{\mathsf{m}} < \left(\frac{1}{\omega}(k_o - 1) + 1\right)\lfloor \log_2 q \rfloor \tag{10}$$

We also call this inequality Criterion (10). This criterion is the reason why lenient batch verification is more efficient than separate verification. That is if $c_{\text{loss}} = 0$, then Criterion (10) is always satisfied since $0 \le k_o$ and $2 \le \omega$. This means that lenient batch verification is always faster than separate verification. However, if either Approximation (7) or Observation (9) does not hold, we may have different results. We will deal with such a special case later in this paper.

Note that when $c_{\text{loss}} \approx c_{\text{gain}}$ batch verification is not recommended even if $c_{\text{loss}} < c_{\text{gain}}$, because the cost of random number generation, exponent number generation, etc. are ignored. Therefore, if $c_{\text{gain}} - c_{\text{loss}}$ is comparable to the decrease of one exponentiation base, that is

$$N_{\mathsf{m}}c_{\mathsf{m}} \lesssim c_{\text{gain}} - c_{\text{loss}}$$

then we ignore this matter, otherwise it is given special attention.

5 Detailed Analysis

5.1 Subgroup of $\mathbb{Z}_p^*$ for Random Prime p

In this section, we examine the order-q subgroup of $\mathbb{Z}_p^*$ without any special conditions of r such that $p = 2qr + 1$. We call such a prime, p, '*random prime*'.

[1] Instead, we assume that no fixed bases, no fixed powers, and no off-line precomputation tables are available.

We show that lenient batch verification is always faster than separate verification for this group, and investigate the conditions under which strict batch verification has an advantage over separate verification.

Let p, q be prime such that $q|(p-1)$, and let G be the order-q subgroup of $\mathbb{Z}_p^*$. We can use Criteria (8) and (10) in this case, which means that lenient batch verification is always faster than separate verification for this group. Therefore, we only discuss strict batch verification in this section. The ideal input of h_0 is an element of G. However, an element of G is practically represented as an element of $\mathbb{Z}_p$. There is an additional cost to ensure $h_0 \in G$. If p is a random prime such that $p-1$ has many small factors, there is no means except $h_0^q = 1$. Suppose that h_0^q is calculated by the window method or its derivation, then we have an approximation of c_{loss} as

$$c_{\text{loss}} \approx N_{\mathsf{m}} c_{\mathsf{m}} + N_{\mathsf{s}} c_{\mathsf{s}}$$

Then Criterion (8) is equivalent to $2 < k_o$. In this case, strict batch verification is useless unless it has three or more common exponentiation bases among all verification items. However, the gain c_{gain} is so large even if $k_o = 0$ that strict batch verification can be constructed if we find an efficient domain test for $h_0 \in G$. For example, if we use the binary method to calculate h_0^q, the loss can be estimated precisely as

$$c_{\text{loss}} = (w_{\mathrm{H}}(q) - 1)\, c_{\mathsf{m}} + \lfloor \log_2 q \rfloor c_{\mathsf{s}}$$

where $w_{\mathrm{H}}(q)$ is the Hamming weight of q. In this case Criterion (10) is equivalent to

$$w_{\mathrm{H}}(q) - 1 < \frac{1}{\omega}(k_o - 1)\lfloor \log_2 q \rfloor$$

If we choose q as $w_{\mathrm{H}}(q) \ll \lfloor \log_2 q \rfloor / \omega$, then we can relax the necessary condition for strict batch verification[2], that is $1 < k_o$.

5.2 Subgroup of $\mathbb{Z}_p^*$ for Special Prime p

In this section, we examine the order-q subgroup of $\mathbb{Z}_p^*$ on condition that all prime factors of r are greater than q. We call such a prime, p, '*special prime*'. According to the results in the previous section, lenient batch verification is always faster than separate verification in this case. Therefore, we discuss only strict batch verification. We show that strict batch verification is always faster than separate verification in this case.

Under this special condition, we can substitute $h_0^{qr} = 1$ for $h_0^q = 1$, and clearly the Legendre symbol (h_0/p) is available for this purpose, in which the computational complexity is $O((\log_2 p)^2)$. We assume that the Legendre symbol costs $\mathcal{L} \times c_{\mathsf{m}}$. Thus,

$$c_{\text{loss}} = \mathcal{L} \times c_{\mathsf{m}}$$

[2] Clearly, it is sufficient that the (quasi)optimal q-th power costs much less than c_{gain}, the necessary condition is independent of $w_{\mathrm{H}}(q)$.

Then Criterion (10) is equivalent to

$$\mathcal{L} < \left(\frac{1}{\omega}(k_o - 1) + 1\right) \lfloor \log_2 q \rfloor$$

If $p \sim 2^{1024}$, then we experimentally observe $\mathcal{L} \sim 10$ (according to GP/PARI CALCULATOR). We can satisfy this inequality for practical p, q (e.g., $p \sim 2^{1024}$, $q \sim 2^{160}$) even if $k_o = 0$ because $2 \leq \omega$. In this case, we conclude that the strict batch verification is always faster than separate verification for practical p, q.

5.3 $\mathbb{F}_p$-Rational Points on $E/\mathbb{F}_p$ ($3 <$ char $\mathbb{F}_p$)

Let $\mathbb{F}_p$ be a finite field, we define an elliptic curve over $\mathbb{F}_p$ as

$$E/\mathbb{F}_p : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \ , \ (a_i \in \mathbb{F}_p) \tag{11}$$

Let $\mathbb{F}_{p^m}$ be the extension of $\mathbb{F}_p$ of degree m. The group of $\mathbb{F}_{p^m}$-rational points on $E/\mathbb{F}_p$ which we denote $E(\mathbb{F}_{p^m})$, is defined as a set of solutions $(x, y) \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m}$ on $E/\mathbb{F}_p$ together with a point at infinity denoted by $\mathcal{O}$.

Groups of elliptic curves are classified by its definition field. In this section we assume $m = 1$. If p is a large prime, we call it '*prime field*.' If p is a prime power such that $3 <$ char $\mathbb{F}_p$, we call it '*OEF*.' We discuss both of them here. Criterion (10) is available in this case. Therefore, we can recapitulate the results of Section 4.2 for lenient batch verification. Thus, we discuss only strict batch verification and show that it is always faster than separate verification.

Although h_0 must be an element of $E(\mathbb{F}_p)$, an element of $E(\mathbb{F}_p)$ is practically represented as $\mathbb{F}_p \times \mathbb{F}_p$. In this case, we can choose curves with a prime order, and all recommended curves over a prime field in [13] have a prime order. Here h_0 is given as $h_0 = (x, y) \in \mathbb{F}_p^2$ and clearly all that is necessary is to check if the given point satisfies Curve equation (11). It costs less than only one elliptic addition on $E(\mathbb{F}_p)$. It is sufficient for Criterion (10) if

$$1 < \left(\frac{1}{\omega}(k_o - 1) + 1\right) \lfloor \log_2 q \rfloor$$

When $q > 4$, this will always be satisfied because we hold $0 \leq k_o$ and $2 \leq \omega$. We conclude that strict batch verification is practically always faster than separate verification in this case. Note that if this check is omitted, attackers could easily cheat the system into performing operations beyond the group, and we should acknowledge that lenient batch verification is useless for any practical cryptographic applications in this case. If we choose compressed representation (See Appendix C), there is less of an advantage for the attacker than in the case of redundant representation, but they can choose x such that we cannot solve y on $\mathbb{F}_p$. Some algorithms to calculate y may not detect this attack and give a false solution, in such a case we must check Curve equation (11).

5.4 Subgroup of $\mathbb{F}_{2^m}$-Rational Points on $E/\mathbb{F}_{2^m}$

On Curve equation (11), if p is a power of 2, we call the groups of elliptic curve '*binary field*', which is the subject of this section. Criterion (10) is valid for this case, then we can apply the results of Section 4.2 for lenient batch verification. Therefore, we discuss only strict batch verification and show that it is always faster than separate verification.

In this case, it is not sufficient only to test Curve equation (11) if we want to ensure $h_0 \in G$, because all non-supersingular curves must have an even order. We can choose a curve whose order can be factorized into a 2 fold large prime and these types of curves have an effective large subgroup domain test. That is

$$T(x) \overset{?}{=} T(a_2) \tag{12}$$

where $T(x)$ is the *trace* of x-coordinate and $T(a_2)$ is the trace of the curve parameter a_2. See [14] for details. All recommended random curves over $\mathbb{F}_{2^m}$ in [13] have this type of order. Slight calculations are necessary for this (and Curve equation (11)).

The situation is the same as in Section 5.3. We conclude that strict batch verification is always faster than separate verification.

5.5 Subgroup of $\mathbb{F}_{p^m}$-Rational Points on $E/\mathbb{F}_p$ $(1 < m)$

On '*binary field*' and '*OEF*', we can chose '*special curves*' called '*Koblitz curve*' [15,16] to accelerate the calculation. In this section, we discuss such special curves and we show that lenient batch verification is not always faster than separate verification in this case. Afterwards, we propose an efficient domain test and investigate the conditions under which strict batch verification becomes faster than separate verification.

Let p be prime or a prime power. Let $E(\mathbb{F}_{p^m})$ be a $\mathbb{F}_{p^m}$-rational point group on $E/\mathbb{F}_p$. Because $E(\mathbb{F}_{p^m})$ must have a small order subgroup $E(\mathbb{F}_p)$, the order of $E(\mathbb{F}_{p^m})$ is essentially a composite. We must chose the curve parameters such that the curve has no small subgroups except $E(\mathbb{F}_p)$. Therefore, m must be prime at least. All recommended curves over $\mathbb{F}_2$ in [13] have a composite order which can be factorized into a $\#E(\mathbb{F}_p)$ fold large prime. Let q be the large prime. Let G be an order-q subgroup of $E(\mathbb{F}_{p^m})$.

The optimal elliptic exponentiation scheme on G uses the Frobenius map ϕ which is a special constant multiplication that costs almost 0 [15,16]. And many elliptic doublings are replaced by ϕ. This does not allow us to apply Criteria (10) because Observation (9) is not true in this case. Instead of Observation (9) we should apply

$$N_{\mathsf{m}} = \frac{1}{\omega} \lfloor \log_2 q \rfloor \ , \ \ N_{\mathsf{s}} = \lfloor \log_2 p \rfloor$$

in this case [15,16]. Assume that $\log_2 q \sim (m-1) \times \log_2 p$, then

$$N_{\mathsf{s}} \approx \left\lfloor \frac{\log_2 q}{m-1} \right\rfloor$$

This means N_{s} becomes very small. If we suppose $c_{\mathsf{m}} \approx c_{\mathsf{s}}$ then $c_{\text{loss}} < c_{\text{gain}}$ is equivalent to

$$c_{\text{loss}}/c_{\mathsf{m}} < \frac{1}{\omega}(k_o - 1)\lfloor \log_2 q \rfloor + \left\lfloor \frac{\log_2 q}{m-1} \right\rfloor \tag{13}$$

We discuss lenient batch verification with the right side of Inequality (13) later in this section.

On the other hand we need the estimation of the loss for strict batch verification. In this case it is not sufficient to check Curve equation (11) if we want to ensure $h_0 \in G$ (however, it is still necessary). We must find an effective test to ensure that h_0 belongs to the order-q subgroup of $E(\mathbb{F}_{q^m})$. We can apply the Frobenius map ϕ for this purpose and we propose an efficient domain test[3]. Assume that $m > 1$ and m is relatively prime to $\#E(\mathbb{F}_p)$, then for given point P on the curve, $P \in G$ is equivalent to

$$\sum_{i=0}^{m-1} \phi^i P = \mathcal{O} \tag{14}$$

(See Appendix A). In general, the computational complexity of a Frobenius map ϕ^i is at most as much as that of a multiplication on $\mathbb{F}_{p^m}$, and can be much smaller if we choose a special representation for $\mathbb{F}_{p^m}$. Therefore, the cost of the domain test is estimated as the number of necessary elliptic additions for (14). We obtain at most

$$c_{\text{loss}}/c_{\mathsf{m}} \approx \lfloor \log_2 m \rfloor + w_{\text{H}}(m) - 1$$

where $w_{\text{H}}(m)$ is the Hamming weight of m (See Appendix B). According to the overlap factor, k_o, the advantage of lenient or strict batch verification becomes apparent.

- $k_o = 0$: If $q \sim 2^{160}$ then the abstract window size, ω, is at most 5 or a similar value in this case. Therefore, if $\omega < m-1$, then the right side of Inequality (13) will become negative. This means if $k_o = 0$, then lenient batch verification is not faster than separate verification on $E(\mathbb{F}_{2^m})$ and most of $E(\mathbb{F}_{p^m})$. Obviously strict batch verification is not faster than that.
- $k_o = 1$: If $m > \log_2 q + 1$, then the right side of Inequality (13) becomes 0, and if m is comparable to $\log_2 q$ then the gain becomes almost 0. This means lenient batch verification is slower then separate verification on $E(\mathbb{F}_{2^m})$. Furthermore, when $q \sim 2^{160}$, to satisfy Inequality (13) for strict batch verification, we need $m < 23$. The gain is so small that we do not recommend strict batch verification in this case.
- $k_o \geq 2$: The right side of Inequality (13) is always positive, and lenient and strict batch verifications are always faster than separate verification for a practical case.

[3] The trace test is also available for some special case of $q = 2$, but finally the same results are obtained.

References

1. Brands, S.: Untraceable Off-line Cash in Wallet with Observers. In Stinson, D., ed.: Advances in Cryptology — CRYPTO'93. Volume 773 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1993) 302–318
2. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme — A practical solution to the implementation of a voting booth —. In Guillou, L.C., Quisquater, J.J., eds.: Advances in Cryptology — EUROCRYPT'95. Volume 921 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1995) 393–403
3. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In Nyberg, K., ed.: Advances in Cryptology — EUROCRYPT'98. Volume 1403 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1998) 437–447
4. Abe, M.: Mix-networks on Permutation Networks. In Lam, K., Okamoto, E., Xing, C., eds.: Advances in Cryptology — ASIACRYPT'99. Volume 1716 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1999) 258–273
5. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation network. In Kim, K., ed.: Public Key Cryptography 4thInternational Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001. Volume 1992 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (2001) 317–324
6. Naccache, D., M'Raïhi, D., Vaudenay, S., Raphaeli, D.: Can D.S.A be Improved ? - Complexity Trade-Offs with the Digital Signature Standard -. In Santis, A.D., ed.: Advances in Cryptology — EUROCRYPT'94. Volume 950 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1995) 77–85
7. Sung-Ming Yen, Chi-Sung Laih: Improved Digital Signature Suitable for Batch Verification. IEEE Transactions on Computers **44** (July 1995) 957–959
8. Bellare, M., Garay, J.A., Rabin, T.: Fast Batch Verification for Modular Exponentiation and Digital Signatures. In Nyberg, K., ed.: Advances in Cryptology — EUROCRYPT'98. Volume 1403 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1998) 236–250
9. Boyd, C., Pavlovski, C.: Attacking and Repairing Batch Verification Schemes. In Okamoto, T., ed.: Advances in Cryptology — ASIACRYPT2000. Volume 1976 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (2000) 58–71
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press (1997)
11. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Brickell, E.F., ed.: Advances in Cryptology — CRYPTO'92. Volume 740 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1993) 31–53
12. Cohen, H., Miyaji, A., Ono, T.: Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In Ohta, K., Pei, D., eds.: Advances in Cryptology — ASIACRYPT'98. Volume 1514 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1998) 51–65
13. NIST: Recommended Elliptic Curves for Federal Government Use (1999) (available at `http://csrc.nist.gov/csrc/fedstandards.html/`).
14. Seroussi, C.: Compact Representation of Elliptic Curve Points over $\mathbb{F}_{2^n}$ (April 1998) Research Manuscript, Hewlett-Packard Laboratories,.

15. Koblitz, N.: CM-Curves with Good Cryptographic Properties. In Feigenbaum, J., ed.: Advances in Cryptology — CRYPTO'91. Volume 576 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1992) 279–287
16. Kobayashi, T., Morita, H., Kobayashi, K., Hoshino, F.: Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic. In Stern, J., ed.: Advances in Cryptology — EUROCRYPT'99. Volume 1592 of Lecture Notes in Computer Science., Berlin; Heidelberg; New York, Springer-Verlag (1999) 176–189 (A preliminary version was written in Japanese and presented at SCIS'99-W4-1.4).
17. Knuth, D.E.: Seminumerical Algorithms. Third edn. Volume 2 of The Art of Computer Programming. Addison Wesley (1997)
18. IEEE P1363/D13 (Draft Version 13): Standard Specifications for Public Key Cryptography Annex E(Informative) Formats (1999) (available at `http://grouper.ieee.org/groups/1363/P1363/draft.html`).

Appendix A

Why $\sum_{i=0}^{m-1} \phi^i P = \mathcal{O}$ Is Equivalent to $P \in G$?

Let ϕ be the Frobenius map on $E(\mathbb{F}_{p^m})$. $\phi - 1$ is an endomorphism of $E(\mathbb{F}_{p^m})$. According to the homomorphism theorem, we can decompose $E(\mathbb{F}_{p^m})$ into

$$\ker(\phi - 1) \oplus \mathrm{Im}(\phi - 1)$$

Note that $\ker(\phi - 1) = E(\mathbb{F}_p)$ and we define G as $\mathrm{Im}(\phi - 1)$, that is, for all $P \in E(\mathbb{F}_{p^m})$, there exists $P_{\ker} \in E(\mathbb{F}_p)$ and $P_{\mathrm{Im}} \in G$ such that $P = P_{\ker} + P_{\mathrm{Im}}$. Thus

$$\sum_{i=0}^{m-1} \phi^i P = mP_{\ker} + \sum_{i=0}^{m-1} \phi^i P_{\mathrm{Im}} = mP_{\ker}$$

since

$$\sum_{i=0}^{m-1} \phi^i P_{\mathrm{Im}} \in \mathrm{Im}(\phi^m - 1) = \{\mathcal{O}\}$$

Suppose that m is relatively prime to $\#E(\mathbb{F}_p)$, then we obtain

$$P \in G \iff \sum_{i=0}^{m-1} \phi^i P = \mathcal{O}$$

Note that m must be a prime when the curve has no small subgroup except $E(\mathbb{F}_p)$. Therefore, the condition is the same as $m \nmid \#E(\mathbb{F}_p)$ for our purposes.

Appendix B

Optimal Addition Sequence of $\sum_{i=0}^{m-1} \phi^i P$

Let $a_0, ..., a_n \in \mathbb{N}$ be an addition sequence of $m(= a_n)$ such that

$$a_0 = 1, \quad a_i = a_j + a_k, \ (j, k < i)$$

This yields the addition sequence of $\sum_{i=0}^{m-1} \phi^i P$ given by $A_0, ..., A_n \in E(\mathbb{F}_{p^m})$ such that

$$A_0 = P, \quad A_i = \phi^{a_k} A_j + A_k, \ (j, k < i)$$

then $A_n = \sum_{i=0}^{m-1} \phi^i P$. This causes the computational complexity of A_n to be $n(m)$ elliptic additions, where $n(m)$ represents the length of the optimal addition chain of m. According to the binary addition chain of m, at most

$$n(m) \leq \lfloor \log_2 m \rfloor + w_{\mathrm{H}}(m) - 1$$

See [17] for more a precise evaluation of $n(m)$.

Appendix C

Representation of Rational Point

In this appendix, we observe that the gain varies with the representation of a group if we use an elliptic curve.

Let h_0 be an element of $E(\mathbb{F}_p)$, which is practically represented as $\mathbb{F}_p \times \mathbb{F}_p$ or $\mathbb{F}_p \times \{0, 1\}$. We call the former *'redundant representation'*, and the latter *'compressed representation'* [18]. If we choose redundant representation, Approximation (7) and Criterion (8) are available for cost evaluation. On the other hand, Approximation (7) is not accurate if we choose the compressed representation. We estimate the gain in the compressed representation and show that the disadvantage is very small.

Since we need y or some replacement before we perform some group operations on h_0 and given h_0 has only x (and 1 bit for y) in compressed representation, we must solve y such that it satisfies Curve equation (11) on $\mathbb{F}_p$. Therefore, c_{gain} becomes smaller than Approximation (7), and unfortunately Criterion (10) is not available for this case. We must find a replacement for Criterion (10).

The dominant task of solving y is a square root on $\mathbb{F}_p$ or a similar calculation, which costs at most as much as one exponentiation on $\mathbb{F}_p$. In general, we may suppose that it works about 10 times faster than the elliptic exponentiation (scalar multiplication) because elliptic curve addition and doubling both require about 10 multiplications on $\mathbb{F}_p$. We assume that an exponentiation on $\mathbb{F}_p$ costs about $\epsilon_{\mathsf{m}} N_{\mathsf{m}} c_{\mathsf{m}} + \epsilon_{\mathsf{s}} N_{\mathsf{s}} c_{\mathsf{s}}$ where $\epsilon_{\mathsf{m}}, \epsilon_{\mathsf{s}} \lesssim 1/10$, because $p \sim q$ in this case. In general, the smaller the characteristic of $\mathbb{F}_p$ is, the smaller $\epsilon_{\mathsf{m}}, \epsilon_{\mathsf{s}}$ we obtain. Thus, we should substitute

$$c_{\text{gain}} \approx (k_c - 1 - \epsilon_{\mathsf{m}}) N_{\mathsf{m}} c_{\mathsf{m}} + (1 - \epsilon_{\mathsf{s}}) N_{\mathsf{s}} c_{\mathsf{s}}$$

for Approximation (7). We obtain the following instead of Criterion (10)

$$c_{\text{loss}} / c_{\mathsf{m}} < \left(\frac{1}{w} (k_c - 1 - \epsilon_{\mathsf{m}}) + 1 - \epsilon_{\mathsf{s}} \right) \lfloor \log_2 q \rfloor$$

Thus, the necessary conditions for batch verification are not so different from those of Criterion (10) except for ϵ_{m} and ϵ_{s}. This representation is more suited to communications.