Applied Mathematics & Information Sciences **2**(2) (2008), 123–133
– An International Journal

New Strategy for Doubling-Free Short Addition-Subtraction Chain

Raveen R. Goundar, Ken-ichi Shiota, and Masahiko Toyonaga

2-5-1 Akebono-Cho, 780-8520 Kochi

Faculty of Science

Kochi University, Japan

Email Addresses: {*raveenrg, shiota, toyonaga*}*@is.kochi-u.ac.jp*

Received November 20, 2007; Revised January 10, 2008

The efficiency and security of most elliptic curve cryptosystems are based on exponentiation. One such method could be the use of short addition-subtraction chain. This paper proposes a new strategy to find sufficiently efficient doubling-free (SPA-resistant) short addition-subtraction chain for an arbitrary integer by utilizing a precise golden ratio. It is termed as the golden ratio addition-subtraction (GRASC) method. The proposed method has attained 12% to 28% reduction in the average chain length compared to other doubling-free addition chain methods known in the literature.

Keywords: Addition-subtraction chain, Fibonacci sequence, elliptic curve cryptosystems, side-channel attack.

1 Introduction

Elliptic curve cryptography was proposed independently by Koblitz [6] and Miller [10] in 1985. It facilitates two parties to generate a secret key for communication across an insecure channel. The most fascinating feature of elliptic curve cryptography is that it utilizes much smaller key of size 160 bits to provide same level of security as other cryptographic standards such as RSA of 1024 bits. The strength of elliptic curve cryptosystems (ECC) is based on the infeasibility of the elliptic curve discrete logarithm problem. The major building block of most ECC is computation of the form kP known as the scalar multiplication or exponentiation, where k is a positive integer (a secret scalar) and P is a point on an elliptic curve defined over a finite field. Therefore, efficient and secure exponentiation methods are vital in ECC. There are many exponentiation methods proposed in literature [1], such as double-and-add using binary scalar representation, triple-and-add using ternary scalar representation, use of addition chains etc. Most of these methods are dependable on the

secret scalar or exponent, hence through side-channel analysis it leaks secret information. This is known as side-channel attack, recently discovered by Kocher et al. [7]. In one type of side-channel attack, known as simple power analysis (SPA), the attackers uses the power consumption to monitor each operations. Due to each type of operations having difference in power consumption, helps attackers to retrieve secret scalar. One way to overcome SPA attack is the use doubling-free addition(-subtraction) chain [3]. It results in a fixed sequence of operations, hence attackers could not detect any information through SPA. Note that inversion of a point in elliptic curve cryptography is cost negligible, hence addition and subtraction operation involves same power consumption. The addition-subtraction chain involving one doubling, that is 2, is unaffected by SPA attack since the computation of $2P$ is required in almost all ECC.

Our contribution will deal with the construction of doubling-free short addition-subtraction chain, enhancing efficiency and security in applications to ECC. Although finding a minimal addition chain is known to be an NP-hard problem [9], we propose a reasonably short addition-subtraction chain involving mostly of Fibonacci pattern. In fact, there has been many strategies proposed in the literature [2,5,8,13] to obtain a sub-optimal chain. Our propose method is dependable on two parameters, therefore it is considered not to be sub-optimal. We experimentally select suitable parameters for a 160 bit integer for the best results. We will make comparison of our proposed method with the other doubling-free addition chain methods known in the literature.

The rest of this paper is organized as follows. In section 2, we give a brief overview on addition chains and Fibonacci sequence. In section 3, we propose a new strategy (GRASC) for finding moderately short addition-subtraction chain. In section 4, we discuss our experimental results and make comparisons with the previous methods in the literature.

2 Background

In this section, we briefly state some classic definitions used in the study of addition chains and an overview on Fibonacci sequence. More details could be cited from [1,4].

Definition 2.1. An addition chain computing an integer k is given by two sequences $v = (v_0, \ldots, v_\ell)$ and $w = (w_1, \ldots, w_\ell)$ such that $v_0 = 1\,, v_\ell = k\,, v_i = v_r + v_s\,,$ for all $1 \leq i \leq \ell$ with respect to $w_i = (r, s)$ and $0 \leq r, s \leq i - 1\,.$ The length of the addition chain is ℓ.

Definition 2.2. An addition-subtraction chain is similar to an addition chain except that the coordinate $v_i = v_r + v_s$ is replaced by $v_i = v_r + v_s$ or $v_i = v_r - v_s$.

Definition 2.3. The Fibonacci sequence is defined as $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$ where $F_0 = 0$ and $F_1 = 1$.

The Fibonacci sequence has many properties [4, 12]. We recall one here by stating the following Binet's Formula.

Theorem 2.4 (Binet's Formula)**.**

$$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}\,, \quad \forall n \in \mathbb{N}\,,$$

where $\phi = (1+\sqrt{5})/2$ *is the positive root of the real polynomial* $X^2 - X - 1$*.*

From the above theorem it is easy to deduce the following classical result.

$$\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \phi\,, \tag{2.1}$$

where ϕ is the golden section and here we term it as the golden ratio.

3 Proposed Strategy (GRASC Method)

In this section, we discuss our propose strategy for finding an efficient doubling-free short addition-subtraction chain by utilizing a precise golden ratio. We term it as the golden ratio addition-subtraction chain method or GRASC method in short.

The last term, v_ℓ in a doubling-free addition chain is maximal if the following condition holds:

$$v_i = v_{i-1} + v_{i-2} \qquad \text{for} \qquad i = 2, 3, \ldots, \ell\,, \tag{3.1}$$

that is, when v is a Fibonacci sequence. Thus, our aim is to maintain a Fibonacci pattern. Our strategy creates chain starting from the last term. As deduced from equation (2.1), the ratio between the two large succeeding terms in a Fibonacci sequence, maintains the value near ϕ, therefore we consider multiplying the last term (in the case of 160 bit integer) by an inverse of a golden ratio to get its preceding term. That is

$$v_{i-1} \approx v_i \times \phi^{-1}\,, \tag{3.2}$$

where $\phi^{-1} = (-1+\sqrt{5})/2$ is the inverse of the golden ratio. Then we follow the Fibonacci pattern working downwards, checking each time the ratio between two succeeding terms to be near golden ratio value, if not, then we take few actions and repeat the Fibonacci pattern working downwards. We continue with this process until we reach some prescribed lower bound, a small term, thereafter we can efficiently find doubling-free short addition chain. We join this short addition chain to the previous chain to complete the overall chain. Experimentally, we found that a 160 bit Fibonacci integer has minimal chain length of 231, whereas GRASC method gives an average chain of length 258 for an arbitrary integer of 160 bit. In fact, GRASC methods gives minimal chain for Fibonacci numbers or

equivalently if (3.2) holds but we do not guarantee for the case of non Fibonacci arbitrary integers to be minimal. We believe that GRASC method gives moderately short addition-subtraction chain since it utilizes mostly of the Fibonacci pattern. The following describes the GRASC method in detail.

We consider making chain starting from the last term, which is the input k. Let u_i denote the reverse of v_i, that is, $u_i = v_{\ell-i}$. To maintain (3.2), we let

$$\begin{aligned} u_0 =& k\,, \\ u_1 =& [u_0 \times \phi^{-1}]\,, \\ u_i =& u_{i-2} - u_{i-1} \qquad \text{for} \qquad i = 2, 3, \ldots \end{aligned} \tag{3.3}$$

If continued with the procedure (3.3), we will not be able to achieve the best result, since u_i will exponentially deviate from $(u_{i-1} \times \phi^{-1})$ as i increases. In order to overcome this problem, we introduce the parameter MAXIMALGAP such that the above procedure (3.3) terminates whenever

$$|u_i - (u_{i-1} \times \phi^{-1})| > \text{ MAXIMALGAP or } u_i \leqslant \frac{u_{i-1}}{2}\,.$$

In such case, we define, new u_i to be the nearest integer of $(u_{i-1} \times \phi^{-1})$. We resume the procedure (3.3) with u_{i-1} and new u_i as the initial terms. Note that it is necessary to include old u_i in the chain between u_{i-1} and new u_i. As a consequence, we have a gap, $g_j = |\text{old } u_i - \text{new } u_i|$ which we include in the storage. Also note that a subtraction is involve, whenever old $u_i <$ new u_i. We introduce another parameter, LOWERBOUND, to cease the procedure (3.3) when $u_i \leq$ LOWERBOUND. Note that the storage initially consists of $1, 2$, and 3. Later we have included all the g_j's in the storage. Once the execution of procedure (3.3) is ceased, we include the last two u_i's of the chain in the storage. Thus, using the storage, we randomly find a short addition chain by avoiding the use of doubling, except for numeral 2. Finally, we join this chain to the third last u_i of the previous chain resulting in a moderately short addition-subtraction chain for the given input k. Note that the storage capacity is dependent on the experimentally selected values for the two parameters.

Algorithm 1 Golden Ratio Addition-Subtraction Chain method (GRASC method)

Input: An integer k, MAXIMALGAP and LOWERBOUND.

Output: Short addition-subtraction chain for k.

1. $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
2. $u_0 \leftarrow k$
3. $u_1 \leftarrow [k \times \phi^{-1}]$
4. $u_2 \leftarrow u_0 - u_1$
5. $v = \{u_0, u_1, u_2\}$
6. $S = \{1, 2, 3\}$
7. $i \leftarrow 2$
8. $j \leftarrow 1$
9. **while** $u_i >$ LOWERBOUND **do**
10. **if** $|u_i - (u_{i-1} \times \phi^{-1})| >$ MAXIMALGAP or $u_i \leqslant \frac{u_{i-1}}{2}$ **then**
11. $i \leftarrow i + 1$
12. $u_i \leftarrow [u_{i-2} \times \phi^{-1}]$
13. $v \leftarrow v \cup \{u_i\}$
14. $u_{i+1} \leftarrow u_{i-2} - u_i$
15. $v \leftarrow v \cup \{u_{i+1}\}$
16. $g_j \leftarrow |u_i - u_{i+1}|$
17. $S \leftarrow S \cup \{g_j\}$
18. $j \leftarrow j + 1$
19. $i \leftarrow i + 1$
20. **else**
21. $i \leftarrow i + 1$
22. $u_i \leftarrow u_{i-2} - u_{i-1}$
23. $v \leftarrow v \cup \{u_i\}$
24. $S \leftarrow S \cup \{u_i, u_{i-1}\}$
25. $w \leftarrow$ a short addition chain including all terms from S
26. **return** $w \cup v$

Note that in step 13 of Algorithm 1, if $u_i < u_{i+1}$ then g_j's will involve subtraction during exponentiation.

Example 1. Evaluate Algorithm 1 for the inputs $k = 131456$, LOWERBOUND $= 10$ and MAXIMALGAP $= 6$.

We begin by letting

$$\begin{aligned} u_0 &= k = 131456\,, \\ u_1 &= [u_0 \times \phi^{-1}] = 81244\,, \\ u_2 &= u_0 - u_1 = 50212\,, \end{aligned}$$

$$\begin{aligned}
u_3 &= u_1 - u_2 = 31032\,,\\
u_4 &= u_2 - u_3 = 19180\,,\\
u_5 &= u_3 - u_4 = 11852\,,\\
u_6 &= u_4 - u_5 = 7328\,,\\
u_7 &= u_5 - u_6 = 4524\,,\\
u_8 &= u_6 - u_7 = 2804\,.
\end{aligned}$$

Since u_8 exceeds the MAXIMALGAP, that is $|2804 - (4524 \times \phi^{-1})| > 6$, we let

$$u_9 = [u_7 \times \phi^{-1}] = 2796\,.$$

There is a gap, $g_1 = |2804 - 2796| = 8$, which we include in the storage. Let

$$\begin{aligned}
u_{10} &= u_7 - u_9 = 1728\,,\\
u_{11} &= u_9 - u_{10} = 1068\,,\\
u_{12} &= u_{10} - u_{11} = 660\,,\\
u_{13} &= u_{11} - u_{12} = 408\,,\\
u_{14} &= u_{12} - u_{13} = 252\,,\\
u_{15} &= u_{13} - u_{14} = 156\,,\\
u_{16} &= u_{14} - u_{15} = 96\,,\\
u_{17} &= u_{15} - u_{16} = 60\,,\\
u_{18} &= u_{16} - u_{17} = 36\,,\\
u_{19} &= u_{17} - u_{18} = 24\,,\\
u_{20} &= u_{18} - u_{19} = 12\,.
\end{aligned}$$

Since $u_{20} \leq u_{19}/2$, we let

$$u_{21} = [u_{19} \times \phi^{-1}] = 15.$$

There is a gap, $g_2 = |12 - 15| = 3$, which we include in the storage. Let

$$u_{22} = u_{19} - u_{21} = 9\,.$$

We stop the above continuous procedure at u_{21}, since u_{22} transcends the given LOWERBOUND$= 10$. Now, we consider the storage which consists of pre-numbers $1, 2, 3$ and additional gap numbers $g_1 = 8$ and $g_2 = 3$. Further, we include $u_{21} = 15$ and $u_{22} = 9$ in the storage. Hence we have

$$\{1, 2, 3, 8, 3, 15, 9\}$$

We exclude the repeated numbers and rearrange it as

$$\{1, 2, 3, 8, 9, 15\}$$

We search for a moderately short addition chain including all numbers from the storage. We already have $1\,,2$ and $3\,$. Further, we insert 5, so that $3+5 \rightarrow 8\,$. It follows that $1+8 \rightarrow 9\,, 5+9 \rightarrow 14$ and $1+14 \rightarrow 15\,$. Hence, this completes our chain utilizing all the storage elements. Thus, we attain the following doubling-free short addition chain.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 9 \rightarrow 14 \rightarrow 15.$$

Finally, we join this chain to the previous chain at u_{20}, resulting in a complete chain for $k = 131456$ with length 28.

Next, we utilize the above chain to compute exponent $k = 131456$ starting from numeral 1. we denote v as the addition-subtraction chain, where $v_{\ell-i} = u_i$. It follows that

$$\begin{aligned}
v_0 = \quad & v_{28-28} = \quad u_{28} \quad = \quad 1\,, \\
v_1 = \quad & v_{28-27} = \quad u_{27} \quad = \quad v_0 + v_0 = 2\,, \\
v_2 = \quad & v_{28-26} = \quad u_{26} \quad = \quad v_0 + v_1 = 3\,, \\
v_3 = \quad & v_{28-25} = \quad u_{25} \quad = \quad v_1 + v_2 = 5\,, \\
v_4 = \quad & v_{28-24} = \quad u_{24} \quad = \quad v_2 + v_3 = 8\,, \\
v_5 = \quad & v_{28-23} = \quad u_{23} \quad = \quad v_0 + v_4 = 9\,, \\
v_6 = \quad & v_{28-22} = \quad u_{22} \quad = \quad v_3 + v_5 = 14\,, \\
v_7 = \quad & v_{28-21} = \quad u_{21} \quad = \quad v_0 + v_6 = 15\,, \\
v_8 = \quad & v_{28-20} = \quad u_{20} \quad = \quad -v_2 + v_7 = 12\,, \\
v_9 = \quad & v_{28-19} = \quad u_{19} \quad = \quad v_5 + v_7 = 24\,, \\
v_{10} = \quad & v_{28-18} = \quad u_{18} \quad = \quad v_8 + v_9 = 36\,, \\
v_{11} = \quad & v_{28-17} = \quad u_{17} \quad = \quad v_9 + v_{10} = 60\,, \\
v_{12} = \quad & v_{28-16} = \quad u_{16} \quad = \quad v_{10} + v_{11} = 96\,, \\
v_{13} = \quad & v_{28-15} = \quad u_{15} \quad = \quad v_{11} + v_{12} = 156\,, \\
v_{14} = \quad & v_{28-14} = \quad u_{14} \quad = \quad v_{12} + v_{13} = 252\,, \\
v_{15} = \quad & v_{28-13} = \quad u_{13} \quad = \quad v_{13} + v_{14} = 408\,, \\
v_{16} = \quad & v_{28-12} = \quad u_{12} \quad = \quad v_{14} + v_{15} = 660\,, \\
v_{17} = \quad & v_{28-11} = \quad u_{11} \quad = \quad v_{15} + v_{16} = 1068\,, \\
v_{18} = \quad & v_{28-10} = \quad u_{10} \quad = \quad v_{16} + v_{17} = 1728\,, \\
v_{19} = \quad & v_{28-9} = \quad u_9 \quad = \quad v_{17} + v_{18} = 2796\,, \\
v_{20} = \quad & v_{28-8} = \quad u_8 \quad = \quad v_4 + v_{19} = 2804\,, \\
v_{21} = \quad & v_{28-7} = \quad u_7 \quad = \quad v_{18} + v_{19} = 4524\,,
\end{aligned}$$

$$
\begin{aligned}
v_{22} = \quad v_{28-6} = \quad u_6 \quad &= \quad v_{20} + v_{21} = 7328\,, \\
v_{23} = \quad v_{28-5} = \quad u_5 \quad &= \quad v_{21} + v_{22} = 11852\,, \\
v_{24} = \quad v_{28-4} = \quad u_4 \quad &= \quad v_{22} + v_{23} = 19180\,, \\
v_{25} = \quad v_{28-3} = \quad u_3 \quad &= \quad v_{23} + v_{24} = 31032\,, \\
v_{26} = \quad v_{28-2} = \quad u_2 \quad &= \quad v_{24} + v_{25} = 50212\,, \\
v_{27} = \quad v_{28-1} = \quad u_1 \quad &= \quad v_{25} + v_{26} = 81244\,, \\
v_{28} = \quad v_{28-0} = \quad u_0 \quad &= \quad v_{26} + v_{27} = 131456\,.
\end{aligned}
$$

4 Experimental Results and Comparisons

In this section, we discuss our results and make comparisons with other doubling-free methods in the literature [11].

We carried out an experiment to analyze the GRASC method using python programming language on a 1.66 GHz Intel Core Duo processor. We randomly selected 10000 integers k of 160 bits and set the searching ranges of the parameters of LOWERBOUND to be between 5 to 23 and the MAXIMALGAP to be between 5 to 15. The experimental results in Table 4.1 shows that it took 209 trials to obtain chains of lengths between 253 to 261 and on an average it took about 3 seconds to find each chain. Whereas according to Meloni [11], in the case of EAC method, a 160-bit integer k will require testing of more than 45000 g's to find a chain of length 270, where g is randomly selected in the range $1 \leq g \leq k$. Hence the best case for a EAC was found to be of length 320.

GRASC length (ℓ)	# inputs k
GRASC-261	15
GRASC-260	389
GRASC-259	2165
GRASC-258	3610
GRASC-257	2555
GRASC-256	1003
GRASC-255	219
GRASC-254	37
GRASC-253	7

Table 4.1: The distribution of chains based on GRASC method for 160 bit integers k.

Our experimental result on Table 4.1 shows that chains of length between 257 to 259 could be found most efficiently using GRASC method. The best case was found to be chain of length 258. Note that GRASC method includes 26 points (worst case) in the storage, see appendix for details.

The data on Table 4.2 shows that GRASC method has attained $28\%, 20\%, 12\%$ and

Methods	Chain length
Fibonacci-and-add [11]	358
Signed Fib-and-add [11]	322
Window Fib-and-add [11]	292
EAC [11]	320
GRAC	258

Table 4.2: The average length of doubling-free addition chain for 160 bit integers

19%, in reduced average chain length compared to Fibonacci-and-add, Signed Fib-and-add, Window Fib-and-add and EAC methods respectively.

5 Conclusion

In this paper we have proposed a new strategy to find an efficient doubling-free short addition-subtraction chain by utilizing a precise golden ratio. The empirical data shows that GRASC method has attained 12% to 28% reduction in the average chain length compared to other doubling-free addition chain methods. Further work may include finding chains of much shorter lengths. One may consider giving formal algorithms for GRASC method to suit implementations in ECC. Also, reducing the storage content can make GRASC method more applicable to small memory constraint devices in ECC. The GRASC method has a unique way of creating chain, hence its real diligence is yet to be discovered.

References

[1] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, CRC Press, 2005.

[2] J. Bos and M. Coster, *Addition chain heuristics*, Advances in Cryptology-CRYPTO'89, volume 435 of Lecture Notes in Computing Science, Springer-Verlag, 1989, 400–407.

[3] A. Byrne, N. Meloni, F. Crowe, W. P. Marnane, A. Tisserand, and E. M. Popovici, *SPA Reisitant Elliptic Curve Cryptosystem Using Addition Chains*, International Conference on Information Technology-ITNG'07, 2007, 995–1000.

[4] D. Knuth, *Fundamental Algorithms*, The Art of Computer Programming, volume 1, Addision-Wesley, 1981.

[5] D. Knuth, *Seminumerical Algorithm (arithmetic)*, The Art of Computer Programming, volume 2, Addision-Wesley, 1981.

[6] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation* **48** (1987), 203–209.

[7] P. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*, volume 1666 of Lecture Notes in Computing Science, Springer-Verlag, 1999, 388–397.
[8] D. C. Lou and C. C. Chang, An adaptive exponentiation method, *The Journal of Systems and Software* **42** (1998), 59–69.
[9] A. J. Menezes, P. C. vanOorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
[10] V. S. Miller, *Uses of elliptic curves in cryptography*, In: Advances in Cryptology-CRYPTO'85, volume 218 of Lecture Notes in Computing Science, H. C. Williams, editor, Springer-Verglag, 1986, 417–428.
[11] M. Nicolas, New Point Addition Formulae for ECC Applications, *Arithmetic of Finite Fields*, volume 4547 of Lecture Notes in Computing Science, Springer-Verlag, 2007, 189–201.
[12] N. Vorobiev, *Fibonacci Numbers*, Birkhuser Verlag, 2002.
[13] Y. Yacobi, *Exponentiating faster with addition chains*, Advances in Cryptology-EUROCRYPT'90, volume 473 of Lecture Notes in Computing Science, Springer-Verglag, 1990, 222–229.

Appendix

Here, we give an estimation of the storage capacity. That is the number of g_j's required in GRAC method for a 160 bit integer. Note that the number of g_j's are same as the number of new u_i being computed.

We have $\left\{u_0 = k\,, u_1 = [\frac{k}{\phi}]\,, u_2 = u_0 - u_1\,, u_3 = u_1 - u_2\,, \ldots\right\}$.

Note that Fibonacci sequence gives the optimum result, therefore we have a desire to maintain such pattern for the GRAC method. Thus, we will check at every step to maintain the property of a Fibonacci sequence given by equation (2.1).

For $i = 1$

$$\left|u_1 - \frac{k}{\phi}\right| \leq \frac{1}{2}\,.$$

For $i = 2$

$$\begin{aligned}\left|u_2 - \frac{k}{\phi^2}\right| &= \left|u_0 - u_1 - \frac{k}{\phi^2}\right| \\ &= \left|k - u_1 - \frac{k}{\phi^2}\right| \\ &= \left|k - \frac{k}{\phi} - \frac{k}{\phi^2} + \frac{k}{\phi} - u_1\right| \\ &= \left|\frac{k}{\phi^2}(\phi^2 - \phi - 1) + \frac{k}{\phi} - u_1\right|\,, \text{ using Theorem 2.4, we get}\end{aligned}$$

$$= \left| \frac{k}{\phi} - u_1 \right|$$
$$\leq \frac{1}{2}\,.$$

For $i = 3$

$$\left| u_3 - \frac{k}{\phi^3} \right| = \left| u_1 - u_2 - \frac{k}{\phi^3} \right|$$
$$= \left| u_1 - \frac{k}{\phi} + \frac{k}{\phi^2} - u_2 - \frac{k}{\phi^3} + \frac{k}{\phi} - \frac{k}{\phi^2} \right|$$
$$= \left| (u_1 - \frac{k}{\phi}) + (\frac{k}{\phi^2} - u_2) - \frac{k}{\phi^3}(1 - \phi^2 + \phi) \right| \text{, using Theorem 2.4, we get}$$
$$= \left| u_1 - \frac{k}{\phi} \right| + \left| \frac{k}{\phi^2} - u_2 \right|$$
$$\leq \frac{1}{2} + \frac{1}{2} = 1\,.$$

Hence, in general we have

$$\left| u_i - \frac{k}{\phi^i} \right| \leq \left| u_{i-2} - \frac{k}{\phi^{i-2}} \right| + \left| u_{i-1} - \frac{k}{\phi^{i-1}} \right| .$$

Assuming MAXIMALGAP to be 11, we will check the number of terms exist before MAXIMALGAP is exceeded. It follows that

$$\begin{aligned}
i = 4: \quad & \tfrac{1}{2} + 1 && = \tfrac{3}{2}\,, \\
i = 5: \quad & 1 + \tfrac{3}{2} && = \tfrac{5}{2}\,, \\
i = 6: \quad & \tfrac{3}{2} + \tfrac{5}{2} && = 4\,, \\
i = 7: \quad & \tfrac{5}{2} + 4 && = \tfrac{13}{2}\,, \\
i = 8: \quad & 4 + \tfrac{13}{2} && = \tfrac{21}{2}\,, \\
i = 9: \quad & \tfrac{13}{2} + \tfrac{21}{2} && = 17\,.
\end{aligned}$$

The MAXIMALGAP is exceeded at $i = 9$, therefore at $i = 10$, a new u_i is computed. Hence, we could infer that in worst case, a new u_i is computed once in every 10th term for GRAC method. Therefore, the maximum number of points in the storage for an average chain of length 258 will be 26 .