How to (pre-)compute a ladder

Improving the performance of X25519 and X448

Thomaz Oliveira[1], Julio López[2], Hüseyin Hışıl[3], Armando Faz-Hernández[2], and Francisco Rodríguez-Henríquez[1]

[1] Computer Science Department, Cinvestav-IPN
thomaz.figueiredo@gmail.com, francisco@cs.cinvestav.mx
[2] Institute of Computing, University of Campinas
jlopez@ic.unicamp.br, armfazh@ic.unicamp.br
[3] Yasar University
huseyin.hisil@yasar.edu.tr

Abstract. In the RFC 7748 memorandum, the Internet Research Task Force specified a Montgomery-ladder scalar multiplication function based on two recently adopted elliptic curves, "curve25519" and "curve448". The purpose of this function is to support the Diffie-Hellman key exchange algorithm that will be included in the forthcoming version of the Transport Layer Security cryptographic protocol. In this paper, we describe a ladder variant that permits to accelerate the fixed-point multiplication function inherent to the Diffie-Hellman key pair generation phase. Our proposal combines a right-to-left version of the Montgomery ladder along with the pre-computation of constant values directly derived from the base-point and its multiples. To our knowledge, this is the first proposal of a Montgomery ladder procedure for prime elliptic curves that admits the extensive use of pre-computation. In exchange of very modest memory resources and a small extra programming effort, the proposed ladder obtains significant speedups for software implementations. Moreover, our proposal fully complies with the RFC 7748 specification. Our estimates suggest that a full implementation of our pre-computable ladder should outperform state-of-the-art software implementations of the X25519 and X448 functions by a 40% speedup when working in the fixed-point scenario.

1 Introduction

Since the last decades, Elliptic Curve Cryptography (ECC) has been largely used for achieving highly secure and highly efficient cryptographic communication implementations. In particular, ECC has become the prime choice for realizing key exchange and digital signature-verification protocols. However, reports released in 2013 suggested that the National Security Agency (NSA) secretly introduced backdoors to internationally-used encryption standards [36]. Immediately thereafter, new revelations [35] indicated that the same agency had tampered the elliptic curve-based pseudorandom number generator standard Dual_EC_DRBG, which was consequently removed from the SP 800-90A specification by NIST [29,30].

These events prompted, in 2014, that the Transport Layer Security (TLS) working group of the Internet Engineering Task Force requested from the Crypto Forum Research Group (CFRG), recommendations of new elliptic curves to be integrated into the next version of the TLS protocol [37]. The requirements for the selection of such curves were based on [5,34], which advocate for a number of design practices and elliptic curve properties, including rigidity in the curve-generation process and simplicity in the implementation of cryptographic algorithms. After a long and lengthy discussion, two prime elliptic curves, known as Curve25519 and Curve448, were chosen for the 128-bit and 224-bit security levels, respectively (see §3 for more details). The RFC 7748 [25] memorandum describes the implementation details related to this choice, including the curve parameters and the Montgomery ladder-based scalar multiplication algorithms, also referred to as X25519 and X448 functions.

The Montgomery ladder and Montgomery curves were introduced in [27]. Since then, the Montgomery ladder has been carefully studied by many authors, as discussed for example, in the survey by Costello and Smith in [12] (see also [7]). We know now how to use the Montgomery ladder for computing the point multiplication kP, where P is usually selected as a point that belongs to a prime order r subgroup of an elliptic curve, and k is an integer in the $[1, r-1]$ interval. Nevertheless, arguably the most important application of the Montgomery ladder lies in the Diffie-Hellman shared-secret computation as described in [25].

The classical Montgomery ladder as it was presented in [27], is a left-to-right scalar multiplication procedure that does not admit in a natural way efficient precomputation mechanisms. In an effort to obtain this feature, and in the context of binary elliptic curves, the authors of [31] presented a right-to-left Montgomery ladder that can take advantage of pre-computing multiples of the fixed base point P. Notice that this procedure was previously reported by Joye in [20]. However, the procedure presented in [31] crucially depended on the computation of the point halving operation. Although this primitive can be performed at a low computational cost in binary elliptic curves, in general there are no known procedures to compute it efficiently for elliptic curves defined over odd prime fields. Hence, it appeared that the finding of the right-to-left ladder procedure of [31] was circumscribed to binary elliptic curves, as there was no obvious way to extend it to elliptic curves defined over large prime fields.

Our contributions In this paper, we present an alternative way to compute the key exchange protocol presented in [25]. In short, we propose different X25519 and X448 functions which can take advantage of the fixed-point scenario provided by the Diffie-Hellman key generation phase. This algorithm achieves an estimated performance increase of more than 40% at the price of a small amount of extra memory resources. In addition, it does not intervene with the original RFC specification and it is straightforward to implement, preserving the simplicity feature of the original design.

The remainder of this paper is organized as follows. In §2 we briefly describe the Diffie-Hellman protocol. In §3 we give more details on the CFRG selected elliptic curves. The Montgomery ladder-based scalar multiplication functions X25519 and X448 are analyzed in §4. Our proposal is discussed in §5 and our concluding remarks and future work are presented in §6.

2 The Diffie-Hellman protocol

The Diffie-Hellman key exchange protocol, introduced by Diffie and Hellman in [14], is a method that allows to establish a shared secret between two parties over an insecure channel. Originally proposed for multiplicative groups of integers modulo p, with p a prime number, the scheme was later adapted to additively-written groups of points on elliptic curves by Koblitz and Miller in [21,26]. Commonly known as elliptic curve Diffie-Hellman protocol (ECDH), this variant is concisely described in Algorithm 1.

Algorithm 1 The elliptic curve Diffie-Hellman protocol

Public parameters: Prime p, curve $E/\mathbb{F}_p$, point $P = (x, y) \in E(\mathbb{F}_p)$ of order r

Phase 1: Key pair generation

	Alice		**Bob**
1:	Select the private key $d_A \xleftarrow{\$} [1, r-1]$	1:	Select the private key $d_B \xleftarrow{\$} [1, r-1]$
2:	Compute the public key $Q_A \leftarrow d_A P$	2:	Compute the public key $Q_B \leftarrow d_B P$

Phase 2: Shared secret computation

	Alice		**Bob**
3:	Send Q_A to Bob	3:	Send Q_B to Alice
4:	Compute $R \leftarrow d_A Q_B$	4:	Compute $R \leftarrow d_B Q_A$

Final phase: The shared secret is the point R x-coordinate

As shown in Algorithm 1, the ECDH protocol is divided into two phases; in the first phase, both parties generate their private and public key pair. The private key d_A (d_B) is an integer chosen uniformly at random from the interval $[1, r-1]$ while the public key Q_A (Q_B) is the resulting point of the scalar multiplication of d_A (d_B) by the base-point P. In the majority of the proposed elliptic curve-based standards and specifications (e.g. [15,9,28], including [25]), the point P is fixed and its coordinates are explicitly given in the documentation. At the implementation level, this setting is usually called fixed- or known-point scenario.

After computing their respective public/private key pair, each party sends her public key to the other. Next, they perform the point multiplication of the received public key by their own private key. The group properties of $E(\mathbb{F}_p)$

guarantee that

$$R = d_A Q_B = d_A(d_B P) = d_B(d_A P) = d_B Q_A = R.$$

As a result, the parties have access to a common piece of information, represented by the x-coordinate of R, which is only disclosed to themselves. [4] Since the public key Q_B (Q_A) is not known a priori by Alice (Bob), the scalar multiplication in the second phase is said to be performed in a variable- or unknown-point scenario.

3 The curves

The [25] memorandum specifies two Montgomery elliptic curves of the form,

$$E_A/\mathbb{F}_p : v^2 = u^3 + Au^2 + u. \tag{1}$$

The standard specification for the 128 bits of security level uses the prime $p = 2^{255} - 19$, and the curve parameter is given by $A = 486662$. This curve is commonly known as Curve25519 and was proposed in 2005 by Bernstein [1]. The point group order is given as $\#E_{486662}(\mathbb{F}_{2^{255}-19}) = h \cdot r \approx 2^{255}$, with $h = 8$ and

$$r = 2^{252} + 27742317777372353535851937790883648493.$$

The order-r base-point $P = (u, v)$ is specified as,

$$\begin{aligned} u_P =& \texttt{0x9} \\ v_P =& \texttt{0x20AE19A1B8A086B4E01EDD2C7748D14C} \\ & \texttt{923D4D7E6D7C61B229E9C5A27ECED3D9}. \end{aligned}$$

The recommendation for the 224-bit security level is to use $p = 2^{448} - 2^{224} - 1$ and $A = 156326$. This curve was originally proposed by Hamburg in the Edwards form as Ed448-Goldilocks [19], but it is referred in [25] as Curve448. The group order $\#E_{156326}(\mathbb{F}_{2^{448}-2^{224}-1}) = h \cdot r \approx 2^{448}$, with $h = 4$ and,

$$\begin{aligned} r =& 2^{446} - 13818066809895115352007386748515\backslash \\ & 426880336692474882178609894547503885. \end{aligned}$$

For this curve, the base-point P is given by

$$\begin{aligned} u_P =& \texttt{0x5} \\ v_P =& \texttt{0x7D235D1295F5B1F66C98AB6E58326FCECBAE5D34F55545D060F75DC2} \\ & \texttt{8DF3F6EDB8027E2346430D211312C4B150677AF76FD7223D457B5B1A}. \end{aligned}$$

[4] Here, we are considering an ideal but unrealistic scenario. In practice, an inappropriate choice of the elliptic curve parameters, the prime p, the order r, the implementation of the scalar multiplication algorithm, among many other aspects, could disqualify this statement.

4 The scalar multiplication operation

Let $E_A/\mathbb{F}_p$ be an elliptic curve and P an order-r point in $E_A(\mathbb{F}_p)$. Then, for any n-bit scalar $k = (k_{n-1}, \ldots, k_2, k_1, k_0)_2 \in [1, r-1]$, the scalar multiplication operation is given by,

$$Q = kP = k_{n-1}2^{n-1}P + \cdots + k_2 2^2 P + k_1 2P + k_0 P.$$

As presented in §2, the scalar multiplication function is used in the two first ECDH phases; first, to generate the public keys Q_A and Q_B and later, in the second phase, to compute the common point R.

4.1 Left-to-right Montgomery ladder

Initially proposed to improve the performance of integer factorization algorithms, the Montgomery ladder [27] is now largely used in the design of constant-time scalar multiplication implementations. This is because its ladder step structure assures that the same arithmetic operations are executed independently of the scalar bits k_i values. A high-level description of this procedure is presented in Algorithm 2.

Algorithm 2 Left-to-right Montgomery ladder

Input: $P = (u_P, v_P) \in E_A(\mathbb{F}_p)$, $k = (k_{n-1} = 1, k_{n-2}, \ldots, k_1, k_0)_2$
Output: $u_{Q=kP}$

$R_0 \leftarrow \mathcal{O}$; $R_1 \leftarrow u_P$;
for $i = n-1$ **downto** 0 **do**
  if $k_i = 1$ **then**
    $R_0 \leftarrow R_0 + R_1$; $R_1 \leftarrow 2R_1$
  else
    $R_1 \leftarrow R_0 + R_1$; $R_0 \leftarrow 2R_0$
  end if
end for
return $u_Q \leftarrow R_0$

If the difference between the points R_1 and R_0 is known, it is possible to derive efficient formulas for computing $R_0 + R_1$ that refer only to the u-coordinates of the operands, a formula that is sometimes named as differential addition [12]. [5] That is the main rationale for Algorithm 2; throughout its execution, the Montgomery ladder maintains the invariant $R_1 - R_0 = P$ by computing at each iteration

$$(R_0, R_1) \leftarrow \begin{cases} (2R_0, 2R_0 + P), & \text{if } k_i = 0 \\ (2R_0 + P, 2R_0 + 2P), & \text{if } k_i = 1. \end{cases}$$

[5] It is also possible to express the u-coordinate of the resulting point $R_i = 2R_i$, for $i \in \{0, 1\}$, using only the u-coordinate of the operand P, an operation known as differential doubling.

In order to avoid expensive field inversions, one can accelerate the scalar multiplication procedure by using projective coordinates, by means of the transformation $u = U/Z$.

In the context of Algorithm 2, the differential addition formula required in Step 6 can be computed as [12,25],

$$\begin{aligned} U_{R_1} &\leftarrow Z_P((U_{R_1} + Z_{R_1}) \cdot (U_{R_0} - Z_{R_0}) + (U_{R_1} - Z_{R_1}) \cdot (U_{R_0} + Z_{R_0}))^2 \\ Z_{R_1} &\leftarrow u_P((U_{R_1} + Z_{R_1}) \cdot (U_{R_0} - Z_{R_0}) - (U_{R_1} - Z_{R_1}) \cdot (U_{R_0} + Z_{R_0}))^2. \end{aligned} \tag{2}$$

where the standard trick of use $Z_p = 1$, saves one field multiplication. Thus, it can be seen that the computational cost of performing the differential addition formula of Eq. (2) is of $3\mathbf{m} + 2\mathbf{s} + 6\mathbf{a}$.

Similarly, the differential point doubling required in Step 6 of Algorithm 2 can be computed as [12,25],

$$\begin{aligned} U_{R_0} &\leftarrow (U_{R_0} + Z_{R_0})^2 \cdot (U_{R_0} - Z_{R_0})^2 \\ T &\leftarrow (U_{R_0} + Z_{R_0})^2 - (U_{R_0} - Z_{R_0})^2 \\ Z_{R_0} &\leftarrow \left[a_{24} \cdot T + (U_{R_0} - Z_{R_0})^2\right] \cdot T, \end{aligned} \tag{3}$$

where $a_{24} = \frac{A+2}{4}$. It can be readily seen that the computational cost of performing the differential doubling formula of Eq. (3) is of $2\mathbf{m} + 1\mathbf{m_{a24}} + 2\mathbf{s} + 4\mathbf{a}$.[6]

A low-level description of the left-to-right ladder on prime elliptic curves in Montgomery form is given in Algorithm 3. [7] When computed with the parameters listed in §3, this algorithm is called X25519 (with $n = 255$) or X448 (with $n = 448$) [25]. The $\oplus$ notation stands for the exclusive-or logical operator, while the symbols $+, -, \times, ^2$ and $^{-1}$ represent the field $\mathbb{F}_p$ arithmetic operations of addition, subtraction, multiplication, squaring and inversion, respectively.

At each iteration i of Algorithm 3, the conditional swap function (cswap) exchanges the values of the R_0 and R_1 coordinates when the bits k_{i-1} and k_i are different. This function is a countermeasure for potential cache-based attacks [22,23], which could reveal the scalar digits (the private key in Alg. 1) by determining the access order of the points R_0 and R_1. The cswap function consists only of simple logic operations, so its cost will be disregarded in our estimations. For more details on the implementation of this function see [25,31].

Cost estimations Let $\mathbf{m}$, $\mathbf{m_{a24}}$, $\mathbf{m_{uP}}$, $\mathbf{s}$, $\mathbf{i}$ and $\mathbf{a}$ represent the cost of a general multiplication, multiplication by the constant $(A + 2)/4$, multiplication by the u-coordinate of the base-point P, squaring, inversion and addition/subtraction over the field $\mathbb{F}_p$, respectively. Then, the computing cost of the left-to-right Montgomery ladder is

$$n \cdot (4\mathbf{m} + 1\mathbf{m_{a24}} + 1\mathbf{m_{uP}} + 4\mathbf{s} + 8\mathbf{a}) + 1\mathbf{m} + 1\mathbf{i}.$$

[6] were $\mathbf{m_{a24}}$ stands for one multiplication by the constant a_{24}.

[7] The description is closely related to [25, §5].

Algorithm 3 Low-level left-to-right Montgomery ladder

Input: $P = (u_P, v_P) \in E_A/\mathbb{F}_p$, $k = (k_{n-1} = 1, k_{n-2}, \ldots, k_1, k_0)_2$, $a_{24} = (A+2)/4$
Output: $u_{Q=kP}$

Initialization: $U_{R_0} \leftarrow 1$, $Z_{R_0} \leftarrow 0$, $U_{R_1} \leftarrow u_P$, $Z_{R_1} \leftarrow 1$, $s \leftarrow 0$
for $i \leftarrow n-1$ **downto** 0 **do**
  `# timing-attack countermeasure`
  $s \leftarrow s \oplus k_i$
  $U_{R_0}, U_{R_1} \leftarrow \mathrm{cswap}(s, U_{R_0}, U_{R_1})$
  $Z_{R_0}, Z_{R_1} \leftarrow \mathrm{cswap}(s, Z_{R_0}, Z_{R_1})$
  $s \leftarrow k_i$
  `# common operations`
  $A \leftarrow U_{R_0} + Z_{R_0}$; $B \leftarrow U_{R_0} - Z_{R_0}$
  `# addition`
  $C \leftarrow U_{R_1} + Z_{R_1}$; $D \leftarrow U_{R_1} - Z_{R_1}$
  $C \leftarrow C \times B$; $D \leftarrow D \times A$
  $U_{R_1} \leftarrow D + C$; $U_{R_1} \leftarrow U_{R_1}^2$
  $Z_{R_1} \leftarrow D - C$; $Z_{R_1} \leftarrow Z_{R_1}^2$; $Z_{R_1} \leftarrow u_P \times Z_{R_1}$
  `# doubling`
  $A \leftarrow A^2$; $B \leftarrow B^2$
  $U_{R_0} \leftarrow A \times B$
  $A \leftarrow A - B$
  $Z_{R_0} \leftarrow a_{24} \times A$; $Z_{R_0} \leftarrow Z_{R_0} + B$; $Z_{R_0} \leftarrow Z_{R_0} \times A$
end for
$U_{R_0}, U_{R_1} \leftarrow \mathrm{cswap}(s, U_{R_0}, U_{R_1})$
$Z_{R_0}, Z_{R_1} \leftarrow \mathrm{cswap}(s, Z_{R_0}, Z_{R_1})$
$Z_{R_0} \leftarrow Z_{R_0}^{-1}$
$u_{R_0} \leftarrow U_{R_0} \times Z_{R_0}$
return $u_Q \leftarrow u_{R_0}$

More specifically, at the 128 bits of security level, the X25519 function costs

$$1021\mathbf{m} + 255\mathbf{m_{a24}} + 255\mathbf{m_{uP}} + 1020\mathbf{s} + 2040\mathbf{a} + 1\mathbf{i},$$

where each operation is performed in the prime field $\mathbb{F}_{2^{255}-19}$. At the 224-bit security level case, the cost for computing the function X448 is

$$1793\mathbf{m} + 448\mathbf{m_{a24}} + 448\mathbf{m_{uP}} + 1792\mathbf{s} + 3584\mathbf{a} + 1\mathbf{i},$$

with the arithmetic operations being carried out in the prime field $\mathbb{F}_{2^{448}-2^{224}-1}$.

5 How to (pre-)compute a ladder

Our proposal for improving the performance of the X25519 and X448 functions focuses in the first phase of the Diffie-Hellman protocol (see Alg. 1). There, the scalar multiplication is performed in the fixed-point setting. More specifically, the point operand is always the base-point described in the [25] document (see §3 for more details).

One possible solution for taking advantage of this scenario was published in [2], in the context of message signing. In short, the authors pre-compute the points $P_{ij} = i16^j P$, for $1 \leq i \leq 8$ and $0 \leq j \leq 63$ and represent the Curve25519 in Edwards form to process the scalar multiplication through a windowed variant of the traditional double-and-add method. In addition to the significant amount of required memory space, the main drawback of this approach is that complex cache-attack countermeasures need to be applied during the retrieval of the pre-computed points P_{ij}, which go against the principle of implementation simplicity promoted in [5,34].

Thus, instead of designing a timing-protected double-and-add algorithm, we suggest using a slightly modified version of the right-to-left Montgomery ladder presented in [31] as explained in the following subsection.

5.1 Right-to-left Montgomery ladder with pre-computation

Algorithm 4 Right-to-left Montgomery ladder

Input: $P = (u_P, v_P) \in E_A(\mathbb{F}_p)$, $k = (k_{n-1} = 1, k_{n-2}, \ldots, k_1, k_0)_2$
Output: $u_{Q=hkP}$

1: **Pre-computation:** Calculate and store u_{P_i}, where $P_i = 2^i P$, for $0 \leq i \leq n$
2: **Initialization:** Select an order-h point $S \in E_A(\mathbb{F}_p)$
3: $R_0 \leftarrow u_P$, $R_1 \leftarrow u_S$, $R_2 \leftarrow u_{P-S}$
4: **for** $i \leftarrow 0$ **to** $n-1$ **do**
5:   **if** $k_i = 1$ **then**
6:     $R_1 \leftarrow R_0 + R_1$ (with $R_2 = R_0 - R_1$)
7:   **else**
8:     $R_2 \leftarrow R_0 + R_2$ (with $R_1 = R_0 - R_2$)
9:   **end if**
10:   $R_0 \leftarrow u_{P_{i+1}}$
11: **end for**
12: **return** $u_Q = hR_1$

The operating principle of Algorithm 4, is to compute $Q = kP$ using the Montgomery differential arithmetic formulas for the point doubling and point addition operations. This is achieved by recording and storing the difference $R_0 - R_1$ in the point R_2 through the whole execution of the procedure. Indeed, in the case that the bit $k_i = 1$, then R_0 is added to the accumulator R_1 (Step 6) and the difference R_2 does not change, since the operation $2R_0 = R_0 + R_0$ is performed in Step 10. On the other hand, if $k_i = 0$, nothing is added to the accumulator R_1, so it is necessary to increase the difference R_2 by R_0 (Step 8) in order to account for the unconditional doubling performed in Step 10. Notice that at each iteration, the accumulator R_1 is updated in the same fashion as it would be done in a traditional right-to-left double-and-add algorithm. It follows that at the end of the main loop, $R_1 = kP + S$.

The reason why the accumulator R_1 must be initialized with a point $S \notin \langle P \rangle$ is because the differential formulas are not complete on Montgomery curves. Hence, one must prevent the cases where $R_0 = R_1$ or $R_0 = R_2$. One can eliminate S by performing a scalar multiplication by the cofactor h, thus obtaining

$$hR_1 = h \cdot (kP + S) = hkP + hS = hkP.$$

Notice that for Montgomery curves, the cofactor h is as little as four. So this last correction does not represent a computational burden. Furthermore, in § 5.4 we show a trick specially tailored for the X25519 and X448 functions, which eliminates the point S at almost no cost, and that allows us to return the correct $R_1 = kP$ result. Nevertheless, we stress that the points S and $P - S$ can be clearly specified beforehand and therefore, this matter should not bring any complications for the programmer.

Given that the difference between R_0 and R_1 is volatile, at first glance the differential point addition formula computed in Steps 6 and 8 of Algorithm 4, requires an extra field multiplication as compared with Eq. (2) of the classical ladder shown in Algorithm 2. This is basically because R_2 is now represented in full projective coordinates, which means that its Z-coordinate value will be in general different than one.

We discuss in the following how to compute the differential addition formula of Algorithm 4, without incurring in any additional cost as compared with the cost of Eq. (2) of Algorithm 2.

5.2 Montgomery differential addition with precomputation

Let $R_0 = (u_0, v_0)$ and $R_1 = (u_1, v_1)$, be two points of the elliptic curve of Eq.(1).[8] Then, the point $R_3 = (u_3, v_3)$, such that, $R_3 = R_0 + R_1$, is determined as,

$$\begin{aligned}(u_3, v_3) &= (u_0, v_0) + (u_1, v_1) \\ &= \left(\frac{u_0 v_1 - v_0 u_1}{u_0 v_1 + v_0 u_1} \cdot \frac{1 - u_0 u_1}{u_0 - u_1}, \frac{u_0 v_1 - v_0 u_1}{u_0 v_1 + v_0 u_1} \cdot \frac{v_0(u_1^2 - 1) - v_1(u_0^2 - 1)}{(u_0 - u_1)^2} \right).\end{aligned} \tag{4}$$

Let us assume that the point $R_2 = (u_2, v_2)$, such that $R_2 = R_0 - R_1$, is known. Then, the addition formulas (4) can be rewritten as the following differential addition formulas,

$$(u_3, v_3) = \left(\frac{1}{u_2} \cdot \frac{(1 - u_0 u_1)^2}{(u_0 - u_1)^2}, \frac{1}{v_2} \cdot \frac{v_0^2(1 - u_1^2)^2 - v_1^2(1 - u_0^2)^2}{(u_0 - u_1)^4} \right) \tag{5}$$

One can perform u-only arithmetic by transforming the above equation to customary projective coordinates as,

[8] Notice that in general an Montgomery elliptic curve has the form, $Bv^2 = u^3 + Au^2 + u$.

$$
\begin{aligned}
(U_3 : Z_3) &= \left(Z_2(U_0U_1 - Z_0Z_1)^2 : U_2(U_0Z_1 - Z_0U_1)^2\right) \\
&= \left(Z_2((U_1 + Z_1) + \mu(U_1 - Z_1))^2 : U_2((U_1 + Z_1) - \mu(U_1 - Z_1))^2\right)
\end{aligned} \tag{6}
$$

where

$$
\mu = \frac{(U_0 + Z_0)}{(U_0 - Z_0)}.
$$

The per-point-R_0 constant value μ can be precomputed and stored since it only depends on $(U_0 : Z_0)$. Computing $(U_3 : Z_3)$ in (6) takes only $3\mathbf{m} + 2\mathbf{s} + 4\mathbf{a}$, by reusing (U_1+Z_1) and $\mu(U_1-Z_1)$ on both sides. Notice that this exactly matches the computational cost of Eq. (2), which computes the differential addition of the classical Montgomery ladder. In Appendix A, a Magma [8] script verifying Eq.(6) is presented.

The corresponding differential addition formulas for Edwards and Huff forms are described in Appendices B and C, respectively. The formulas give exactly the same performance as the formulas presented for the Montgomery form. Therefore, they are not included in the main body of the text.

5.3 Differential addition formulas in Algorithm 4

In the context of Algorithm 4, the differential addition formula required in Steps 6 and 8 can be computed as,

$$
\begin{aligned}
U_{R_3} &\leftarrow Z_{R_2}((U_{R_1} + Z_{R_1}) + \mu(U_{R_1} - Z_{R_1}))^2 \\
Z_{R_3} &\leftarrow U_{R_2}((U_{R_1} + Z_{R_1}) - \mu(U_{R_1} - Z_{R_1}))^2,
\end{aligned} \tag{7}
$$

where,

$$
\mu = \frac{u_{R_0} + 1}{u_{R_0} - 1}.
$$

Once again, notice that the μ-values can be pre-computed and stored since they only depend on the u-coordinates of the points 2^iP.

Timing attacks Notice that no side-channel countermeasures are required to retrieve the values $\mu_i = \frac{u_{2^iP}+1}{u_{2^iP}-1}$ from memory, since they are public and do not have any direct correlation to the sensitive information contained in the scalar k.

Also, the addition performed in Step 8 is not a dummy operation. The correct value of the R_2 coordinates must be maintained in order to perform further additions in Step 6. Moreover, since $k_{n-1} = 1$, a computational fault induced at any iteration of Algorithm 4 would produce a wrong resulting point Q.

5.4 Implementing the pre-computable ladder

Before presenting a low-level description of the known-point scalar multiplication using Algorithm 4, we must examine the point S selection and how to optimize the processing of the scalar k.

Strategies When selecting the private key k (Alg. 1, Step 1), presumably to facilitate the programming effort, the X25519 specification [25] recommends to generate 32 bytes at random as $k = K_0 + K_1 2^8 + \ldots + K_{31} 2^{248}$ with byte-words $K_i \xleftarrow{\$} [0, 255]$, and to perform the following operations:

$$K_0 \leftarrow K_0 \wedge 248, \qquad K_{31} \leftarrow K_{31} \wedge 127, \qquad K_{31} \leftarrow K_{31} \vee 64,$$

where the symbols $\wedge$ and $\vee$ represent the logical conjunction and disjunction operators. For the X448 function, 56 randomly-chosen bytes are required, which are further processed as

$$K_0 \leftarrow K_0 \wedge 252, \qquad K_{55} \leftarrow K_{55} \vee 128.$$

Those procedures are equivalent to compute, respectively,

$$k'' \xleftarrow{\$} [0, 2^{251} - 1], \qquad k' \leftarrow k'' + 2^{251}, \qquad k \leftarrow 8 \cdot k'$$

and

$$k'' \xleftarrow{\$} [0, 2^{445} - 1], \qquad k' \leftarrow k'' + 2^{445}, \qquad k \leftarrow 4 \cdot k'.$$

Consequently, we decided to process only the bits of k' in the main loop of our function. At the end of the algorithm, as we eliminate the point S from the accumulator by multiplying it by h, we will have the correct resulting point $Q = h \cdot (k'P + S) = kP$. In order to obtain a non-invasive procedure with respect to the RFC specification, we simply start processing the scalar from the $(\log_2 h + 1)$-th bit of k.

Point S selection In the Curve25519 setting, we could select an order-8 point S. However, because of its elegant u-coordinate, we decided to choose the order-4 point:

$$\begin{aligned} u_S &= \texttt{0x1}, \\ v_S &= \texttt{0x6BE4F497F9A9C2AFC21FA77AD7F4A6EF635A11C72} \\ &\quad\ \texttt{84A9363E9A248EF9C884415}. \end{aligned}$$

The point $P - S$ is given by:

$$\begin{aligned} u_{P-S} &= \texttt{0x215132111D8354CB52385F46DCA2B71D440F6A51E} \\ &\quad\ \texttt{B4D1207816B1E0137D48290}, \\ v_{P-S} &= \texttt{0x5199331F1F5630BBFA49B1B1B02B207B493D0A63B} \\ &\quad\ \texttt{B4F8F01C011242F9C6E9E7C}. \end{aligned}$$

For the Curve448, the order-4 point S is given by:

$$u_S = \texttt{0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE}$$
$$\texttt{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE},$$
$$v_S = \texttt{0x45B2C5F7D649EED077ED1AE45F44D54143E34F714B71AA96C945AF01}$$
$$\texttt{2D1829750734CDE9FADDBDA4C066F7ED54419CA52C85DE1E8AAE4E6C}.$$

And the (u, v) coordinates of $P - S$ are:

$$u_{P-S} = \texttt{0xF0FAB725013244423ACF03881AFFEB7BDACDD1031C81B9672954459D}$$
$$\texttt{84C1F823F1BD65643ACE1B5123AC33FF1C69BAF8ACB1197DC99D2720},$$
$$v_{P-S} = \texttt{0x45CD0137F88682464AE12E4E2CFCEA7E9360F6FE1E04AE1C5065F397}$$
$$\texttt{533F2282EE2643E610A0CC8E9B07D43D47C9658D05E22F0F077395DD}.$$

Algorithm Next, in Algorithm 5, we present the low-level details of our approach. Again, the term n represents the bit length of $\#E_A(\mathbb{F}_p) = h \cdot r$ and $q = \log_2 h$. [9] The pre-computation phase (Step 1) consists of computing and storing the values $\mu_i = \frac{u_{P_i}+1}{u_{P_i}-1}$ for the multiples $P_i = 2^i P$. These $n - q$ field elements are computed a priori from the base-point P and their values are listed in Appendix D for both, Curve25519 and Curve448. Assuming that the architecture is byte-addressable, the memory space required for Curve25519 is approximately $(255 - 3) \cdot 32B \approx 8KB$, while in the Curve448 setting, we need $(448 - 2) \cdot 56B \approx 25KB$.

The conditional swap function is identical to the one used in Algorithm 3. However, in this case the inputs are the coordinates of the accumulator R_1 and the difference point R_2. Moreover, the s variable that controls the swap is set to one, since the Montgomery point additions, in terms of memory location, are always performed as $R_1 \leftarrow R_1 + 2^i P$ throughout the algorithm. Also, given that the most significant bit k_{n-1} is always equal to one, it is unnecessary to include another couple of cswap functions after the main loop. At the end of the algorithm (Steps 16-23), we must perform q consecutive point doublings to process the least significant bits of k and to eliminate the point S from the accumulator R_1.

Cost estimations The cost of the Algorithm 5 can be estimated as

$$(n - q) \cdot (3\mathbf{m} + 2\mathbf{s} + 4\mathbf{a}) + q \cdot (2\mathbf{m} + 1\mathbf{m_{a24}} + 2\mathbf{s} + 4\mathbf{a}) + 1\mathbf{m} + 1\mathbf{i}.$$

If the Curve25519 is used, then $n = 255$ and $q = 3$. As a result, the fixed-point scalar multiplication would cost

$$763\mathbf{m} + 3\mathbf{m_{a24}} + 510\mathbf{s} + 1020\mathbf{a} + 1\mathbf{i},$$

[9] For the sake of simplicity, in the remaining of this paper it will be assumed that h is a small power of two.

Algorithm 5 Low-level right-to-left Montgomery ladder

Input: $P = (u_P, v_P), S = (u_S, v_S), P - S = (u_{P-S}, v_{P-S}) \in E_A/\mathbb{F}_p$, $a_{24} = (A+2)/4$
$k = (k_{n-1} = 1, k_{n-2}, \ldots, k_1, k_0)_2$

Output: $u_{Q=kP}$

Pre-computation Let $P_i = 2^i P$. Compute and store the values $\mu_i = \frac{u_{P_i}+1}{u_{P_i}-1}$, for $0 \leq i \leq n-q-1$
Initialization: $U_{R_1} \leftarrow u_S$, $Z_{R_1} \leftarrow 1$, $U_{R_2} \leftarrow u_{P-S}$, $Z_{R_2} \leftarrow 1$, $s \leftarrow 1$
for $i \leftarrow 0$ **to** $n-q-1$ **do**
  `# timing-attack countermeasure`
  $s \leftarrow s \oplus k_{i+q}$
  $U_{R_1}, U_{R_2} \leftarrow \mathrm{cswap}(s, U_{R_1}, U_{R_2})$
  $Z_{R_1}, Z_{R_2} \leftarrow \mathrm{cswap}(s, Z_{R_1}, Z_{R_2})$
  $s \leftarrow k_{i+q}$
  `# addition`
  $A \leftarrow U_{R_1} + Z_{R_1}$; $B \leftarrow U_{R_1} - Z_{R_1}$
  $C \leftarrow \mu_i \times B$
  $D \leftarrow A + C$; $D \leftarrow D^2$
  $E \leftarrow A - C$; $E \leftarrow E^2$
  $U_{R_1} \leftarrow Z_{R_2} \times D$; $Z_{R_1} \leftarrow U_{R_2} \times E$
end for
for $i \leftarrow 0$ **to** $q-1$ **do**
  `# doubling`
  $A \leftarrow U_{R_1} + Z_{R_1}$; $A \leftarrow A^2$
  $B \leftarrow U_{R_1} - Z_{R_1}$; $B \leftarrow B^2$
  $U_{R_1} \leftarrow A \times B$
  $A \leftarrow A - B$
  $Z_{R_1} \leftarrow a_{24} \times A$; $Z_{R_1} \leftarrow Z_{R_1} + B$; $Z_{R_1} \leftarrow Z_{R_1} \times A$
end for
$Z_{R_1} \leftarrow Z_{R_1}^{-1}$
$u_{R_1} \leftarrow U_{R_1} \times Z_{R_1}$
return $u_Q \leftarrow u_{R_1}$

where the arithmetic operations are over $\mathbb{F}_{2^{255}-19}$. In the Curve448 context, $n = 448$ and $q = 2$. As a consequence, we have the following cost in terms of $\mathbb{F}_{2^{448}-2^{224}-1}$-operations:

$$1343\mathbf{m} + 2\mathbf{m_{a24}} + 896\mathbf{s} + 1792\mathbf{a} + 1\mathbf{i}.$$

These results show that, our approach saves more than 25% of general field multiplications. In addition, it completely eliminates the multiplication by u_P [10] and drastically reduces the number of multiplications by the constant $(A+2)/4$. In addition, it saves half of the field squarings and half of additions/subtractions.

[10] In fact, given that the difference of the point operands $P_i - R_1$ is variable, the $\mathbf{m_{uP}}$ operations were changed into two general multiplications and were included in the $\mathbf{m}$ operation count.

For the programmer, the only extra effort is to organize the pre-computed values in the memory and load them during the main loop execution, since the remaining field and logic operations are very similar to ones presented in Algorithm 3. In the next subsection, we present a comparative based on the arithmetic of state-of-the-art software implementations.

5.5 Comparison

In this part, we present a more concrete analysis of the performance efficiency of Algorithm 5. For this purpose, we measured the field arithmetic cost of different state-of-the-art constant-time software implementations of the Diffie-Hellman protocol on Curve25519 and Curve448. After that, we computed the ratios of $\mathbf{m_{a24}}$, $\mathbf{m_{uP}}$, $\mathbf{s}$ and $\mathbf{i}$ to $\mathbf{m}$, which are considered the most representative field arithmetic operations for scalar multiplication implementations. As a result, we were able to show the practical savings of our proposal in terms of general field multiplications $\mathbf{m}$.

Regarding the X25519 implementations, we selected the code from Bernstein et al. [2], which represents the $\mathbb{F}_{2^{255}-19}$ elements in radix-2^{51}, the AVX2 approach from Faz-Hernández and López [16] and the curve25519-donna library from Langley [24]. [11] For the X448 function, we considered the original implementation of Hamburg in [19]. The source code of [2,19] were downloaded from the eBACS [3] web page, the [16] implementation was shared by its authors via personal communication and the curve25519-donna library was retrieved from its GitHub repository [24].

Every field arithmetic code was compiled with the clang/LLVM compiler version 3.9 with optimization flags `-O3 -march=haswell -fomit-frame-pointer` and further benchmarked in an Intel Core i7-4700MQ 2.40GHz machine (Haswell architecture) with the Hyper Threading and Turbo Boost technologies disabled. The ratios are presented in Table 1.

The cost of the $\mathbf{m_{a24}}$ operation in the Bernstein et al. implementation was estimated as follows. After analyzing the assembly code, we concluded that $\mathbf{m_{a24}}$ takes 10 `movq`, 5 `mov`, 5 `shr`, 5 `add`, 4 `addq`, 5 `mulq` and 1 `imulq` machine instructions. Next, we added its latencies [17] and, to calculate a "lower bound" of our speed improvements, we applied an aggressive throughput of 0.25. Finally, given that the $\mathbf{m_{uP}}$ is similar to the $\mathbf{m_{a24}}$ operation, we also assumed a similar cost. In Table 2, we present the performance improvements of our proposal in terms of the general field multiplication.

The above comparison suggests that about 36.01 to 44.04% of speed-up can be reached in the first phase of the ECDH protocol by using Algorithm 5. When considering the complete Diffie-Hellman scheme, the improvement ranges from

[11] The benchmarking reports in [3] shows that the library of Chou [10] currently holds the speed record on computing the scalar multiplication over Curve25519. However, the author decided to embed the field arithmetic functions into the ladder step, in a single assembly code. Isolating the field operations would be impractical and could alter the author's original intentions.

Table 1. Ratios of selected arithmetic operations to the general field multiplication in state-of-the-art software implementations

Implementation	Ratios to **m**				
	$\mathbf{m_{a24}}$	$\mathbf{m_{uP}}$	**s**	**i**	**a**
Bernstein et al. [2]	$0.23^{\dagger}$	$0.23^{\dagger}$	0.76	203.29	< 0.1
Faz-Hernández and López [16]	0.28	0.41	0.96	84.33	< 0.1
Langley [24]	0.60	$1.00^{\ddagger}$	0.82	192.55	< 0.1
Hamburg [19]	0.24	$1.00^{\ddagger}$	0.75	405.00	< 0.1

† Estimated

‡ The general field multiplication (**m**) is used to implement this operation

Table 2. A comparative between Montgomery-ladder approaches in the fixed-point scenario

Implementation	**Estimated costs** †		**Diff.**
	Mont. ladder left-to-right (Alg. 3)	Mont. ladder right-to-left (Alg. 5)	
Bernstein et al. [2]	2116.89**m**	1354.58**m**	-36.01%
Faz-Hernández and López [16]	2260.48**m**	1337.77**m**	-40.82%
Langley [24]	2457.95**m**	1375.55**m**	-44.04%
Hamburg [19]	4097.52**m**	2420.48**m**	-40.93%

† Because of its negligible cost, the field addition/subtraction operation was not included

18.01 to 22.02%. In practice, these estimated savings can be further improved if we take into consideration compiler optimizations and the machine throughput. Moreover, while the field addition/subtraction cost is imperceptible if measured separately, it constitutes a significant part in the whole protocol execution timings.

6 Conclusion

In this work, we presented an alternative way to compute the elliptic curve Diffie-Hellman protocol with Montgomery ladders. Particularly, we focused on the key-generation phase, which can be characterized as a fixed-point scenario. For this phase, we assumed that the relevant multiples of the base-point can be pre-computed off-line, which helps to boost the computation of the scalar multiplication via a right-to-left variant of the Montgomery ladder. As a result we achieved, in the Curve25519 setting, estimated performance improvements that range from 36 to 44% of speed-up, at the price of just $8KB$ of memory space. Our proposal carefully minimizes coding modifications with respect to the specifications given in the RFC 7748 memorandum.

Currently, we do not have a high-speed software implementation of our approach. However, we provide in Appendix E, a Magma script of our X25519 alternative function, which shows that the generated public keys are in accordance with the test vectors listed in [25, §6.1]. In the near future, we intend to work on software implementations targeting different architectures to evaluate the real optimization speed-ups that can be accomplished by our approach.

We also would like to explore the potential savings that our ladder approach can bring for digital signature protocols and other elliptic-curve based protocols. Finally, building on the work of [32], we would like to explore a Montgomery ladder variant, which can be applied to prime elliptic curves equipped with efficient endomorphisms such as the Four$\mathbb{Q}$ elliptic curve [11]. For that kind of elliptic curves, the ladder variant presented in [32], allows for an important saving in the number of required point doubling operations when working in the fixed-point scenario.

Acknowledgments

We would like to thank Daniel Cervantes for his useful comments and Armando Faz-Hernández for sharing with us insightful comments about the point multiplication software library reported in [16]. We also thank Peter Dettman for benchmarking this proposal in Java [13] and providing useful comments.

References

1. D. J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *Proceedings of PKC 2006*, volume 3958 of *LNCS*, pages 207–228. Springer Berlin Heidelberg, 2006.
2. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.
3. D. J. Bernstein and T. Lange. eBACS: ECRYPT Benchmarking of Cryptographic Systems. `https://bench.cr.yp.to`. Accessed in Mar 2017.
4. D. J. Bernstein and T. Lange. Explicit-formulas database. `https://hyperelliptic.org/EFD/g1p/auto-edwards-yzsquared.html`. Accessed in May 2017.
5. D. J. Bernstein and T. Lange. SafeCurves: choosing safe curves for elliptic-curve cryptography. `http://safecurves.cr.yp.to`. Accessed in Mar 2017.
6. D. J. Bernstein and T. Lange. A complete set of addition laws for incomplete Edwards curves. *Journal of Number Theory*, 131(5):858–872, 2011.
7. D. J. Bernstein and T. Lange. Montgomery curves and the montgomery ladder. Cryptology ePrint Archive, Report 2017/293, 2017. `http://eprint.iacr.org/2017/293`.
8. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
9. Certicom Research. SEC 2: Recommended Elliptic Curve Domain Parameters, 2010. Version 2.0. Standards for Efficient Cryptography. `http://www.secg.org/sec2-v2.pdf`.
10. T. Chou. Sandy2x: New Curve25519 Speed Records. In *Proceedings of SAC 2015*, pages 145–160. Springer International Publishing, 2016.

11. C. Costello and P. Longa. Four$\mathbb{Q}$: Four-Dimensional Decompositions on a $\mathbb{Q}$-curve over the Mersenne Prime. In *Proceedings of ASIACRYPT 2015*, volume 9452 of *LNCS*, pages 214–235. Springer, 2015.
12. C. Costello and B. Smith. Montgomery curves and their arithmetic: The case of large characteristic fields. Cryptology ePrint Archive, Report 2017/212, 2017.
13. P. Dettman. Comments on "A note on how to (pre-)compute a ladder". Crypto Forum Research Group, April 2017. `http://tinyurl.com/l986vyj`.
14. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
15. ECC Brainpool. Standard Curves and Curve Generation, 2005. Version 1.0. `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`.
16. A. Faz-Hernández and J. López. Fast Implementation of Curve25519 Using AVX2. In *Proceedings of LATINCRYPT 2015*, pages 329–345. Springer International Publishing, 2015.
17. A. Fog. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs. `http://www.agner.org/optimize/`, 2016.
18. P. Gaudry and D. Lubicz. The arithmetic of characteristic 2 Kummer surfaces and of elliptic Kummer lines. *Finite Fields and Their Applications*, 15(2):246–260, 2009.
19. M. Hamburg. Ed448-Goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625, 2015. `http://eprint.iacr.org/2015/625`.
20. M. Joye. Highly regular right-to-left algorithms for scalar multiplication. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 135–147. Springer, 2007.
21. N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of computation*, 48:203–209, 1987.
22. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proceedings of CRYPTO 99*, volume 1666 of *LNCS*, pages 388–397. Springer Berlin Heidelberg, 1999.
23. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In *Proceedings of CRYPTO 96*, volume 1109 of *LNCS*, pages 104–113. Springer Berlin Heidelberg, 1996.
24. A. Langley. curve25519-donna. `https://github.com/agl/curve25519-donna`. Accessed in Mar 2017.
25. A. Langley, M. Hamburg, and S. Turner. Elliptic Curves for Security, 2016. Request for Comments. `https://tools.ietf.org/html/rfc7748`.
26. V. S. Miller. Use of elliptic curves in cryptography. In *Proceedings of CRYPTO 85*, volume 218 of *LNCS*, pages 417–426. Springer Berlin Heidelberg, 1986.
27. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48:243–264, 1987.
28. National Institute of Standards and Technology. FIPS PUB 186-4: Digital Signature Standard (DSS). Federal Information Processing Standards, 2013. `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`.
29. National Institute of Standards and Technology. NIST Removes Cryptography Algorithm from Random Number Generator Recommendations, 2014. `https://www.nist.gov/news-events/news/2014/04/nist-removes-cryptography-algorithm-random-number-generator-recommendations`.

30. National Institute of Standards and Technology. Special Publication 800-90A Rev.1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 2015. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf.
31. T. Oliveira, D. F. Aranha, J. López, and F. Rodríguez-Henríquez. Fast Point Multiplication Algorithms for Binary Elliptic Curves with and without Precomputation. In *Proceedings of SAC 2014*, volume 8781 of *LNCS*, pages 324–344. Springer Berlin Heidelberg, 2014.
32. T. Oliveira, J. López, and F. Rodríguez-Henríquez. The Montgomery ladder on binary elliptic curves. Submitted to Journal of Cryptographic Engineering, 2017.
33. N. G. Orhon and H. Hisil. Speeding up huff form of elliptic curves. Cryptology ePrint Archive, Report 2017/320, 2017. http://eprint.iacr.org/2017/320.
34. K. Patterson. Formal request from TLS WG to CFRG for new elliptic curves. Crypto Forum Research Group archives, 2015. https://mailarchive.ietf.org/arch/msg/cfrg/Hvihr_yQhVB_Qdl-mtwTdVbHGiU.
35. N. Perlroth. Government Announces Steps to Restore Confidence on Encryption Standards. *New York Times*, 2013. https://bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards/.
36. N. Perlroth, J. Larson, and S. Shane. N.S.A. Able to Foil Basic Safeguards of Privacy on Web. *New York Times*, 2013. http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html.
37. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, 2017. Internet-Draft. https://tools.ietf.org/html/draft-ietf-tls-tls13-19.

A Differential addition formulas

In this appendix we present a Magma verification script for the improved differential addition formulas.

```
RF<A,B>:=RationalFunctionField(Rationals(),2); P:=PolynomialRing(RF);
PR<y0,y1,x0,x1>:=PolynomialRing(RF,4);

//(x3,y3):=(x0,y0)+(x1,y1) Source: Explicit Formulas Database (EFD).
x3:=B*(y0-y1)^2/(x0-x1)^2-A-x0-x1;
y3:=(2*x0+x1+A)*(y0-y1)/(x0-x1)-B*(y0-y1)^3/(x0-x1)^3-y0;
//(x2,y2):=(x0,y0)-(x1,y1)
x2:=B*(y0+y1)^2/(x0-x1)^2-A-x0-x1;
y2:=(2*x0+x1+A)*(y0+y1)/(x0-x1)-B*(y0+y1)^3/(x0-x1)^3-y0;
//Proposed alternative formulas.
x3t:=(1-x0*x1)/(x0-x1)*(x0*y1-y0*x1)/(x0*y1+y0*x1);
y3t:=(y0*(x1^2-1)-y1*(x0^2-1))/(x0-x1)^2*(x0*y1-y0*x1)/(x0*y1+y0*x1);
//Differential addition formulas.
x3d:=(1-x0*x1)^2/(x2*(x0-x1)^2);
y3d:=(y0^2*(1-x1^2)^2-y1^2*(1-x0^2)^2)/(y2*(x0-x1)^4);

//Check modulo the quotient relations.
QR<x0,y0,x1,y1>:=RingOfFractions(quo<PR|
    B*y0^2-(x0^3+A*x0^2+x0),B*y1^2-(x1^3+A*x1^2+x1)
>);
QR!(B*y2^2-(x2^3+A*x2^2+x2)); QR!(B*y3^2-(x3^3+A*x3^2+x3));
QR!(x3-x3d); QR!(y3-y3d); QR!(x3-x3t); QR!(y3-y3t);

RF<A,B>:=RationalFunctionField(Rationals(),2); P:=PolynomialRing(RF);
PR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=PolynomialRing(RF,8);

//((X2:Z2),(Y2:Z2)):=((X0:Z0),(Y0:Z0))-((X1:Z1),(Y1:Z1))
X2:=(Z0*Z1-X0*X1)*(X0*T0*Y1*Z1+Y0*Z0*X1*T1);
Z2:=(X0*Z1-Z0*X1)*(X0*T0*Y1*Z1-Y0*Z0*X1*T1);
Y2:=(X0*T0*Y1*Z1+Y0*Z0*X1*T1)*(Y0*Z0^2*T1*(X1^2-Z1^2)+T0*(X0^2-Z0^2)*Y1
     *Z1^2);
T2:=(X0*T0*Y1*Z1-Y0*Z0*X1*T1)*T0*T1*(X0*Z1-Z0*X1)^2;
//((X3:Z3),(Y3:Z3)):=((X0:Z0),(Y0:Z0))+((X1:Z1),(Y1:Z1))
X3:=(Z0*Z1-X0*X1)*(X0*T0*Y1*Z1-Y0*Z0*X1*T1);
Z3:=(X0*Z1-Z0*X1)*(X0*T0*Y1*Z1+Y0*Z0*X1*T1);
Y3:=(X0*T0*Y1*Z1-Y0*Z0*X1*T1)*(Y0*Z0^2*T1*(X1^2-Z1^2)-T0*(X0^2-Z0^2)*Y1
     *Z1^2);
T3:=(X0*T0*Y1*Z1+Y0*Z0*X1*T1)*T0*T1*(X0*Z1-Z0*X1)^2;

//Precomputation.
mu:=(X0+Z0)/(X0-Z0);

//Projective differential addition formulas with precomputation.
X3d:=Z2*((X1+Z1)+mu*(X1-Z1))^2;
Z3d:=X2*((X1+Z1)-mu*(X1-Z1))^2;
```

```
//Check modulo the quotient relations.
QR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=RingOfFractions(quo<PR|
    B*Y0^2*Z0^3-(X0^3*T0^2+A*X0^2*T0^2*Z0+X0*T0^2*Z0^2),
    B*Y1^2*Z1^3-(X1^3*T1^2+A*X1^2*T1^2*Z1+X1*T1^2*Z1^2)
>);
QR!(B*Y2^2*Z2^3-(X2^3*T2^2+A*X2^2*T2^2*Z2+X2*T2^2*Z2^2));
QR!(B*Y3^2*Z3^3-(X3^3*T3^2+A*X3^2*T3^2*Z3+X3*T3^2*Z3^2));
QR!(X3/Z3-X3d/Z3d);
```

B Twisted Edwards differential addition with pre-computation

Bernstein and Lange introduced efficient differential addition formulas for Edwards curves in EFD [4] with links to the arithmetic of Kummer lines [18], and with the following note: *"The following formulas are the outcome of a discussion in 2009 among Daniel J. Bernstein, David Kohel, and Tanja Lange. The core ideas were published by Pierrick Gaudry in 2006."* (2006 Gaudry "Variants of the Montgomery form based on Theta functions").

This appendix adapts the same formulas to the present setup, with the same assumptions that $r^2 = d$ and $ry^2 = Y/T$. The letter 'T' is used for $(Y_i : T_i)$ in order to produce a simple verification script that uses the embedding of E in $\mathbb{P}^1 \times \mathbb{P}^1$, see [6]. The numbering in the subscripts are also modified to match the formatting of this paper. The formulas then read:

$$\big(Y_3 : T_3\big) = \Big(T_2\big((T_1 - Y_1) - \epsilon(T_1 + Y_1)\big)^2 : Y_2\big((T_1 - Y_1) + \epsilon(T_1 + Y_1)\big)^2\Big) \quad (8)$$

where

$$\epsilon = \frac{(1-r)(T_0 + Y_0)}{(1+r)(T_0 - Y_0)}.$$

Computing $(Y_3 : T_3)$ in (8) takes only 3**M**+2**S**+4**a**, assuming that ϵ is precomputed. The following Magma [8] script verifies the differential addition formulas.

```
RF<a,d>:=RationalFunctionField(Rationals(),2);
P:=PolynomialRing(RF); K<r>:=quo<P|P.1^2-d/a>;
PR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=PolynomialRing(K,8);

//((X2:Z2),(Y2:Z2)):=((X0:Z0),(Y0:Z0))-((X1:Z1),(Y1:Z1))
X2:=X0*Y1*T0*Z1-Y0*X1*Z0*T1; Z2:=Z0*T1*T0*Z1-d*X0*X1*Y0*Y1;
Y2:=Y0*Y1*Z0*Z1+a*X0*X1*T0*T1; T2:=Z0*T1*T0*Z1+d*X0*X1*Y0*Y1;
//((X3:Z3),(Y3:Z3)):=((X0:Z0),(Y0:Z0))+((X1:Z1),(Y1:Z1))
X3:=X0*Y1*T0*Z1+Y0*X1*Z0*T1; Z3:=Z0*T1*T0*Z1+d*X0*X1*Y0*Y1;
Y3:=Y0*Y1*Z0*Z1-a*X0*X1*T0*T1; T3:=Z0*T1*T0*Z1-d*X0*X1*Y0*Y1;

//Assumed input coordinates.
rYY0:=r*Y0^2; TT0:=T0^2; rYY1:=r*Y1^2; TT1:=T1^2; rYY2:=r*Y2^2; TT2:=T2
    ^2;
//Precomputation.
epsilon:=(1-r)/(1+r)*(TT0+rYY0)/(TT0-rYY0);

//Projective differential addition formulas with precomputation.
rYY3:= TT2*((TT1-rYY1)-epsilon*(TT1+rYY1))^2;
TT3:=rYY2*((TT1-rYY1)+epsilon*(TT1+rYY1))^2;

//Check modulo the quotient relations.
QR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=RingOfFractions(quo<PR|
```

```
    a*X0^2*T0^2+Y0^2*Z0^2-Z0^2*T0^2-d*X0^2*Y0^2,
    a*X1^2*T1^2+Y1^2*Z1^2-Z1^2*T1^2-d*X1^2*Y1^2
>);
QR!(a*X3^2*T3^2+Y3^2*Z3^2-Z3^2*T3^2-d*X3^2*Y3^2);
QR!(r*Y3^2/T3^2-rYY3/TT3);
```

C Extended Huff differential addition with pre-computation

The affine addition formulas for the extended Huff curve $H : y(1 + ax^2) = cx(1 + dy^2)$ are given in [33] as

$$\begin{aligned}(x_3, y_3) &= (x_0, y_0) + (x_1, y_1) \\ &= \left(\frac{(x_0 + x_1)(1 - dy_0y_1)}{(1 - ax_0x_1)(1 + dy_0y_1)}, \frac{(1 - ax_0x_1)(y_0 + y_1)}{(1 + ax_0x_1)(1 - dy_0y_1)}\right)\end{aligned} \tag{9}$$

Assuming that we have $(x_2, y_2) = (x_0, y_0) - (x_1, y_1)$, the affine addition formulas (9) can be rewritten as the following differential addition formulas

$$(x_3, y_3) = \left(\frac{1}{x_2} \cdot \frac{x_0^2 - x_1^2}{1 - a^2x_0^2x_1^2}, \frac{1}{y_2} \cdot \frac{y_0^2 - y_1^2}{1 - d^2y_0^2y_1^2}\right) \tag{10}$$

As in [33], we use

$$\mathcal{H} = \left\{\big((X : Z), (Y : T)\big) \in \mathbb{P}^1 \times \mathbb{P}^1 : YT(Z^2 + aX^2) = cXZ(T^2 + dY^2)\right\},$$

the embedding of H into $\mathbb{P}^1 \times \mathbb{P}^1$, for efficiency purposes. The affine differential addition formulas in (10) translates to the projective differential addition formulas (11), as follows:

$$\begin{aligned}\big((X_3 : Z_3), (Y_3 : T_3)\big) &:= \big((X_0 : Z_0), (Y_0 : T_0)\big) + \big((X_1 : Z_1), (Y_1 : T_1)\big) \\ &= \Big(\big(Z_2(X_0^2Z_1^2 - Z_0^2X_1^2) : X_2(Z_0^2Z_1^2 - a^2X_0^2X_1^2)\big), \\ &\qquad \big(T_2(Y_0^2T_1^2 - T_0^2Y_1^2) : Y_2(T_0^2T_1^2 - d^2Y_0^2Y_1^2)\big)\Big) \\ &= \Big(\big(Z_2(Z_1^2 - \kappa X_1^2) : X_2(\kappa Z_1^2 - a^2X_1^2)\big), \\ &\qquad \big(T_2(T_1^2 - \lambda Y_1^2) : Y_2(\lambda T_1^2 - d^2Y_1^2)\big)\Big)\end{aligned} \tag{11}$$

where $\kappa = Z_0^2/X_0^2$, $\lambda = T_0^2/Y_0^2$, $\big((X_2 : Z_2), (Y_2 : T_2)\big) = \big((X_0 : Z_0), (Y_0 : T_0)\big) - \big((X_1 : Z_1), (Y_1 : T_1)\big)$, $X_0 \neq 0$, and $Y_0 \neq 0$. Computing $(X_3 : Z_3)$ in (11) takes only 4**M**+2**S**+1**D**+2**a**, assuming that κ is precomputed. Analogus comments apply independently to $(Y_3 : T_3)$, assuming that λ is precomputed.

A better operation count can be achieved if $a = \pm 1$:

$$\begin{aligned}(X_3 : Z_3) &= \Big(\big(Z_2(X_0^2Z_1^2 - Z_0^2X_1^2) : X_2(Z_0^2Z_1^2 - X_0^2X_1^2)\big)\Big) \\ &= \Big(Z_2\big((X_1^2 - Z_1^2) - \kappa'(X_1^2 + Z_1^2)\big) : X_2\big((X_1^2 - Z_1^2) + \kappa'(X_1^2 + Z_1^2)\big)\Big)\end{aligned} \tag{12}$$

where

$$\kappa' = \frac{(X_0^2 - Z_0^2)}{(X_0^2 + Z_0^2)}.$$

Computing $(X_3 : Z_3)$ now takes only $3\mathbf{M}+2\mathbf{S}+4\mathbf{a}$, assuming that κ' is precomputed. Analogus comments apply independently to $(Y_3 : T_3)$, assuming that $d = \pm 1$ and $\lambda' = (Y_0^2 - T_0^2)/(Y_0^2 + T_0^2)$ is precomputed. The following Magma [8] script verifies the proposed differential addition formulas.

```
RF<a,c,d>:=RationalFunctionField(Rationals(),3); P:=PolynomialRing(RF);
PR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=PolynomialRing(RF,8);

//((X2:Z2),(Y2:Z2)):=((X0:Z0),(Y0:Z0))-((X1:Z1),(Y1:Z1))
X2:=(X0*Z1-Z0*X1)*(T0*T1+d*Y0*Y1); Z2:=(Z0*Z1+a*X0*X1)*(T0*T1-d*Y0*Y1);
Y2:=(Z0*Z1+a*X0*X1)*(Y0*T1-T0*Y1); T2:=(Z0*Z1-a*X0*X1)*(T0*T1+d*Y0*Y1);
//((X3:Z3),(Y3:Z3)):=((X0:Z0),(Y0:Z0))+((X1:Z1),(Y1:Z1))
X3:=(X0*Z1+Z0*X1)*(T0*T1-d*Y0*Y1); Z3:=(Z0*Z1-a*X0*X1)*(T0*T1+d*Y0*Y1);
Y3:=(Z0*Z1-a*X0*X1)*(Y0*T1+T0*Y1); T3:=(Z0*Z1+a*X0*X1)*(T0*T1-d*Y0*Y1);

//Projective differential addition formulas without precomputation.
X3d:=Z2*(X0^2*Z1^2-Z0^2*X1^2); Z3d:=X2*(Z0^2*Z1^2-a^2*X0^2*X1^2);
Y3d:=T2*(Y0^2*T1^2-T0^2*Y1^2); T3d:=Y2*(T0^2*T1^2-d^2*Y0^2*Y1^2);

//Check modulo the quotient relations.
QR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=RingOfFractions(quo<PR|
    Y0*T0*(Z0^2+a*X0^2)-c*X0*Z0*(T0^2+d*Y0^2),
    Y1*T1*(Z1^2+a*X1^2)-c*X1*Z1*(T1^2+d*Y1^2)
>);
QR!(Y2*T2*(Z2^2+a*X2^2)-c*X2*Z2*(T2^2+d*Y2^2));
QR!(Y3*T3*(Z3^2+a*X3^2)-c*X3*Z3*(T3^2+d*Y3^2));
QR!(X3/Z3-X3d/Z3d);
QR!(Y3/T3-Y3d/T3d);

RF<c,d>:=RationalFunctionField(Rationals(),2); P:=PolynomialRing(RF);
PR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=PolynomialRing(RF,8);

//Assumption.
a:=1;

//((X2:Z2),(Y2:Z2)):=((X0:Z0),(Y0:Z0))-((X1:Z1),(Y1:Z1))
X2:=(X0*Z1-Z0*X1)*(T0*T1+d*Y0*Y1); Z2:=(Z0*Z1+X0*X1)*(T0*T1-d*Y0*Y1);
Y2:=(Z0*Z1+X0*X1)*(Y0*T1-T0*Y1); T2:=(Z0*Z1-X0*X1)*(T0*T1+d*Y0*Y1);
//((X3:Z3),(Y3:Z3)):=((X0:Z0),(Y0:Z0))+((X1:Z1),(Y1:Z1))
X3:=(X0*Z1+Z0*X1)*(T0*T1-d*Y0*Y1); Z3:=(Z0*Z1-X0*X1)*(T0*T1+d*Y0*Y1);
Y3:=(Z0*Z1-X0*X1)*(Y0*T1+T0*Y1); T3:=(Z0*Z1+X0*X1)*(T0*T1-d*Y0*Y1);

//Precomputation.
kappadash:=(X0^2-Z0^2)/(X0^2+Z0^2);

//Projective differential addition formulas with precomputation.
X3d:=Z2*((X1^2-Z1^2)-kappadash*(X1^2+Z1^2));
Z3d:=X2*((X1^2-Z1^2)+kappadash*(X1^2+Z1^2));
```

```
//Check modulo the quotient relations.
QR<X0,Z0,Y0,T0,X1,Z1,Y1,T1>:=RingOfFractions(quo<PR|
    Y0*T0*(Z0^2+X0^2)-c*X0*Z0*(T0^2+d*Y0^2),
    Y1*T1*(Z1^2+X1^2)-c*X1*Z1*(T1^2+d*Y1^2)
>);
QR!(Y2*T2*(Z2^2+X2^2)-c*X2*Z2*(T2^2+d*Y2^2));
QR!(Y3*T3*(Z3^2+X3^2)-c*X3*Z3*(T3^2+d*Y3^2));
QR!(X3/Z3-X3d/Z3d);
```

D Pre-computed constants

Tables 3 and 4 list the pre-computed values $\mu_i = \frac{u_{P_i}+1}{u_{P_i}-1}$ related to the multiples $P_i = 2^i$P for the functions X25519 and X448, respectively.

D.1 X25519

Table 3. Pre-computed μ_i-values related to multiple points $2^i P \in E_{486662}(\mathbb{F}_{2^{255}-19})$, for $0 \leq i \leq 251$

i	μ_i-value
0	0x5FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3
1	0x5142B2CF4B2488F4D5A9A5B075A5950F82EBEB2B4F566A346B8220F416AAFE96
2	0x4B5ACA80E36011A42A58D9183B56D0F489CF7820A0F99C416AAEBC750069680C
3	0x306912D0F42A9B4A1E45BB03FF67BC34F4A2E616E1642FD7329132348C29745D
4	0x174E251A68D5F222AA512FE82ABAB5CE04F50E13DFEEC82FFF886507E6AF7154
5	0xC59888A51E0482E379EEC98B4E86EAA1743E3370A2C02C5CF96700D82028898
6	0x2938218DA274F972C1C20D06231F7614ACAEF0D58E9FDC84FBCBF1D699B5D189
7	0x69C1627C690913A996FCC9EF4015C56BCC541C22387AC9C2F6AF49BEFF1D7F18
8	0x1AD7A7C829B37A79095E4B1A8EA2A229FDB8C4F29E087DE97A86FD2F4733DB0E
9	0x3DF7B4C84980ACBB19CA31BF2BB42F7467BEDDA6CCED2051342D89CAD17EA0C0
10	0x3D867C6EF247E668C215CDA00164F6D8B91E440366E3AB85A8C6444DC80AD883
11	0x7529D871B0675DDFA0FD9B95CC9F4F71FD2C4748EE0E5528C7DD582BCC3E658C
12	0x75E7FC8E9E4986032DCE6CCD4A7C3B621233011B91F3DA82B8F568B42D3CBD78
13	0x6EBE0DBB8C83B56AC249C1A72981E29BF1A8CA1F29FF7A452F4F13F1FCD0B6EC
14	0x2F0EEF79A2CE9289BDC41F68B59C979A65A2DCD5BF93935F7114FA8D170BB222
15	0x3780AA4BEDFABB80F294B0C19CFEAC0D2930BC09EC49632242ECBF0C083C37CE
16	0x41B883C7621052F80CE931732DBFE15AE7CB4BEB2E5722C556C17D3E7CEAD929
17	0x1D45BF82322225AAC936E03CB4A9B2122936BE086EB1E351DBF75CA0C3D25350
18	0x20FFDB5A4D839581C5D73FBA6832B1FCE212201C304C9A72E81AB1036A024CC5
19	0x5166408EEE85FF499D4935467CAAF22E6C2B25CA8B164475A283D367BE5D0FAD
20	0x6A621892D5B0AB335259729241159B1CB3E433C67EF35CEF3C67BAA2FAB4E361
21	0x61AB3443F05C44BFEAA17B7762281DD1532AA10E1208923F20B74A387555CDCB
22	0x295A407A01A7858023758739F630A257131C6C1017E3CF7F257A6C422324DEF8
23	0x7D10C8E81B2B47002AFCFC92731BF83D19D775450C52FA5DF8C443246D5DA8D9
24	0x3196D36173E629755412EFB3CB7ED4BB993748867CA63957C8E0271F70BAA20B
25	0x256DBF2D04ECEC2A34CD942E11AF3CB47CC8CD2B395C848DE5BCAD141C7DFFC
26	0x5F1A87BB8C85B19B47F12E8F4E72C79FCAD4DD83C0850D10875AB7E94B0E667F
27	0x23A05A2F7D2C56272ADE09FE5CF77AEE12C7CE55188790657AE9D0B6437F51B8
28	0x183ABADD1013984574B4C4CEAB102F64F77498DD8AD0852D5908E128F17C169A
29	0x2701E635EE204514BE2F8F0CF8FC40D1D5C5EF9599386705B165BA8DAA92AAAC
30	0x60D9944CF708A3FA5B894FFF0B3F060EF223868764A8C1CE629FA80020156514
31	0x79B958150D0208CB6F7709594C7A07E1EBF16A633EE2CE63AEEA001A1C7A201F
32	0x131384427B3AAEECD88768E4904032D8E3A34EDFF3FDC84D24B55E5301D410E7
33	0x2FE2A94AD8A7FF93B8A2B5B250634FFD14DC4739ADB4C5298405E51286234F14
34	0x743629BDE8FB777EA4B561D6CF3D63052843CE40F0BB9918EC5C57EFE843FADD
35	0x1FC223E28DC88730A401760B882C797AED981828B101A651343EDD46BBAF738F

Table 3. Continued from previous page

i	μ_i-value
36	0x23F7EED4437A687C91CCAC3D09E9239CB637F78F052C6FA448604E91FC0FBA0E
37	0x2959894FCAD81DF5FDBF177988BBC58629D641B63189D4A75173B1118D9BD800
38	0x40D158894A05DEE824E20B0134F92CFB4148995AB26992B9AEBC8EF3B4BBC899
39	0x2A608BD8945524D73DC0BF95AB8FFF5F26BAC77873187A7946B00B1185AF76F6
40	0x26218D7BC9D876B98E98A4F383BD11B27C4BC21C0388439C26449588BD446302
41	0x7DDE05734AFEB1FA5C217736FA2793743C2D29A86FB6606FE3081542997C178A
42	0x394FAF38DA245530E6053BF89595BF7AE4F7803E1980649C3BF10E3906D42BAB
43	0x48F222A81D3D6CF772670CE330AF596FFBC778E9CC6A113C7A8EFB58896928F4
44	0x7F89AE31DA8A7417BC21165C1FA14835A20ECC7213B5595F01FCE410D72CAA7
45	0x705EDB91A65333B6A5A5590A96D3A904D43E330FC631629305D2C2B4C6830FF9
46	0x621C0DA3DE544A6D8F4B71CEEDC4A40B3240CFCA9E0AAF5D048EE15E0BB9A5F7
47	0x2667FCFA7EC836358A72EB524F276394CE8375B010C9144592872836A08C4091
48	0x3780CEF5425DC89C25DD9AFA9F86FF34061B47FEEE7079A57F4C173345E8752A
49	0x321A967634FD9F22C78C5F1C5FA24B503E1EF379AC575ADA1A46035A513BB4E9
50	0x6D9284169B3B8484C189218075E914363DCA84D64C506FD0946707B8826E27FA
51	0x26EC449FBAC9FBC43EC7C86FA783EF4733EEC9A30C4F9B753A67E840383F2DDF
52	0x5A238AA0A5EFDCDD3E23B0D306FC121C81168CC762A3478C5C0F38CBA09B9E7D
53	0x5CA9938EC25BEBF988FBEA0B0ADCF99A36F8C77F7C8832B51BA26121C4EA43FF
54	0xF5034E49B9AF46619346A65D3224A081DBC4797C2CD893BD5436A5E51FCCDA0
55	0x2A75381EB6026946FB2FABC6A7341679E58B08FA867A4D88F23C3967A1E0B96E
56	0x25ABBBD8A660A4C47CF7036761E9338866B1F6C681F2B6DCC80A3BE4C19420AC
57	0x3EA988F75301A441F826842130F5AD28684950FC4A3CFFA991EA12BA14FD5198
58	0x75952B8C054E5CC7444D6D77B44599951746EB4A0530C3F3C978109A695F8C6F
59	0x77FEA47D81A5D71FD01469DF811D644B66C346202F2647D8A3703F7915F4D6AA
60	0x4B80BE3E9AFC3FECB6E91A28E8009BD66EEEB4B9CE2F881AC5E9529EF57CA381
61	0x7C54699F122D400EA920BDD7BAFFB24D1B4AFCB453C9A49D7E3773C526AED2C5
62	0x61B68649320F712CBEA450E1DBD885D5E0B074CE2952ED5EEF46C8E14FA94BC8
63	0x445DC4758C17F77025232973322DBEF4BD06320D7D4D1A2D8A485F7309CCBDD1
64	0x4ADBE867C65DAF991EFEBEFDC053DB34ED6FE82175EA059FDB0434177CC8933C
65	0x697427F6885CFE4D1EC69B688157C23CE5E991856DD040503ACD71A2A90609DF
66	0x49E5B7E17C2641E128399D658FD2B645A03D28D522C536DDD7BE7B9B65E1A851
67	0x2E003258A7DF84B1D13C3CCBC382960F5078F0A25EBB67786F8C3A98700457A4
68	0x59256173A69D2CCC33EE0673FD26F3CBC1EEAA652A5FBFB28AD1F39BE6296A1C
69	0x445B652DC916694FBDAACB805831CA6FD9FC19527C87A51E41EA07AA4E18FC41
70	0x790A2D94437CF586A1823AAFE04C314A1EDC282DE11B9964CE92A3A7F2172315
71	0x2F7A89891BA319FEBF70903B204F51698922A5672284527671C447FB93F6E009
72	0x3E88363C14E9355B5253EC44E4323CD1ED9A4ED4427BDCF402A08EB577E2140C
73	0x3EE141579555C7AB2030BD12C93FC2A21AE0391610A23390AA66C14277110B8C
74	0x5682250F329F93D0674F1288F8E112173CCDD88607F17EFE9214DE3A6D6E7D41
75	0x47069B48ABA16B65930B1B5BFCC4E836E4CF86F1014DEBF6CF00B136D2E396E
76	0x221B1085368BDDB5CDFA50F54E00D01DB24DB91A97D0FB9E0D4CE4AB69B20793
77	0x61EFA662CBBE3D42EEE8A903E0663F0953C56563BD122F93E7E59468B1E3D8D2
78	0x29275B5D41D29B275DEADACEC9F049739BF80AD51435F2312CF8DDDDDE6EAB2A
79	0xA63BF2F1673BBC7CAE80DD9A1C420FDB9AAB96B054905A7CFDE0F0895EBF14F
80	0x6F3F7722C8F192F8CAC8351560D52517672A81E804822FAD092F6E11958FBC8C
81	0x68E122157B743D69894D1D855AE523592C7557A438FF9F0DF8BA90CCC2E894B7

Table 3. Continued from previous page

i	μ_i-value
82	0x3C66A115246DC5B22121154710C0A2CE3F2CDECD95798DB9D87E5570CFB919F3
83	0x6144735D946A4B1E9610C2EFD4078B67BA7143C36A280B16CBEDC562294ECB72
84	0x149DFD3C039E8876F93CB1000E10413C0211DB8C2041D81B536F111ED75B3350
85	0x5FAFDA1A2E4B0B35DAFDE43B1F13E038B66E15E93C837976D479DDE46B63155B
86	0x34FCE5E43F9B860F5938906DBDD5BE863972050BBE3CD2C23600BBDF17197581
87	0x3EBAD76FB814D25F33DCABEDD2E131D3828DABC53441DF6575A8A4CD42D14D02
88	0x1F768AEAD23299945ADB16E76CEFCF25D12F7AA51690F5AD4906F566F70E10F
89	0x4C3BFF2EA6F66C23CE1C0B80D4EF486A3CD30628EC3AAFFD2B6CC77B6248FEBD
90	0x23B20565DE55E3EF5EEFA966DE2A701D61B19B286E372CA73F2EC4094AEAEB5F
91	0x2A52FEE23F2BFF56A532CD8A9DCF1D6707B2D4CE27C2874FE301CA5279D58557
92	0x7B3052CB86A6EC6657B85E9C82D37445BBC7AC20FFBD75948624EFB37CD8663D
93	0x185F8C2529781B0AAF4F6D052E1B003A2CB68043D28EDCA03482F0AD2525E91E
94	0x4CC4B8DD0E297BCE563EC36E357F4C3A9407B2416853E9D6AA41DE5BD80CE0D6
95	0x72F4A0C4A0B9F09910F9A366CDDF4EE11811F16E67058E37A2FC1A52FFB8730E
96	0xBA53CBC968A80892102E7F1D69EC345693B3AF74E970FBA8C16C06F663F4EA7
97	0x40D2A72AB454CC60B9886314844006B14C6824BB51536493CA3D9DC7FEA15537
98	0x6BA10D12EE51D03344094BB64330EA91B9D648DEBDA6575936A1B712570975
99	0x4502D4CE4FFF0E0BA1039F9DD56019900EB12F4C38CC05B019228468F5DE5D58
100	0x6E15C6114B502EF040727064C416D74FD0F6544C6DD3B93CEB2054106837C189
101	0x705B3AAB41355B444A497962066E604311256C7419F2F6B14DF2A398CFB1A76B
102	0x3777AA05C8E4CA4D3BBF33B0E0575A8800076BD622DDF0DB365EF536D797B1D8
103	0x9963437D36F1DA3B1F0B00B8ADC98676FDA4149DBAE5AE2392745C85578DB5F
104	0x625F3CE26604249F6736D86C87CE8FCCCCB5F6641F135CBD7E824E90A5DC3853
105	0x1A394360C7E23AC335292E9C764B63050C05E70A2E351469AF8AC8059502F63F
106	0x13A3DA3661206E5EB5FBF5D03B973F9B62065ABD43C2B74FD5C6D53251183264
107	0x7FBA1F495C8379872088CE1570033C6818E30912205016C5C6BD5837725D94E5
108	0x6691FF72C878E33CE4F9B49AD2FAB3511735157B34023FC55A8C7423F2F9079D
109	0x49B92B9D975C743EB86205D9E9E5BDAF8DD4BF1D8956CF4122C2ADEDC5EFF3E
110	0x470E9DBC88D5164AB0032C34B20DCD6D72A0FFACC6F3A553A5379730B0F6C05A
111	0x48C14F600C5FBE8EB3321BD16DD80B43B65466711F6C81A2B19CF10CA237C047
112	0x31602627C3C9BC10D45F19B0B3128395B66E3904A4FA7DA666451C264AA6C803
113	0x566D4FC14730C50900F52E3F67280294EB20C46756C717F73120DC4832E4E10D
114	0x22E8C3843F69CEA7216730FBA68D6095C1E926DC7159547A7E3A5D40FD837206
115	0x7773C12F89F1F3F35534C26AD6BA2365B6E4350E84D1581633D074E8930E4B2B
116	0x3A81907FA093C291508E862F121692FC5B9897A81999CE568CBA404DA57962AA
117	0x32A5C1D5CB09A44C5B9D151C9F1F4E8910D8CC10673FC5030DDED0FF4725A510
118	0x25E496C722422236BEE595CE8A9DF2E55F85EB7CC1B485DB1E0AA442B90541FB
119	0xE878A01A085545CE488DE11D761E35234E75A7ED2A433885EDF3C46CD0FE5B9
120	0x67A9958011E417949EA37A487AE80D672B4D1843C7DF899ABA493C77E021BB04
121	0x68BECAA181C2DB0DBB4DA8D483CA46C147E33F7D8D6BA6D44B58051A6697B065
122	0x6E228363B5EFB56951C6C7C4796E73A2F95EB14A2C93C99B8D8980E90B989AA5
123	0x1DC6BC087160BF9B3CDE15A004CFAFA9777EB47DEC8170EEC6BBC0B02DD624C8
124	0x48FBC3BB00568253D8DA03188BD15B9A18E9FC677A68DC7F2E07E043EEC34002
125	0x47531EF114DFBB18F63EAF0BBF154478D3565B82A058E2AD57547D4CFB654CE1
126	0xAA7EFA85682844E85E135C63ADC0C2B5507D546CA8E83F3E1EC630A4278C587
127	0x2CFE53DEA02E39E897B6D92E39BB786832B4E9701FBE3FFA72691BA8B3E1F615

Table 3. Continued from previous page

i	μ_i-value
128	0x269BB0360A84F8A097134556A9832D0627FF66C910E29831687392CD85CD52B0
129	0x5CD6ABC198A9D9E07AEE91E8C6EFA4723734A48C9B597D1B706E55457643F85C
130	0x7215C371746BA8C8904659BB686E3772D8C6EB893402E1380E04DE06CB3CE41A
131	0x5C847085619A26B9266FD5809208F2949514B7516394F2C5FD12A97EEAE4A2D9
132	0x63B3D2D69E5E9E110BB47692D3BE4673C905B934A2ED25452985410FED694EA
133	0x117C554FC4F45B7C2B1641917B307614EFB6C4AE10F41891472726EEDDA57DEB
134	0x1015E87487D225EAD7E803F4171B282701DBD82050017939C07CF3118F9D8812
135	0x6B1640FA6E37524A0B94D43D1C9CF45750DB91C294A7BE2DC58DE3FED23ACC4D
136	0x4B38395F3FFDFBCFB8C46F760777A296200B1C59FA4D3151692F346C5FDA0D09
137	0x50FDF64E9CDA043287AD8F263B78B98260D50582BEC8ABA618D25E00BE54D671
138	0x15473C9BF03101C70EEBBA9242D9DE71EF1E9B0EF2A3133B90F567AAC578DCF0
139	0x7CF931C1FF733F0B2DA0B9615348BA1FB678E7666E6F078E7C77E8AE56B78095
140	0x35D90C991143BB4CC13AEEA5F91CB2C0E9708CF42B87D73226B357F50A0A366C
141	0x1FBD9ED379561F243875A8C473B38C31659E58451972D25147C1C404A9A0D9DC
142	0x6135FA4954B72F2772E73B5D8C4045957EF8DFE3CD2A2DCA11FABC6FD41EC28D
143	0x5E9BF806CA477EEBBE3350ED5AC3F9293F55698C1F095D88CCFC32A2DE24B69C
144	0x6632A1EDE5623506D1AFCFB35A6393F15376F63565E1F9F4E9CE8FB63C309F68
145	0x5D588B60A38CA72A66305A1249ECC3C756CB3281DF04CB1F0B7D6C390C2DED4C
146	0x1AAF1AF517C36731EC219C48FBD2160486EEB44B3C8A3EECA6ECBF78E8E5F42D
147	0x76CFA1CE1124F26B8027F51FFBFF94A6208280622B1E2ADBC306A2836769BDE7
148	0x253B7F082128A274DBBC207F531561AF377C4D58F8C29C318EB00562422ABB6
149	0x356F97E13EFAE57652D17436309D42534860E1ABD64628A93D1F091CB62C17E0
150	0x1D34AE93032885B80C776128BED92C983E6B45BB1DD878CCD351E11AA150535B
151	0xF81A0290654124A66124C6F97BDA770985348C33C9CE6CE4BA0488CA85BA4C3
152	0x52A148199FAEF26BFF08D03F93D8C20A811009FD18AF9A2D9ED09CA6569B86FD
153	0x7AA1F15A1C0D549C0D987F041A359704205801873961A703E03F9DC2D8D1B73
154	0xA4556E9E13D95A2808A7A6399897B596D0A024F934E4239DFD46CE08CD27224
155	0x77DB28D63940F7215DA643CB4BF300359B0E8548FE7751B8D21A991FE9C13045
156	0x340D053E216E4CB59EC3E7787D1DCF745229419AE8C411EBFC5EEB614ADC9011
157	0xCF1C37134273A4C140A69245CA575EDC0FAEC2871A10A94CAC7AF39B48DF2B4
158	0x7D9A62742EEB657DA1E806BDAACBE74F57EAEE7CCB4930B0C8EE306AC224B8A5
159	0x58654FEF9D2E0412E6B9F383EF0D7105885CCA1FDDB36E2E9EB6B6EF546C4830
160	0x30684DEA602F408D497D723F802E88E1942DE5DF9B31816EA905C4FFBE0E8E26
161	0x28C3C9CF53B98981B30B8E049D77CA15AEFB6E6F5B151DC421E5A278A3E6CB34
162	0x4A7F11464EB5642F7468C7423A5432580D317CA897022274287FB721556CDD2A
163	0x604003575F39F5EB24C515ECF87C1A88D865986EA92129A1A237A4774D193AA6
164	0x74FCD91047E21901026DF551DBB85C202B98CEDE465E4B7847B9F189570A9B27
165	0x67FE5438BE812DBE5FF1CBBE3AF6CF440CB0074E478519F613E2A90A23C1BFA3
166	0x482F144F1F610D4E4173915B7F0D2AEA054DFB2F32283787D13CF64FA40F05B0
167	0x7E93D0F1F0916F01DCD7F455B049567C5D0AE1929E70B990F6210201B47F8234
168	0x585A73EA2CBF1705027145D14B8291BDBE8391BF6F74C62FDD79CBF18A7DB4FA
169	0xF0433B5048FDB8A2F482EDBD6D551A710FC01A5742857E7485CA03E928A0DB2
170	0x273B24FE3B367D7513033AC001F6669788B4C9D38CD4819A60DA2E8DD7DC6247
171	0x4B3AE7D103DEE548D1726FDFC8B23DA7281514A494DF49D5C6E8F66A31B3B9D4
172	0x74621888FEE66574ACC63CA34B8EC145FF5C5CF186E3C61CC6256E19CE4B9D7E
173	0x694283A9DCA7502ED6A50EB5EC2647BEF0BF8E3263A962E956F409645290A1E

Table 3. Continued from previous page

i	μ_i-value
174	0x63005E2C19B7D63A4F002AEE13397EAB42B7C8EA09FC5353769B963643A2DCD1
175	0xB696063A1AA89EEACE09390C537C5E1966C7F6DB12A99B7CA6736DA63023BEA
176	0x1A5FB4C3E18F9D97A6A5A93D5B717F71432A9F9F938C8BE8EBB03E97288C56E5
177	0xA10263C8AC27B588DAFE4D867C46A20EE202A43FC02C4A01C94E7AD1C60CDCE
178	0x6F151B6D9BBB1E91CE8472ACC212C71A856AF87BBE9277C5D0DEA9DFE4432A4A
179	0x1F81B702D2770C4237AE66A6FD4609CC7D211CB7FBF8FAEC26776C527CEED56A
180	0x29571073C9A2D41EC964F8EB17BEB4F8E1DD89FE29744E9D2FB0B057EAC58392
181	0x78CD6EC4199A01FA33EB2D75FCFD3C62DF6369B65B22830A948A18981C0E254
182	0x7F69EA9008689FC268C4E8338431C97832142B78E2C74C524A584A41AD900D2F
183	0x4DE71B7454CC29D28CD7D5FA25359E94FD78072D04A832FD52F2C81E46A38265
184	0x44E6BE345122803C81004B71E33CC1910AAD37DFDBC09C3A42EB60AD1EDA6AC9
185	0x21EDB518DE701AEE49C8C4281AF60C29F5D57C32150DB00803FE8388BA1920DB
186	0x5EC3AD712B92835824DD5248CE520A83A4460D99C166D7B87FB63E418F06DC99
187	0x498194C0FC620ABB12BC8D6915783712A4F64A77D82570E315022A5FBD17930F
188	0x56307860B2E20989E4D5C81AB24A5484785C6BD9193E21F038A2D9D255686C82
189	0x59F2F014979983EF1E60C24598C71FFF22F1834643350131429D55F78B4D74C4
190	0x2CEAFD4E5390CDE7B346E15274491C3B3E22A854D636A18E46A47D56EB494A44
191	0x61A24FE0E56192C4CBDAB89085D304C34B9074BB50818E23BA8A8538BE0D6675
192	0x2D668E097F3C9766E6B4153ACAFCDD69DD7D8C35A567E4CACB7615E6DB525BCB
193	0x90D52BEB7C3F7AEFBC83606492FD1E55D9F4E527CD4B967A57E7E265CE55EF0
194	0x7EF04287126F15CCA1C49548E2C555041F266A2599DA44C009B9515A1E7B4D7C
195	0x1FD96EA6BF5CF788884D6236A5DF32918B4AB9EEC4E0277BFED1659DBD30EF15
196	0x7C22D676DBAD85D89FE113BF285A2CD561D849507E6052C142A161981F190D9A
197	0x5895319213D9BF64CD40A9C2B09001504C05B2ECE996F5A582E770ED2BFBD27D
198	0x537C659CCFA32D62CCE30BAA48205BF0B50C491258E2188CE7CC5D703FEA2E08
199	0x725F71C40B51957504D29B8E56A8D1B0FE9BED1FA4D6ACA437B6623A98CFC088
200	0x585F1974034D6C17883ADA83A6A1652C8367B14469DDC18B28C7F89CD0339CE6
201	0x3E035C9DF0954635D88C9DA6B4C0526AE63B4863E7C3521789CFB266F1B19188
202	0x2D8C2CC811E7F6904B5C999B151D671CDD684532E4CFF40DDD9D5412FB45DE9D
203	0x2C018DC527356B3033979624F0E917BEA464C5DF464AAF407F54BE1D90055D40
204	0x747201618D08E5A94EC42C4EF9B59F173FF3D96691652D3A5415024E330B3D4
205	0x59810BC09A02F7EB9C4DD40051E227FF66415F2FCFA661194D6CA48ACA411C53
206	0x5E8CE0A71E9D112F32025C9B93B359EA441C5AB99FFEF68E2A7EB171B3DC101D
207	0x18C8DF11A83103BA345EAD5E972D091ED271BA752F095D55BFCCCB92429503FD
208	0x6E42E400C5808E0DB8CAC52D56C52E0BC5D1F4CB6660E37E90CD949A9AED0F4C
209	0x51F27F054C09351B189DC8C9D683A51D0C4F1F0BE39ECDCAA3B46966EEAEFD23
210	0x547DC829A206D73DC8CCF79E555CB8E8587EA95BB3DF1C964C487CCD2A320682
211	0x228F4660E2486E1D28535B6F91463B4DE96D54732000D4C6B822A6CD80C39B06
212	0x4322E1A7535CD2BB79952A008221E7388CD8330045EBCA6E98799538DE8D3ABF
213	0x5EC362C0AE5F7266DD0E2DF409260F4C2016E4D84F3F5EC7B114C11819D1801C
214	0x43BBF8DCBFF9CCD3F25F7105436B02D27CC8D950274D1AFBC0462B18B8B2B4EE
215	0x4ED5558F33C609615E55FA18B42931F5B0714DA8F69D3583B6AD1767A039E9DF
216	0x69ACCA274E9CAF610249A4FD813FD7A6593DDF1F8081D3571FE37901C647A5DD
217	0x9A62DAB892895271DF4C0AF01314A6083423FC20E7E1EA0047BA3EA330721C9
218	0x4296E8F8BA3A4AAD3BE28779ADC994A0E94B5DC654B56CB6A5B325A49CC6CB00
219	0x4980A319601420A849B96853D7A7084A2E4D598BFF59594A328689761E451EAB

Table 3. Continued from previous page

i	μ_i-value
220	0x538B4CD66A5D4EDA05C90B4D43ADD0D78A5318DB7100FE969565B9E12F552C42
221	0x25FFFAF6C2ED141908A36EB5FD4B9550592C9AF26F618045F4E94FC3E89F039F
222	0x5129CECCEB64B773DDEB34A061615D99EEECBFB4B1D5476B34434459CC79D354
223	0x715F42B546F06A97D2E2FCE306BEDAD5772F9C7CF14C0B3BEE43215894993520
224	0x487C02354EDD9041680BD77C73EDAD2E0DA17115A49741A9434ECDCEDA5B5F1A
225	0x7F650B73176C444DF3A68BD48A2A5A056A32AA3E857E302B8EFEFF3A70ED9C4
226	0x1CA941E7AC9FF5C456D90319C9F9496479E053C18B09FB36E38B9B1626E0CCB1
227	0x3391F78264D5AD8695DFDA14CABB437D8488CF3282B3330549C4DF29162FA0BB
228	0x27FEDAFAA54BB592E9834262D13921EDD58A58D73259A946729AE06AE2B5095D
229	0x78CEB3EF3F6DD938802C8ECD5D7513FD5F025742499EE260A99DC5B829AD48BB
230	0x5064B164EC1AB4C89436D11A0537CFE77B9EDB44828CDDA3C342F44F8A135D94
231	0x5344467A481130441B930D7BDFA1BB341F31EA3ED90D25FC7020ECCFD37EB2FC
232	0x2A77A55284DD40D82348698AC8FC4F00E385DC1A50114CC870073170F25E6DFB
233	0x785768EC9300BDAF1428D01E33BF1ED3C235DF96DDDFD6E4FE06AFE0C98C6CE4
234	0x4804A82227A557BC645B426F3D1D58AC61BDB8BFE5CE8B809702E57A91DEB63B
235	0x4172F257D4DE63E2C39C9EC3F9E1C29368D6501A4B3A69358E57048AB44D2601
236	0x16F649228FDFD51F2C34BB6090B7D90D040D3017418F2391D368B450330C6401
237	0x313034806C7FFD0F594AAA68E77A36CDE28CCF91CDC11E72BEA6818E2B928EF5
238	0x56F68341D797B21C26CCFF352B37EC719A3B464018E95128A9D27AC2249BD65
239	0x7046C76D4DAE743BD3E7222C6EAF5A60FABDBCB6553AFE155E79D6757EFD2327
240	0x3ED8E9800B218E8EC103053A302BDCBB19992518574E1496660BE872B18D4A55
241	0x6065510FE2D0FE3498A35FBE391A7793EFE9FB684633C0837B0B9239FA75E03E
242	0x526028809372DE352C43ECEA0107C1DDB4584548DA87E52755CB668548ABAD0C
243	0x5FF20E3482B29AB613F6B105B5CF709B5BEE1A4D017E98DB3415C56AF9213B1F
244	0x250386B124FFC207899FC38FC4B5C515FC7D73CA3A70E2060AA29C75CC2E6C90
245	0x46B23975EB018FC716694FC58F06D6C19913149DD6DE60CE54EA28D5AE3D2B56
246	0x7FA20801A0806968ABEEE5B52FBD3ADB5D92475A8F7253DE470A6A0FB4B7B4E2
247	0x4B065A251D3A2DD20FB4CD8DF212744EB3E840C12F4660C376F3FAF19F7714D2
248	0x3147C0E50E5F6422A2DD242EB09AF7596ADF39DF882C9CB15CEBDE383D77CD4A
249	0x4BDEE0C45061F8BAE640CE4D13E5DA08F8D13479C33FC962164CA5101D1350DB
250	0x2DFFB2AB1D70710E58942F6BB2A1C00B5514D7B6437FD98AD7C46DC1A4EDB1C9
251	0x2FBBA1CD0126E8C980697F95E2937E3A8EBCBA8B7806167CCDFCF2FC18B6D68

D.2 X448

Table 4. Pre-computed μ_i-values related to multiple points $2^i P \in E_{156326}(\mathbb{F}_{2^{448}-2^{224}-1})$, for $0 \le i \le 445$

i	μ_i-value
0	0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF80000000 000000000000000000000000000000000000000000000001
1	0x289DED68857A5E30EBB712BCE84346051D99E5E7FCD7DCD2948F8EEEFFD75A06 00968505A7A501AE3A811ABA3408956CE924A6860DAAFA37
2	0xE8C3D58719E2C57618E1E49266F480A4233926EEA48FD46C76A38091AE1B1ABF C1611B9A46403C9EF71E2A5A76A4E73680624A322B087DF1
3	0x39F9D871F2C563B041A9973CA99EFDBD897BFFF422DDCB61C2F98AD74409F640 7D4CE09266E2AD5DFFCD3B2BDFBED3650D893F207F0C9FB
4	0xAA586154CC43DF406809EA883E5FCDEA160F66EA17F487EBD8A157DA8EC0FBA9 44F0D0EF4D18ABFA671C214C37CECBE0A6842C81F797354C
5	0xB91E4BBE2685992A9D345859AFE86E4D14D109C127156DE9BF025DF585765283 9608DD9325CB29E6B35EDD018549D13FA737635ED759E78D
6	0x6701BE70B8F0788642EDB79318FDAD881F2E8D29E0EE88074A2E83C5099EB1D1 8716C522F4FB61BDC63C938F0185DF8E820D5618E294CE8D
7	0x1E4FCB85845F554B35CB3DEE0CAF6C30A3BF9A5963EED63730B6430A96DE831D 6ABF7CA7D2D7AEA316688030DB2BBB27DA2335C7942D62DF
8	0xC7EB67FE64EAEC6181E3F0BF93D72446E84A06EFADF41AD0EC6B3CEEC3674349 A54336CD94B767DEF00A47FBEBF170C1870DA9A48968049D
9	0x5ABBEB7A4CE7AEC8E49692DB6516C057ABA6893F4EAB543831DA7DA7437293B4 A47501A6C9DB789404C3CFA18F4B4684FE9BFD2CCF794595
10	0x1DA0AB47683D56F8D36C8D096063E83076FD42430D38935ABCFC837E09D17108 156592A759C9E574F5EBA15522CA2120186AE097C386FEBE
11	0x609C84662AE4BDAC9439F3778166A65B1E94B5752C006592E8C7EA11F642647B 98CDDF1939091AB19BE18F4FFCE669E3388F88ACB11C5038
12	0xAC0D1DA5DF174A998C37C400CDF4F38F9F897AA0706E00A87A0B9B64163EBAF1 EDDB7B2B9675C4A2599E113FA9D2B4A0AA8A46703D9B6B5
13	0xAE9552296FD0D7B593A9DA50623CBA1E70CB108E3B459DC982B97CB48EB8F6EE 9340B1EDD3F408D6DD44DCA6BA66C66FA3F05094ABAB0AF2
14	0x958D00013D107FBD7EFC7DC96CDECD2FB5A9E718CDD542921117F19F15A796AE 956CED94C3E1961CBAAEF7DF92E891229E8744A823696814
15	0x4B8E0CD2CE607FF316A55B101667D501FEA741F386BDC4FC9F7FF08605794978 7B1AF64575EE1619A3B7E1479C136DD1F7F284AFAE72763B
16	0x34823BCF1F22AC41F23337889C8BC06B09E57DB74A6BF5625F779A2A6DB73AA8 BF98984D430E68E04A9B7744B748B9681491740F4A99074A
17	0x632B706460B9505D788618F6F60FE4782960B3FCEE1930D99D5B2A207373194D BB53D26ED930FBB9DE9BBE4A42196B1DADC2E324278C300
18	0x17FBFAB5331115915717035454EFD13038936141925917 7C62F7FB48ED37AE2B 7786F7054A6AE7A240D8DCC98F425C539F5D9C2FDF34F544
19	0x5D1A794E4EE8F6B4F04552DDB65940AF330CA44E2682B0DF87079EB47883EEE0 B1C0E3029AEECCB7F7191889A8296ACE9472F19CB4D4BB29
20	0xCC2C936F379647A6641D63D49E16D2E5C97A29AFF48D06992522DA3A11EB5CE5 9AE78CB060FDDD86571CF02B5BAB1FBCB1754F6B4C4D3327

Table 4. Continued from previous page

i	μ_i-value
21	0xDE2CE732B569B245426964E8FF76730639FC854E9370785DF91EADC97233D571 2655A0BAA7C44EBC8CB4C754766385235891F275867494C1
22	0x11E7E84EA5CCFB8C33E6961B4F3FC8B39302C2DCEC18E2B700701AD65C17B0DB DF11F592ADDB07929CE375242EEE603A0EE9324933D816DE
23	0x981E66C180F83B2C08373E8E29733A5B102B220B03C29DDE062BFB9F9B9D082C B2505438A2F98EF877AE1D029A4E1A6A7C0DDFF852747A05
24	0x2CC87481B851636C41E9D3B28355C2F5F5B3A26AB6B6C9ADC24B766BD6606F53 40AF5B844A0CF1FE92E7D687060CC081412B8C63CD3ED6D8
25	0xED8B716980708E992D8746807D544B4CC2051838AC92B4F81C1D660190879ED5 6D2880963311B6400C7528622DC96A2CA858B007EE1CAFB2
26	0xA9547972BF92BF1231500F77380677391BEDFD07B107292FF4E7C23677789C1D CBA03AC78FB3307E7997A6465ADA420FFF8E7536716BA737
27	0x5F703510ADFF5F9273CD596D8E6B7633631CB2806D5422ECB7C4CA6FAE51AC85 D38854AB12F6120C6E4ABC946D64FDED2D46801A6B464DBD
28	0x75E649151C7618785EA69A356565AB07C0D33BBDA14AE8D5ECEDAE2992B8E6FC C3B3427AB0FE0B2EE80C5A6569F0D929FA634304C8DE84B3
29	0xE752E722969244D9BA9F76963FE86FBB3F0D5E6575962238BDF8E7053C144BC0 F029C05D989F27AA9D6A94A85D833A646811B8C3A992B345
30	0x1CB1E939830626C74705E81B9CD8A6ACCB4683081FCB6489835036117C526DE7 3E7AA2C308724600A09D4F7D678713F4DC7DDF479095A3F4
31	0x6B40E6B930122804B02094C97F101A854B45C18885A60DF75F224603840E2827 A99BAC1624688BF5A2524F4F959685F1A39C30BF42C8E811
32	0x7E15A4BCA7A636F4664EF244432AB319BC22081F2E60CD39809B9D61EB62F579 2E9C0F7A26D953CC3AD530291C9BFD768D81DB42B6571BAC
33	0x28A44C0F23163CBE96BE6EAF7D59D8FD3D25781E94DE8A2CE4E1D891F8554490 063499C54EDE0F82E68D6084A076EE1673C378FFDE8A90C1
34	0xE2EE32134BD4603379379C4075F625C4CD50B7AE9E268BDC23D4F5F515235096 F13BB371FC0EA47F51AA9C500D5AB88924F4B603573511BA
35	0xE93BE8D103787F0264FA2B48BA89A0A391BA2BF5874EDAA32C4615542E82CF6F 9C5CD9B673F4B20354165A1E4C5870EAB32368F05956B843
36	0x871D3C28BCA0B73B7FD27B527125946FCFE6380EA70442FF2400F604E37C75C2 4B096D158EAB7FA6BA6A2BB7FF443A107E9240CD471A415
37	0xC83B4DA32110391F5BBEDE74E2F77DD3413935F4344FD03199559677DB7177F6 7545A080A59686A8BFD89775D4BD3DB89DF2F16F52D70E26
38	0x2084BB69F7B5D68E123D30A21504D6C52DA2281E0DAD4086FCF32B55AA0B7A39 AD2881C5E86517CF13D077A42896EB6A189BAFB2452A0004
39	0x29998C83DCDF3BFAAC96879188520F22D78752395CC43D960DBA44FA7CA521FD FFA1E4FB70B56B57A97CC3793136C61EE98081FB2C777AD7
40	0x7E7C8F595ABCC6C418788686F4F0EC53E812951B626435C06E68CD719AFCEB2D DDC9FBA5EBE9C4A710AA6BCE974C80D862E117838D3A6B98
41	0x57F603D724AB5CC9CC8A90C0D5136F1953D52D087D0ABBA80334B31E67A52B19 7E140E9630A4ED3B17ACED01012ECF0BBE1FDC83C51ED77F
42	0xA1FCFDA83AC91D952828A13DAEBC9CEC34A120AE66C2301CE0F40C676E8C876E 7F5A280DDF95BEA867AF2D2405AD596F71F8A65F96B4CB49
43	0x5DFD06055755CDEEA6DE7F7EB40CA7DE376AC54A1A3438BA42E4A9FB5DA99218 A451D80111F23F1CB927C7700562162EBB679902ED9D2BE2

Table 4. Continued from previous page

i	μ_i-value
44	0x59643F1547032036989DE8E64A6FE8F2B3C3DEDCDA9087469AD126C7E987D3C6 205810DAE2BBF248EC9CDA87F835570C9ABE7B2C2ECF322F
45	0xBAC02C503F3C87CBF7AD4E855617187F4F309401102EC22878793250D2624211 8A407738416A818F0ACC175E6E3A1BF43FAA97670A4971D4
46	0xEE0F7D700AC9A3F6B39AE6F1A8521AB0298A9D1B55659C4CAE447362C1C0DD60 24A5555B34479DCD6959327A5295D47A385B578E658E30BC
47	0xD1FD92232C2EEF832B1C83C203B1293612AAA7A101E4EF609799BC577BC47936 D45367C5DF24B5B70BB7BD923F68DC7F1BEB00FA8EE5D377
48	0xBEC4CD6207AB216A0D8B49B9E5085FEAA69923F04AD57D3E8B907B33DE43E386 BCA9E37169F6493352F1CE444D0858EE76188AFA5903EF94
49	0xAEDC2D512C8FFE660372592CAB26100EC65605B2AE28E7B0EB4DDCBEAF88E68D 9C34B166299CDCD1D683627148B2F80C7C9E4EFA1470BF61
50	0xEB361183393DFC9A2181624BAB2BC1B9DA4E1AB84DB23D5C48DC06D1FB6908B0 EC520786ACFEC2473CB149DBD9E222A6F10B655C8BF163DC
51	0x719F9A2BB206CA42945ADC6F36A364BBF7A00D94CA338E4B6DEAA99681F8F1E3 42D998A3D34AA06C63F65E98C53527D7C35460F9EF17F394
52	0xF55877F11E4E33734EDE2A784D119509F4091616823662FED496C7C3C65C420A 1EB2044F2E88DE052AFE3A7B67A8ADAA33615DF1FA6DBEC
53	0x41361F686E3EB9427B4E45CD60908A1114933D788201F9B64F7F4A22372AD150 AA63E3027E40D30AEF464AB8815ACFA7964DC048B00FA1B3
54	0xCCC9DBD96EC2A08313C6156D36A1FCD6D5ABA8E64E731F9828604E853FEB297F AD64EC5817DBF00855F49A1A594A7B70299E93794F0E7313
55	0xEFEFAB86C76FA344F4AE6ADBC522F429696279E5F23DA2F324656C78081A9178 92442838C06EEF3FEEB80DB5D068B3193ECF99DCA352D8FF
56	0xF56119B0FECD6E9F7D242468F140222BC309F57B28E9A4CF84C9E8943B7102C0 D17605FB3BD0E23A82882013FFCAF5FBDED0175D099A54BF
57	0x148606312B49E952403FE66DF3AED9C90586CE1256498002B4F5561D77B51FDB 690691BC2C1E012B0D0EEAEDA4455292747B6B95B35163E8
58	0x9118815E6CB61CADC8B40C5B2A8A9243A345A3F1925E6224805B9AD77A5A7E91 73DEB6FA4603F7C88E7D2CA41BE8991CA511DFFFD23B09F5
59	0x9A204F34253849DC367DCF0AD0B07538D2A2E23FE3EC711EDD7A7433F2CA2298 F2C84D27F9CDB6DAAD0778E72129741772597AE32BCAB396
60	0xE44B264CFBCCC2AFD384EB2F974E52DFF2984D9ACAC9F2A8917490C5F494B60F F372E61382398F81BA9648BD0A5E4EF6243B3D4F0D046E2B
61	0x847F4900CA362B47AAD3A0815C7C414C2ABC15E0695D4410AC671312E6CA49D4 03C8A9B538997B8393F8E41562E0BB3DD9979789DFB9E12
62	0x74D01DDB4BAEAAF75A313EC10575203B63BE98C7723575F12DB307E21642668B A1D9C2B172966A80225B7B8056DCD332930994F58209EC7F
63	0xF94D0035A964E730BE93334E5C42C3F4E68E12821F14D1CC9A05F146ED1966AF 86AFF49B0F9176491F98FE41674E4D4B153F595DE6745C36
64	0x9511992988D2D84E4A612C9938F6D0079DCB6AD1870A8FE56BF562B237BA1057 2FDC5649F27EE3FBA94A85059CAF766BA3FA7CE3B71EC11F
65	0xAAA8980422EC3C8E85DF70C4AC91F89A21BC122336B63978600F455CEAC5C5C0 8C5A4E182C684140DE9C329523626593464CA7D6D6D5DC6F
66	0x25BE77B500330D3BA74209C07081E8258459E1BB10EBC9F6CA342EC1A73EEE8A 458C48A5B05FFCD9334814041CF8D43EFAC56C4B5B1052CC

Table 4. Continued from previous page

i	μ_i-value
67	0xBCEF05A28AA4B87D81213E4AF60336997700E9E884022ABA29DF0854685FE447 95D6806AA2DE553D1BD6CA096A59A9AF91CAE5EF7DBC1FCD
68	0x7BE1534A733FD958DA7FF8B1036AD897FBD723A56708BBE7121E319736374A5D A4D314A04C4BC0959BF2BA70518DA18A4383B7D9665F597C
69	0x85186295B574AC35AE2AF9AC77DBB1DBDCD4F9448980AA6805839CEBE74DEB03 47BFFECA328B589C092D097D4C3E23F7ED5E566C78988355
70	0x5120B1314D360D53C44F7650F4601A15EEF236AEB3467BC7B189B056983FCAF4 5187C568A0EB2E715927A69C9C311266D157E6D7107E82AB
71	0xDC6E71981A509A0DE9DDB8481BACBDBEF2E2E4027C94262429E24DD93D370AB1 33A4AFF46C193CAF4363B6C65FE0A5C56F02E55E1B8C1DF7
72	0x8579757C1187452494A798AD8AF24C5F3F673BDD65DA9513CD676BE122FC9788 4FAB239892A3527652DEF3B83EF87DA6FE6C2866B3B1409F
73	0xE5EA2BE481229D906C416CD15CAC8772B04F9EAE6DC014F818B385365271C0F7 4F0139E5DE12393AB387BF6598A4C996725744A93BF9BF70
74	0xC9AAEB7B475BAF48D21649C91B547418217F6653FE170A5EC9E6B3C4030B73EC 5AE934C75531D945E9D0F1327B7296B5396A49D9651AEAFC
75	0x7BFAF2D971706F87B23AF166CC32BC5A6DED7937F70E38CD45866D15FB45A963 8DD32C8D7D1D1D5AB01D02AF88834E34219E77875AF50447
76	0x82C5649A70B6AD046B18C2C0910C77C47A8FE1F3E076E32E9D637F532491FCBD 3F6D002EEDAEEF7A13FD80543066BEC832ECE44061DC0768
77	0x99F8FB4C1CF9D14401A2A994CF10C06D3665C14F80A3BA077AD10955C4010447 0A24C67F70621CC13AD29CE038DA4FB5065BFC9ED813EA7F
78	0xBBE015434EB1A79A3976718804C04774816096509CB975BFE26556BB9ECBA761 511AA4F452AFA593D261B9C6C1522EEEBBB95B43EE6BB96C
79	0xE125710886AD627A9007FC4039BEF3EBC57D6845CF79B5B87FBF1C6AB4CFB3B7 5BD926B8286E8ECF1A063BEBE373AD40820F17F9BBE2CEBB
80	0xE05FE39E06285AEA680A42958A12FE893C2EE423050DEA4D71443F39EAC508F2 E35A5208E93D80EA416AB2C21BF9769597A14EC1EFAC469B
81	0x42E6EFC1C7013B313080CF7B6F519972D4765390820258BD7867945500167A1D A45DBB7D24C4F087EAB3C8BEED1BC51C06B9E79B986C6100
82	0x5B88F8CF3C2EFD83DF9A9E6B70AE4FAB84FD004B2DD3185866AE15F95A18692C 536ABCB3B8EA29E2EFE53E43B068C0992412C6794BAF35E2
83	0xFB23F08A1285D1960072F2D52954618DF0AF579913BC2AB1C32DFBF9069FA800 001848AED44C78221B8A73B6E70BA1C964DFCD13853190A4
84	0x426BDB2DD14D23868D14D822C2EBDF75F9145CD8D19174B79F73BF2DAE3E17D3 9D3926350D4AC4E6AFF5F41AD193A5EB9B087592D9DBC769
85	0xFDDD74C430880C58954ABB0EF43720587F0A2B482F6DD13AB98879654973A69E 4D13469E286D34E28BA1E3E33ACCAE78062BBA71F8C6DA8C
86	0x5737789D69F5C9A6F46B565E99917015814112944548379F2CC4FFEF9BEEE571 F506604656EB03DE90AB4A3E710EE309EDA3A92D894E136D
87	0xEA3FC1AF75565D8E71A4019E3890F06B644D626AB47FE7866ECC5F51C3835623 6980D68C94877C907B53DE694CF0C5967C66F437E5751F3
88	0x5922D923D766B797C77BEE5CB4433AA95DF2C9BAAA7F1E1EDBD4D849C7B49093 2531188C8871B1D870F2B89B14A8136C2A626058B7A6300E
89	0x812CD0230248E0DAF51E24E5E99A03519BC9DB17011151FB3C5C9C978FE739CB 643624958454A64F5CB26F618F79CB6DD4782089F9A61928

Table 4. Continued from previous page

i	μ_i-value
90	0x54B6CFD047119B1F74E1D3F0D3F30AE281B8121A9C24D6E66C4F2D54EB27C215 8186218CA27665107125 8D339E576628D20C2A8591E41B42
91	0x1B8F3BAA1CCE47FD3A1DE4B172EDE163096686A79D50D604987F44E283EB6FD9 7A1D88BE55A5B5618770EAC7B386E8B0C0AE2DD410A24526
92	0x68CAF74382E5CD49F8D3B9B8A99133C51B1C5A8A7608F8C4CC79A1A71BF8F223 BB7AC6A9B34822AE9010973561EA38B72EE09CC2B64A3CF4
93	0xD9EAB6B225DC45C2B65773F8D50D38378FD5BA64734491ECF18FB9C2ED8A0B23 6DBEAB47F7A74A38BB82A9FFCA9C0E7C566457FE44AEBA38
94	0x77BB8825A5641DA8431BF8C30726E3FE5BB0ACD74D8E2D69E4A8F454ABD4AAE1 E49E192FE56D6B2D2C372C34CC651A701DE6CC4D401AE9BF
95	0xFB1D4F8EE50BBF08CE8880DE9E646CB668326B7D8D86DA1D5D752AF50D2F6DB3 114404DDA02C14D3CCD17B3CB5485811F4180479A06ECD92
96	0xC39E16B58F2AA6B232C0E2D4B8AEC18A48BCB66E43A62C40B09087ED297013B1 34467AD7DEDE48FF956781A596D3D57C668D377FBF85E8AF
97	0xC029E8B0A72A2EDF1216742602ABD36E07DE14947B5DDF1B3825A2BBBB546C59 6A7E7AFCA357FBA03E8794FEB8A811C1962CEF74D6D24DFF
98	0x7C8308AE27C355120B2BA04EC4241DE05F304349019F97841F2155D2E600F31C 6329273BB0B30A39CFE611C539833B04E02345BAD30B386A
99	0x10C6975638A4DF65E91E13A485CBB57D5B8AD9BC84512C7C1082EF687135C30E 0671D4541A48765E24D300D5F0287B7ADECFDF5A9B71A5D
100	0xA7BB5F9790A0ADAB39746E3D9D7D092C20B8E5668562E15DF5944900B4BE273D 9DA6F984BE7652863C40BBF956CF961045FC3D71A02D538F
101	0xF1B7B9F7BF932F650C62C76A7316F6A45CB90441968514000B56296F0C074D58 9B64474F549E006DB9FC495B283DEAA87A83461608992DDB
102	0xD8CC64A9A55DFCB7B7C27A7984A7F357A02EE47E13E5CC001FAF92116BA0E1B4 504CAD451B0C898A9F5FCBC53FBF149408CBD23246B36DEB
103	0xDA7E27B79AEF95421AC57B4CFBDF628C28479EB1F05D757A85796D001D2822ED A1DB45B9FF58EB7FD9D59488FA873FFD383E78348B12C4A9
104	0xECABDA480327B1D781A1D216EE7D69B68058804397621E97EA411F78B8AC7DF7 8FF96D4FB0CBA7907826E462FE0AB72A9C7966A117C0D5CA
105	0x90A6681A6AA52F8C5CB26E16759B5BFA01AB80FB275C507BB55EC1C60917BEE9 12114EE7AEF96A4C7DAC16BDFCB20C7F9066C18C8E40ED5B
106	0x29D03C30B99D6B78DF42997C1D486D14D4F54921D4C4F67ADCB570F67E0D5FDC 54EA6DA08275A0E89615C1A303D8D1A9124C879696B6FE75
107	0x200F6E0B014930F1AC4DF9FCE7159201248A696139EE99C282D9E34E6DED8D0F 5B209A147217D367725A0276E4109672BDDD7491F3D8D685
108	0xE809599EF27795B7EA415C2F332CD865439DD885166D5D7A80DD05D63099EC92 6EFDFBCFDFC5EB005B589F3C4155E3E8CCF529435E9425D8
109	0x9D73BC619D05BCEDF1005B0D3BC2B895AE0BF9A2B3620B815799D009B55B7D1C 3418426DFCF9F5C0B866BF4FDD9915D417E560CD8003A000
110	0x5DD8DF9E2015DC6052F8E2033B45AF9E1877F617D5D8B492AD345BCB8AA1CCD5 39A7A36E74D1E04DC0887FB3AB54A82A89C6E985E529DBD4
111	0xD6FC16905873C78F8776C03456C82921BEA3337E7FCDB8C0AAED8F84074FF594 7854287004AE43DA027DDAC27EBA1E1EF56C8E151AACC102
112	0x325CB5129020686DE58D00B455BF0591FF6C0DF07B16E94BD38602095D701946 D4C26FA4BEFD2C86EC61C6C489A847F60C25913ADFE13FB8

Table 4. Continued from previous page

i	μ_i-value
113	0x47C58815E49D7CC0EF47EE07534905F1F4D861442B75D03B56F1ADFDA0C38F23 68DABFB7AB7FA226D2363B9E3C4B5FCF5C1CBA8751C84580
114	0xEB0B6DCFB59BE109CAF41B0CA15F96333130596E6BF099E0746A738514D857E8 209B67E98D50CB99B0CF75D36EA6D24CFED8D3B98009B52D
115	0x4E810B78570BD419F751DF44AC3D325DAD166DD950FBC9BC0375A3778D736C97 505E7651A9508664D2333FFEC04F824F43103EACA950DE1A
116	0xA7344FB510A585AD5E5CFD4EEE2D83B33AFD5E35C1622FC8C8A98DBFA90E74D3 A20B6C2E0D5044DE1601988F6AFADA1FB31D62556CF12432
117	0x9FDC1C64DA7E8360D9FD1F137C2D4D2EAA4BF5D3CB1E64744F1776D78890D6CF 654B72E92865891862B4B90C88DC83CFCDC22C54E66F58D1
118	0xD538C903A617F3CB80EA8DF95E5DA65FA178B804686CFE1EF4CD8F8FF8F771D8 FFF0258DE722F2EA959544CAF1AC1FC9155A30C11BACEFAD
119	0x9C106A5D48CFE96A38E0B261BAD71FEA1DFF1D1723B534FA6B8F6CBAC7D9F33E 7F4A54232DE439199F9C59804E04A317BFAAE538C3311479
120	0x94B1CA311B2C90B1974545738BDCAC0D907C9BD132893A5FE2AF4AACD31651C8 2E740BBE9A81F4631DDEA825445237ACFD343C3F1C13BBC8
121	0xCF64D48606C30E5935FFBD17D042E83989C5C5F6C253556783E079F021C1EC0D 886444EB3E94D642DB767EB414D4E4F37E95A51410DD2807
122	0xB1AE248C0274C64602015D125450BF0288BD6E4D539EEE346C79969C4BAA0DB3 06AC9A6DCCBE6615B5ABB3979AD76E103196E29B57DBA891
123	0x9F9691844734994FFFCD8A76E0A7C8B8D2F7FCA31E1833105D3345F64F4BD266 943B02436FC46473A5DCA44F6A7936A58A339C7DACEA22F5
124	0xC8CC8C78C3B0740B8D27353D3FCC0D9BA80FA2C676E162F2FF790B87FFEDC125 B53DB8CD5A56B3334256D0DE35D2D680E2ACBC4A483B488A
125	0xE96C842147456FC30C4418AFA9DE86A5EB43086B1794BF93DF75B7D0C1899147 F83D5A1864BA6F8E8FE5764A619B3CFDFB84249A80C9AFC
126	0x24F0D1F3E4BEAC12A4515E4F8FF674DCF8F89F57E61F393723E0EEA29659995B 28E0B1BA7DFB10613AE049074049B5D4CE19DB44BCEC66DF
127	0xACF6DFD8B9EF18C3A4297BC115B69861E7ACEAA890F457E131CB50F4181A8B96 C24FB1AEFF908161C7579D96CDC3288A4ED69FDF5F9A3E1E
128	0xDC4BACCF65C0449CE9FC59D857E9294C9292387FEB398793DCF17EA08F48A615 2EDC245327EEF9AA307ABB0E8B3848EE31FECAB4F16CA8CA
129	0x69D708F0546660A7BA5111E5C510B0B7AD3A6831B21BBD9973DB237EBF3CC527 C1B92937C1234F5C18E2914B2665465547C62F868F37D0A3
130	0xC7938EEE5AE2C7DE7A21603C4FD3EA49E73591EBEAB905B79DB4AD026174EADF 9F504C09BB047D7541E4A7B052CE4CA34BF50AEBCC794189
131	0x513091493AF83565189B896A673A4EAA4309D5BA0FF780BAB5F2B6179FB1FF8F 17BB50A95967D90EF86DEAE200F62CDEB2AF0DB3D0EDE800
132	0x6860290B0DC65B6E2AC8358D66EFF9E4705985777C367581ECA0B67B4F413252 43D5AD112E509D50C29065E633B97E58E476507D0EE9258D
133	0xE8EF4EC18C147F029F9946E5C0B4B8E041CF51DE42A0CB0B367062E095B79BC4 39D035730031C007F00EE0E4DDEA73259A517D563AB0CEF8
134	0x12A0221E1590A557EEC75C9DDF009FCA07B56355E9650940B830622F09E3027C 97C8E9B97B0CCC2A55C79A4A772800CAC8188BBE0A570110
135	0x29D88D18F30754E6ECCCC1DF92CA7150162AF0EAE50669DDE230B14C427CCF39 C7E1BFB332BC2343916E6D3BCEA86B4312D8B927EFC92D71

Table 4. Continued from previous page

i	μ_i-value
136	0xF92AFDD783A307E66A88C367F41C5F11D1E4D7643908196E06C77C6F08BD66AC DF452D1C0CAD1181ADC97DB61217C6B0675E2EFA8F485198
137	0xDDEC2BA25CDA0026A72A423F893E8B766B984290AFFE8222C5E3FF944D1E05F3 B4F7BB222040C502B8146BB4B05820AA8B43FFDFD9EBED87
138	0x43A2599D4D1FE63B4235E73E36B50B2DB16318139ED1FD36706574956DE62C79 973D8B473F5AAFE76173503ECB25E788A7D0B8460D16215C
139	0xB962A92A7D0F3AD0283632280F957BC152EE981E98BED48771876C076A9B5FA1 73B549B7570AF84EFBB370C0311F9E1C05F8E09E247C0C8C
140	0x2F541B5654A631E8F3F42E64976F17A573EDF5CF327A7A4A48CAFE8017195D23 FB2798A5214B71FF72FBDDE429F62E0BCAC34F97718E697
141	0x7052570A6EC5BE1F61BC7565590A5B8F17587E764C7A63BF98128B96E0E28959 32D01CB4C7D06DD46C0FCFB269814DC5026447E10BC58572
142	0x3264AA975ECFF0BD7A5CB541725F97FE4DC08DE418FAB3801292478A29323123 688AB4C5A40C76D25D80CCAB8307669C256C0262BB771C92
143	0x1010895DDD8E4CDD2EDB7CDA58DA197609B5818432B4D34C3454B32B235D7FEE D2755BD4B64301FB5CE3EC62324E0BC6A97E757AC1753F35
144	0x742347163FCCB926FA1684962DD5B138C06BC31175CBD7DA1FF7A62D1DB6B39A D28FD577699F77F04CA45234338FEF24DFA89237086A2557
145	0x1FA2BA2A005AE459AE19248760C0483E8F27FEF32ACEAD1775CE19F9F717DD47 4E365F976A4CC4B7FA9DE6CA50637D1CFAA539DD1C2A619B
146	0xDA286DF22B7EEE57D8A9B7C23A9D02EF19D7605E77BAD2066BE8FF946EFC2EF1 473BCF36185529344BBB020E6A470F397C2C6E9F60428826
147	0xAF65C7D7EE3B5AA949EDD7DC53FF94032949270E9B23A340B730EC10A1DB7CD9 509987859217388679655 50EEE70742F45E223981192D9DF
148	0xB0F4D1A708EFBDA8DC4B1540E54B58DB208D0C2DC18E3F14642A3CADAE566561 753AE815D6D2D9C6A0BF2992CB527B0032BE0CEE03BEBDA0
149	0x3B38D035DA141CCD09251404860618F0CB79BA87B51C993C132CB92ED42FC06E B665B6FF150393E38F1D5E34E1B8E13F651FAC6694050FA9
150	0xBEF9D10AA7B16A8BCDA3CB1AB1D3AADEADE028834751AB295995F5F8E6447480 5B126C6B7B06BE3602202F5AD3698F1E4837D3B427A3041B
151	0x5E107C2AFFDC1CC12EB167B4716850675A85CC637F4FDA450C955B576DFB0409 6E20992CB4FE74792FCF5BDAE04480CEB8496D74CA852FCE
152	0x451F9295943EDB5495D3EF3D7070AAD3C8F5B364B32FA7D9684FEE1D405EBF4D EEE6DED1403E655489EBA27E5954300B9E243022D4F3DEA6
153	0xD9482B59906BE574EA9F6A58F4AB97145A147F0FD3881554F2D07A31215A15DB 6FAE80DF410845469EAEC842661B1EE06F78A5969D1D8022
154	0x33A0FB667603DC6F96B9727F1B1B85A3A4DF700AE65C1E5EBEAE7BDBCCE66E7F EF48EBC1601A82DC72F6AD24D598467CB08CC250F5042412
155	0xC850D791F83DAE4E9E1140963EDD223C17EB2A4EA68D700D8C135C047533BE3A 1ABAA8B6C3CFA8B4ADD1483D7624A5BBCA865868BCBB4037
156	0xED8073F6343B779E6CDFBF2ED2F36C7A74477F511CD633EC7D694C9FA91B47B7 3ED7B52146A4DC9949495BF49F9B7F1873E400CF19DECAD8
157	0x437093B2B137500F9E49F43716E606A815066866851CD2AACEAE659C7A1D7B08 CD76594EB6D3515A247B5167B0527F03F4E483C2C316FB16
158	0x5AF97CDCFA5E21B043FE2B264DC7B0144525DACA7D9E41B951342D7BB9CBC5AB AE23E6ABFDC32E96192A5C5F61EC097BD635C88DE5163A18

Table 4. Continued from previous page

i	μ_i-value
159	0xC1349A9148A1DE9CD02F479CF5EEA5D48143FA3BFCF6E53CDC12646EB2A71AF6 3E759BFD3AC5A6BCC6F150BB55D241F60D1CCDF51742AD94
160	0x5A06934CF07E173020EE1E52E870F5165A94CC598DAD49A7243A393CCC1F93A0 23CAA05C5F7A61CD15358FB60A2B4F84FB4E98C988B31B37
161	0xF959AF1D2DDFA295A5BCCA5E354F3E6EBF15D656CD3C8606B8107A4E95B6FB52 FF01DADE0E6C79C5F6D2D919FC2A39223A32B43ED73BBA81
162	0x1AC2C2698CEF282FC2858A4FE67D6DF74C813CC18F7C5EB05D864BBE555AD22B 0DA807E13081F2B23EBFB4BE2E4E6BF509DD7FC9FAE2E569
163	0x29B3F98D96A12FF496BF867B25DDB66E5303BA5994163C6F808020BEF6001D43 10FCCE92619B15ECD12EAF1646FE40A19F2FF7C21E0EE4D9
164	0x52F739FD6D4C09667AB2FE6195D301C7CBBDECDE02351BE77F1DD2E8755C2B1F 8182B20DD4A5680372AB6A4A1688531863D2ABCE6E60043B
165	0x454B72DE58B03829A3AF76A807307B0C767B4808F93451CD9DBF169E56F0658D A0381307B6C4D0BE113146111C8011C5FEC15235F85FC142
166	0xB3E3C7626CECCDE9C7715FE86589860DA41D47C6ADE70C6B244E7FE05647F89E 4C12AF0CE9EF2209647485C5ED35C8371ADF9FB4C0A716D4
167	0x4299FA467DA2DE9DFE3D7DD48DDE00157AFD7CC14AD23DE177AAEB567BAFF0ED D3DA24B42167EBEEB1C26531595057B6D3F1815E386EAB5
168	0x73A2A0F3CAC6EA5FBCBBB447BFCAC14EE53A3AAF1F3582BCE31ECE3EB65C2338 EB24EB7BF10ABE7308D26D479B022581701EE54E9CE65316
169	0xA4347881B09353EC3D63E18FB726D31A8774B1E6DD0D073B975D1199FA480574 D84FCB0480A9CE9EFEA51EBAA5E4A52E43B6C7E44E7AA4EF
170	0xFDEA4B556D184C821B6D89CAC1966A5AE104480F93EB71C17F2832D81200B830 3665DF4856B1AD773742AA6F48163766658B0E9D58D4E7E1
171	0xAA1F67CE2FF262354B9071B2CDA3D8A1A90F03D4F1DD2AF3A3A9BDB7E3A885A4 54F9EF2B2303BD026912C76166D48818156FBEB1B4531A62
172	0xA22DAB501641732BD930F19D8131CC23A150D530AE2207CB80C34284252E8739 9A396FF9BF4FD9F18537C9EE7E4C973A9437877E43CD1E9E
173	0xED87E22C8616D0E5A183DBE0FCCB0E33601FDEB9808A6515814242CCA4A7C0FE C18CDEB814C50A737D9F6EB793042BE814BFB94EA14F69A4
174	0xF08C327DE00886B9145A652941B63911E730722AB3D2A059B7014DF677C828EB 33839970BCBB06B2E3C64519E7ABDB1E6D4C0DF66BEBC07A
175	0x22A726E31C2528FD14890483E58B2AC308D39E23CA18156470BFA3F1C75321D9 0719150023878E3B82E1D7C375C9B30F45F527059E30115C
176	0x698BB63C258E6D656C53772D2A760B426B5F04D19E087A4E8313B23AE675E920 17E0724E6F123D510D33BDF91B3C8A8E09EA94996075D4DC
177	0xEA86771FF8CFC3AFA89C3E122A656E19BABD99E93061863B8BE97A2A837F4C1B F2B1E0CFE3E9CF539DC4CBA3388A14C94C8F6CDCD02849B9
178	0xF1693E089B8523D64AC4CF2AC27204902301145535648D01C256D6500C817474 F6BEB60FAF0BC39DBF039F27AFB0BF659A2F754FFE7DE30
179	0x91AC5527E94CA1F852569D216C7BD4F5CEB1F092615135252455DF9C21B05A11 77F0EB0DEF333CA1AD629D053346BAC2A8A32FB240BA0984
180	0xAFAD54D7B1D2FD5489D6868AE645BDD163D197374C7B9182A97EB0806F6D5783 448C087468E2DA8BCC5C04B6CB8B1F3D827DFA0AD2303E03
181	0x2A1C06409885BD0CEC6AD1F39F9AB42AC84FEBB6BFDECFAE78EA6C13E5B59CF7 3136D761F762765C600268E6B3BF584287F9639F4962338B

Table 4. Continued from previous page

i	μ_i-value
182	0x125A1D17FB94B0300E1E93798C1947F0C6592208683D4593ACBCFDCB12E48FAE 546F23AB09CC5B969624CF761697E8035654AED581E73B74
183	0x54C343381BE8067A83E5E93BAA0FBA08DA0F820CD22F69CF72C07D88FC19B99F 0A13D4F464C140551BA3EBE1344D23693744FF851840FF85
184	0xAD57594F6AD99DF059D0F62AA222CF985CF0B9AB5DF1D8F731E871F6A906B6D9 0BCA9637B8D37E68FECC71A8F97C6C22C5EE28E66EB3AA14
185	0xF995564013F339535F85109F09F02E973F3FDB591C01EDEAB7B960AA16CFA133 9C85F9C7B67186F308DE0FF96262A2494D232E4C383D2BD7
186	0xB2EBFC81C56A3936064F0E49248172BA556C13F4082FFCC651212D32B671F452 6D5FA19E8DB530A716094CBFA01D20D4AC8E781FC6DAE3B4
187	0xA391F75D139ABF05BFDB149EE356A2BD2CBCF4E8E098D9A5E29C2B64D61A243D 134BF5160594FD4634FD0530C969C940E8B9B4EA6CC88D7D
188	0xB2073E566F1F24E5C5A2F3385098A40DC282BC13CE99446EE55F746F01723625 AA06839AF8670438C6A295711315A60F40052615F327CB8B
189	0xA1D1D0148952478AA9FD260BE82066647EF74BD4451617E2814DA8940B135707 04B3FF6B54EE1F5AE88BE889025707110BD7B10760320FBD
190	0x7BD422BAE4E0CA3C34B2CE207ACBF15102641AAEA1FC1F4FBA1A8423BF7CC69C 05D15D3D54C592C343BE5C55C1222416048F474420E603DC
191	0x27CEFD01B12F16D4B2550AC0E75B47ADAA71FBB50C6BC172969A25D2B60B6348 B9F3524B8D9CC7F0614D6FBCAB200C79F0AE61DED7CAFDCA
192	0xD7A07F9FBEF8A4690C6D9F716D447446EC92A4C732CB47A953912D99CDB98BCA E6518DA0E0BBED421F817BC436C3374A49C60957AB4B1164
193	0x5DA1CE4EAA810851547FB9E41C44573214C8A84E6E2EB1E591E4F62CC61E067A 920B64834B412A982BD64A20746E387F5D3FA5C891B6335F
194	0xC99A317AFE24E23F7261E484DB45386427B7E6E0BB7F0EB979A254BB399FAE3C 7ABD697FFBA60628077E08795641007352B1B57C5E9BD55C
195	0x52E0CF7D6441EB352AE823DE7A051075976C8FEF0FBF3C3775A861922AC01A83 E84675BC2D63A30B588B3B827B3C78795BE59624EEA4D2A1
196	0x6AE6D67BCFE0C491E3CCCA024EDA8B0F2AB5FC4E9BE182FD5100E8B985AB1138 DC01CE9FC92CA88C8EFEB9D8459809E18C24F3AE9349D04A
197	0x7CA84CEADC428E437AED1635010FD2CFCDEBE49E8EADD073575EB881FC2130B7 12BE59CE2D9BC34C31D705C861E0E0042E520CE3AAD711FE
198	0xD5FDB193374922056604835681D8BB6B52F39F84B2EA1C82129DC59C6983B2C8 E4C136CA8259B8C103D07F6CC43C193EDD76E50B76CC2637
199	0x749CA1BB5972FEF068765BC60BDA79D67E2DFB6879A1E3D3222D9A43F8942031 0EDB42293D9E51D87C522B0A4637BA19785DE6BE03BF96AF
200	0x92F5BA03F2C89B825B58E1C6D6E52BA2D38E36BF4D859AB06A83549112C11F5A 40A5EF7F9A53C3AC5BDC1332C546AB82CFDAF0BDC5ECD90A
201	0xA4F7A91507D2E055B58EBDFC6951D9767A6476307D598BC9B0ADD8774BD13E0F 5E75F8D6D7D2005CE7F4F0F8785B3A8656C0B4E7DE7E21FB
202	0xF765CC2F2BED9B1E8017F01E396BEE964338836BF462505F1ADB3ABFD2CA1AAB 0D74348A7307FACC4831AAA24F8228E9CDDD153C2B2F2DAA
203	0xB19260B819E7B58A196B08D07718554A229E58F6548C146AB42753D66E7664A6 9C953200A4D38211C43954DE4C8A17347EEBD10D9C23B303
204	0x4DAA7411D38AE1173518E220693CA84BB3BC6A272CCCACB3CD1F3B39D6D02D8B C293C74656AB64B3F9C2B50A6470F65BD47F67DE12F3EC66

Table 4. Continued from previous page

i	μ_i-value
205	0x69CAD7DB5A824BB191E95D6305B9770684DF52F74596CA17998FD6E5FE786A4B EB5B587FE500D74D7551FDA6DB1B321F7DD0F5FD7E16C496
206	0xFED4215F2D97FA4A43D9B260DEA7C498436396712E3032EE5DC7BB85627ECB68 FAB2D63D6D8EFF9897359BF121F3CA119FBDEA873838D302
207	0xC5E51AECE40C31BD740AF77BEFBF3A826D30BAC5ADFFAF96D711B9077C02CD0E D942C80C215600C3CF1DDE40BA56ABC7C378B5070729203B
208	0x2D4CD912C5FED66692C50EF33033C639888EEF19A60843C4476191B2BD5A0A21 BB8D68A824A8485349B2A8DB9FCBF7BC85EFBF121628F2CC
209	0x8E328DC47CC40B581AB8846F074F5DC356287B98394531F152F51105324DFE51 78BAD3507E24926AAC557BAE383539F1654E5A9CBF073430
210	0xA6E63B122EC79586E0130170BE701CB0D65DEC72FE0BA137F7C907D40C3879C6 79F90112B1CEA48D24C01D5A2AB62CFAF69D2BC5CE98AF7E
211	0xE15522CF79F764AD0230086808EB0CA10E89E37F76CEFE2F60F35A133F5BC888 AB8823FCCD1ACC0B247053AA0DB443808567B14E0678224
212	0xEDEDFF428FF276A9035A1F71FB5053493050F407521A940BCE73458F9E9A4631 326E1571BE64BB6AC31DEF07A45B07B3A676019A2CE1495E
213	0x51EA03CDDA8167BAD54080A1F1857051564A72D916C246E443D7DD58B9309531 A088FCCD615064AD8CFFD103CBFAA247B4A2EE1C91DB08ED
214	0x4822A95C8B3CA23CAA6C65A678BC897A04F2475B7005AF2FD1CE1E9E1C12553A F7C100DE0350E55C217BE83D6FFFF1E185726137EFF05042
215	0x1BF820A6C2A320ACCBFBA85A74E4D07D9C7516643478F1425FAA63F1FCC9F6AB 9002E8C88FFD1D4378773B839DDDCED4945AE3602F8F8C7E
216	0x94534445C3AFE76B76FB98D83B2D3949AF3A97D0679ECC58D8DC93585D921B2F 370BF05C86543A67B76741F18DEEE53187B234B5561E4F00
217	0x9CCD083FC4EAA42A5BD6E35D1DC8DD60F17368CFD931697BDD90686650B00204 33E402B41E213ADECD9931D46B9F1427583BBDD37590BD30
218	0xD8209351575AB32015113653875E321F61A64D2D61723D8E2F7ACA94330288D0 7306CB94C7DD19DFE99577A35EF5DF09050C1BC155BF1A6B
219	0xB76446052C905AAD98C6A401C886F64E57A9B298170A4DB219694A982674E965 27E20AD8A7BD237153F82A75B029301A99FB0C7DABF456B7
220	0x7941EF0E24181D231398A375674A897B6199D66C096D13025B69B49B8CF94267 BF26FB78FA7404B0A3E3E7E8A8D5F79E158C1CF8FCDF917B
221	0xA215764B230D88E5B0CBF9C5D7F82493EE2D3672E154CCF6AEE0BC613F8A0C35 3B7F660007E91B8735A3E9DB30BD0B3C761EBFCB08AB3386
222	0x70E7D309ECF9F09D85249673305ED1C558C39E000A01C6B4AE351460BB720100 496F4A943A0A6DA9024EC0E21585151B3724066DD583DB37
223	0x82C3412D6A7145801477A0B023A03CC0D64EA9C27437CC9A2CB18D5ABC0D8DC8 DD5EA1096FA931A672F12CCD0AEB45A2D6BD71E98796BDF3
224	0x4F4CCF17AB58637A313CB270FAC12FA60C4378F43E050BCC9E20E546C1C96925 7E447AEFDFC3D600BAF8D9CB88089C12A90CC13E8AEC4437
225	0xFAC9346ACF07D8F83EBF0F0C428A0E389078B82FC3E88B8BEB4208AA0091A779 C453EA26AA37AB1F11C311F1DD7079A90AEC8B755931082B
226	0x7C6EFE63EC3A941AAED8F43BF78A3083FB0E54BA52FFB26A611DAB72D8579F39 B4C94DF98DF18A193992B53ADE2A9A9FCCA6C5234D76E011
227	0x28A7ED0BF613231DA84E9DF60AA17F1DAACEAB7B5A4733B30471D07D760FE566 2B44485830D40F63C4555D5896114A06AD2BF44558E70DC

Table 4. Continued from previous page

i	μ_i-value
228	0x16D3F57FD145CA0EA3430D0AD216432739480C2491DA6E03FE77AAB48E5B5538 C385AF1F7BEA03222B6FE63B2AB32A79FEA25D2189B351B9
229	0x76C9E2531CD5D29DD178D623C2109FFC781A9F8115E058D953DF4E58DE857E10 3BF8D54DD1C7589BABFE66D8062B505B789306F8389B323D
230	0xF32899D288FDD2EA4A00DAE702D17B6033F488F360C8F7CE1AF814B6A894342B 7AEC65703BDA4607C8A326CB98A6A1AF154B18717B381E98
231	0x8CAEF7A8B3C36C5594D7E603CA332F8C9796FA483839CB95FDFD3064EA712B01 A8442A8E8C6A263B21F3FF776BBE6BB43B8BEBC2261D2428
232	0x65CF2DCE102DCC7267880EEAE269AD70F848926B757D2102042FE4EBCFBFB937 8CD66DA7EF74699B627CD6067120C5AEB8F774EBB48A62EA
233	0x8F37E968565B4398D940F5B67761E67CC0935AD24F9E011A63EF94A059068B0F A5692751B688867B94CAA9CC8AC4974965C55CE56457F349
234	0x3BE0A77BE599A15AE56B41E85AD5DBE1973EE204231BA987CC4A84C21F3661BA 493D3D0D6CA01E7BA7B24D8BFDE86D071089391B05D287EE
235	0xA63289A084712DAA93033E81235C5F31E7E72FE62B9B5451CED5C006471DD8D0 2AA5B40C8CD0EE10A5885F11AFE46AC541777F77DD6AA3EA
236	0xDA24FFE2D3635C85AC6D99C2CD32E52CAFF9B11FB815DF9384DDC75AC6792E82 4090CA28D16C05542E18D2AE37FB57CAEC7D934E9213A355
237	0xB0F6B288DD0A5B554427E22E6AB9EE60454BD5E9E64051C565C11628AAFC5F89 5645D76B2792A3DB727500959ECB9E5A002E5CC65FD8D2D
238	0xB17CC048AE95B8697A4CB59964EA1E5096D23A3ECB65FF291F4E09728C5F7F10 7D32E68BEEF5021B60EAF9100F1B659B3B843710598CB22D
239	0xB13C57C20F638D1A93529589A3984573628E441A83359BEA54E5CF9266A5EA58 4DF2DDE41FB54DFD0F130DDC17D17612CB90F3EA3E4F7CCD
240	0xEE077C4BC3CF14DDC758D244613583BE6E4DB20C76B0F832A222F9F6998909DF 0F0C4E920A4893EE0281752E0AE55614A252F6B751DFB472
241	0x22664FBE9E7BE473DDF89E22E0D96CB940162C9AE5E7F7900C2CFD82D4A27674 0950FC1729BF125A5B6792D12B4CD197BD282EA7CBEE038B
242	0x95F0F508BEBE9B779C20F4D680DEF231889F50C4EF4BAD2BC30FE1FF4B298159 ABBEE6AEF7D965D7816CB63E97C40AEA8E773A83B8BE10A4
243	0x8FB4CBC700A7832DE004C4C07D324670C04B9A0D35FC92FFFC966B39DF214E2C FD322BA40D244605116DE11B7451758CFC83D8B37C7B7C0E
244	0xCDC575666E8209D748C5F0BCDEB639517B0DAB2AA5FEB86D6121FE89EA3AF078 7D221901A42C7B06D1E6AFB6B3DEAC4DAA852CDD0E7F5AA9
245	0xD53623DFA35E9959C7EEFE6026A90ED5A985F2EE27A5E8A6A0F4E8EE456FF67A 3C2F3D1CD256E9139B7AE7680F2AFA27E161DEA86C2BDFEE
246	0x8F11D19C99C888A1360106E140C848AD5FDA6E6F7274C86F737D3931512A392F FE9A68CCC79B363D4DA67D486B75E8B1F70AA478D56F6615
247	0xD61A0BC05A1CD5874DA6A13F8C729AB393FF50CC62448E62DD6EA021A40C0A09 E5AE17C479268AA9ED1B36D42D2EB182D02173BDCFCBC5AB
248	0xA6F9823A9767E9243AE345532A23C895AA92C3D55E07F4A990C7377A31CF11BF 5505D9CA47276D0AC83F29210E0BE8A92080E5961F6A3667
249	0xB1C31B65684695D8EE15A35697CDFC6E1B8F8128EFB8007EFA01FDFB4E24027B F64764AEC2EC620D8DBF989D3A72C9617F9E967826D9E92
250	0x80DCFD31B9B00110A9B32EC7A68BAF700587263A486851795B2BC40FE63478C9 F5A654169529358866B5DD6503AEA15B444546BBEF5C94D6

Table 4. Continued from previous page

i	μ_i-value
251	0xC47F0F5263C4F812CD2F1164D7727EE78F4F986598ED5113E57E9863F8511CB4 46E99D2A07F473B06DE52A2760978A640504A375D6ADF225
252	0xC02EBED3670E304CC216146D34B6D1E4A0A9F03DD97D687B380FF6BD5F6C15F8 E45495FFB16D67709693F049733183259A658AC76385D95C
253	0x95E2C7088A37EAAB187097B1810AC3A8F9A48B6077C472F93F8EF696E45D9F3F 6C25E094294282DCB6C58BF6F2C1EA9303AECD14C8526B06
254	0x82D7F98550FA773F7836C387F13B0CFB5F6328A2E4ACA2DBC567FF056FCD569B E644D7521C8765951E95E65B7A1A502DAD8272765FD94994
255	0x18F043A1B1DB311FE0762045B6B4D6413794522B5278BEB28AD43DD9751359AD 51D4EA18E50F7CC1E985C89F7EBCCDFD6ACBED2DD8E0C064
256	0x82DD585516023517991 6B2D8B0EF368D6AF5BAEE89141B498AB6837B6532B13B E600E6EF9306F0B180CBA417922EC3E5683E6EDDA9A4A974
257	0x5335A0EDDC8903C146B8EC2808BF045317A630565732D04D29487731207BD0FA 9F0EB9D964F84295D2FD599E8430FA3EC5BB4236F4E89536
258	0xC93F06C08E90599AC7FA3D524466898F158C4DCEE3E75918E5505EEFBE292186 551463452EA31A61FD5AD90DEA7523C8685FBAC5D89E7FC5
259	0xB85A1F3DF61D1D0880EB5404EB97F126DB9BB54FA3778D0B9967C881EE08F311 F465B80BD971FAC24D270447FDCEE3D59F8C1D5016D9048E
260	0x82F1572E245B3E8372CFC84BFF7B96B670BA4DE6978816E483EE3D4D469DE073 B4108764797E64FB95A24E273C60FD9E0DAD5CB9EE9D9CB7
261	0xE73FD6448B08C528B6B73D3D7F95D4F5B4AF45D29E2A518A4F20036F68E84003 AE57B1C0EF5A4361F8AD15EB9D6E54F0C7BB444F51407130
262	0xA8CE6461AE8F4930C638E151E3C6B2B0D459ED042D4C975DE0355F1946557CC4 5AE58D350F23AFD448838A6A21440118F86320CB670452A9
263	0xF3B5AAE7D0426E692A202D03C963CAF847189B4ADD61776403691FE12834107D ACA1CE48C826AF8FB1CF26F3EB7F5BE727B9DFD33C83649B
264	0xC59E5C15DC4FABF7518CF4CD854FE11823DC3C4FAA912F659A5B2E52941CC42D 10047EC3344F0766C64395DD6FA530D835E33E0F570C31EA
265	0x8745628C9BE6C7D800CFEA992F51938A057DD07C075FA3D82601378770344A90 50174A16C6BE9B58E97957A21FE7FE7D042F82E717235754
266	0x56EE478C48702C2B9F3967C32E68905525083835B9F15EFEE62B431B27FD9701 79BDA7FFABA7C2991E1A90E65B630DE0772930E76629FDC3
267	0x655511AB8D1C9E3AC7D95F40605E420C0100DBCA4505494A6E1A58342E7B83B1 3F60F3152CF53F6ABA47F40EFE079B390AFD42E6B57032F7
268	0xCEFC2AEE31BF70DF8C685FE4EE8C04F0E22830CC44CC8F773E7FA8E2602D0A23 27A235C5159E1098395C6D3BFEA26D75908934AA0DB74087
269	0x5A405A77A8D10E2F9F76D61AD1E2A6FD7B762AE8995EB905CCD19AFCFF0B0595 764D8A92EDA122F05E8D0660545946991A7C3B58D9A0F292
270	0xB11D0B086DE4134C5AE41FC53643F86B1970F9F2B525D81DA67D20473DC2C966 D028A9226AA26B0B1C3EEC3F5D10A78B036D6CA6A93FF9FF
271	0xC0CDB3EBFB9BCAB1DFAA2A09DAB3F61A676BCA7F2DA2F035F5FE9CB5ECE794EC F3AD2EB6DA107DB163C036ED300B88FCC5317DECFB9B203F
272	0xEF9A9A1E18E592F4FC720678E6D4B94D2CE198EF43BFE0743EF5AF933283A5B8 0961747459AF6BC46B6C9F8A5359176F0483BA400894DCD
273	0x4500B549AC7635A96AAA0446907F2B3C10F30495BFF30AD2C1AA3E8D1B2F7F98 865D1D41ED58C86D6761C3229638A1788B7EE92E50B57BDC

Table 4. Continued from previous page

i	μ_i-value
274	0xF9F457492C5F711C7CA3805299FE0CDF23749AC109CD99CCB5BA903A764966FB 7658D210F9E08632A7A5CE0674C5AA1678302A4A847993B1
275	0x635CED05B2EF6B8DF7A215929554FCED069AD61DF0E2BC572F9E7AA64610C244 473E9F0AAD1507BFA0AE901AA208BCE3CFED9825E652BF6D
276	0x360CDCBD907AF0E96D6D789007BB018CA0B0B3A92AB6E3A6D7E38E09559B3E4D 93CA26F8D752E57473076281E6277E0248E1F83A3A270BBC
277	0x552906CAC7DD273909C3C06AD444476EBF91087C5F24DD63C0D5FBE2CC507FB2 2BC3A542BDF7A7E462FB80224747270CD85300400F12207F
278	0xF36A655CADFDC9A4DA2F61A10F7C79C4E3E4A641D2D5C324E6DCE9896420F97D B00F03DB1FE88FE5DC51755B66ABC9E9040FF3D2B0582876
279	0xCC8B150460263946E9492F95D8B545191AC2A06A236D994C79CDF13AF8F511A9 44F36E965564889EADAB2D5FB2D217D5CFEA593B7A72ABD5
280	0xF9CC852E16CEE6CB57A18CB7ADAB99503743FD75EB1F4635F4E54F344CB081E3 E1C6D439416CDFC0329634A6448A85E5AB91024154F00C66
281	0x350F0C43A47A4B8B1F2E87E7BB3F9BA7042BE3ADF195F49B9732DA540C97995F 02B35609FE4E3DA6C9698FBB2D48CE3E1D2E17A29E3FD163
282	0xEB1950A9DF9881D22F92791BF076742A27C6ED38785AEBBC4AB4B91988188580 93E72ED6F7B1028B8FCA7B356CC04E1FF1D0C1A12CE0DC05
283	0x333E49AF19520193E163420FA726AE67C858A3646E828244D255A0257BAC243A A92CEF37057F59E0C0E09DFC6CB3A2F8B4B6813A11B4E7B5
284	0xC1B11405836034F27E620D228E9865FCDF79A0FF1AA7849D0C577D6B2874F413 63F3534F5D74C34E7D4CE2906062249DBFDF5FAF66E68505
285	0x484EE3992B0A33796A7BA9953D611CA95EC50D88AA6FF84A22487DC4F0996B04 15D817F1B3B0AD5223B48BC90EBCD94C6888BC5CDFCC2264
286	0x808CEF66496B1D720A331E5724E28104EC6DB4E0546EDAFE226AA34BC65AFEC8 34A11411C946D914B58159C4587D173B43160F12A6A9B6B
287	0xE7BF54DDEB12844B87DADB3C391C773B81DE2EE13161CBC3E4A1E550FA2776DC 2C4780D6431F423AAC945FD93AC07832C8B8B49A9F5CA916
288	0x177B12558AFBDE8A8AA9CDF2BDB789E4AB8B3A81BFD2BDF8018EA19751457FBE F1B9C7D9439C019C8E9FE4A52B2134A68C137E3D2173CBBF
289	0x1635DA76B3826E502A845C26F5527EA1076CC4792E7B8CEEEC3F22EE1B779E08 90E256427146918825360DB18415D81C523151E64D58951B
290	0xEC7D72DE9E871FE767C5FDB1C86306571CBAEA2C4F089B64CE15551832D14587 958FEC757952DCE144B30F9865E8AECAB28A7DC087996F0A
291	0x92FE4CEA32E1AA6A0ECE9BDED8196B12CEE9CFEC6F2AEC1F5A992C5059FF0F85 B2EB89D3B222EADAF2FC1009C58C0BAC96E846C22C4D8B36
292	0x4C9CAE1965FE439F6B7D7CD156887FF21B05423DBEA49A726D2C7F593F53E034 229AD5005E2BD5A4313D2BAB0B42C1D1AB0B03265370175F
293	0x7B43EB90E9D21D57EC6DD9659B1BE6792049DBA25C3DBCFDE81DC12B78E17A6B 3860CAC2ABF2095D376ACCE59441573C3BF2F2FCA9207CF3
294	0xA3AA5E979E43289A57FC1E1BD85839DEAA1BEB5031C4DE9A0461BA52F7233D29 01D96E2A59E5387CE4D1E53A9A2D7DE783B81018192E8B2A
295	0xF8886F7C71B1FFD7CBDCDFB8C186C59367AEBA5F442ABEF7AB156BDAF571A98C 2B735E2AFCFE1C6E599A4EB1A78C3EC7406B9939C4740A97
296	0xBE2C10F408DC2DC6460B0076ACA902DEA8D5DA2F015B3D3BC9672A788E970CAF 4976702AAA3777B40C57D6BF7194DFFE56AC1624BFB1893F

Table 4. Continued from previous page

i	μ_i-value
297	0x6087BD24D3937246F20E05E7B506F9269F8278EF09E0FA6ED4765222ACE90AE2 9D8377862D0E654E73203ABE10FCAE63543ED5C691F907CB
298	0x613A910CE9ACABE41FAFF38F4C2C479EAF28289F5121D631DCEA6668B6469E6D BC897C39A2889F442617D669D62DBE8C7380606C7B845514
299	0x1B9D5611A9D7A614C1F641236266637F0EADE4E1D72D2C351FEFB087B240FD4D 07BF7F740017714EFC112C7F11C5E642CEB1C3E94B5537B0
300	0xE2F230FF68592CE4D177C4A097BC618B00353FDF52D21FDB2AB834591BC8AE41 B28D046BBB6EE2A0F28F4901EDEE5F785DC91B4F18428202
301	0x234E34CAD5EBB7E8E537464F4B91AD906F4E235B4CB333F3CDF7CA7ED9158AEA 030B876921F5B3610BF268AE6CECA0DAEDD0AE55DBB10879
302	0xDF991507D656E6E83575EA73652BE12CD40CA01BEC30E94F2027DF067DD53F7D 55D5BDC5F75C7F46BAE0C23FD0495E368056AC70FDDD0209
303	0x96AD2BB6DB12E9BB5C368A4F80CF00962779CC57C61E56125A75FD9D16AE5342 A498ECFD425A409CCA4EDC86DF587C0016CF8C63D0B7104E
304	0xD6524AE4BE562B45AF2879960D060C66E56FF4ABC7B9EE433214E3F511865F10 004A4DA44E271BD5ADC8D68B9F6A601B089402C6FD1F3292
305	0xFBB6076597B68B4BF4806C2B9C72CC55691713C42334608E755FC693196BCAE6 3808B2F56C89B6CD2983D10AA030C60F2582CA60EF691615
306	0xA282885E21B71D698D123D4CF17EF75901065DC9639339B0897FCDB85C2F0408 8452791D1C15E3CE7435E79382DFFB25E5A367A628749901
307	0xC7CD21A56088A00C8F4A7B9A747851399A1E2B6F688CC040D3FAA0632D382632 729DD68037E58923480EDE6E515C8D8ECF9D0D419AAF698E
308	0x3C744BE226FEC7797C5F7426BA112397E65ABDA3EC0F677876129AB6D807EE15 44C7622AB95068629C42D2093F25D094C6A89BF14737F3A
309	0x57F4FAE273916912F86FA49614168D58BC3CE710D28A254A9A90A9622F2546D9 F451425387D4EA24D8B178391F4932A5E03EC5A3419023A5
310	0x89BCA1C0B40CAB745383089F28C3955E4BF75B37CF451C900CCB4BBF9C334662 693E8FD1CE0AB800B03013C2EA096301A63897DC963F4551
311	0x2F2E55DD54DCF2FAE971F2C9FBFAE8B324A57D3CEADD25ADA6778BABE73B2389 083009CAB1D0F8F4464D3512EE65207A172160651551B106
312	0x5643BB84C70F7F3239C880BA19E489A29D1E543ECC762A885CB4999B330B3258 ABCE6DDE2CCA518C9F2A611F53F11862EACF8652FB96967F
313	0x275D38C4F08BBA171F12E09959C4DBCADB426B977A5070FB3219C52B5DCD2389 5215F18031EDFC9802DDE0617E2B70284F78A0A942B4079B
314	0x37F12844B1BE819C739456FBD78A8536BB1570B55EB2DDA3D8CF38C348809DC4 CD07D5D244EFE74D24D76E3656A0763661045BF62BDC1D29
315	0xF5CE0DF971BC5E0FC8A304F5945A9E28213BDE72CEC4261283BABDB0F369B2CC D8C6871BF7527156F1E9C181CB9D7A70CDC2EAE2D25CCFBF
316	0xBCC47EED910923D375052866D072E0780AA279CE0CC5BF7295D7CE2AF56C96D1 55F03CE39AA7F45A0A939E63D66C2BD9EC40B4F485009793
317	0xA5FFEF29E16377C837470F066CAFF495DAB3EB7D44961ABE240F18D24D5C2093 38BA1F8592A865EE08FB6D44ECDAF6E7B418E6A6F79AC277
318	0x12CF19E6B0C1250C1C51211370E2868D877C49DE51ABB147526E3ED6963BAD42 F7CDDE5FDCAA29ABACB59883AE9AA5F2059B229047D91216
319	0x2869B36ADF288C3346FA298D89BA6C83D8189FD8D2DBE6DA9821122D7E394976 5FE297D9C524CA60BB1926F43CED578C1FD12F4B8F97BCF0

Table 4. Continued from previous page

i	μ_i-value
320	0xC2D1D92526AE025D45E919EFF04108741BADD32DDA4E4920BE514B5C49A093D0 233BADAD0F40708124A1DFAD8CB584BB831EBA7CD6BC1786
321	0x9782DBEFC1EA61E053059350C17C3B84B949835D715F0BB71B45B12515BD6908 88B7D67270365DAFF22652A0B3754CD0B34DF883B23E8148
322	0xE073CB3F19BCADA5FD006B7DD18A2C59287AA97F60564A2415BF6C28605444AB 9B8E3EFD7B3AD7AEB6A9BCF3EF37FFEF7A238A4D3AC3D500
323	0x148537A47F04F950B6815013C70FFD21E79C66F96A10843CF054EA264DEC38F0 E2FD5DE4889029BF002D5C5AED9E9DC29CF52B4D06669ECF
324	0xC88D164423D538864E7183091753461E3AAD15A5B60E9577E497A3F92C3ECE21 043D175AAB8F8E9A8AE04D1B1AB90C98320C39A8C74987DB
325	0x648501A22210DCE2016CEBA0EF8FFCEDF07C6AD4B8142E4C0B204A6955CB11D8 81CE9463B93DEB9E4D6A7CF4F3F2126CB7C0C3BB08995E05
326	0xD0A079216EC16ECABD8BA1BF43DC2FB440263E695F74E875BAB0EF1E363F03BC 10234AF529444F27DBCB6C996588677DA625C655279CE158
327	0x28691F9DDB1F06EDB827F117F22E05B645B36AEACCB9708D41D28911C96EC5C0 D1BAFB791CDAF2B1971D7BCE5866D2747D6E6E222D4A9378
328	0xB27EA247EE69B83E6BA936F7A757CDCB188ECF643D70BE4FC2EAC1C6FE76BA8C 4A64BDC59C5C03D32E2DED7C601CE1AC8D3092E111A81293
329	0xB9ACB65CE1865596B11DFDF5BC9D7B5D69AEA8340FC392804DD6A228DF3BAF8B 6C1D2E87FF38D6AFF985360B6259AF82F7AFCDEDF53AAF96
330	0xB74078AA4ADDF453D2F7B70FE54E8B75E719EA2261A0ECF927DF1FC641926865 7764448516FC7A3B77A8090E98203FDA6333A87D28379E50
331	0x9BE1AF5DB9046F83E897DAFBB28F91E55A982AAA5EA46929112ACED675D97884 39D45E308490EA3DB162226E7D6578E6DAC3B6BA79E389D4
332	0xCA998862A94062DD7372C188B049A362BAC814B0429D719B373A6036F5252D60 E615EDCB383392361FF4135535D365DF8983C49C48E1E48B
333	0x86D00182FC192BC61FCB0F1EA0BAC1F60D4B29E658F2054B2462D93799E7F3CD 1238E43E6616310622540BAC8D4DB4E3C6452F46DAEA7B47
334	0x28DBEAFAA92B4879E1622E9467F9560EE8B8D2B230B9260DD2D1EC894000A4B5 005D6066ED762E5D9086BB7952E8EF61B418A8FB2077AEA2
335	0x231B92F076D979AC6F5D9204AE7EE505BFCEDF30DB27FAFBD34B2E8AB5BC17DB FF86D009B82048C1F97A169B404A24418EF27BD43B0F135
336	0x3CC6366F2C2CF2362BD819B3019B629B9775D3D3F3EB5E63395385DF23035D6C 618FE9CD93ACB106F17D368CD3A81A0EA61D7E61C54A54F9
337	0xFCD42C72447E54EB105A0537FF7175E3BE8F5BBDE77E8B8D4DBAFFDE1BBF5029 51CEC6DCE1BB2E5AE5C1AFC1EE9A4602BFF8B0B610437405
338	0x3F69235925C950FE48AB1DDA35A03EE961BCC7A7E80790C32E923D29B9C38403 B8DF31ADA7C3DCBDC197A68D7FADA0502DA19008B21C1203
339	0xCA1270928D650083791920434E32391266F93B76DF253F20696EB1E6011E8D01 421979396ACDC3602AD9D65B2AD6776894F916BAB918E
340	0x8C33626FAB890B55441A2685FF41FC18C2128D7A9C2F0F02A6F79D422E03546E 91ED61439325A226D1C3903CD9F985EED1A4CB332621BB1F
341	0x96237A2F633F958A25F2153B4BA655076A2AB495F35D26F4BF215B29D370E204 C7E7C3ACFB93E771D562E03E24A637F235818764705D3CE0
342	0xA376B8AC30F13788BEC2855B667001267CD22BDF842B62FEC7332BFD09E33464 3C2F390EF0F853B9DA99FF84EE2302F7675EB3A713D93D0D

Table 4. Continued from previous page

i	μ_i-value
343	0x756540491DCFDB393D61C898ED18DAB54408786000F40FD0178908E4DF4A5392 74CCC46DC68798AA8CB86FEF11D69D6243BDE43555F0016E
344	0x4A05ACB21DCAED7CCB3BF994144EC559377BA3B3FFE2FD05278724865AF9C568 D35DA526CCE6FB438B22BE575B491620FD95D2065FAE108E
345	0xB4E374BFCBC54F516DB42B3E82812D145B2B84F482595160F227D08A1EEEDAD5 8D7B5D841EBB2F1D86F543E1EE2992411EFEBF89EB81D1F4
346	0x4918161108E9E5583001344C36E5CBBFAE082FCB62339B17590D49D61E910542 857E466B491F938482F0EEEAAB96005CA815F04B3222F19
347	0xD161102522767991A38BBAB12128C63272D341CD87E4DFA4ED6BBB39FAC5240D EFAFB1D2FBD09C2A85185A03985662BC79BB9CD981D72A5F
348	0xD02C6E8A78D23E32CC6CF0923D8230DE0CDE49584A4C322150BBF83FEED45EE3 9A91030381052B486B7D56773C33F12222DDEBDD9732B573
349	0x8795BF2F6D0924DD5E45EC902C62B7FE93003FA88A074D58C63A481D739FD9EC E7926302BC208612BD816C0A148DDE9A88574C10BA1F7F0
350	0x5FA681BD81873D443FA3BFC79D96179C9B8C72371FC803C012F190F7975CD36B 85ACE2BAAE740E3E2FC2E844E5A09B54102C94D143818934
351	0x18092A63122795241465BD8C13B9F02A1726C5FD0FED14D4A8F056FB244AEB04 2E9A5D3BEAF867AA7B3B1153C4A65BF558765B826278FFE4
352	0x8AB3C2B4CD1C59AE1F5EADE89363B7B8BE5EB1FB9D5973995A4C80DFDEB762B5 A96E1B61096207634A7917E0A5396A49F2966A00E0E75CE1
353	0x38D515FA35B3782E7629D19DDC272D62FF4274783FD34DFA4B0125FD0B38A59E 855210914D5B9677F7A723BB5F4012B37E639C704CA15378
354	0xE862D73E25A3522EEC72AC12AA3306B60826668F61D6B3342AE98CE686E40841 A47330B21854A4272DAD25C982E838020DF15EF0CE43C78B
355	0x35BAB824ABAB716D8F7059FF2F870012F667115B31035221A22754C51897AE83 A166F05EDC5924E9D8CF007855C4D94AEC84CC40C889BBC4
356	0xAF96969E5C3D1795AF7F265C203080640502ECFABA68264EDC7BEBC9276C592E 25A13899FC9901D1342FD0C022452BE3C2E9960614A74369
357	0x9D899D1034CBAEF9A94D0F5C7024E01307DDA2907F00E6C579AC558E78271505 52BD3904F66C04B8D43EA891440B6D9BDE98E2322921BAA6
358	0x98FD943BE3E2AA3F827E7582F96E36BA89EC2B21595E459381C5B21952930334 09F23AFF48755EEE8231C78D286499EDCF1D96C49FB3E012
359	0x8E40088EE9FB5B3D4F71C39FE7C8519D0675517E9F31E7DFC540F42261F383CA 3A28B5423E132B0F660AA1A95AC3D7942815FB613175DB7C
360	0xEDB18A270BDF073EB1A9F9914C8B744CF86F56B69D6B5199F0D3692F8F50DEFF CFAA194B70E782059B9F3D5A897CF6687393255D50AD31F7
361	0xADB56AE726DB2231C8A33147EFACECD3E4953760544B84E27534AE57DDAAF804 5FD5B78730197EC725AD5300EB8A15439B44E4D248D967FB
362	0xBCC02A569E9DE5DF746E07477E4E846B4923F7BE7FE941F5808331784DC766B3 6CF5667BFFBE22F21FFA6F995142ACD11A3C72E4955C0944
363	0xD122DC6185E2D0F189CF688454FBBDAB6DFFA69E87DB6DE6E26B7B7D9FE7CF67 63DA23090B9540C3C9AC9FFBBEC4D0B3DF83FA7C8C8010AC
364	0xE7CA63A290FA805C2BA8C70FC725F93859317E52038D8E308CB89DC26280116B F9791675869E5187C08A00A321EB43437201636A8CFB48A8
365	0x2BC14757BCEC53C6A623367DAA7FC25C8746B4DC6849FB2597115E2636E7B838 CCD5B7729139F34A2B7502FF074650C50F7D1020D9B55DF1

Table 4. Continued from previous page

i	μ_i-value
366	0x48E6C1D7305649FB0B942DB92F3C982D78C7903F331A16682506681EB6CC7045 BE3D621F306C762417A0E067FA09191E788F0BDA957D5D25
367	0x936AACCAD27929CBE885911D13DF846BCF58D1844E5A834813E7E95ED3DE5FD2 8F818D6EF23FCF27312EAEAFFA98841F500BFC678EE0D17A
368	0x65C96228EC75E752EFB5603A3F2C0FD8DF65766226A19576AB188D6E0B1E1E71 57CD8B5C4E02B40DE74E873301E4C4AE16F889661CE8253
369	0xB4BEA1C62CE0B60191325B896D080A35E8EC0EDC4E416B3FF7AAE28F241E7F54 D220711DF7382336DBF211F0D2D3F6D3159D24BF3EBF7E98
370	0xD5DAF4AEC3848A7BCE5DA93F69452FF4A7F70F7A35CF4C3C0A78BB9006C98D64 CA373493EF2FAC4682D2ED9EB3482C87438A68DC1C1EF229
371	0xB50FF08DC3F07034EC933190480F6EBF89006176C650760A32A3CA8CC0EC69F2 7A85BE1A0E51953DC0CE6A6A5F180A5CD9067B054395BACA
372	0xAA760CC4A14CB969A3769D6D6FC09CFE62CA9BD77B6A5A8FF9AD8DF9F9B7791E 0D5BEE04B36DD7AE153C9A1EB8ACC9F68AEDF19918E223BE
373	0xD9C9A81B9881E368BD7DBC4FB83F1A1DC5D82E1924F6D4CE58C47C8F8F0DDE2E D5AF00CF1F73752C490CA5F2D3B1A2A997CD70453FFE061C
374	0xA5A89CFD1689D2CD9707B20CE09C9F596EF91014A9DFA2EA5FEE707B72CE9CD9 B9F8CA978CCBCC8062C15E7138B98DD0E7E4386C24789539
375	0xA1E22C361636074B160B67998934039CC7BA079C0A9A19C5F9E18EC3182EF702 662B690811B7CAC47E32F7C9E9689985E2DA387C59F1615A
376	0x246A66B25BFD97DDEA754BD501A34CF140E81EA4D7B79DDB8118636CFD9A9F00 2E7B57144D567D0BD368A5D31F796F04298E24A827911070
377	0xDA6489C1C93B7F86B542685F7CD8D8773EA3CA769AEDD13DBA325F9F321E35BA 04B5B7FA664750D828E840FE051515937FEAD31B73FE1172
378	0xBAC745E9CD9E0871DD643A25A687C67D2953D5A824C547E28B1F3D43822AB197 492F1B6C54F6CA2A90D1E406393EE5E8B8FD6AFC3117D4D5
379	0xA73CD1ADE402CEF752BB376A4B84607EC6984F4CEA4D91F74981DB3CC7616868 42364A4E49D371E31EC03E71369EE3E631048D7A136D7237
380	0x3A70C297727A3B3CC3C13112C107A8422726B900F5C398E979D24FA00EB8C3EA 5D3644513808555F98ED3B7329ACA9E50CF1E808F82D0651
381	0x1019FA21527837DFDEB736CAFCFF2339F01D9D5C002A9E95199804DF14EADA9F 53AF299766F5F23A6D13E2AECD453CC7872FA9BC32BAB2F4
382	0x59831415A0D65676C2009ED1D09174379FEC489A5A54294B1DF0691A356C0A15 4280B83B46C8BCD7609DD4743485669BF7F2216274732050
383	0x4DAE519EF2227F003C1947E4DA3EFE9DE0925C364F9DF220287E65079814DAE1 D759464B4692A9E29F4B6A06568C1E6F1DBD07232022F8C2
384	0x21B0E9D04CC671AB219128A1633FEBB37CDD1F55BB2BED44C8BDEE1F0AEEFFE9 E8BC7A57B467240ABDF6F1FBA6215ED8CCCCADB6101B8CBE
385	0xB59F955EFB6E5D6D511C6198C509E5D9F1F357BA032B9AEA8BD7DD7EEBDF7C2E 067283A84341789B161AEDE414F6F9FD7BA489EE2B77C9AD
386	0xD04E6EB329216D15148AB94C3B4EE26AC3504F26F78D8F52A4021054AE8C7F48 88103A884768B79336836B4BBF70C3C710425EA3951EE223
387	0xBC96D3924B145A1924D8575B0FC6B84F648C1D4B8F6685FA9EF3E9CD053620C2 D8A808FEBD01BDA26BF46FEA3C68CA543EFD20B73F188A4C
388	0x9F801E26019E3E497FF87195375C5660B7C879F3EB3A0A348EF2D0DB5F264263 EAE63069AC06A631608A6B479A7F65DF6F1E64CA6CC93FA9

Table 4. Continued from previous page

i	μ_i-value
389	0xB002D58DCD6FF2CAC5F418B30C50863956CF0DE88029133D31987BF41AEAF1CEB32B07D80AFE39A95CC447A926E560D08EB528B0AA9A16F8
390	0xC93043ED75F04145A2407A24EC122C1D6734D717E5CBC2413E29C4663B04E509A63637B7AA769373465BB99D41D62442E46A1A6DAC3F39BF
391	0xD7D803C16B0130FC7352DE2F4D4567653D47C6B544F31FC0FA2BF0A1DFD31B71080434B4A737EF1B0DF29542D7B2A4DDBC0254AD5B47F680
392	0xA6102D891A04C7CB223C6077C31F97EF6E6A12676C5776EC1AD82992329F509FF8F8632A891E365D742471CBFC5834E53E0F17013E55671C
393	0x8F9A7422CFDB7AC324358B347ABDEE7F523DFF1423145F438D6F3A5313667A6E582AF8E7FB0D86E7DB1C6B70D1AB835D653CD91626A9F512
394	0xCBB0EA8B9F683C9361A3D25BA68E1C470D2AEA7F3FEA7BDC0961B625CC8C701A78509A88944ED7B94F05D4801DBF0BEE787D4E39E5FC20DB
395	0xFB86B53B79BA2DB73B946CE6E82AB08DDBDD1EF812C57ACC82B93D9E1D3B44E340DCCBCDAADC43414551F787ACE4E03A2C4C8BAE190254FF
396	0x93CDC4CE4CE81FFABB47EB8797B210A0F3FF1D0547CF88FFE65252435626A86A81CE0F414E1ED8946A6EA616ACB786902B85CD6CDD138882
397	0x1C54236F2D09633EA27930835A40C8EC990F4AC3C528C60151ED537674250744EF861837ECF93A72654A129C7D290B59B37929898DF511F4
398	0xF24F8142A8826DAB8A21FC87DB3289A0CD615A3082E15B55F5C35365E7A759BD0BE764C10FF5C27AA0FD50857E2E8366877E77301C3B6F5C
399	0x2AAB973F28ECA18BBABA17F53B4BD2F8DD7FCFCA4E6AE13247ABFDAC47240CFFB72515FE740146D57B04D15AA1715DBABF9656F61888EBB3
400	0x9F30005BBA386C19625822EF4047D672B56FDB237E9CAB887B9A918D6D4CA06241C5B25C1B7473C94A0C39C87D044D24F264B3C65EC93851
401	0xF5AC3C52B2CDF0BCDC4B97180D0648BD2BD20A018D2545155E9A7E9F6C15E0279EB39D084AB5229AE4CB77E3F39513E4BEBFAEE99B3BBB65
402	0x75625225A188EAEF97B885D143BEB1D89BC4FB931470B42EAF163513A9EDBB637ECB03883EA52079A3210036CBA315E9D076F65C6B47ED3E
403	0x3F22012D042EB909AE42ADA1E3F424B5CF2FDEF8C3CF5AD331856AB5F79B4A02F7E5756A5CFCE28A894A33C530110089372B7E6223AA4F8E
404	0xAD0949025729FF94ACC7EED2BF72D6B5E9D0CEAAB701BD35C4CB9EC2B65781FFC6B97D4CE1A2FAB71590B23DFDBA2488BAB5FD507DFB1641
405	0x8F84333F4299476F39D6152A646BA3086B55C8FAED385AE1074E0C3E54A982566E623003678A05C7B61867B037A8D98C0BAF6BBF1B78A527
406	0xB7CF10A103D2CF6CC89384263E66943A27C1C96E8B2F356092AE5039FF8CF97EC621673413D2CEB0FEC3DA1EC4E1103A72C5A40E812945BC
407	0x7E5CFE739CFFA62DACA6A5E524400F5B47385577AC14BF985A0929D7FCCEB4833472CF718BE0E2D9F975B2340D078188441773C5F619DA64
408	0x95C53612CC00A26ABF9CC38E0965B3CE832C37E087A3059C27595BECADF1EEF17A46BE9A0BB5D9AC507216D9F838CCAF938CBCB268CC331E
409	0xF01BEE4E06B3606F103C9947CF88A7BE313E8B54CE012792F9EF97A43E13EE81E1E43C6226E0A3B2AC5354082B6B5B232FEB3C9C14021136
410	0x35C8A36A8E2AF13FEEAD9B1F4C428637E93330E193150F73EC81C46507395169796E6316EFC282F0A8655FFA091CA3FE73F6E622CF664959
411	0xE9C3347045696AA3D5D379C833CFDA1476468B14850E8938BF78A913ABB99F3140EBAA2FE7E1EBF4CDDC3AAB6AE406B2788E9CD2DDF1491E

Table 4. Continued from previous page

i	μ_i-value
412	0x67BCDF7F49CF7289C8A04CB9E270A73C38D8EEA1381804CA56506972E9605518 6DBE32E6F822004A42CCBE3AFBE06BC8F4E9CC24E978E75E
413	0xDD9CCF67FCC6EDC721BE507C7100FE2A2FA5AF5DCE653751F617FE7032F6C3DD ED7B31CF61244DAF740C9C26B63199A8E336B2BD0BCB171D
414	0xB7280CCF14A3C735AEC17A35B6CEADF2728E620E9BFAC92A070431263F6A2952 0A57F0B6F76D9C99CADE4C3587620BEBD801B1D0BBF83E26
415	0xCEC856746A30BBA4241FBFCAF292923D9B3803DAA1FD4D6C01D5F5B27E71D9C9 AD4EB138414ECC7A5B7A33AF99375DCF9724D2203A32F1A6
416	0xDD246D4EE5CD3DA4C5C7C8311299430BC705ADF1918A7303B234E7EFC1A3CF0E E381A6EBF45D07E9A6AAC755F3EBF6BE8579B1C3C9EDF427
417	0x7AD2CDBF1FCF7FE793737815C6861253A397D77693B49E82106E0240CB980FF1 398E83D477CDE6A3576BEDEB119D1AB5AC75452C95D05461
418	0x96F1BC9DFEF6D18366E65E3D293FEF7749C9EE27F8E0803A39EFA627EB250413 F9923D1B64BD246376C1564169C3E13A49CAFF69D5DC21C2
419	0x24E3385B78F76CC96C0DAA465EDD809FCCA26F7C0CF999608E297B53A3E41E17 37D7F0E4A72CCE668ADC2D62528C29C7F963F33C2699C66B
420	0x46B40D49641104883BA72A21F06FE9682E6102C267A23F4D3BCD500C077226C5 B71E5A0915CDDBEDD89B1F3BDF9682AB7672E1082F6A754B
421	0xF7D4A532E0BDB8BE4C86BB09B1A3307A0EC835AA7F064B7EE8CB9AFECB515236 36BCB03A4FF0BC7CB07E6EF30A45CB17F09F5A2BC5F38563
422	0x292183C704FF8058A5736A6BD19363B2D933A7C2619C217237308C63B20EB1DC 0E7E96C7507C98E982F251B16D2B0FD52CB2224E6837507D
423	0x87943E6E1C7674DDF91C4DBA41F9F492BFDFE3A49E1CD1434EA88DC97808CFFF 716890283BCB9B28B6A9E6B390E8176BC958C93999BC0429
424	0xECE649540C6E5096D8BEE4DE0FABC32A319A9C2E558E53E47C713046F19C22F1 65F2A6EF4DDB5DD88AF28CBCB9AA196B4DC587FBD7C5DAE4
425	0xF32C0C28C993FCF3ECD16207D8F4643D6CC0C53635454EDB4EACFE6FCD7BAF74 BA0EF181F0A49D4564913175866FACAEC2DDADDAA55A1D3C
426	0xC4136809417D002EE7F02DA401952026730607C93990405BB51035B2DAD3AF50 487BF9E6A8FB69EB478B424C755BF1A4F16941B65C2B690C
427	0xF2639B551910B2980D23DD9109D6A215EB681B678C1A6E00177664F2862FC5D6 8FBE5B9B2B197DA417EC2E573C316109CA364FC9CF7838BE
428	0x83BE0C877707B8F580225770A7B12B937157F41AB53F3CE0C06A9C56237A9176 3B8D1AB3AA12B614560965A674FB57C041FA3A2AEA931F51
429	0x67F782D816039BEC73D81B39737955A571489DCFD9C8FC9A4FEFAD1E0C352DAE 2A1A7A895F040A82999A6860D2B4A66A86136338520E9B07
430	0x25268A46ADAC6707F5A73052E3691371338C4C9C777A08A6C94332C1D3FDF36F 288E59209CC6B02561E2D192202D952B827840C921426752
431	0xDBA34D32047CEFA6368530ADA834A5E66459A0FFD2E512C356E5419344E42C5C 13F1ED82E6F83922CA7016342DED348D2DD5671A2C486DA
432	0x917B70F888CB951DAEE86E8E11CFAD350D090F25DDA72126916200939F1D6351 CD525708A3D873C6E36E7E310EDA6F93E5864B61A464E9D0
433	0xF9631F1E7444CDBEF02BBCCE04F01B52B8CFEE2260E9D4EBB64364C41EE44E36 4C8B991EB743AEEAAE5035B6E86554902C9DDA8DD95C190C
434	0xCB66FB824ED569734D69AD8551B4FC1A4147919193AF5DA89BC05E2B85A2BBD8 0C06935F759F6078221ADDDB0EFDDB1BA931B6B294953FE1

Table 4. Continued from previous page

i	μ_i-value
435	0x877BEFAED8B5B4896F0E142F5B6FE869196FEA8102EFF569D4365D256F7F4C48 822C41DAB442B72AD603244F764F41FDAB6E74C451C7BE7C
436	0x12D4CD008282F3664E220B875668A36EEAFE54DB1FA02108DBE2D10A1150941E 196E18767610AA0043BFF75EDC6FC1379001E0C98BA091E9
437	0xA2EE7213ABE1603769C7CB039FFA9A0759C358242036419A6AFD9C92B447F16D 2EA782906F5E4167DE65351BC7DCC018F8E04C6477248E25
438	0x325B1F730AA00D1CEE639DF1CFDD1B57C249C98364C91BD3ED5B0501EE8C8A34 6B83F36BDFDAE8210627D40683171DF61A378ECB13FB7874
439	0xDC9825BF94786F39DA7D545A5007D930D9DC80DA75C64DF6A3A87271C76FF7FF 3BA8CB2CDE2280BC8844ACDDE923395A99EBF56AC01ECE29
440	0x2A53F4945A42F3BB857D3BE7611536E119C855A7DA70B5D43CDE83700430947D EFAEDB57FE4FFFD8379B9C16C7160A7372110FFB44622601
441	0x36E15965BB060DBE20264548E461885B3F4454548BAF645AFB661F3AA44AFA6D 0FC82594CE8DF57D99988D35ED50CEC07723E1830BA66014
442	0xB29ACE8138F93EEFC8E377DCE679C7A710B94C028C7AB491EA6430B0115D69BB AB7E0144B7EC9E6B6F4437317559249E1B11EEE3A3CE98B7
443	0x6EA17BF89F18CE6959BC1193353320F961BB16BF6E5CBDCD49DFBFB2DB90FF92 C10800F0C79204ED245F195C6B933A89E1AFB410D0761291
444	0x7B5C2EC3A449966C008AAD4DD31A767F75252EB78D1402EC8CAE5968A18837A6 E2E5909A1A89BF3E110B7BC956DE4DA1D299E6F8ADAFF4A1
445	0xDD0260321B81E3A34E7CB684F509982CBAC9A0FC4B10C817E10FC664D84D9D50 F433701A109A262B4085B08889C3E363B11401A8DAF75326

E Magma code

For the sake of illustration, we present in the following script an implementation in Magma of the X25519 function based on Algorithm 5.

```
/* X25519: fixed-point scenario */
clear;

/* field */
Fp := GF(2^255 - 19);

/* curve */
E := EllipticCurve([Fp | 0, 486662, 0, 1, 0]);
h := 8;
a24 := 121666;

/* base-point */
P := E![9,
    147816194475895447910205935684099868872646061346164752\
    88964881837755586237401];

/* point S of order 4 */
S := E![1, 48802004052532134862652268456126542835229456083994414\
     50108585062254396887963 7];

/* P - S */
D := E![0x215132111D8354CB52385F46DCA2B71D440F6A51EB4D1207816\
     B1E0137D48290,
     0x5199331F1F5630BBFA49B1B1B02B207B493D0A63BB4F8F01C0\
     11242F9C6E9E7C];

/* pre-computation of the constants (u_{Pi} + 1)/(u_{Pi} - 1) */
M := []; aux := P;
for i:=0 to 251 do
    m := (aux[1]+1)/(aux[1]-1);
    Append(~M, m);
    aux := 2*aux;
end for;

/* X25519 */
function X25519(uP, uS, uPS, M, k, a24)
    /* init */
    K := IntegerToSequence(k, 2);
    U1 := uS; Z1 := 1;
    U2 := uPS; Z2 := 1;
    s := 1;
```

```
    /* main */
    for i:=1 to 252 do
        /* timing-attack countermeasure simulation */
        s  := (s + K[i+3]) mod 2;
        TU := s*U1 + (1-s)*U2; U1 := s*U2 + (1-s)*U1; U2 := TU;
        TZ := s*Z1 + (1-s)*Z2; Z1 := s*Z2 + (1-s)*Z1; Z2 := TZ;


        s := K[i+3];

        /* addition */
        A := U1 + Z1; B := U1 - Z1;
        C := M[i] * B;
        D := A + C; D := D^2;
        E := A - C; E := E^2;
        U1 := Z2*D; Z1 := U2*E;
    end for;

    for i:=1 to 3 do
        /* doubling */
        A := U1 + Z1; A := A^2;
        B := U1 - Z1; B := B^2;
        U1 := A * B;
        A := A - B;
        Z1 := a24*A; Z1 := Z1 + B; Z1 := Z1 * A;
    end for;

    /* projective to affine */
    Z1 := Z1^-1;
    u1 := U1 * Z1;

    /* end */
    return u1;
end function;

/* Diffie-Hellman: key pair generation phase */

/* Alice (dA) and Bob (dB) private keys according to Sec. 6.2 of
    RFC7748
   after the preprocessing described in Sec. 5 */
dA_RFC := 0x6A2CB91DA5FB77B12A99C0EB872F4CDF4566B25172C1163\
         C7DA518730A6D0770;
dB_RFC := 0x6BE088FF278B2F1CFDB6182629B13B6FE60E80838B7FE17\
         94B8A4A627E08AB58;

/* Alice (QA) and Bob (QB) public keys according to Sec. 6.2 of RFC7748
     */
QA_RFC := 0x6A4E9BAA8EA9A4EBF41A38260D3ABF0D5AF73EB4DC7D8B7\
         454A7308909F02085;
QB_RFC := 0x4F2B886F147EFCAD4D67785BC843833F3735E4ECC2615BD\
```

```
         3B4C17D7B7DDB9EDE;

/* alice */
QA := X25519(P[1], S[1], D[1], M, dA_RFC, a24);
print "chk (QA == QA_RFC)?", QA eq QA_RFC;

/* bob */
QB := X25519(P[1], S[1], D[1], M, dB_RFC, a24);
print "chk (QB == QB_RFC)?", QB eq QB_RFC;
```