Journal of Computer Sciences

Literature Reviews

A Review on Heuristics for Addition Chain Problem: Towards Efficient Public Key Cryptosystems

[1]Adamu Muhammad Noma, [1]Abdullah Muhammed,
[2]Mohamad Afendee Mohamed and [1]Zuriati Ahmad Zulkarnain

[1]Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, Serdang, 43000, Selangor, Malaysia
[2]Faculty of Informatics and Computing, Universiti Sultan ZainalAbidin, Besut, 22200, Terengganu, Malaysia

Article history
Received: 03-12-2016
Revised: 03-03-2017
Accepted: 27-05-2017

Corresponding Author:
Abdullah Muhammad
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia,
Serdang, 43000, Selangor,
Malaysia
Email: adamnoma@yahoo.com

Abstract: Field exponentiation and scalar multiplication are the pillars of and the most computationally expensive operations in the public key cryptosystems. Optimizing the operation is the key to the efficiency of the systems. Analogous to the optimization is solving addition chain problem. In this study, we survey from the onset of the addition chain problem to the state-of-the-art heuristics for optimizing it, with the view to identifying fundamental issues that when addressed renders the heuristics most optimal mean of minimizing the two operations in various public key cryptosystems. Thus, our emphasis is specifically on the heuristics: Their various constraints and implementations efficiencies. We present possible ways forwards toward the optimal solution for the addition chain problem that can be efficiently applied for optimal implementation of the public key cryptosystems.

Keywords: Heuristics, Addition Chain Problem, Modular Exponentiation, Scalar Multiplication, Public Key Cryptosystem

Introduction

Public Key Cryptosystems (PKC) are substitution-based encryption systems for which security depends mostly on Integer Factorization Problem (IFP), Discrete Logarithm Problem (DLP) or Elliptic Curve Discrete Logarithm Problem (ECDLP). Every entity has two distinct keying materials: One kept secret while the other in public domain. Any party wishing to communicate with other accesses the communicating party's authentic public key in the domain. Two parties employing Secret Key Cryptosystem (SKC) utilizes the capability of the PKC to exchange the keying materials securely in public channel, provided they authenticate one another. An entity communicating with many others has access to their respective public key in the public domain, as at when needed. Only one entity is in possession of a given private key and hence any message it signed can be used to trail back to it: Which serves as both non-repudiation and authentication mechanisms. Therefore, PKCs offer broader functionalities than SKC by facilitating key exchange and an elegant digital signature mechanism that also facilitates authentication and non-repudiation. Implementation-wise, PKCs depend on extensive field (modular) exponentiation or scalar multiplication.

In the given Field F_p, exponentiation x^e mod p and point scalar multiplication eP are the basis and most resource consuming operations in the PKC that render their utilization feasible only for special purpose: Likes key exchange and digital signature. For the system to be computationally secured, the integer e size should be around 160 to 2048-bit. While the PKC security depends on the size of e, the operations are highly involved with large-sized e. Two basic approaches to optimize these operations are design of efficient multiplication hardware or utilizing some optimization algorithm. From the latter point, the exponentiation/scalar multiplication involves chains of repeated multiplications/additions, thus has an additive properties with respect to the e. Fortunately thus, the operation can be abstractly approached as an Addition Chain Problem (ACP) with a view to finding the corresponding shortest (or optimal) Addition Chain (AC). The AC for the e represents the sequence of multiplications or additions required in the respective operation. The ACP is generally regarded as NP-hard.

This paper surveys ACP as applies to optimizing modular exponentiations and scalar multiplications in various PKC. Primarily, we concentrate on state-of-the-art heuristic works on the problem and their implications

on the PKC. We discuss only theoretical findings that have a significant impact on the heuristic. The paper is organized thus: We proceed in this section with a brief description of PKC basics and the roles of exponentiations/scalar multiplication in each. The section is rounded with a technical description of modular exponentiation and scalar multiplication. In the next section, we focused on the heuristics and metaheuristics for the ACP. Section 3 concludes the survey and present recommendations based on.

Public Key Cryptosystems (PKC)

In PKC, an entity A is associated with pair of keying materials k_{pr} and k_{pu}. While k_{pr} is held secret as its private key, k_{pu} is made public for any entity wishing to communicate with the A. The needed conditions are that k_{pu} is authentic that of A and k_{pr} is computationally intractable with the knowledge of only k_{pu}. Basic PKCs include Diffie-Hallman key exchange system (Diffie and Hellman, 1976a), RSA (Rivest *et al*., 1978), ElGamal-based system (ElGamal, 1985) and Elliptic Curve Cryptosystem (ECC) (Koblitz, 1987; Miller, 1986).

Diffie and Hallman (1976b) pioneered the solution to key distribution problem in SKC, called the Diffie-Hallman key exchange scheme. Basically for two parties A an B to exchange a secret key $k_{A,B}$:

- Large primes p,q/(p-1) *and* q's generator a is agreed on (may be supplied by a trusted third party)
- A and B choose a uniformly random and secret numbers x_A, $x_B < q$ respectively
- A evaluates $y_A = a^{x_A} mod\ p$ and sends to B, similarly B sends back $y_B = a^{x_B} mod\ p$
- Both then compute $k_{A,B} = (y_B)^{x_A} mod\ p = (y_A)^{x_B} mod\ p$, at their respective ends

An adversary is left with the problem of evaluating either $x_A = d\log_{a,p}(y_B)$ or $x_B = d\log_{a,p}(y_A)$ called Diffie-Hallman Problem (DHP), that is a DLP. Currently the fastest DLP algorithms are Pollard rho (Pollard, 1975) and number field sieving (Joux and Lercier, 2003). Approximately 2048-bit p and $q \approx p/2$ are considered computationally as secured as SKC such as AES (Adrian *et al*., 2015; Dahshan *et al*., 2015; Robshaw and Yin, 1998).

Note of the field exponentiations involving very large exponents, each of an expected order of $\approx 2^{1000}$ that are involved in the algorithms: $a^{X_A}, a^{X_B}, Y_A^{X_B}$ and $Y_B^{X_A}$.

ElGamal (1985) proposed full featured DLP-based cryptosystem, which has been the basis for such DSS and S/MIME email security (Stallings, 2011). The PKC is in principle similar to Diffie-Hallman algorithm except that A makes available the set $\{a, q, p, y_A = a^{x_A}\}$ in public domain as its required public keying material. An entity B wishing to communicate with the A partitions its message M into $0 \le M_i < p$; And for each M_i, it selects a random exponent $1 < x_{Bi} < q$, then generates and sends a key/message pair (C_1,C_2): Where $C_1 = y_{Bi} = a^{x_{Bi}}$ and $C_2 = y_A^{x_{Bi}} \times M_i$. When the pair reaches A the message is recovered as $M_i = (C_1^{x_A})^{-1} C_2$.

RSA is developed by Rivest *at al*. (1978), The algorithm (Koc, 1994) consists of pair of public key (e,N) ($N = p \times q$: Where p, q are large primes of fairly the same bit-length and that $1 < e < \phi = (p\text{-}1)(q\text{-}1)$, $\gcd(e,\phi) = 1$) and a private key (d,N) $1 < d < \phi$, $e \times d \equiv 1\ mod\ \phi$). A message M is RSA-encrypted as $Y = M^e\ mod\ N$ and decrypted as $Y^d \bmod N = (M^e)^d\ mod\ N = M$.

Note that e and N are open to public while d, p and q are kept secret. The problem of determining d given the e and N is that of computing $e^{-1}\ mod\ \phi$. Thus N has to be factored into q and p, hence an IFP (May, 2004). As p and q grows large the factoring is assumed to be computationally infeasible. Given the present computing power, its assumed that 1024-bit p, q and therefore about 2048-bit N are sufficient (NIST, 2013).

The field exponentiations in the RSA has been the most expensive operation that renders it less attractive for general purpose data encryption, apart from the less rigorous key exchange and digital signature. One means of reducing this operation is in utilizing well-known e = 3,17 and 2^{16+1} as public key. However, the first 2 are vulnerable to attacks using Chinese Remainder Theorem (CRT) (Boneh, 1999). Furthermore, d is such that $d > n^{1/4}$ for the system to be secured (Wiener, 1990; Boneh and Durfee, 2000). However, employing CRT along with Fermat theorems to factor d reduces the exponentiation length by factor of 3 to 4 (Stallings, 2011).

Elliptic Curve Cryptosystem (ECC) (Miller, 1985; Koblitz, 1987) is yet an emerging general purpose PKC, requiring less key size while giving similar functionality and level of security. For a prime p>3, an elliptic curve over the field F_p is a set of solutions (x,y) for an equation of the form:

$$E: Y^2 = X^3 + AX + B \tag{1}$$

such that $A,B \in F_p$ and $4A^3 + 27B^2 \neq 0$.

The set of points on E with coordinates in F_p is the set $E(F_p) = \{(x,y): x,y \in F_p, y^2 = x^3 + Ax + B\} \cup \{0\}$. O is a point at infinity and represents the identity in E.

ECC over binary extension field F_{2^m} also exists and is applicable to cryptographic systems. But, for the general discussion, ECC over F_p suffices. Refer to (Robshaw and Yin, 1998; Koblitz *et al*., 2000; Hankerson *et al*., 2004) for details on the ECC.

The set of points satisfying the Equation 1 forms an additive group under special addition, hence the basis for its use in the cryptography: That is for any points P,Q and R in E, $P + Q = Q + P$;$P + O = P$;$P + (-P) = O$ where -P: $(x, y) = (x, y)$ and $(P + Q) + R = P + (Q + R)$.

For any two points $P_1(x_1,y_1)$, $P_2(x_2,y_2)$ where $P_1 \neq -P_2$ and $P_1, P_2 \neq O$: $P_1(x_1,y_1) + P_2(x_2,y_2) = P_3(x_3,y_3)$. Such that $x_3 = \lambda^2 – x_1 – x_2, y_3 = \lambda(x_1 - x_3) – y_1$ and:

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} if\ P_1 \neq P_2 \\ \frac{3x_1^2 + A}{2y_1} if\ P_1 = P_2 \end{cases} \quad (2)$$

Note that the operation, even as termed point addition, actually involves some additions, subtractions multiplication and inversions in the field: Thus, referred to as Elliptic Curve Addition (ECADD) and Elliptic Curve Doubling (ECDBL) when $P_1 \neq P_2$ and $P_1 = P_2$ respectively.

Furthermore, operationally-wise cryptosystem based on elliptic curve involves repeatedly adding point P in the form of $P + P + P + \cdots + P = eP$, that is e times, called scalar multiplication. For example ECC version of Diffie-Hallman key exchange by two parties A and B involved agreeing on a unique E: $y^2 = x^3 + ax + b$, the prime p over which E is defined and a point P on E. Thus (E,p,P) are public. Party A privately selects a scalar e_A, evaluates e_AP and pass it to B. Similarly party B passes e_BP to A. Both parties evaluate e_Ae_BP at their ends and the x coordinate serves as their common secret key.

The security of the ECC also depends on the size of e. It is determined by the difficulty of evaluating e given P and $Q = eP$, which is ECDLP. The fastest known algorithm to solve ECDLP in $E(F_p)$ is estimated to take $\sqrt{p}$ steps (Hoffstein *et al*., 2014): Thus it is fully exponential algorithm. Consequently, it is assumed that around 160-bit ECC provides equal security to 1024-bit RSA (Maletsky, 2015). Therefore, the ECC provides similar functionality but with superior advantages in term of less key size: Translating to bandwidth savings and faster implementation.

Note, scalar multiplication is isomorphic to AC. Thus a number of addition/doublings in it is minimal with optimal AC for the scalar e. Another good aspect of optimizing its operations is that an inversion of $P(x_1, y_1)$ is virtually cost-free as compared to multiplicative inversion in other PKC.

Field Exponentiation versus Scalar Multiplications as Addition Chain Problem

Exponentiation, evaluation of power (Knuth, 1998), involves computing $y = x^e$ given x and positive integer e. A basic feasible method for the computation is the square-and-multiply algorithm (Knuth, 1998): e is expressed in binary form and x is held as the partial result; Beginning from and skipping the Most Significant Bit (MSB), the result is squared par bit and if the bit is 1 it is again multiplied by x. For example, x^{13}: $13_{10} = 1101_2$ and therefore $x^{13} = (((x^2x)^2)^2)x = x\text{-}x^2\text{-}x^3\text{-}x^6\text{-}x^{12}\text{-}x^{13}$ On the other hand, evaluating eP involves series of addition of the P e-times. In a similar manner to x^e, this can be achieved by series of addition and/or doublings. E.g., $13P = 2(2(2P + P)) + P = P\text{-}2P\text{-}3P\text{-}6P\text{-}12P\text{-}13P$, requiring equal number of additions as that of the multiplications in the x^{13}. Additionally, the numbers of operations correspond respectively to that of terms in the sequences x/P, $x^2/2P,\ldots x^e/eP$: The respective powers/coefficients form a sequence of AC for e. Thus finding shortest AC corresponds to that of the number of operation required. Therefore, henceforth, we disambiguate problem of minimizing both modular exponentiation and scalar multiplication as ACP.

Definition 1

Given an integer e, a sequence A: $a_0 = 1$, $a_1 = 2$, $a_3,\ldots, a_r = e$, is said to be an Addition Chain (AC) for e if for all $i \geq 1$, $a_i = a_j + a_k$, $i>j\geq k$. The length of the AC is denoted as r.

The shortest r for which there exists an AC of e is denoted by $l(e)$.

Other variants of AC are star-chain (Brauer, 1939), Addition-Subtraction Chain (ASC) (Morain and Olivos, 1990) and Addition Sequence (AS) (Yao, 1976).

Definition 2

An a_i, $i>0$, of an AC A is said to be a star if $a_i = a_{i-1} + a_k$, $k<i$; an AC A is said to be a star-chain if all $a_i \in A$, $i>0$, are star; the shortest length of star-chain for an integer e is denoted as $l^*(e)$.

Definition 3

An Addition Sequence (AS) of an integer vector $E = \{e_i: 1 \leq i \leq s\}$ such that $1<e_1<e_2<\cdots<e_s$ is a sequence $1,2,a_2,\ldots,a_L = e_s$ satisfying the properties of an AC in Definition 1 and that all e_i, $1\leq i\leq s$ are in the sequence.

Definition 4

The length of e (denoted as) $n(e)$ is define as the minimum number of bits to represent e in binary form as $e = (e_{n-1}e_{n-2}\cdots e_0)_2$, while $H(e)$ is the number of non-zero bits (weight) in the binary form. We use n to indicate an arbitrary n-bit integer.

The problem of finding shortest AC exists and persists for over hundred years (Dellac, 1984). Scholz (1937) raised the fundamental question of what is the least number of multiplications to evaluate x^e given x and e. Knuth (1998) once described ACP as a mathematical problem of which no conjecture is safe. The problem

received ample efforts from both theoretical studies (Brauer, 1939; Downey *et al*., 1981; Mignotte, 2011; Thurber, 1973, 1993; Yao, 1976; Knuth, 1998) and algorithmic perspective in search for the optimal AC (Bahig, 2006; 2011; Bos and Coster, 1990; Clift, 2011; Thurber, 1999), to mention but a few. ACP is an NP-hard, while its generic ASP is proven as NP-complete problem (Downey *et al*., 1981). Therefore, heuristic (Bos and Coster, 1990; Gelgi and Onus, 2006; Koc, 1995; Lee *et al*., 2006; Park *et al*., 1999) and metaheuristic (Cruz-Cortés *et al*., 2005; 2008; Jose-Garcia *et al*., 2011; León-Javier *et al*., 2009; Osorio-Hernández *et al*., 2009; Dominguez-Isidro and Mezura-Montes, 2011; Dominguez-Isidro *et al*., 2015) have become alternative approaches in searching for near optimal solution for the ACP.

We are motivated by the potentials of deterministic heuristics and metaheuristic in searching for near optimal solution for the ACP, especially as applied to various PKC. However, we observed various fundamental issues that are needed to be considered/addressed, at first in selecting particular heuristic/metaheuristic for the problem; secondly for the appropriate method to be able to search for a nearest optimal AC and for the result to be most suitable for PKCs; and for the search process by the heuristic to be as efficient as possible. While ACP heuristics/metaheuristic for PKC is central to our survey, we consider both the time and space efficiencies in solving the problem: This is particularly in considerations of the varying platforms, with varying constraints, of deploying PKC. Therefore, in the remaining of the survey, we review those various heuristics for the ACP with emphasis on their implementation efficiencies; effectiveness in finding the (near) optimal AC; as well as their implication constraints in the context of the PKC, with respect to both speed and memory requirements. We study only those theoretical studies and/or exhaustive methods for which results have direct implications for the formulation of the heuristics. We use the terms method and algorithm interchangeably.

Addition Chain Heuristic

Algorithms for finding (near) optimal AC can be classified as exhaustive or heuristics. The exhaustive approaches (Bahig, 2006; 2011; Clift, 2011; Thurber, 1999) normally end up generating optimal ACs for some large set of integers. It takes a few days to months of computer cluster operation to do so. Normally, the generated optimal ACs are for small integers far from the ones utilized in PKCs.

An AC heuristic algorithm *alg* generates AC for an integer e, $A_{alg}(e) = a_0,a_1,\ldots, a_r = e$with length $l_{alg}(e) = r$, such that $l_{alg}(e) \geq l(e)$. The objective of *alg* is to minimize $l_{alg}(e) \rightarrow l(e)$ and it is said to find an (shortest) optimal AC for e if $l_{alg}(e) = l(e)$. In addition to this objective, we consider the algorithm efficiency in finding the solution and amount memory required when applying the solution in various PKC.

Binary Method

In binary method, a binary form of e is processed in a sequence of doublings and an optional addition, depends upon the given digit value that is 1 or 0 respectively. For n-bit $e = e_{n-1}e_{n-2}\ldots e_0$, in binary form, the method generates an AC $A_{bin}(e)$ having length $l_{bin}(e)$ as presented in Algorithm 1.

Algorithm 1: Binary Method
Input $e = e_{n-1}e_{n-2}\ldots e_0$; **Output**: $A_{bin}(e) = \{a_0,a_1,\ldots,a_r = e\}$.
1: $A_{bin}(e) = \{a_0 = 1\}$, $i = 1$
2: for j: n-2 to 0 do
3: $A_{bin}(e)$: = $A_{bin}(e)\cup\{2a_{i-1}\}$ and $i = i +1$
4: if $e_j = 1$ then $A_{bin}(e)$:= $A_{bin}(e)\cup\{a_{i-1}+1\}$ and $i = i+1$
7: end for
8: return $A_{bin}(e)$

$$l_{bin}(e) = n(e) + H(e) - 2 \quad (3)$$

On the average, $l_{bin}(e) = \frac{3}{2}n(e)$.

Binary method is efficient in terms implementation and utilizes minimal memory resources. It can also be applied directly in PKC implementation, without the need to find the AC for the given exponent/scalar a priory. Algorithm 2 shows the procedure in the (field) exponentiation.

Algorithm 2: Binary Exponentiation
Input x, p, $e = e_{n-1}e_{n-2}\ldots e_0$; Output: $y = x^e \bmod p$.
1: y:= x
2: for j: = n-2 to 0 do
3: y:= $y \times y \bmod p$
4: if $e_j = 1$ then y:= $y \times x \bmod p$
7: end for
8: return y

Point scalar multiplication $Q = eP$ is similar to Algorithm 2 except that x is replaced by P, y by Q, squaring in step 3 by ECDBL and multiplication in step 4 by ECADD. Note, only 2-units of storage are needed for the base x and the (partial) result y.

However, the method is least effective in finding optimal AC. But Knuth (1998) proves that for e with $H(e)\leq 4$, $l_{bin}(e) = l(e)$, except the following categories of $e = 2^A +2^B +2^C +2^D$, $A > B > C > D$ having $H(e) = 4$:

- A-B = C-D (e.g.: $e = 15_{10} = 1111_2 = 2^3+2^2+2^1+2^0$)
- A-B = C-D +1 (e.g.: $e = 23_{10} = 10111_2 = 2^4+2^2+2^1+2^0$)

- *A-B* = 3, = *C-D* = 1 (e.g.: $e = 39_{10} = 100111_2 = 2^5+2^2+2^1+2^0$)
- *A-B* = 5, *B-C* = *C-D* = 1 (e.g.: $e = 135_{10} = 10000111_2 = 2^7+2^2+2^1+2^0$)

all having $l(e) = n(e) +1 = l_{bin}(e)$-1.

Considering its efficiency, the binary method may be a good candidate as an initial solution for other more effective heuristics/metaheuristic and that needless for search for optimality for the *e*'*s* with $H(e)<4$.

M-ary (2^k-ary) Method

m-ary or 2^k-ary method is an extension for the binary method. The method is traceable to Brauer (1939), but witnesses various enhancements (Knuth, 1998; Thurber, 1973; Koc, 1995; Park *et al*., 1999). An *n*-bit $e = e_{n-1}e_{n-2}\ldots e_0$ is padded (where necessary) with trail of 0s to the multiple of *k*. It is then partitioned into $w = \lceil n/k \rceil$ fixed *k*-bit words: m_i, $i = w$-1,…,0, m_{w-1} being the Most Significant Word (MSW). Thus, $0 \le m_i = (e_{ik+k-1}e_{ik+k-2}\ldots e_{ik}) \sum_{j=0}^{k-1} e_{ik+j} < 2^k$ and $e = \sum_{i=0}^{w-1} m_i 2^{ik}$. Initially the values $x, x^2, \ldots, x^{2^k-1}$, corresponding to all possible x^{m_i}, are pre-computed. The algorithm proceed by scanning the most significant *k* bits m_{w-1}, raising the corresponding $x^{m_{w-1}}$ to the power of 2^k as the partial result. This is followed by subsequent scanning of the remaining m_i, $i = w$-2, *w*-3,…,0 each time multiplying the partial result by x^{m_i} and raising it to the power of 2^k as: $x^{2m_{w-1}}, x^{4m_{w-1}}, \cdots, x^{2^k m_{w-1}}, x^{2^k m_{w-1}}.x^{m_{w-2}}, x^{2^k\left(2^k m_{w-1}+m_{w-2}\right)}, \cdots, x^{\left(2^{n-k} m_{w-1}+2^{n-2k} m_{w-2}+\ldots+2^k m_1\right)}.x^{m_0} = x^e$. That is beginning from $x^{m_{w-1}}$, *k*-time squaring are performed, followed by multiplying the partial result by the next x^{m_i} until the last x^{m_0} is multiplied by. Let $A_m(e)$ be AC generated by *m*-ary for *e*, the process, in the form of addition chain, is presented in Algorithm 3.

Algorithm 3: *m*-ary method.
Input *e*,*k*; Output: $A_m(e) = \{a_0, a_1, \ldots, a_r = e\}$.
1: $A_m(e) := \{1, 1+1 = 2, 2+1 = 3, \ldots, 2^k-1\}$
2: pad *e* with trailing 0s so that *k* divides *n*(*e*)
3: partition *e* into $w = n/k$ *k*-bit words m_i: $i = w$-1…0
4: $r := 2^k$-2; $a_r := m_{w-1} \in A_m(e)$
5: for $i = w$-2 down to 0 do
6: for $j = 0$ to *k*-1 do
7: $r := r + 1$
8: $A_m(e) := A_m(e) \cup \{a_r := 2a_{r-1}\}$
9: end for
10: if $m_i \neq 0$
11: $r := r + 1$
12: $A_m(e) := A_m(e) \cup \{a_r := a_{r-1} + m_i\}$
13: end if
14: end for
15: return $A_m(e)$.

From Algorithm 3, step 1 requires 2^k-2 additions to pre-compute all possible values m_i up to 2^k-1. In step 8, there are $(w$-1$)\ k = (\lceil n/k \rceil$-1$)k$ doubling. While in step 12 an addition is performed only when $m_i \neq 0$. Having *k*-bit m_i in which each bit is equally likely to be 1 or 0, $\Pr(m_i = 0) = \Pr(e_{ik+k-1} = 0 \times \ldots \times e_{ik} = 0) = \prod_{k-1}^{0}(1/2) = 2^{-k}$. Thus $\Pr(m_i \neq 0) = 1\text{-}2^{-k}$. Therefore, for the w-1m_i, an average of $(w\text{-}1)(1\text{-}2^{-k}) = (\lceil n/k \rceil\text{-}1)\ (1\text{-}2^{-k})$ additions are performed. Hence, on the average *m*-ary AC length for *n*-bit *e* using *k*-bit partitions $l_m(n,k)$ is:

$$l_m(n,k) = \left(2^k - 2\right) + \left(\left\lceil n/k \right\rceil - 1\right)k + \left(\left\lceil n/k \right\rceil - 1\right)\left(1 - 2^k\right) \quad (4)$$

An even shorter length is obtainable when the pre-computation is delayed until after the partitioning: The corresponding additions is reducible by 2^k-1-max(m_i): max(m_i) being m_i with the largest value. E.g., an *m*-ary optimal AC length for 62-bit *e* = 3606984084510991957_{10}, is found to be 83 using $k = 3$.

According to Knuth (1998), computing all $2^i q : q < 2^{k-i}, i > 0$ can be omitted in the pre-computing stage, thereby saving $(1+ n)$ mod *k* additions. Similarly, Thurber (1973) generalized that all even $m_i > 2$ can be computed when needed at the addition stage, by adding their respective odd halves before the last doubling. For example, given $e = 30_{10} \mapsto 11110_2 \mapsto 011\text{-}110$ (3, 6) and $k = 3$: Instead of 1-2-3-4-5-6 $\mapsto$ 3-6-12-24-30 (24+6) with length 9, the AC is generated as: 1-2-3 $\mapsto$ 3-6-12-15(12+3)-30 having length 7. Consequently, $e = 3606984084510991957_{10}$ have an AC-length 81 and Equation 4 reduces to:

$$l_m(n,k) = 2^{k-1} + \left(\left\lceil n/k \right\rceil - 1\right)k + \left(\left\lceil n/k \right\rceil - 1\right)\left(1 - 2^k\right) \quad (5)$$

From (5), the additions reduces with increase in *k*. However, the length of the pre-computation increases exponentially with the *k* size (thus henceforth we refer this as *k*-constraint). Thus, Koc (1995) estimated optimal values of *k* to be around 3 to 7 for $n \le 2048$*e*s.

m-ary can also be efficiently applied as binary method, but requires additional 2^{k-1} units of memory to hold the pre-computed values. Therefore, it is suitable for one-time and infrequent exponentiation/scalar multiplications. The choice of *k* is important to achieving both optimal computation and memory utilization.

Window-based Addition Chain Methods

Window-based methods (Bos and Coster, 1990; Thurber, 1999; Koc, 1995) are enhancements of *m*-ary

that form partitions m_i of arbitrary length of 0s. In the partitioning process, the leading zeros in a given k-bit Non-zero Window partition (NW), $m_i \neq 0$ are also carved out and merged with subsequent zeros encounter to form a Zero Window partition (ZW). The ZW takes any arbitrary length, until a non-zero bit is again encountered. In other word, only NW is restricted to an odd value with $n(NW) \leq k$. Thus, pre-computation of the odd values up to 2^k-1 costs $(2^k\text{-}2)/2\text{+}1 = 2^{k-1}$ additions. The number of NWs is also reduced on the average. The method is also referred to as Constant Length Non-zero Window (CLNW) (Park *et al*., 1999). The AC algorithm is presented in Algorithm 4.

Algorithm 4: CLNW
Input: e,k; Output: $A_w(e) = \{a_0,a_1,\ldots,a_r = e\}$.
1: $A_w := \{1,2,3,3\text{+}2 = 5, 5\text{+}2 = 7,\ldots, 2^k\text{-}1\}$
2: Partition e into m_i: $i = w\text{-}1, w\text{-}2,\ldots,0$ such that if $m_i \neq 0$ then $MSB(m_i) = LSB(m_i) = 1$ and $n(m_i) \leq k$
3: $r := 2^{k-1}$; $a_r := m_{w-1} \in A_w(e)$
4: for $i = w\text{-}2$ down to 0 do
5: for $j = 1$ to $n(m_i)$ do
6: $r := r + 1$
7: $A_w := \{A_w(e)\} \cup \{a_r := 2a_{r-1}\}$
8: end for
9: if $m_i \neq 0$
10: $r := r + 1$
11: $A_w(e) := \{A_w(e)\} \cup \{a_r := a_{r-1} + m_i\}$
12: end if
13: end for
14: return $A_w(e)$

$LSB(x)$ and $MSB(x)$ indicate the most and the least significant bit of x respectively. The equivalent right-to-left version of Algorithm 4 is structurally similar except that step 6-8 are preceded by 9-12, a_r is initialized as m_0 and all the loops are reversed.

The Variable Length Non-zero Window (VLNW) version further restructure NW having a fairly long trail of zeros within, e.g.: 1100001. Constructing NW is terminated upon encounter of a predetermined q consecutive zeros; Transition to ZW begins with the q zeros. The strategy minimizes $n(NW)$ and/or maximizes the weight; and the proportionate number of ZWs is also maximized. On the average, the number of NW is also reduced (Park *et al*., 1999). Both the CLNW and VLNW are referred to as Sliding Window Methods (SWM).

Empirically, an SWM AC length for an n-bit e utilizing k-bit NW is evaluated thus: Beginning from the MSW, let there be $m_s,m_{s-1},\ldots,m_1,s \leq w$ NWs in the partition of e; the pre-computation stage involves generating AC consisting of all odd numbers up to 2^k-1 at the cost of 2^{k-1} additions, in which only the unique NWs are needed. But in practice the largest NW in the given partition may be less than 2^k-1. Therefore, when the pre-computation is deferred until it is known, the cost reduces to $\left\lceil \frac{\max(m_i)}{2} \right\rceil$. Beginning with m_{w-1}, there are $n\text{-}n(m_{w-1})$ doublings and s-1 additions corresponding to the remaining NW. Thus, the AC length $l_{swm}(e,k)$ is given as:

$$l_{swm}(e,k) = \left\lceil \frac{\max(m_i)}{2} \right\rceil + n(e) - n(m_{w-1}) + s - 1 \quad (6)$$

Equation 6 is independent of partitioning parameters and is, in fact, generic for the family of window-based methods (Binary, m-ary and SWM).

We denote $l_{cw}(e,k)$ and $l_{vw}(e,k,q)$ as the length for an AC generated by CLNW $A_{cw}(e,k)$ and VLNW $A_{vw}(e,k,q)$ respectively. Proceeding with example for e = 3606984084510991957, exhausting all optimal windows parameters k = [3,7] and q = [1,4], we get $l_{cw}(e,3) = l_{vw}(e,3,2) = l_{vw}(e,4,2) = l_{vw}(e,4,3) = 77$.

According to Koc (1995) analysis, thought biased in respect $n(e)$ (Park *et al*., 1999), for n = [128,2048] CLNW and VLNW gain 7-3% and 8-5% fewer additions respectively over m-ary. The gains depend on the selective (k,q) parameter values, even as in general SWM share the same k-constraint.

Note on Windows-Based Addition Chain Methods

Window-based AC algorithms are state-of-the-art feasible methods for exponentiation/scalar multiplication: Consequent to their implementation efficiency and minimal memory requirement. However, none of the methods guarantees any (near) optimal AC. The resulting optimality is also tightened to the choice of optimal window parameters. We regard the methods as optimal approaches when an exponent/scalar e is not known and/or frequently changed in the course of running an application. However, in PKCs, an e is normally utilized multiple times and, at times, an application partially chooses random e. In this case, a general parameter for the n-bit class of e may not be suitable particularly for it. Therefore, a yet adaptive means of determining suitable parameters for the individual exponent is needed, for the method to be even more effective. Upon preliminary empirical analysis, we present in Table 1 the range of optimal windows parameters that result in the most optimal SWM AC.

Table 1. Optimal parameter range for SWM

n	k	q
32-127	2-4	1-4
128-255	3-5	2-4
256-511	4-6	2-5
512- 639	4-6	3-5
640-1023	5-6	3-5
1024-2048	6-7	4-6

Moreover, to reduce the negative effect of the *k*-constrains, Bos and Coster (1990) proposes finding AS for the sequence of the unique NW. The AS-length depends on that of the largest NW max(m_i), an NW with the longest expected AC and the number of the NWs s. Yao (1976) related the upper bound AS-length L for a sequence vector $V = (v_1,v_2,\ldots,v_s)$, $v_1<v_2<\ldots<v_s$ as:

$$L(V) \le \lg(v_s) + c \times \sum_{i=1}^{s} \lg(v_i) / \lg(\lg(v_i + 2)) \quad (7)$$

where, c is constant and (henceforth) $\lg(x) = \log_2(x)$. The worst case scenario occurs when $s = 2^{k-1}$, that is $m_i \neq 0$, $i = 1,\ldots,2^{k-1}$, in which $L = 2^{k-1}$. As demonstrated in (Nedjah and Mourelle, 2005; Bos and Coster, 1990; Cruz-Cortés *et al*., 2008; Dominguez-Isidro and Mezura-Montes, 2011; Dominguez-Isidro *et al*., 2015), an even better result is expectedly obtainable with a well-formulated AS algorithm.

Heuristics for Addition-Subtraction Chain Problem

A class of algorithms that is related to the family of windows methods is that of Addition Subtraction Problem (ACSP). ACSP is a special case of ACP in which subtraction (inversion as applied to exponentiation) is also allowed.

Definition 5

Given an integer e, the sequence A: $a_0 = 1$, $a_1 = 2$, $a_3,\ldots,a_r = e$, is said to be an Addition-Subtraction Chain (ASC) for e if $\forall i \ge 1$, $a_i = a_j \pm a_k$, $i>j\ge k$.

ASC heuristic algorithms are proprietary to scalar multiplications as in ECC due to the negligible cost of inversion in its field. We demonstrate the strength of the subtraction with the class of $e = 2^n-1$. These es exhibit the longest binary AC-length of $2(n-1)$, having n-1 doubling and n-1 additions, as: 1,2, 2+1, 2^2+2^1, $2^2+2^1 + 1,\ldots,2^{n-1} +\cdots+ 1 = 2^n-1$. But by admitting subtraction in the AC, the same binary method produces shortest AC with length n +1 by performing n doubling and a subtraction, as $1,2,2^2,2^3,\ldots,2^n$, 2^n-1. In general any sequence of consecutive k non-zero bits $11\ldots1 = 1\times2^{k-1} +1\times2^{k-2} +\cdots+1\times2^0 = 2^k-1$ could be recoded with k +1 bits as $100\ldots-1 = 1\times2^k +0\times2^{k-1}+0+\cdots -1\times2^0 = 2^k-1$ (henceforth -1 is coded $\bar{1}$). The former has $H(e) = k$ while in the latter $H(e) = 2$. Therefore, the idea is to reduce the number of additions due to $H(e)$ that follows the doublings. In this regard balanced ternary $(\bar{1},0,1)$ recoding (Knuth, 1998) is re-introduced to minimize the non-zero density in any given e where possible.

Various signed binary (ternary) recoding methods exist, with Non-Adjacent Form (NAF) (Eğecioğlu and Koç, 1994; Reitwiesner, 1960; Morain and Olivos, 1990) being canonical that minimizes $H(e)$ from asymptotic $n/2$ to $n/3$. A *kNAF* is a signed equivalent of m-ary and reduces the density asymptotically to $1/(k+1)$ (Okeya, 2004). Similarly, in (Laith and Kuo, 1997), the m-ary version of Modified Signed Digit (MSD) is proposed, having the same non-zero density as the *kNAF* but demonstrating an improved empirical result: This is possibly due to the m-ary lookup table with sparse bits. Mutual Opposite Form (MOF) (Okeya, 2004) is similar to NAF, but with the advantage of having left-to-right version. The left-to-right recoding eliminates the need for two parses and additional n-bit memory required in the NAF process is reduced to 1 (or k-bit for its *kMOF* version). Basically for an n -bit e:

$$MOF(e) = (e \ll 2) \ominus e \quad (8)$$

All the operations in Equation 8 are bitwise.

On the other hand, Complementary Recoding (CR) (Balasubramaniam and Karthikeyan, 2007) is easier as 1+ bitwise complement of e are bitwise-subtracted from $2^{n(e)+1}$. That is:

$$CR(e) = 2^{n(e)+1} \quad \ominus\bar{e} \ominus 1 \quad (9)$$

Note that Equation 9 is also equivalent to $2^{n(e)+2}$-e. However, CR unconditionally results in additional 1 bit to a recoded e and is only effective in reducing the $H(e)$ when $H(e)>n/2$. As n-bit e has its (n+1) -bit recoded CR(e) related as:

$$H(CR(e)) = n + 1 - H(e) \quad (10)$$

Therefore, CR rather increases the non-zero bit density when $H(e)>n/2$. But this is not the case for other recoding methods. For example, given $e = 24066_{10} = 101111000000010_2$: $H(24066) = 6$ and $CR(24066) = 10\bar{1}0000\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}\bar{1}10_2$. Therefore $H(CR(24066)) = 10$. But the $NAP(24066) = 10\bar{1}000\bar{1}000000010$. As for suitable recoding for method for the formal SWM, it is an open problem (Win *et al*., 1998).

Other Heuristics for Addition Chain Problem

Various other deterministic heuristics for ACP exists, that result in minimal length AC than binary family. Famous among them are Factor and Power-tree methods.

Factor method factors e as $e = yz$. Note that $x^e = x^{yz} = (x^y)^z$. Thus, AC for $e = yz$ reduces to the AC for y followed by that for z, wherein 1 is replaced by y and a prime e is decomposed into $e = yz +1$. The AC length l_f by the method can be obtained recursively as:

$$l_f(e)=\begin{cases} l(y)+l(z)\ if\ \ e=composite \\ l(y)+l(z)+1\ if\ 2<e=prime \\ 1\ if\ \ e=2 \end{cases} \quad (11)$$

Therefore, the method recursively factors e into its smallest prime factors and subsequently evaluates the corresponding ACs as described.

Power-tree also recursively generates the AC for e, prior to which at least all that of $e/2$ must have been generated as the base (Knuth, 1998). Let the AC and its length be denoted as A_{pt} and l_{pt} respectively, then:

$$A_{pt}(e)=A_{pt}(b)\cup\{a_j+b\} \quad (12)$$

such that $a_j \in A_{pt}(b)$, $a_j + b = e$. Thus:

$$l_{pt}(e)=l_{pt}(b)+1 \quad (13)$$

for some $e/2 \le b < e$ for which the AC is already generated.

Therefore, Power-tree method is a semi-exhaustive algorithm. Moreover, the direction in which the b is searched for, in the previously generated results, determines the optimality of the corresponding AC: Generally forward scan from the smallest to the largest ones yields better result. To study it, we apply the method to generate ACs for e = 1 to 4096. Optimal accumulated AC length for the set is 54,408. Forward scan in method yields 54,812 (99.26% optimal), while the result for the backward scan is 61,457 (87.04% optimal).

Both the factor and the power-tree methods, though effective, are only feasible for relatively small integers. For a large number, IFP in itself is an NP-complete problem (Mohamed *et al*., 2011).

Gelgi and Onus (2006) propose improved versions of both Factor and Power-tree methods: The proposed Factor heuristic searches for the pair factors x,y: $e = xy$ whose ACs already computed to have shortest length summands; Whereas in the power heuristic (called Dynamic heuristic) given e, instead of searching for AC for the first b whose $a_i = e-b$, the algorithm searches for the one with the shortest AC-length among the already generated ACs.

Our preliminary investigations on the proposed Dynamic heuristic revealed that its optimality is again a function of the base ACs up to that of value δ that are needed to be pre-computed and, for the algorithm to guarantee any (near) optimality, the base ACs of at least 1 to $e/4$ are needed to be evaluated by some exhaustive method. For example we apply the method to evaluate ACs for e = 1 to 4096, the resulting accumulated AC-length is presented in Table 2.

Table 2. Accumulated AC-lengths for 1 to 4096 due to varied δ and e = 4096 in Dynamic heuristic

δ	$e/32$	$e/8$	$e/4$	$e/2$
Cum. AC-length	54,613	54,556	54,482	54,410
%Opt	99.62	99.73	99.86	≈100

Cum: Cumulative; % Opt: Percentage Optimality

The method may not be feasible for large e with n of order 128 to 2048, utilized in PKCs.

Mani (2013) proposed division-based AC method. It is an attempt to design an AC algorithm that is as efficient as metaheuristic-based algorithms (to be detailed later) while avoiding the processing overhead, by emulating windows-based methods. e is successively divided by 2 while the quotient is tracked; the quotients' sequence structure determines the selection of one base ACs, to be utilized in building the e's AC, out of 9 pre-defined optimal star-chains; based on the partial AC generated by the quotient, the AC is then systematically built on the selected base AC. The method is potentially as efficient as windows-based methods. However, we observed that it results in little optimal AC than SWM.

Mohamed *et al*. (2011) proposed factorization-based Decomposition Method (DM). e is initially factored into powers of its primes factors $p_1,p_2,\ldots,p_m$ such that $e = p_1^{n_1} \times p_2^{n_2} \times \ldots \times p_m^{n_m}$. For each p_i, a rule is applied, that is similar to (and with same result as) the binary method, to compute its AC. AC for $p_1^{n_1}$ follows from that of the $p_1 : 1,2,\ldots,a_{r_{p_1}}$ as a base, by repeatedly applying the base AC structure while substituting the current last element a_r as the next first element $a_0 n_1$-times. Thus, resulting AC-length equals $n_1 \times l_{p_1}$. Similar procedure is applied on the remaining $p_i^{n_i} : i = 2\ldots m$, by substituting the last element of the AC corresponding to $p_{i-1}^{n_{i-1}}$ as the first element of p_i's AC. Therefore, AC-length for e $l_{dm}(e)$ due to DM $A_{dm}(e)$ is given as:

$$l_{dm}(e)=\sum_{i=1}^{m} n_i \times r_{p_i} \quad (14)$$

where, $r_{p_i} = l_{bin}(p_i)$. The algorithm results in more optimal ACs for large integer as compared to window-based methods. However, it is most suitable for smooth integers since prime factorization is an NP-complete problem. Additionally, an even better result is obtainable if a more optimal algorithm is utilized in evaluating the ACs for the p_i instead. For example, $l_{dm}(e = 1553^9 \times 5521^{13} \times 9281^5) = 418$. But 5521 has an AC with length 15 (1-2-4-8-9-17-26-43-86-172-344-688-1376-2752-5504-5521) which can be utilized, instead of $A_{bin}(5521)$ having $l_{bin} = 17$, to obtain shorter AC having length 392 (26 less). Even on applying SWM, $l_{swm}(5521, k = 3) = 16$, a 405-length AC is obtained.

We generalized that windows-family methods are efficient in terms of implementation but least effective in finding (near) optimal AC; there are feasible windows-based heuristic that generates more optimal results for specific classes of integers (Mohamed *et al*., 2011) or proprietary to ECC (Balasubramaniam and Karthikeyan, 2007; Chang *et al*., 2003; Joye and Yen, 2000; Morain and Olivos, 1990; Okeya, 2004); Others, like factor and power methods, are feasible only for small-sized integers far from those utilized in PKCs; Exhaustive methods have been the most effective but often finding optimal results requires months of computer-cluster efforts (Bahig, 2006; 2011; Clift, 2011). There is yet to be any algorithm *alg* that can answer that for any given $eA_{alg}(e)$ has $l_{alg}(e) = l(e)$. However, many studies have been carried out at establishing theoretical optimal ACs, at least for some classes of integers, without necessarily finding the exact AC sequence. Even as the study are yet to yield the objective in general but have been able to establish some bounds, cost function and specialized set of integers which are useful in searching for (near) optimal ACs. The resulting studies are valuable in designing yet effective heuristics for the ACP.

Theoretical Results and Asymptotic Bounds for ACP

Downey *et al*. (1981) prove that ASP is an NP-complete problem. ACP, even yet to be proven as an NP-complete, is an NP-hard problem that is believed to be complete as well. An upper bound for $l(e)$ is given (Knuth, 1998) as:

$$l(e) \le \lfloor \lg(e) \rfloor + H(e) - 1 \tag{15}$$

According to Thurber (1973), for all e such that $n(e)<17$:

$$l(e) \ge \lceil \lg(e) + \lg(H(e)) \rceil \tag{16}$$

$l(e)$ lower bound is (Schönhage, 1975):

$$l(e) \ge \lceil \lg(e) + \lg(H(e)) - 2.13 \rceil \tag{17}$$

Therefore, from (15) and (17):

$$\lceil \lg(e) + \lg(H(e)) - 2.13 \rceil \le l(e) \le \lfloor \lg(e) \rfloor + H(e) - 1 \tag{18}$$

For all e such that $H(e)<4$, $l(e) = \lfloor \lg(e) \rfloor + H(e) - 1 = l_{bin}(e)$.

Equation 18 along with (7) are viable tools in controlling windows-based ACP heuristics, to search efficiently toward the optimal results: Equation 7 can be utilized to evaluate a partition without generating the actual AC while using Equation 18 to decide on its optimality or otherwise.

In respect of the cost functions:

$$l^*(e) \ge l(e) \tag{19}$$

Huge number of e exist for which $l^*(e) = l(e)$. It was widely conjectured that for any e there exists star-chain(s) with $l^*(e) = l(e)$. However, Hansen (1957) proves the otherwise; as exemplified in (Knuth, 1998). Notwithstanding, ACP heuristics generate (near) optimal star-chains and the chain is sufficient for practical purpose. Bahig (2011) even proves that for any e there exists optimal AC(s) such that the last four (4) steps are stars. He further proves that for all e: $n(e) \le 2^{18}$ the last half elements of their optimal ACs are star; and finally conjectured that all last $\left\lfloor \frac{l(e)}{2} \right\rfloor$-steps are star. We observed that in optimal ACs for any e all steps after $a_m \ge \left\lfloor \frac{e}{2} \right\rfloor$ are star and are well-defined. Consequently, heuristics should revert to local-search after generating the a_m.

According to Thurber (1993), there exist from a very few to a large number of minimal ACs for any given e, referred to as MNC(e). The NMC depends on the cardinality of e and, to some extends, pattern of 1s in its binary form. However, he concluded that NMC(e) is erratic with respect to the e. Similar studies on star subsets of the optimal AC (i.e., $l^*(e) \subset l(e)$) may assist in optimizing ACP heuristics: This is in view of the fact that the algorithms generate the star-chains; Additionally, it is easier to find optimal (star) chains for e with large NMC as compared to their counterparts.

Knuth (1998) backed theoretical findings with many empirical results that aid in benchmarking ACP algorithms. Among the results are for set small of es for which the ACs known but considered special for the uniqueness of their optimal ACs. Refer to (Knuth, 1998) for the list. Similarly, $c(r)$ is defined as smallest e for which $l(e) = r$. Furthermore, Knuth (1998) compiled $c(1)$ to $c(27)$ while Clift (2011) found up to $c(38)$. Dominguez-Isidro *et al*. (2015) reported another set of es termed *"hard"* because of the relative hardness in finding their optimal AC by deterministic heuristics. The set consists of 3243679; 3493799; 3459835; 3235007; 3230591; 3182555; 3440623; 3704431; 3234263; 3352927; 3926651; 3922763; 2948207, each having $l = 27$. While the results do not contribute to the development of any ACP heuristic, they serve as test-beds for evaluating ACP heuristics. Recently, exploiting solution for ACP witnesses a huge turning point where efforts have since been shifted towards exploring various metaheuristic.

Metaheuristic Algorithms for ACP

Cruz-Cortés *et al*. (2005) formulate GA model for ACP. Variable-length chromosomes represent the AC; a_i elements of the AC are mapped against genes in the chromosome; the AC-length represents the chromosome fitness. Initially, a set of star-chains for e are randomly generated. A one point crossover is utilized; whereby two ACs (parents) generate a pair of new upspring. A crossover point is selected at random in each of the parents and all the elements (genes) before the respective points are copied into generating child chromosomes. The AC pair being generated (children) and parent ACs are then interchanged (crossed); and the rule applied to generate the remaining elements of the parent ACs are utilized to generate that of the children ACs. The process is followed by mutating the children, in which in each two points i,j: $2 \leq j < i < (l_{GA}-2)$ (where l_{GA} is an AC length) are selected at random. An element a_{i+1} is replaced with $a_i + a_j$; and all the remaining upper elements (genes) are replaced by randomly alternating between doubling ($a_i = 2a_{i-1}$) and star ($a_i = a_{i-1} + a_j, j<i-1$) steps. The union of the parents and children are ranked in order of their fitness and the first half fitted ones are selected for the next iteration (generation). The process is repeated for a number of generations and, finally, the fittest valid AC(s) is returned.

Utilizing 100 parents par generation for 300 generations (as reported), a total number of 60000 ACs for e are generated and space corresponding to 200 ACs is required, through the life-span of the algorithm. It generates (near) optimal ACs, than deterministic heuristics earlier detailed, for small sets of integers; at the expense of the estimated memory/processing overheads. To utilize the resulting AC in PKC, memory units corresponding to the AC-length is required.

Osorio-Hernández *et al*. (2009) proposed an improved version of the GA, in which only valid ACs are retained in the population; Two-point crossover is utilized; and Lucas ($a_i = a_{i-1} + a_{i-2}$) is introduced among the steps, at $Pr = 0.2$. Double step is given $Pr = 0.7$. To generate the valid AC, whenever the last $a_i = a_{i-1} + a_j > e, a_j$ is replaced by a_k such that $a_i = a_{i-1} + a_k \leq e, k<j$. The a_k is sequentially searched down the AC. Additionally, 4 mutants per child are produced, whereby the fittest among them is selected. Thus, 240000 ACs are evaluated and memory corresponding to 200 ACs needed. An even more optimal result is realized consequent to the mentioned improvements.

Nedjah and Mourelle (2004; 2006) applied distributed multi-agent ant system ACO, while León-Javier *et al*. (2009) use the PSO. Both are similar to GA. In the ACO, a similar set of parameters to the GA is used but the improved GA discussed performs comparably better. PSO evaluates 30,000 ACs and requires memory space equivalent to 30 ACs. But reported PSO results are for es with $n<12$: These are by far smaller than ones utilized in PKC. Similarly, the GA-based heuristic was reported by Rodriguez-Cristerna and Torres-Jimenez (2013), in which factorial number system is used in the AC representation. The algorithm is fortified with special neighborhood and distribution function. The reported results are also for relatively small integers. Nedjah and Mourelle (2006) reported a promise-looking result from ACO, for up to 1024-bit integers. However, the report may either be incomplete to justify the result: For example, the reported Table 2 showing 1024-bit integers having average AC-length of 1022 is contrary to Equation 17. Moreover, considering the large number of integers in the given range, sampling characteristics of the represented set has to be stated clearly. For example, Kunihiro and Yamamoto (2000) show that optimality of AC for e is a function of $H(e)$: Those e with very few and those with almost all non-zero bits in the corresponding binary forms represent the integers for which their optimal ACs are easy to obtain; Whereas those es having on the average $n/2$ bits density constitute the most difficult set for which to find the optimal ACs. Cruz-Cortés *et al*. (2008) and Dominguez-Isidro *et al*. (2015) opined that metaheuritics, generally, return (nearest) optimal ACs for relatively small to medium integers. In their respective Artificial Immune System (AIS) and AC Evolutionary Programming (ACEP), SWM is integrated to handle large integers. This followed from a proposal pioneered by Bos and Coster (1990).

Bos and Coster (1990) show that computing AS corresponding to the NW in SWM gives room for an even larger k. An algorithm called *Makesequnce* is applied to generate the AS. Given a vector of NW $V = \{v_1, v_2, \ldots, v_s\}$, the algorithm begins with *protosequence AS* = $\{1,2, v_1, v_2, \ldots, v_s\}$. Four rules are defined for the process:

- Approximation: if $v_i + v_j = v_k - \varepsilon . i \leq j < k$ and ε is small positive integer then insert $v_i + \varepsilon$ into the sequence. E.g., 49-67-85-117 ⟼ 49-50-67-85-117 (where 49+67 = 117-1)
- Division: if v_i is divisible by a small prime p then add $\frac{v_i}{p}, \frac{2v_i}{p}, \ldots, p$ into the sequence. E.g., 17-48 ⟼ 16-17-32-48 (where $p = 2$ and 2 divides 48)
- Halving: take a small number t that occurs earlier in the sequence and insert $v_i - t, \frac{v_i - t}{2}, \frac{v_i - t}{4}, \ldots, \frac{(v_i - t)}{2^u}$ for some u. E.g., 14-382⟼14-23-46-92-184-368-382 (where $t = 14$ and $v_i = 382$)
- Lucas: construct Lucas sequence where necessarily so that v_i is last element of the Lucas sub-chain. E.g., 4-23⟼4-5-9-14-23

And finally, remove any redundancy. Some weighting function where attached in selecting the rule, but (as reported) it does not yield any better result. For a set of NW with the largest window $v_s \leq 1000$:

$$L(NW) \leq \frac{3}{2}\lg(v_s) + s + 1 \quad (20)$$

The algorithm is demonstrated having a good result for 512-bit integers, in which about 2/3 of the bits are non-zero. But the criteria for applying the four (4) rules is left open. Therefore, the heuristic is left as a framework with the hope of yielding a nearly optimal result. Cruz-Cortés *et al*. (2008) adopted the approximation rule in an AIS-based AC algorithm, wherein it is called Insertion.

AIS is also nature-inspired swamp intelligent-based AC algorithm. Its simplified version for the AC mimics the pattern matching between antigen and antibody and clonal selection principles. A cloning-mutation is applied to a generation of antibodies and mutated clones with the highest affinity are retained for the next generation. An immune memory mechanism is used to retain some subset of higher affinity clones for future immunity. e is mapped to an antigen, the antibody and it affinity being the corresponding AC and its length $(A_{ai}(e),l_{ai}(e))$ respectively. Cloning involves creating new AC from stimulated existing one with higher affinity (shorter length) and is followed by *hypermutation* (perturbation of the new AC). A cloned AC with low affinity is receptor-edited (replaced) with ones with higher affinity. The process is repeated for a number of generations and those clones with the highest affinity are finally returned.

Analytically, the algorithm generates a population of N of ACs for e. Best P among them are selected for cloning/mutation. Beginning from the one(s) with the shortest length, first batch from the P is utilized to clone another N ACs; then $N/2$ are cloned, utilizing the second batch; until all the P ACs are exhausted. Each of the cloned AC is then mutated into a pair. If e is even, best d among the $2P$ ACs are kept for later reference and finally best N ACs are selected for the next generation. The process is repeated for say g generations. Therefore, the algorithm generates and evaluates at least gPN ACs and $N + d$ AC-units of memory is required. For example given $g = 25, N = 45$, $d = 0.1N$ and $P = 0.25N$: 12375 evaluations are made, requiring memory resources equivalent to 49 ACs.

AIS utilizes least resources among the metaheuristic discussed. It also returns competitively good results for small integers. As for the large ones, the algorithm partitions e just as in SWM detailed. However, in this case an arbitrary larger MSW m_{w-1} is utilized (normally 6 to 20-bit). Then AIS AC algorithm is applied to evaluate the AC for the MSW. The AC serves as an input, along with the remaining unique NWs, for the Insertion-based AS heuristic to generate the sequence and the AC-length is determined as:

$$l_{vw}(e,k,q) = L(NW) + n - n(m_{w-1}) + w - 1 \quad (21)$$

The process is repeated with varying length MSW, but normally fixed k,q at 6,q respectively. The AC with the shortest length is returned.

In the AIS-based sequence generation, given ordered *protosequence* from the $NW\{1,2,m_1,m_2,\ldots,m_s\}$, the algorithm inserts m_s-m_{s-1} that is not in the sequence and sort it again; the procedure continues with the pair $(m_{s-1}, m_{s-2}),\ldots, (m_i,m_{i-1}),\ldots, (m_0,m_1)$, inserting m_i-m_{i-1} that does not exists and sorting the sequence to maintain the order. The estimated AS-length is given as:

$$L(NW) \leq \frac{4}{3}\lg(m_s) + s + 2 \quad (22)$$

ACEP (Dominguez-Isidro *et al*., 2015) utilizes the same method while adopting Evolutionary Programming (EP) model (Dominguez-Isidro and Mezura-Montes, 2011) for generating the initial AC.

EP is an evolutionary computation algorithm in which generations are produced asexually without a parental selection process. In the model, the process of solution encoding, initial population generation and the probability function are similar to that of GA (Osorio-Hernández *et al*., 2009), where N individual ACs are generated. t-mutants are created from each of the N ACs and the best among them is selected as the respective offspring. In the replacement process, each AC competes with q random ACs among the union of the N parent ACs and their offspring; while the number of wins (number of times individual AC-length is shortest among its competitors) is recorded. The ACs are then sorted according to their wins and first-half with highest frequency of wins are selected for next generations. The process is repeated for *MAXGEN* number of generations. Thus, the algorithm generates $N \times q \times t \times MAXGEN$ AC in the process and requires N AC-units of memory. For example given the value for N, q, t and *MAXGEN* are 100,10,4 and 23 respectively, 92,000 AC are evaluated.

ACEP reported slightly better result than AIS for small integers, while both are roughly equal for large ones. But in respect of computing/memory resource AIS is comparably better. To be implemented in PKC, both requires memory units equivalent to the respective AS-length for the NW vector. However, the integer sets utilized by both are insufficient to generalize their effectiveness behaviors-being very small.

On a general note: Metaheuristics applied on ACP are largely nature-inspired, evolutionary or swamp intelligence-based, that are also population-oriented;

with ACEP considered the most successful one reported so far. Jose-Garcia *et al*. (2011) employed trajectory-based Simulated Annealing (SA) but exhibited less promising results. The metaheuristics could be regarded as second most optimal approaches for ACP-superior ones being exhaustive methods. In relation to PKC, the ACs are generated/determined independently before being applied. They are also less efficient as compared to window-based methods: This is partly obvious as they are a sort of compromise, but partly it is due to the population-dependence. The resulting optimality degenerates as the integers become relatively large (Dominguez-Isidro *et al*., 2015). Consequently, it is generally believed that SWM still remains the most feasible optimal method for large integers, albeit with the k -constraint (Bos and Coster, 1990; Cruz-Cortés *et al*., 2008; Dominguez-Isidro *et al*., 2015). Therefore, hybrid of metaheuristic-SWM have emerge as an alternative approach, in which the metaheuristic overcome the k-constrain in the SWM. Worthy of nothing is that finding optimal AS for the NW vector, on which the hybrid depends, is an NP-complete problem (Downey *et al*., 1981). Thus expecting high-quality solution requires well-formulated algorithm for the ASP.

Conclusion

Based this survey on various works/studies on the ACP, following are highlights of issues worth considering, apart from optimality, specifically when designing ACP heuristic for PKC:

- Implementation memory requirement. Binary method is least effective in respect of the optimal exponentiation/scalar multiplication operations but is the most economical in terms memory resource utilization. It requires two units of memory only: One for the intermediary result whiles the other for the base. SWM, in general, are next to binary, requiring 2^{k-1} units for the pre-computed windows. As for the other metaheuristic, it depends on the implementation. But it is obvious that during the AC search/generation population-based algorithms require much memory, depending on the number of population; this is apart from the memory required to store the intermediate results corresponding to the elements in the AC when implemented
- Processing overhead in search for the optimal AC. Again binary method and SWM utilizing fixed windows parameters (k,q) can be applied with negligible processing overhead. However, considering the poor performance of binary in terms of optimality and that SWM using fixed parameters does not guarantee optimal result, it is worth at least searching for the optimal window parameter for a fairly large integer in PKC and when a known and large integer is to be utilized at least twice it is more economical to evaluate its (near) optimal AC prior to the actual exponentiation/scalar multiplication: Most of the PKCs keys are in this category. In fact, for the likes of RSA, the corresponding AC may be pass/stored along with the other key parameters as is the tradition with CRT of the private key d
- Multiplication versus squaring. Squaring is generally regarded as more efficient than multiplying two different numbers. As such, an AC with more proportionate doubling than addition will be faster in term of its corresponding exponentiation. However, this is different in ECC's scalar multiplication where both ECADD and ECDBL involve some number of multiplications, squaring and inversions
- AC/AS representation for very large integers with hundreds to a few thousands ACs lengths. The AC sequence encoding is another issue worth given attention to. A good candidate for this is star-chain, where only one "parent" needs to be remembered for the child element to be reconstructed. Additionally, smaller integer structure can be used to only keep track of the AC elements and their parents' positions. For example, given all 512-bit integers, their star-chain can be structurally represented using 10-bit integers, say, as 2,3(1),5,9,10(4),...,511(x): That is for a 511-lngth AC beginning from 1 as its 0^{th} element; where a number in bracket is omitted doubling is implied e.g., 1,4, 6 and 7; the number in bracket points at the second parent of the given element; and where absent a Lucas is implied

We conclude that metaheuristic-based AC algorithms provide a competitively nearer optimal solution to the ACP in a practically reasonable period of time, albeit with some. However, for the large integers, the edge between the metaheuristic and SWM is still little. Therefore, there is a need for a yet efficient hybrid of the heuristic/metaheuristic that is capable of evaluating (to the nearest) optimal AC for any arbitrary integer. This is achievable by exploiting the analytical works for suitable boundaries and cost function; then integrating the deterministic SWM into some efficient metaheuristic-not necessarily complex population-based ones – to arrive at the most optimal compromise and achieve the nearest optimal result. We observed that the following approaches could be a success path to an optimal AC heuristic:

- Investigating the star elements distribution in the star-chain could provide a good cost function for a yet successful algorithm
- The probability distribution of the star elements in the optimal ACs is expected to be a key to effectiveness of an algorithm by simulating the optimality

- Instead of population-based metaheuristic, it may be better to apply trajectory-based one such as great deluge algorithm (Duek, 1993) that was successfully applied in other similar problems (McMullank, 2007; Duek, 1993), by giving it tight boundaries and good cost function
- There is yet the need for well-formulated ASP heuristic
- For very large integers, considering the efficiency of SWM but with its *k*-constraints, an algorithm in the point 4 could be integrated to effectively handle sequence of larger-sized windows
- Point 1 to 5 could be tailored to the specific case of the ECC

Acknowledgement

This research was partially supported by the Malaysian Ministry of Higher Education [Grant No: FRGS/1/2015/ICT03/UNISZA/02/1] and [Grant No: FRGS/1/2014/ICT03/UPM/03/1].

Author's Contributions

Adamu Muhammad Noma: Carried out the review process of the materials selected by the other authors, that are included in the review and forwards same for further review by the authors.

Abdullah Muhammad: Contributed in the design and flow of the review and in the selection and evaluation of the relevant heuristic-based works included.

Zuriati Ahmad Zurkannain: Coordinate the review process and ensure both the heuristic and security aspect are relevant and professionally reviewed

Mohamad Afendee Mohamed: Initiated the framework for the review process and the selection and evaluation of the PKCs works included in the review.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Adrian, D., K. Bhargavan, Z. Durumeric, P. Gaudry and J.A. Green *et al*., 2015. Imperfect forward secrecy: How Diffie-Hellman fails in practice. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Oct. 12-12, ACM, pp: 5-7. DOI: 10.1145/2810103.2813707

Bahig, H.M., 2006. Improved generation of minimal addition chains. Computing, 78: 161-172. DOI: 10.1007/s00607-006-0170-6

Bahig, H.M., 2011. Star reduction among minimal length addition chains. Computing, 91: 335-352. DOI: 10.1007/s00607-010-0122-z

Balasubramaniam, P. and E. Karthikeyan, 2007. Elliptic curve scalar multiplication algorithm using complementary recoding. Applied Math. Comput., 190: 51-56. DOI: 10.1016/j.amc.2007.01.015

Boneh, D. and G. Durfee, 2000. Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans. Inform. Theory, 46: 1339-1349. DOI: 10.1109/18.850673

Boneh, D., 1999. Twenty years of attacks on the RSA cryptosystem. Notice AMS, 46: 203-213.

Bos, J. and M. Coster, 1990. Addition chain heuristics. Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology, Aug. 20-24, Springer, New York, pp: 400-407. DOI: 10.1007/0-387-34805-0_37

Brauer, A., 1939. On addition chains. Bull. AMS, 45: 736-739. DOI: 10.1090/s0002-9904-1939-07068-7

Chang, C.C., Y.T. Kuo and C.H. Lin, 2003. Fast algorithms for common-multiplicand multiplication and exponentiation by performing complements. Proceeding of the 17th International Conference on Advanced Information Networking and Applications, Mar. 29-29, IEEE Xplore Press, pp: 807-811. DOI: 10.1109/aina.2003.1193005

Clift, N.M., 2011. Calculating optimal addition chains. Computing, 91: 265-284. DOI: 10.1007/s00607-010-0118-8

Cruz-Cortés, N., F. Rodríguez-Henríquez, R. Juárez-Morales and C.A. Coello-Coello, 2005. Finding optimal addition chains using a genetic algorithm approach. Proceedings of the International Conference on Computational Intelligence and Security, Dec. 15-19, Springer-Verlag, pp: 208-215. DOI: 10.1007/11596448_30

Cruz-Cortés, N., F. Rodríguez-Henríquez, R. Juárez-Morales and C.A. Coello-Coello, 2008. An artificial immune system heuristic for generating short addition chains. IEEE Trans. Evolut. Comput., 12: 1-24. DOI: 10.1109/tevc.2007.906082

Dahshan, H., A. Kamal and A. Rohiem, 2015. A threshold blind digital signature scheme using elliptic curve dlog-based cryptosystem. Proceedings of the IEEE 81st Vehicular Technology Conference, May 11-14, IEEE Xplore Press, pp: 1-5. DOI: 10.1109/vtcspring.2015.7145653

Dellac, H., 1894. Question 49. L'Intermédiaire Math.

Diffie, W. and M.E. Hellman, 1976a. New directions in cryptography. IEEE Trans. Inform. Theory, 22: 644-654. DOI: 10.1109/tit.1976.1055638

Diffie, W. and M.E. Hellman, 1976b. Multiuser cryptographic techniques. Proceedings of National Computer Conference and Exposition, Jun. 7-10, ACM, pp: 109-112. DOI: 10.1145/1499799.1499815

Dominguez-Isidro, S. and E. Mezura-Montes, 2011. An evolutionary programming algorithm to find minimal addition chains. Proceedings of the Congreso Internacional de Ingeníería Electrónica, Instrumentación y Computación, Minatitlén Veracruz, Jun. 22-24, México, pp: 1-5.

Dominguez-Isidro, S., E. Mezura-Montes, N. Cruz-Cortés and F. Rodríguez-Henríquez, 2015. Evolutionary programming for the length minimization of addition chains. Eng. Applied Artif. Intell., 37: 125-134. DOI: 10.1016/j.engappai.2014.09.003

Downey, F., B. Leong and R. Seith, 1981. Computing sequences with addition chains. SIAM J. Comput., 10: 638-646. DOI: 10.1137/0210047

Duek, G., 1993. New optimization heuristics: The great deluge algorithm and the record-to-record travel. J. Comput. Phys., 104: 86-92. DOI: 10.1006/jcph.1993.1010

Eğecioğlu, Ö. and Ç.K. Koç, 1994. Exponentiation using canonical recoding. Theor. Comput. Sci., 129: 407-417. DOI: 10.1016/0304-3975(94)90037-x

ElGamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory, 31: 469-472. DOI: 10.1109/tit.1985.1057074

Gelgi, F. and M. Onus, 2006. Heuristics for minimum Brauer chain problem. Proceedings of the 21st International Conference on Computer and Information Sciences, Nov. 01-03, Springer-Verlag, Istanbul, Turkey, pp: 47-54. DOI: 10.1007/11902140_7

Hankerson, D., A. Menezes and S. Vanstone, 2004. Cryptography Basics. In: Guide to Elliptic Curve Cryptography, Hankerson, D., A. Menezes and S. Vanstone (Eds.), Springer-Verlag, New York, pp: 2-3.

Hansen, W., 1957. Untersuchungen über die Scholz-Brauers chen addition sketten und deren verallgemeinerung. Göttingen, Wiss. Prüfungsamt, Schriftl. Hausarbeit zum Staatsexamen.

Hoffstein, J., J. Pipher and J. H. Silverman, 2014. The Elliptic Curve Discrete Logarithm Problem. In: An Introduction to Mathematical Cryptography, Hoffstein, J., J. Pipher and J. H. Silverman, (Eds.), Springer, New York, pp: 310-312.

Jose-Garcia, A., H. Romero-Monsivais, C.G. Hernandez-Morales, A. Rodriguez-Cristerna and I. Rivera-Islas *et al*., 2011. A simulated annealing algorithm for the problem of minimal addition chains. Proceedings of the 15th Portugese Conference on Progress in Artificial Intelligence, Oct. 10-13, Springer-Verlag, Lisbon, Portugal, pp: 311-325. DOI: 10.1007/978-3-642-24769-9_23

Joux, A. and R. Lercier, 2003. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. Math. Comput., 72: 953-967. DOI: 10.1090/s0025-5718-02-01482-5

Joye, M. and S. Yen, 2000. Optimal left-to-right binary signed digit recoding. IEEE Trans. Comput., 49: 740-748. DOI: 10.1109/12.863044

Koblitz, N., 1987. Elliptic curve cryptosystems. Math. Comput., 48: 203-209. DOI: 10.2307/2007884

Koblitz, N., A. Menezes and S. Vanstone, 2000. The state of elliptic curve cryptography. Designs Codes Cryptography, 19: 173-193. DOI: 10.1023/a:1008354106356

Koc, C.K., 1994. High-speed RSA implementation. Technical Report TR 201, RSA Laboratories, Redwood City, CA.

Koc, C.K., 1995. Analysis of sliding windows techniques for exponentiation. Comput. Math. Applic., 3: 17-24. DOI: 10.1016/0898-1221(95)00153-p

Knuth, D.E., 1998. Evaluation of Powers. In: The Art of Computer Programming, Knuth, D.E. (Ed.), Addison-Wesley, USA, pp: 461-485,

Kunihiro, N. and H. Yamamoto, 2000. New methods for generating short addition chains. IEICE Trans. Fund. Electr. Commun. Comput. Sci., 83: 60-67.

Laith, C. and W. Kuo, 1997. Speeding up the computations of elliptic curve cryptosystems. Comput. Math. Applic., 3: 29-36. DOI: 10.1016/s0898-1221(97)00017-5

Lee, Y., H. Kim, S. Hong and H. Yoon, 2006. Expansion of sliding window method for finding shorter addition/subtraction-chains. Int. J. Net. Security, 2: 34-40.

León-Javier, A., N. Cruz-Cortés, M.A. Moreno-Armendáriz and S. Orantes-Jiménez, 2009. Finding minimal addition chains with a particle swarm optimization algorithm. Proceedings of the 8th Mexican International Conference on Artificial Intelligence, Nov. 09-13, Springer Berlin Heidelberg, pp: 680-691. DOI: 10.1007/978-3-642-05258-3_60

Maletsky, K., 2015. RSA Vs ECC comparison for embedded systems. White Paper, Atmel.

Mani, K., 2013. Generation of addition chain using deterministic division based method. IJCSET, 1: 553-560.

May, A., 2004. Computing the RSA secret key is deterministic polynomial time equivalent to factoring. Proceedings of the Annual International Cryptology Conference, (ICC' 04), Springer Berlin Heidelberg, pp: 213-219. DOI: 10.1007/978-3-540-28628-8_13

McMullank, P., 2007. An extended implementation of the great deluge algorithm for course timetabling. Proceedings of the International Conference on Computational Science, (CCS' 07), Springer Berlin Heidelberg, pp: 538-545. DOI: 10.1007/978-3-540-72584-8_71

Mignotte, M., 2011. A note on addition chains. Int. J. Algebra, 5: 269-274.

Miller, V.S., 1985. Use of elliptic curves in cryptography. Proceedings of the Conference on the Theory and Application of Cryptographic Techniques, (ACT' 85), Springer Berlin Heidelberg, pp: 417-428. DOI: 10.1007/3-540-39799-x_31

Mohamed, M.A., M.R. Md Said, K.A. Mohd Atan and Z.A. Zurkarnain, 2011. Shorter addition chain for smooth integers using decomposition method. Int. J. Comput. Math., 88: 2222-2232. DOI: 10.1080/00207160.2010.543456

Morain, F. and J. Olivos, 1990. Speeding up the computations on an elliptic curve using addition-subtraction chains. Inform. Theor. Applic., 24: 531-543.

NIST, 2013. Digital Signature Standard (DSS). FIPS 186-4

Nedjah, N. and L.D.M. Mourelle, 2004. Finding minimal addition chains using ant colony. Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Aug. 25-27, Springer Berlin Heidelberg, pp: 642-647. DOI: 10.1007/978-3-540-28651-6_94

Nedjah, N. and L.D.M. Mourelle, 2005. Efficient pre-processing for large window-based modular exponentiation using ant colony. Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Sept. 14-16, Springer Melbourne, Australia, pp: 640-646. DOI: 10.1007/11554028_89

Nedjah, N. and L.D.M. Mourelle, 2006. Towards minimal addition chains using ant colony optimization. J. Math. Model Algor. 5: 525-543. DOI: 10.1007/s10852-005-9024-z

Okeya, K., 2004. Signed binary representations revisited. Proceedings of the 24th Annual International Cryptology Conference, Aug. 15-19, Santa Barbara, California, USA, pp: 123-139. DOI: 10.1007/978-3-540-28628-8_8

Osorio-Hernández, L.G., E. Mezura-Montes, N. Cruz-Cortés and F. Rodríguez-Henríquez, 2009. A genetic algorithm with repair and local search mechanisms able to find minimal length addition chains for small exponents. Proceedings of the IEEE Congress on Evolutionary Computation, May 18-21, IEEE Xplore Press, pp: 1422-1429. DOI: 10.1109/cec.2009.4983110

Park, H., K. Park and Y. Cho, 1999. Analysis of the variable-length non-zero window method for exponentiation. Comput. Math. Applic., 37: 21-29. DOI: 10.1016/s0898-1221(99)00084-x

Pollard, J.M., 1975. A Monte Carlo method for factorization. BIT Numerical Math., 15: 331-334. DOI: 10.1007/bf01933667

Reitwiesner, G. W., 1960. Binary arithmetic. Adv. Comput., 1: 231-308. DOI: 10.1016/s0065-2458(08)60610-5

Rivest, R.L., A. Shamir and L. Adleman, 1978. A method for obtaining digital signatures and public-key cryptosystems, ACM Commun., 21: 120-126. DOI: 10.1145/359340.359342

Robshaw, M.J.B. and Y.L. Yin, 1998. Elliptic curve cryptosystems. RSA Laboratories Technical Note, RSA Data Security, Inc.

Rodriguez-Cristerna, A. and J. Torres-Jimenez, 2013. A Genetic Algorithm for the Problem of Minimal Brauer Chains for Large Exponents. In: Soft Computing Applications in Optimization, Control and Recognition: Studies in Fuzziness and Soft Computing, Melin, P. and O. Castillo (Eds.), Springer, Berlin, Heidelberg, pp: 27-51. DOI: 10.1007/978-3-642-35323-9_2

Scholz, A., 1937. Jahresbericht. Deutschen Mathematiker-vereinigung. Auhfgabe, 252: 41-41.

Schönhage, A., 1975. A lower bound for the length of addition chains. Theor. Comput. Sci., 1: 1-12. DOI: 10.1016/0304-3975(75)90008-0

Stallings, W., 2011. Overview. Cryptography and Networks Security Principle and Practice, Stallings, W. (Ed.), Pearson, New York, pp: 9-29.

Thurber, E. G., 1973. On addition chain l(mn)≤l(n)-b and lower bounds for c(r). Duke Math. J., 40: 907-913. DOI: 10.1215/s0012-7094-73-04085-4

Thurber, E.G., 1993. Addition chains-an erratic sequence. Discrete Math., 122: 287-305. DOI: 10.1016/0012-365x(93)90303-b

Thurber, E.G., 1999. Efficient generation of minimal length addition chain. SIAM J. Comput., 28: 1247-1263. DOI: 10.1137/s0097539795295663

Wiener, M.J., 1990. Cryptanalysis of short RSA secret exponents. IEEE Trans. Infotm. Theor., 36: 553-558. DOI: 10.1109/18.54902

Win, E., S. Mister, B. Preneel and M. Wiener, 1998. On the performance of signature schemes based on elliptic curves. Algorithmic Number Theory, 1423: 252-266. DOI: 10.1007/bfb0054867

Yao, A.C., 1976. On evaluation of powers. SIAM J. Comput., 5: 331-336. DOI: 10.1080/02522667.1992.10699117