Compact Encoding of Non-adjacent Forms with Applications to Elliptic Curve Cryptography⋆

Marc Joye[1] and Christophe Tymen[2]

[1] Gemplus Card International
Parc d'Activités de Gémenos, B.P. 100, 13881 Gémenos, France
marc.joye@gemplus.com
[2] Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
christophe.tymen@gemplus.com

Abstract. Techniques for fast exponentiation (multiplication) in various groups have been extensively studied for use in cryptographic primitives. Specifically, the coding of the exponent (multiplier) plays an important role in the performances of the algorithms used. The crucial optimization relies in general on minimizing the Hamming weight of the exponent (multiplier). This can be performed optimally with non-adjacent representations. This paper introduces a compact encoding of non-adjacent representations that allows to skip the exponent recoding step. Furthermore, a straightforward technique for picking random numbers that already satisfy the non-adjacence property is proposed. Several examples of application are given, in particular in the context of scalar multiplication on elliptic curves.

Keywords. Public-key cryptography, non-adjacent forms, elliptic curves, smart-cards.

1 Introduction

Most public-key cryptographic primitives rely on group exponentiations (or multiplications for additively written groups). We refer the reader to [3] for an excellent survey on exponentiation techniques. Many implementations are based on the square-and-multiply methods or one of their numerous improvements, for computing g^x. One of these uses Reitwiesner's recoding algorithm [11], which requires the knowledge of g^{-1}. This supposes that the value of g^{-1} is already available (i.e., precomputed) or easily computable, as is the case for elliptic curves [8].

Reitwiesner's algorithm expresses exponent x with the set of digits $\{-1, 0, 1\}$. Because of the sign "−", each digit is usually encoded with two bits and so, *twice* the size of the usual binary representation of x is needed to store its Reitwiesner's representation. That's why it is suggested to compute it "on the fly" when needed [4]. In this paper, we exploit the non-adjacency property of

---

⋆ Some techniques presented in this paper are patent pending.

Reitwiesner's representation to encode the exponent with the *same* efficiency (up to 1 bit) as the binary representation.

Moreover, we present a straightforward method to pick a r-bit random number so that it already satisfies the non-adjacency property. In this case too, NAF-recoding is thus useless. We prove that the numbers we generate in this way are *exactly* the elements of $\{0, 1\}^r$; there is therefore no loss of security for cryptographic applications. We then extend our method to τ-NAF recoding for use on Koblitz curves and conjecture, based on numerical evidences, that the numbers we generate are *almost perfectly* distributed in the set $\{0, 1\}^r$.

The rest of this paper is organized as follows. In Section 2, we briefly review elliptic curves and Koblitz curves. Section 3 presents our new encoding method. In Section 4, we explain how to pick a random number satisfying the non-adjacency property. Section 5 highlights the benefits of our techniques in a Diffie-Hellman key exchange and for El Gamal encryption on a smart-card implementation. Finally, we conclude in Section 6.

2   Elliptic Curves

Up to a birational equivalence, an *elliptic curve* over a field $\mathbb{K}$ is a plane nonsingular cubic curve with a $\mathbb{K}$-rational point [12, Chapter III]. Elliptic curves are often expressed in terms of Weierstraß equations:

$$E/\mathbb{K} : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \tag{1}$$

where $a_1, \ldots, a_6 \in \mathbb{K}$. In characteristic $\mathrm{Char}(\mathbb{K}) \neq 2, 3$, the equation may be simplified to $y^2 = x^3 + a_4 x + a_6$, and in characteristic $\mathrm{Char}(\mathbb{K}) = 2$ the equation (for a non-supersingular curve) may be simplified to $y^2 + xy = x^3 + a_2 x^2 + a_6$.

2.1   Scalar Multiplication

Together with an extra point $\mathcal{O}$, the points on an elliptic curve form an Abelian group. We use the additive notation. The computation of

$$[k]P := \underbrace{P + P + \cdots + P}_{k \text{ times}}$$

is usually carried out through the binary method. However, noting that if $P = (p_x, p_y)$ its inverse is given by $-P = (p_x, -p_y - a_1 p_x - a_3)$, we see that the computation of an inverse is virtually free. So we can use a binary *signed-digit* representation for k, i.e., $(\ldots, k_2, k_1, k_0)$ with $k_i \in \{0, 1, -1\}$. This speeds up the scalar multiplication by a factor of 11.11% compared to the binary method [8] (see also § 3.1), on average.

2.2   Frobenius Expansions

In [6], Koblitz suggested the use of so-called *anomalous binary curves*. These are elliptic curves over $GF(2^n)$ given by

$$E_n : y^2 + xy = x^3 + x^2 + 1 \quad \text{or} \quad \tilde{E}_n : y^2 + xy = x^3 + x^2 + 1 \ . \qquad (2)$$

The Frobenius map, $\varphi : (x, y) \mapsto \varphi(x, y) = (x^2, y^2)$, satisfies the characteristic equation $T^2 - T + 2 = 0$ for E_n, and $T^2 + T + 2 = 0$ for $\tilde{E}_n$. It corresponds to multiplication by $\tau = (1 + \sqrt{-7})/2$ on E_n, and by $-\bar{\tau} = (-1 + \sqrt{-7})/2$ on $\tilde{E}_n$.

Since multiplication by τ is cheap (i.e., $[\tau](x, y) = (x^2, y^2)$), it is advantageous to write k in τ-adic expansion (i.e., $k = \sum k_i \tau^i$) when computing $[k](x, y)$ on E_n. Note also that since $\tau^n = 1$, we can first reduce k modulo $(\tau^n - 1)$ [7]. The advantage resides in that the norm of k, as an element of the Euclidean domain $\mathbb{Z}[\tau]$, is $k^2 = O(2^{2n})$ and the norm of $k \bmod (\tau^n - 1)$ is smaller than $\text{Norm}(\tau^n - 1) = \#E_n = O(2^n)$ by Hasse Theorem [12, Theorem 1.1, Chapter V]. Consequently, k reduced modulo $(\tau^n - 1)$ has a τ-NAF representation of length almost equal to the length of its binary representation.

3   How to Represent the Multiplier?

3.1   Non-adjacent Forms

A *b-non-adjacent form* (or *b*-NAF in short) of length n for an element x in a ring $\mathbb{A}$ is a sequence of digits $(d_{n-1} \cdots d_0)$ such that

$$x = \sum_{i=0}^{n-1} d_i \, b^i \qquad (3)$$

and

$$d_i \cdot d_{i+1} = 0, \quad \forall i \ . \qquad (4)$$

Equation (4) captures the property of non-adjacency. In the sequel, a *b*-NAF representation for x will be denoted by $\text{NAF}_b(x)$. Moreover, the integer associated to a *b*-NAF, say d, will be denoted by $\vartheta(d)$; i.e., if $d = (d_{n-1} \cdots d_0)$ then $\vartheta(d) := \sum_{i=0}^{n-1} d_i \, b^i$.

Reitwiesner [11] proved that each integer has exactly one 2-NAF representation. More importantly, he proved that the 2-NAF minimizes the Hamming weight amongst all the binary signed-digit representations. NAFs are thus particularly suitable for fast exponentiations [3]. See also [8,13] for applications to elliptic curves. Another representation minimizing the Hamming weight is described in [5].

3.2   New Compact Encoding

An r-bit integer has a 2-NAF representation of $(r + 1)$ digits in $\{0, 1, -1\}$ [11] and hence needs $2(r+1)$ bits to be encoded, that is, *twice* more than the binary representation. However, we can exploit the non-adjacency property (Eq. (4)), that is, a '1' or a '−1' is always followed by a '0'. We therefore suggest the following simple right-to-left encoding:[1]

$$
\mathcal{R} : \begin{cases} 01 \mapsto 01 \\ 0\bar{1} \mapsto 11 \\ 0 \mapsto 0 \end{cases}, \qquad \mathcal{R}^{-1} : \begin{cases} 01 \mapsto 01 \\ 11 \mapsto 0\bar{1} \\ 0 \mapsto 0 \end{cases} . \tag{5}
$$

With conversion $\mathcal{R}$, a 2-NAF representation requires only one bit more than the binary representation to be encoded. For example, by right-to-left applying $\mathcal{R}$, $\mathrm{NAF}_2(29) = (100\bar{1}01)$ is encoded into 101101:

$$(\underline{1}\,0\,0\underline{\bar{1}}\,\underline{01}) \mapsto (\underline{1}\,0\,\underline{11}\,\underline{01})$$

Moreover, the inverse right-to-left transformation, $\mathcal{R}^{-1}$, unambiguously gives back the NAF representation.

It is also possible to design a left-to-right encoding. The previous encoding implicitly assumes that the NAF begins with a '0'. For the left-to-right transformation, we add an artificial ending '0', e.g., $\mathrm{NAF}_2(29) = (100\bar{1}01.0)$. Then we left-to-right apply transformation

$$
\begin{cases} 10 \leftrightarrow 10 \\ \bar{1}0 \leftrightarrow 11 \\ 0 \leftrightarrow 0 \end{cases} . \tag{6}
$$

(Note that if the last obtained digit is '0', we may discard it.) Again with our example, $\mathrm{NAF}_2(29)$ is left-to-right encoded into 100111:

$$(\underline{10}\,0\,\underline{\bar{1}0}\,\underline{1.0}) \mapsto (\underline{10}\,0\,\underline{11}\,\underline{1.0})$$

and the inverse transformation gives back the NAF representation.

3.3   Frobenius Expansions

On a Koblitz curve E_n (see Eq. (2)), it can be shown ([3, Theorems 5 and 6]) that every (rational) integer $0 \leq k < \#E_n$ has a (non-unique) τ-NAF representation

$$
k \equiv \sum_{i=0}^{n+1} k_i\, \tau^i \pmod{(\tau^n - 1)}, \quad k_i \in \{0, 1, -1\}, \tag{7}
$$

and that this NAF representation has $1/3$ of nonzero digits, on average. The same also holds for the twisted curve $\tilde{E}_n$.

Consequently, the proposed encodings (5) and (6) can be used to efficiently store k (or more exactly $k \bmod (\tau^n - 1)$) in the computation of $[k](x, y)$ on an anomalous binary curve, as well.

---

[1] For convenience, we write $\bar{1}$ for -1.

4   How to Pick a Random Number?

4.1   Generating a Random Integer

Cryptographic applications frequently require random numbers of a given length. We show here how to obtain an r-bit random number already in the non-adjacent form. Our technique is based on transformation $\mathcal{R}^{-1}$ (cf. Eq. (5)).

First, pick a random number k in $\{0, 1\}^r$. Next, right-to-left apply $\mathcal{R}^{-1}$ to the binary representation of k. If the most significant digit of the resulting representation is $\bar{1}$, pad the representation with '0' until position $(r - 1)$ and add a '1' in position r. The representation resulting from the whole transformation is referred to as the $\mathcal{R}^*$-*representation*.

Here is an example to illustrate the technique. Let $r = 3$. So the set of 3-bit numbers, their 2-NAF representations and their $\mathcal{R}^*$-representations are respectively given by

k	$\mathrm{NAF}_2(k)$	$\mathcal{R}^*(k)$
0	(0)	(0)
1	(1)	(1)
2	(10)	(10)
3	$(10\bar{1})$	$(100\bar{1})$
4	(100)	(100)
5	(101)	(101)
6	$(10\bar{1}0)$	$(10\bar{1}0)$
7	$(100\bar{1})$	$(10\bar{1})$

From this example, we see that:

Theorem 1. *Transformation $\mathcal{R}^*$ induces a permutation on the set $\left\{\mathrm{NAF}_2(k) \mid k \in \{0, 1\}^r\right\}$.*

Proof. By construction, transformation $\mathcal{R}^{-1}$ uniquely maps the representation of a r-bit number k to a binary signed-digit representation. When the leading digit is $\bar{1}$, a '1' is added in position r; the corresponding representations are the only ones having exactly $(r + 1)$ digits. Therefore, each r-bit binary representation is transformed by $\mathcal{R}^*$ into a different $(r + 1)$-digit representation. Noting that all these $\mathcal{R}^*$-representations verify the NAF property (Eq. (4)), are all different, and represent numbers smaller than 2^r, the theorem follows.   □

4.2   Generating a Random Multiplier on a Koblitz Curve

Solinas proposed in [13] an algorithm to compute a τ-NAF of length less than $n + 2$ for a multiplier on a Koblitz curve (cf. Eq. (7)). This algorithm involves two steps: an Euclidean division in the ring $\mathbb{Z}[\tau]$ and the computation of the τ-NAF itself. The idea here is simply to generate directly a random NAF of the required length using transformation (5), and then to use the associated integer as a multiplier.

Let P be a base-point on a Koblitz curve E_n. For some primes n, one has $\#E_n = 2 \cdot \text{prime}$ (or $\#\tilde{E}_n = 4 \cdot \text{prime}$) [6]. So w.l.o.g. we suppose that P has prime order p and that $p \geq 2^{n-2}$.

Now, in order to randomly generate a multiplier for P we proceed as follows. We pick a random x in $\{0,1\}^{n+3}$. Next, using transformation $\mathcal{R}^{-1}$ we set $g := \vartheta(\mathcal{R}^{-1}(x))$ and use it as a random multiplier when computing $[g]P$ on E_n. (Note here that the length of x is chosen to be $n+3$ since the leading digit of $\mathcal{R}^{-1}(x)$ is always 0.) The arising question is to determine the distribution of g in $[0,p)$. First, it is clear that $\mathcal{R}^{-1}$ generates all the NAF representations of length smaller than or equal to $n+2$. Hence, from the discussion in § 3.3, $\vartheta(\mathcal{R}^{-1}(x))$ reaches all the elements in $\{0,\ldots,p-1\}$; or equivalently

$$\{0,\ldots,p-1\} \subseteq \left\{ g = \vartheta(\mathcal{R}^{-1}(x)) \mid x \in \{0,1\}^{n+3} \right\} .$$

To estimate the quality of g, we bound the statistical difference of $\vartheta(\mathcal{R}^{-1}(x))$ where x is considered as a random variable uniformly distributed in $\{0,1\}^{n+3}$. The following argument does not conclude but gives evidences to conjecture that the distribution g is close to the uniform one.

The statistical difference of g is defined as

$$\delta(g) := \frac{1}{2} \sum_{i=0}^{p-1} \left| \Pr(g = i) - \frac{1}{p} \right| . \tag{8}$$

The next lemmas respectively give a bound on the statistical difference in term of exponential sums and a formula to approximate it numerically. Their proofs are given in appendix. In the sequel, we let $\Im$ denote the usual complex number verifying $\Im^2 = -1$, whereas i is merely used as an index.

Lemma 1. *Let p be a prime and let X be a random variable in $[0,\ldots,p-1]$. Then,*

$$\delta(X) \leq \frac{1}{2} \sqrt{\sum_{i=1}^{p-1} |E[\chi(iX)]|^2}$$

where $\chi(x) = e^{\frac{2\Im \pi x}{p}}$.

Lemma 2. *$E[\chi(ig)]$ is equal to a_0, where $(a_0,\ldots,a_{n+2})$ is the sequence recursively defined by*

$$a_j = \frac{1}{2} \left[a_{j+1} + \cos(2i\pi\tau^j/p)\, a_{j+2} \right] \quad \forall 0 \leq j \leq n , \tag{9}$$

with $a_{n+1} = \frac{1}{2}[\cos(2i\pi\tau/p) + 1]$ and $a_{n+2} = 1$.

Letting $f(i) = E[\chi(ig)]$, Lemmas 1 and 2 imply that

$$\delta(g) \leq B := \sqrt{\sum_{i=1}^{p-1} f(i)^2} . \tag{10}$$

The numerical computation of this sum becomes untractable for large values of p. Nevertheless, considering numerical experiments, we conjecture an estimated bound for $\delta(g)$.

Theorem 2 (Conjectured). *On a Koblitz curve E_n, for $n \geq 100$, we have*

$$\delta(g) \leq 2^{-n/5} \ .$$

Table 1 summarizes different values of $B_{\text{samp}} := \frac{p-1}{N_{\text{samp}}} \sqrt{\sum_{j=1}^{N_{\text{samp}}} f(i_j)}$ for random points i_j. The values are obtained using various n and p corresponding to Koblitz curves. The list of the primes used is given in appendix. We took $N_{\text{samp}} = 10000$. Of course, N_{samp} is completely negligible compared to p, and these experiments do not prove anything about the exact value of the bound. Nevertheless, we remark that $B_{\text{samp}} \leq 2^{-n/\alpha_n}$ where α_n seems to decrease as n grows. This fact indicates that our conjecture seems reasonable.

Table 1. Estimated bounds for $\delta(g)$.

n	$\lfloor \log_2 B_{\text{samp}} \rfloor$	α_n
109	-23	4.7
113	-27	4.2
131	-39	3.6
163	-40	4.1
233	-75	3.1
239	-80	3.0
277	-93	3.0
283	-87	3.3
359	-120	3.0
409	-158	2.6
571	-196	3.0

5   Applications

This section gives a non-exhaustive list of practical applications of the new encodings (Eqs (5) and (6)), namely the Diffie-Hellman key exchange and the El Gamal encryption. Finally, we discuss the implications for a smart-card implementation.

5.1   Diffie-Hellman Key Exchange

Let E be an elliptic curve over $\mathbb{F}_q$ and let G be a base-point on E. To exchange a key, Alice and Bob choose a random number x_A and x_B, respectively. Alice computes $Y_A = [x_A]G$ and sends it Bob. Likewise, Bob computes $Y_B = [x_B]G$

and sends it to Alice. Their common key is $K_{AB} = [x_A \cdot x_B]G$. Alice computes it as $[x_A]Y_B$ and Bob as $[x_B]Y_A$.

The advantage of using the proposed encodings is that randoms x_A and x_B can be seen as NAF representations and so the computations are speeded up since subtraction has the same cost as an addition over an elliptic curve. Moreover, if uniformity is desired, Section 4 shows that the proposed encodings can easily be adapted to meet this additional requirement (under a reasonable conjecture for Koblitz curves).

5.2   El Gamal Encryption

Again, let E be an elliptic curve over $\mathbb{F}_q$ and let G be a point on E. The private key of Alice is x_A and her public key is $Y_A = [x_A]G$. To encrypt a message m for Alice, Bob first represents m as a point $M \in E$. Next, for a randomly chosen k, he computes $C_1 = [k]G$ and $C_2 = [k]Y_A + M$. The ciphertext is the pair $\{C_1, C_2\}$. To decrypt $\{C_1, C_2\}$, using her private key, Alice recovers $M = C_2 - [x_A]C_1$ and so message m.

Here too, the proposed encodings are advantageous since the NAFs do not have to be computed. When computing multiples of points on E, Alice and Bob just consider the binary expansion of the scalar multiplier as a representation given by encoding (5) or (6). Note that since the mappings are 1-to-1, there is no loss of security; moreover as before, if desired, uniformity can be achieved.

5.3   Smart-Card Implementation

The proposed methods are particularly suitable for smart-cards. A direct generation of the multiplier avoids the precomputations proposed in [13]. Furthermore, the RAM space needed for the scalar multiplication is smaller, because the multiplier can be generated on the fly, and does not need to be stored beforehand. Besides, the code space required for the routines that precompute the NAF is quite large. Consequently, the proposed encodings enable to save a non-negligible amount of space in ROM or EEPROM.

6   Conclusion

We proposed a new method to encode in a compact way the non-adjacent form of an integer. We gave several applications of this encoding, including the generation of a random multiplier in the context of elliptic curves. For Koblitz curves, we gave an argument to conjecture a bound on the distribution of this generator. Finally, we exposed practical examples of this encoding for some widely used cryptographic schemes.

References

1. W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. 22, pp. 644–654, 1976.

2. T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. on Information Theory*, vol. 31, pp. 469–472, 1985.
3. D. M. Gordon, "A survey of fast exponentiation methods," *J. Algorithms*, vol. 27, pp. 129–146, 1998.
4. IEEE Std P1363-2000, *IEEE Standard Specifications for Public-Key Cryptography*, IEEE Computer Society, August 20, 2000.
5. M. Joye and S.-M. Yen, "Optimal left-to-right binary signed-digit recoding," *IEEE Trans. Computers*, vol. 49, pp. 740–748, 2000.
6. N. Koblitz, "CM-curves with good cryptographic properties," *Advances in Cryptology – CRYPTO '91*, LNCS 576, pp. 279–287, Springer-Verlag, 1992.
7. W. Meier and O. Staffelbach, "Efficient multiplication on certain non-supersingular elliptic curves," *Advances in Cryptology – CRYPTO '92*, LNCS 740, pp. 333–344, Springer-Verlag, 1993.
8. F. Morain and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains," *Theoretical Informatics and Applications*, vol. 24, pp. 531–543, 1990.
9. P. Nguyen and J. Stern, "The hardness of the hidden subset sum problem and its cryptographic implications," *Advances in Cryptology – CRYPTO '99*, LNCS 1666, pp. 31–46, Springer-Verlag, 1999.
10. A. Pinkus and S. Zafrany, *Fourier Series and Integral Transforms*, Cambridge University Press, 1997.
11. G. W. Reitwiesner, "Binary arithmetic," *Advances in Computers*, vol. 1, pp. 231–308, 1960.
12. J. H. Silverman, *The Arithmetic of Elliptic Curves*, GTM 106, Springer-Verlag, 1986.
13. J. A. Solinas, "An improved algorithm for arithmetic on a family of elliptic curves," *Advances in Cryptology – CRYPTO '97*, LNCS 1294, pp. 357–371, Springer-Verlag, 1997.
14. J. H. van Lint, *Introduction to Coding Theory*, GTM 86, Springer-Verlag, 3rd edition, 1999.

A   Proof of Lemmas 1 and 2

Similarly to [9], we make use of the Fourier transform in our proofs. We first recall some basic notions (see [10, Chapter 1]).

Let p be a natural number. For each $m = 0, \ldots, p-1$, we define a vector in $\mathbb{C}^p$ by

$$\boldsymbol{u_m} = \left(1, e^{\frac{2\pi \Im m}{p}}, e^{\frac{4\pi \Im m}{p}}, \ldots, e^{\frac{2(p-1)\pi \Im m}{p}}\right) \ .$$

We also define $\boldsymbol{e_m} = \frac{\boldsymbol{u_m}}{\|\boldsymbol{u_m}\|} = \frac{\boldsymbol{u_m}}{\sqrt{p}}$. The system of vectors $\{\boldsymbol{e_m}\}_{m=0}^{p-1}$ is an orthonormal system in $\mathbb{C}^p$ with the standard inner product.[2] So, for every vector $\boldsymbol{v} \in \mathbb{C}^p$, we can write

$$\boldsymbol{v} = \sum_{m=0}^{p-1} \langle \boldsymbol{v}, \boldsymbol{e_m} \rangle \boldsymbol{e_m} \ .$$

---

[2] That is, $\langle (a_0, \ldots, a_{p-1}), (b_0, \ldots, b_{p-1}) \rangle = \sum_{j=0}^{p-1} a_j \overline{b_j}$.

For $m = 0, \ldots, p-1$, the coefficients $FT(\boldsymbol{v})_m := \langle \boldsymbol{v}, \boldsymbol{e_m} \rangle$ are called the *Fourier coefficients of $\boldsymbol{v}$*. The sequence $FT(\boldsymbol{v}) = \{FT(\boldsymbol{v})_m\}_{m=0}^{p-1}$ is called the *transform Fourier of $\boldsymbol{v}$*.

Lemma 1. *Let p be a prime and let X be a random variable in $[0, \ldots, p-1]$. Then,*

$$\delta(X) \leq \frac{1}{2} \sqrt{\sum_{i=1}^{p-1} |E[\chi(iX)]|^2}$$

where $\chi(x) = e^{\frac{2\Im\pi x}{p}}$.

Proof. We let $\boldsymbol{a} = (a_i)_{1 \leq i \leq p-1}$ and $\boldsymbol{u} = (u_i)_{1 \leq i \leq p-1}$ denote the sequences defined by $a_i = \Pr(X = i)$ and $u_i = \frac{1}{p}$, respectively. The Cauchy-Schwarz inequality yields

$$\sum_{i=0}^{p-1} |(\boldsymbol{a} - \boldsymbol{u})_i (\boldsymbol{u})_i| \leq \sqrt{\sum_{i=0}^{p-1} |(\boldsymbol{a} - \boldsymbol{u})_i|^2} \sqrt{\sum_{i=0}^{p-1} |(\boldsymbol{u})_i|^2}$$

$$\Longleftrightarrow \delta(X) \leq \frac{\sqrt{p}}{2} \sqrt{\sum_{i=0}^{p-1} |(\boldsymbol{a} - \boldsymbol{u})_i|^2} = \sqrt{\sum_{i=0}^{p-1} |FT(\boldsymbol{a} - \boldsymbol{u})_i|^2},$$

by noting that the Fourier transform is an isometry when p is prime.

Furthermore, since $FT(\boldsymbol{a})_0 = \frac{1}{\sqrt{p}}$, $FT(\boldsymbol{a})_i = \frac{1}{\sqrt{p}} \sum_{j=0}^{p-1} \Pr(X = j) \chi(ij) = \frac{1}{\sqrt{p}} E[\chi(iX)]$, and $FT(\boldsymbol{u}) = (\frac{1}{\sqrt{p}}, 0, \ldots, 0)$, we finally obtain

$$\delta(X) \leq \frac{\sqrt{p}}{2} \sqrt{\sum_{i=1}^{p-1} |FT(\boldsymbol{a})_i|^2} = \frac{1}{2} \sqrt{\sum_{i=1}^{p-1} |E[\chi(iX)]|^2},$$

as required. ☐

Lemma 2. *$E[\chi(ig)]$ is equal to a_0, where $(a_0, \ldots, a_{n+2})$ is the sequence recursively defined by*

$$a_j = \frac{1}{2} \left[a_{j+1} + \cos(2i\pi\tau^j/p) \, a_{j+2} \right] \quad \forall 0 \leq j \leq n,$$

with $a_{n+1} = \frac{1}{2}[\cos(2i\pi\tau/p) + 1]$ and $a_{n+2} = 1$.

Proof. Recall that g is obtained from x by right-to-left applying $\mathcal{R}^{-1}$ to $x = (x_{n+2} \cdots x_0)$, that is, by right-to-left applying the rules

$$\begin{cases} 01 \mapsto 01 \\ 11 \mapsto 0\bar{1} \\ 0 \mapsto 0 \end{cases}.$$

If we denote by $y = (d_{n+1} \cdots d_0)$ the result of this transformation, the independence of the x_i's implies that for $j > 0$

$$\begin{cases} \Pr(d_j = 0 | d_{j-1} = 0) = \Pr(x_j = 0) = 1/2 \\ \Pr(d_j = 1 | d_{j-1} = 0) = \Pr(x_j = 1; x_{j+1} = 0) = 1/4 \\ \Pr(d_j = \bar{1} | d_{j-1} = 0) = \Pr(x_j = 1; x_{j+1} = 1) = 1/4 \\ \Pr(d_j = 0 | d_{j-1} \neq 0) = 1 \end{cases},$$

and

$$\begin{cases} \Pr(d_0 = 0) = \Pr(x_0 = 0) = 1/2 \\ \Pr(d_0 = 1) = \Pr(x_0 = 1; x_1 = 0) = 1/4 \\ \Pr(d_0 = \bar{1}) = \Pr(x_0 = 1; x_1 = 1) = 1/4 \end{cases}.$$

In particular, the sequence of digits d_i forms a Markov chain. Now, for $d \in \{-1, 0, 1\}$ and $1 \leq j \leq n+1$, we define

$$x_{j,d} = E\left[\chi\left(i \sum_{l=j}^{n+1} d_l \tau^l\right) \mid d_{j-1} = d\right].$$

By convention, we also define $x_{n+2,0} = 1$ and $x_{0,0} = E[\chi(ig)]$. Using Bayes's formula and the definition of a Markov chain, we have

$$x_{j,0} = \sum_{d' \in \{\bar{1},0,1\}} \Pr(d_j = d' | d_{j-1} = 0)\, \chi(i\,\tau^j d')\, E\left[\chi\left(i \sum_{l=j+1}^{n+1} d_l \tau^l\right) \mid d_{j-1} = 0; d_j = d'\right]$$

$$= \sum_{d' \in \{\bar{1},0,1\}} \Pr(d_j = d' | d_{j-1} = 0)\, \chi(i\,\tau^j d')\, x_{j+1,d'}$$

$$= \frac{1}{4}\left(x_{j+1,\bar{1}}\, \chi(-i\,\tau^j) + 2x_{j+1,0} + x_{j+1,1}\chi(i\,\tau^j)\right).$$

Furthermore, $x_{j,1} = x_{j,\bar{1}} = x_{j+1,0}$ if $j \leq n$, and $x_{n+1,1} = x_{n+1,\bar{1}} = 1 = x_{n+2,0}$ because of the non-adjacency property. Therefore, for $0 < j \leq n$,

$$x_{j,0} = \frac{1}{2}\left(x_{j+2,0} \cos(2i\pi\tau^j/p) + x_{j+1,0}\right). \tag{11}$$

Bayes's formula applied to $\chi(ig)$ yields

$$E[\chi(ig)] = \Pr(d_0 = \bar{1})\chi(-i)x_{1,\bar{1}} + \Pr(d_0 = 0)x_{1,0} + \Pr(d_0 = 1)\chi(i)x_{1,1}.$$

Consequently, as $x_{1,1} = x_{1,\bar{1}} = x_{2,0}$, the recursion formula (11) holds for all $0 \leq j \leq n$. It remains to compute $x_{n+1,0} = E[\chi(i\,d_{n+1}\tau^{n+1}) | d_n = 0]$. By definition of partial expectation, this is equal to

$$x_{n+1,0} = \Pr(d_{n+1} = \bar{1} | d_n = 0)\,\chi(-i\tau^{n+1}) + \Pr(d_{n+1} = 0 | d_n = 0) + $$
$$\Pr(d_{n+1} = 1 | d_n = 0)\,\chi(i\tau^{n+1})$$

$$= \frac{1}{2}[\cos(2i\pi\tau/p) + 1],$$

as $\tau^{n+1} \equiv \tau \pmod{p}$. Thus, setting $a_j = x_{j,0}$ for all $0 \leq j \leq n+2$ concludes the proof. $\qquad\square$

B   Values of p and τ Used in Table 1

n	p, τ
109	$p = 3245185536584267014874486564614677$
	$\tau = 1384233455896981573696930343929817$
113	$p = 5192296858534827627896703833467507$
	$\tau = 3126605487954413221319732774018522$
131	$p = 68056473384187692693232012949340998512977$
	$\tau = 19651107411586109242203251508094536395677$
233	$p = 34508731733952818937173779311385127605709409888622521263 2087024741343$
	$\tau = 2598851043790259083579160746211533789223151387669521214510026908255781$
239	$p = 22085588309729804119791218759286481494821656132170984887480219215362213$
	$\tau = 11282863073195994094887783155777768965281647003338630860570162193181852$
277	$p = 6070840288205403346623318458823496583257511049878650876488417556189162211\backslash$
	$\quad 65064650683$
	$\tau = 35399204550743338457406852595283913963265820174574971604878094514304600611\backslash$
	$\quad 18244376699$
283	$p = 3885337784451458141838923813647037813284811733793061324295874997529815821\backslash$
	$\quad 9704422603873$
	$\tau = 1622537357594321742305829784866063775146998334908836481845747957062258441\backslash$
	$\quad 1461142295062$
359	$p = 5871356456934583069723701491973342568439206372270799668110818246094859171\backslash$
	$\quad 2441244948823651724787481656489986631$
	$\tau = 1622537357594321742305829784866063775146998334908836481845747957062258441\backslash$
	$\quad 1461142295062$
359	$p = 5871356456934583069723701491973342568439206372270799668110818246094859171\backslash$
	$\quad 2441244948823651724787481656489986631$
	$\tau = 1802336455319286892936377811824359533150445662632888008361051662683364461\backslash$
	$\quad 40502957029835041564559810928091169119$
409	$p = 3305279843951242994759576540163855199142023414821406096423243950228807111\backslash$
	$\quad 28924191050673258457774580140963665906177313586711$
	$\tau = 9537745491735000985208826110701086823200291728364758559856030400028993221\backslash$
	$\quad 9131717954822608799115149503556964162824684187434011$
571	$p = 1932268761508629172347675945465993672149463664853217499328617625725759571\backslash$
	$\quad 1144780212268133978522706711834706712800825351461273674974066617311929\backslash$
	$\quad 68242161709250355573368527667311$
	$\tau = 1737617153452667100450620654361010136575696214949435748424058322350852001\backslash$
	$\quad 3406707707278390660980311506772309793318508751262778482693856154191237\backslash$
	$\quad 91259488944630714673206344593711$