Complexity Reduction of the Baby-Step Giant-Step Algorithm

Fahad Alhasoun and M. A. Matin,

Department of Electrical and Computer Engineering University of Denver Denver, USA

Summary
The security of cryptosystems depends on the hardness and difficulty of solving the discrete log problem. One of the well known algorithms to solve the discrete log problem is the Baby-Step Giant-Step Algorithm. This algorithm has $\sqrt{n}$ space and time complexities in the cyclic multiplicative group Z_n^*. In this paper, a modified algorithm is proposed to reduce the space complexity by 50%; the algorithm uses properties of φ function in order to decide what elements are useless for the algorithm. The proposed algorithm of the Baby-Step Giant-Step Algorithm has $\sqrt{n}$ time complexity and $\sqrt{n}/2$ space complexity to find a solution for the discrete log problem in the cyclic multiplicative group Z_n^*. The proposed algorithm can find a solution with time complexity of $\sqrt{n}/2$ if it runs with a space complexity of $\sqrt{n}$ meaning it will run twice as fast as the conventional algorithm with the same space complexity.

Key words:
Cryptosystems, cryptography, discrete log problem, Baby-Step Giant-Step Algorithm, Cyclic Multiplicative Groups

1. Introduction

Information is becoming an asset to everyone as the world is evolving into information technology age. Controlling access to data is of high importance as network systems are booming rapidly around the world. In order to achieve a high level of secrecy for data exchanged across insecure channels, several methods in cryptography are to be used as it offers a great deal of information secrecy. Several encryption schemes are built on the knowledge discovered in a branch of mathematics called group theory; the discrete log problem is one of the major problems in group theory that provide a solid base for several public key cryptosystems [1-2]. In public key cryptosystems, two keys are chosen for a particular cyclic multiplicative group in order to perform exchange of encrypted information; the two keys are known as the private key and the public key [1]. Public keys are exposed to the public while private keys are kept as secretes. For a message to be sent secretly, a sender encrypts data with a public key while the receiver uses the private to decrypt as shown in figure 1 [1].

Manuscript received May 5, 2011
Manuscript revised May 20, 2011

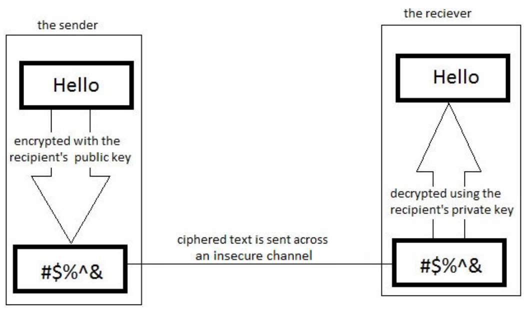
the sender
Hello
encrypted with the recipient's public key
#$%^&
ciphered text is sent across an insecure channel
the reciever
Hello
decrypted using the recipient's private key
#$%^&

Fig.1 Public key cryptosystems.

The security of public key cryptosystems depends on the hardness and difficulty of solving the discrete log problem [1-2]. Given a cyclic multiplicative group Z_n^* where $n \in Z^+$ and $\beta \in Z_n^*$. The discrete log problem is defined as:

Given $g,\ \beta$ and $n,\ x \equiv \log_g \beta \bmod n$ (1)

Where g is a generator in the cyclic multiplicative group Z_n^* [3]. Finding a solution to the discrete log problem attracted the attention of researchers around the word especially after the remarkable work of Whitfeld Diffe and Martin Hellman in 1976 [3]. They came up with Diffie-Hellman key exchange scheme which is presumed to be unbreakable unless the discrete log problem is solved [3]. The trivial approach to finding a solution for the discrete log problem is by doing exhaustive searching testing for all possible powers of the generator; this way is only possible for small integers because it is impossible for exhaustive searching to find a solution of the problem when the integers in the problem are large [4].One of the well known discrete log problem algorithms is the Baby-Step Giant-Step Algorithm. This algorithm was purposed by Daniel Shanks in 1971 [5] and is able to solve the discrete log in $\sqrt{n}$ iterations with a space complexity of $\sqrt{n}$ [6]. This paper provides a modified version of the Baby-Step Giant-Step Algorithm such that the space complexity of the algorithm is reduced by %50. The algorithm runs as fast as the Baby-Step Giant-Step Algorithm with a space complexity of $\frac{\sqrt{n}}{2}$.

2. Mathematical Approach

2.1 Baby-Step Giant-Step Algorithm

The Baby-Step Giant-Step Algorithm is one of the algorithms to find a solution to the discrete log problem. It can solve the discrete log problem for any cyclic multiplicative group [5]. Let us assume that we have a cyclic multiplicative group Z_n^* that has a generator called g. Generators in a multiplicative group produce every element in the group, i.e., every element $x \in Z_n^*$ can be expressed as:

$$x \equiv g^y \ mod\ n \qquad (2)$$

Where y is an integer such that $0 \le y < \varphi(n)$ [7]. The function φ(n) is defined as the number of x elements in Z_n^* that have $x < n$ and $\gcd(n, x) = 1$ [7]. This would imply that the function φ(n) gives the number of elements in a cyclic multiplicative group Z_n^*. Therefore, since every element x within the cyclic multiplicative group Z_n^* can be expressed in terms of a power of the generator g [7], the cyclic subgroup $< g >$ has φ(n) elements and can be expressed as the following:

$$\{ g^0, g^1, \dots\ , g^y, g^{y+1}, g^{y+2}, \dots, g^{\varphi(n)-1}\}$$

The Baby-Step Giant-Step Algorithm has two steps. First, the baby step calculates the values of g^y for $0 \le y < \lfloor\sqrt{n}\rfloor$ and stores the values into a table. After that, the giant step would be able to reach $g^{y \bmod \lfloor\sqrt{n}\rfloor}$ in at most $\lceil\sqrt{n}\rceil$ iterations where $g^{y \bmod \lfloor\sqrt{n}\rfloor}$ would be within the table of the baby step since $y \bmod \lfloor\sqrt{n}\rfloor < \lfloor\sqrt{n}\rfloor$ [1,5]. Therefore, the algorithm needs $\lfloor\sqrt{n}\rfloor$ entries in the table of the baby step and at most $\lceil\sqrt{n}\rceil$ iterations to reach an entry in the baby step table [6]. If we have a cyclic multiplicative group Z_n^* with a generator g and we have the discrete log problem $x \equiv \log_g \beta$. The following illustrates how the Baby-Step Giant-Step Algorithm solves for x.

```
Set $m \rightarrow \lfloor\sqrt{n}\rfloor$;

For $0 \le j < m$
    Calculate $g^j$ and Store $(j, g^j)$ in the baby step table;
End for

Calculate $g^{-m}$ and Set $v \leftarrow g^{-m}\ mod\ n$;

For $i = 0,1,2,3, \dots$ do

    $t \leftarrow \beta v^i\ mod\ n$;
    If $t \equiv g^j$ (for any $j$ in the baby table)
        Return $im + j$;
End for
```

Alg.1 Baby-Step Giant-Step Algorithm [5].

In order to visualize the operation of the algorithm then let us put all g^{im} where $im < \varphi(n)$ on a line.

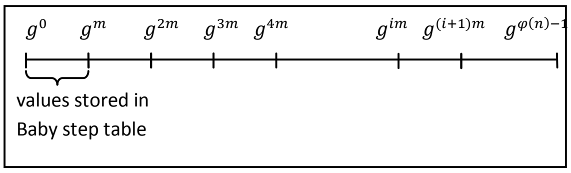

Fig.2 Baby step table values.

Figure 2 shows the values as elements in the cyclic subgroup $< g >$. The baby step computes the values g^x in the subgroup $< g >$ where $x < m = \lfloor\sqrt{n}\rfloor$ and stores both the power and the element generated in the baby table [4]. If we take any random $\beta \equiv g^x$, then the giant step would behave as shown in figure 3 below.

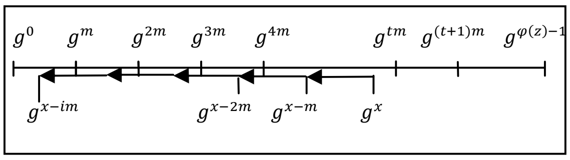

Fig.3 Giant step iterations.

Figure 3 shows how the giant step aims to move $\beta \equiv g^x$ backward towards a value that is less in the power than g^m, i.e., g^{x-im} where $x - im < m$ [6,8]. The giant step moves g^x backward towards the baby step table by reducing its power every iteration by a factor of $(-m)$ [6,8]; this ensures that for every value of x, g^{x-im} will eventually hit a value in the baby step table [5], a hit occurs when $g^{x-im} \equiv g^j$, which means that $j = x - im$. Therefore, the algorithm returns the value $im + j = im + (x - im) = x$. The Baby-Step Giant-Step Algorithm lacks the intelligence of where the algorithm can be modified in a way where the input problem is altered to reduce space or computational complexities [9].

2.2 Modified Baby-Step Giant-Step Algorithm

The proposed Modified Baby-Step Giant-Step Algorithm relies on the fact that the function φ always produces even integers. To prove φ always produces even integers, assume we have a composite integer n , then φ(n) can be expressed as:

$$\varphi(n) = \varphi\left(p_1^{x_1} * p_2^{x_2} * \dots * p_n^{x_n}\right)$$

$$= p_1(p_1 - 1)^{x_1-1} * p_2(p_2 - 1)^{x_2-1} * \dots *$$

$$pnpn-1xn-1 \qquad (3)$$

Where $p_1, p_2, \ldots., p_n$ are the prime factors of n and $x_1, x_2, \ldots, x_n \in Z$ [7]. φ is even because

$$\varphi(n) = p_1(p_1-1)^{x_1-1} * p_2(p_2-1)^{x_2-1} * \ldots * p_n(p_n-1)^{x_n-1}$$

$$= (p_1-1) * [p_1(p_1-1)^{x_1-2} * p_2(p_2-1)^{x_2-1} * \ldots * p_n(p_n-1)^{x_n-1}] \quad (4)$$

For any prime p, $gcd(p,x) = 1$ for any $x < p$ [10]. Therefore, we can say that for $x = 2$, $gcd(p,2) = 1$ meaning $p = 2n + 1$ for some $n \epsilon Z$.
And this means that $2 \mid p - 1$ because:

$$p - 1 = 2n + 1 - 1 = 2n \quad (5)$$

Therefore, $p - 1$ is even for every prime p. Back to (4), we can apply the finding in (5) to the prime factor p_1 expressing

$$p_1 - 1 = 2n \text{ for some } n \epsilon Z \quad (6)$$

Since p_1 is a prime factor of n. $\varphi(n)$ can be expressed as

$$\varphi(n) = (p_1-1) * [p_1(p_1-1)^{x_1-2} * p_2(p_2-1)^{x_2-1} * \ldots * p_n(p_n-1)^{x_n-1}]$$

Applying equation (6) we get

$$\varphi(n) = 2n * [p_1(p_1-1)^{x_1-2} * p_2(p_2-1)^{x_2-1} * \ldots * p_n(p_n-1)^{x_n-1}] \quad (7)$$

Equation (7) shows that $\varphi(n)$ is always a multiple of 2 which implies:

$$\varphi(n) \text{ is even } \forall n \in Z. \quad (8)$$

If we know that for a cyclic multiplicative group Z_n^*, with a generator g, where $\varphi(n)$ is even as shown in (8), then we can reduce the size of the baby step table by 50%. Assume we want to find x in the following discrete log problem:

$$x \equiv log_g \beta \quad (9)$$

The modified algorithm will initially square β in the following equation $x \equiv log_g \beta$ then solve the problem of:

$$2x \equiv log_g \beta^2 \quad (10)$$

When solving for 2x in congruence (10), we know that $\beta^2 \equiv g^{2x}$, and therefore when generating the baby step table, we do not need to store any entry where the power is odd, i.e., $g^i \equiv g^{2s+1}$ where $s \in Z$ because we know β^2 is of an even power 2x to the base g. Ensuring the power is even can help us reduce the baby step table as shown in the following modified algorithm.

```
Set $m \to \lfloor\sqrt{n}\rfloor$;
If ($m\ mod\ 2 \equiv 1$)
    $m = m - 1$; m has to be even

For $0 \le j < m/2$
Calculate $g^{2j}$ and Store $(j, g^{2j})$ in the baby step table;
End for

Calculate $g^{-m}$ and Set $v \leftarrow g^{-m}\ mod\ z$;

Set $\partial \equiv \beta^2$
For $i = 0,1,2,3, \ldots.$ do

Set $t \leftarrow \partial v^i\ mod\ n$;
If $t \equiv g^{2j}$ (for any j in the baby table) meaning we hit
a value

If ($\beta \equiv g^{im+2j}$)
Return $(im + 2j)$;
Else
 Return $(im + 2j) + (\varphi/2)$;
        End if
 End if
End for
```

Alg.2 Modified Baby-Step-Giant-Step algorithm.

It is mandatory that φ produces an even integer as proven in (8). Otherwise, β^2 can be of an odd power in terms of g which makes the modified algorithm unable to find the power of g. We can prove that β^2 is always of an even power in terms of g by expressing it as:

$$\beta^2 \equiv (g^x)^2 \equiv g^{2x\ mod\ \varphi} = g^{2x-b\varphi} \text{ for some } a,b \in Z \text{ where } 2x - b\varphi < \varphi \quad (11)$$

By the finding in (8) we know that φ is always even meaning $\varphi = 2i$ for some $i \in Z$. The power $2x - b\varphi$ can then be expressed as:

$$2x - b\varphi = 2x - b(2i) = 2(x - bi) \qquad (12)$$

Which implies:

$$\beta^2 \equiv g^{2(x-bi)} \text{ where } 2(x - bi) < \varphi \qquad (13)$$

proving β^2 is always of an even power in terms of g. The key to reducing space complexity in the baby step table is the prior knowledge of the nature of the power 2x in $\beta^2 \equiv g^{2x}$. Squaring the power makes the power even and will help us create a smaller baby step table. In order to apply the purposed algorithm, let us consider Z_{41}^* where $g = 6$ and $\beta = 12$ then we can write the discrete log problem as:

$$x \equiv log_6 12 \qquad (14)$$

By using our modified algorithm, equation (14) can be solved as:

$m \rightarrow \lfloor\sqrt{41}\rfloor = 6$

$\frac{m}{2} = 3$ meaning we will create a baby table of 3 entries.

Table.1 baby table of the modified algorithm.

j	0	1	2
g^{2j}	1	36	25

$\partial \equiv \beta^2 \equiv 12^2 \ mod\ 41 \equiv 21$

this step makes sure the power is even as shown in (13).

$v \equiv g^{-m} \equiv 6^{-6} mod\ 41 \equiv 20$ the obtained value is used to reduce the power by $(-m)$ per iteration.

Table.2 The giant step iterations.

i	0	1	2
$\partial v^i \equiv \beta^2 * g^{-im}$	21	10	36

From table 2, we can see that on the third iteration in the giant step, i.e., $i = 2$, the value of $t \equiv \partial v^i \equiv 36$ which is the same value found in table 1 where $j = 1$. This implies that $\beta^2 * g^{-im} \equiv g^{2j}$. Hence, the solution of the algorithm will become as follows:

$x = [\frac{(im+2j)}{2} + (\varphi/2)\,] = [\frac{(2*6+2*1)}{2} + (40/2)]$ as $(\beta \neq g^{(im+2j)})$.

The returned value of $x = 27$. The correct answer is proved by computing $6^{27} mod\ 41 \equiv 12$.

2.3 Analysis of the Modified Algorithm

To show how the modification of the algorithm reduces the size of the baby table of Shank's Algorithm, let us consider:

$$x_s \equiv log_6 \beta \text{ in } Z_{41}^* \qquad (15)$$

where $(g = 6)$ is a generator in Z_{41}^*. To show the difference between the conventional algorithm 1 and our modified algorithm 2, the plot of all solutions to congruence (15) is shown in figure 4.

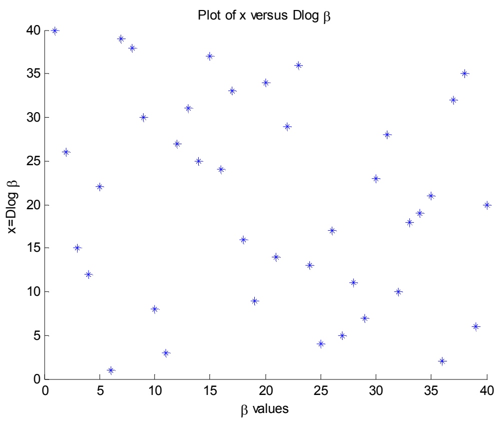

Fig.4 Scatter plot of $x \equiv \log_6 \beta$

Figure 4 shows all the possible solutions (β, x), for the congruence (15). For instance, for any point (β, x) on the plane, we have

$$6^x \equiv \beta \ mod\ n \qquad (16)$$

The conventional Baby-Step Giant-Step Algorithm stores all the values of $g^x \equiv 6^x$ where $x < \lfloor\sqrt{41}\rfloor$, i.e., all the pairs $(x, 6^x)$ located inside the rectangle in figure 5 are the entries of the conventional baby step table.

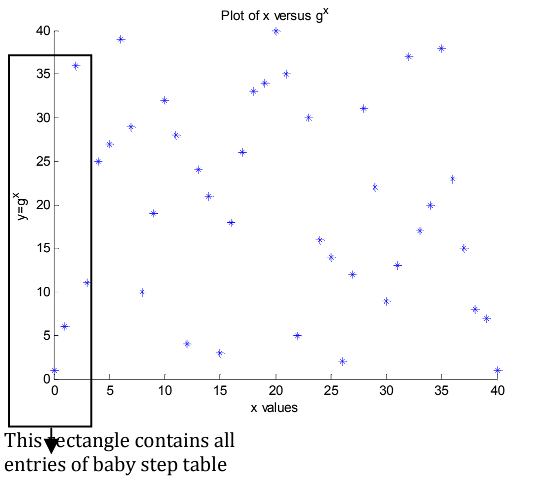

Fig.5 Scatter plot of x versus 6^xmod 41

The modified Baby-Step Giant-Step Algorithm stores half of the values inside the rectangle shown on figure 5. This is because β^2 is known to hit a pair $(x, 6^{2x})$ on the plane where the power is even meaning we can discard all the pairs $(x, 6^x)$ where x is odd. The plot of the values of 6^{2x} in the modified algorithm is shown in figure 6.

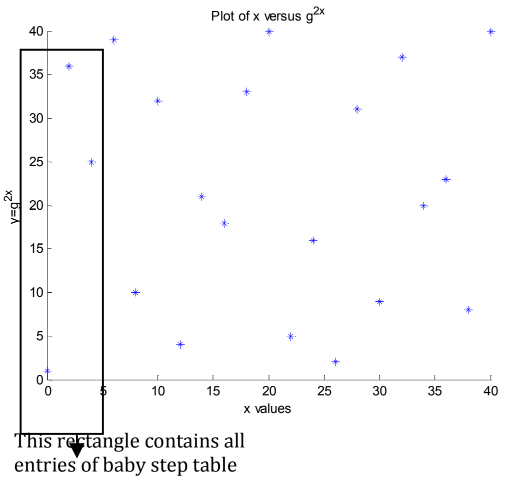

Fig.6 Scatter plot of x versus 6^{2x}mod 41

The pairs $(x, 6^x)$ inside the rectangle on figure 6 are the pairs that are going to be stored in the baby step table of the modified algorithm. We can see that half of the values stored in figure 5 were useless to the algorithm when squaring $\beta \equiv g^{x_s}$.

2.4 Speeding Up the Modified Algorithm

In this paper, we also examine speeding up our modified algorithm 2. The modified Baby-Step Giant-Step Algorithm can run twice as fast as the conventional Baby-Step Giant-Step Algorithm with a space complexity of $\sqrt{n}$. Algorithm 1 would need $2\lfloor\sqrt{n}\rfloor$ entries in the baby table to run as fast as our faster modified algorithm 2. The modified algorithm mainly changes the giant step to reduce the power of g^x by a factor of $(-2m)$ in every iteration rather than $(-m)$. It is important to take into consideration that there is a trade off relation between space complexity and time complexity. The following is the Faster Modified Baby-Step Giant-Step Algorithm where we can find a solution with time complexity of $\sqrt{n}/2$.

```
**Set** $m \rightarrow \lfloor\sqrt{n}\rfloor$;

**If** $(m \bmod 2 \equiv 1)$
  $m = m - 1$; m has to be even

**For** $0 \le j < m$
    Calculate $g^{2j}$ and **Store** $(j, g^{2j})$ in the baby step table;
**End for**

Calculate $g^{-2m}$ and **Set** $v \leftarrow g^{-2m} \bmod n$;

**Set** $\partial \equiv \beta^2$
**For** $i = 0,1,2,3, \ldots$ **do**

**Set** $t \leftarrow \partial v^i \bmod n$;
**If** $t \equiv g^j$ (for any j in the baby table) meaning we hit a value

        **If** $(\beta \equiv g^{(i(2m)+2j)})$
               **Return** $(i(2m) + 2j)$;
        **Else**
               **Return** $(i(2m) + 2j) + (\varphi/2)$;
        **End if**
 **End if**
**End for**
```

Alg.3 Faster Modified Baby-Step Giant-Step Algorithm.

By enlarging the baby step table to be of size $\lfloor\sqrt{n}\rfloor$, we are certain that g^{x-im} would hit an entry in the baby step table in $\lceil\sqrt{n}/2\rceil$ steps. This is valid because we are reducing the power of g^{2x} in multiples of $(-2m)$ rather than $(-m)$. We are certain that g^{x-im} would hit a value in the baby step table for some value $i < \lceil\sqrt{n}/2\rceil$, where i represents the number of iterations the algorithm needs to iterate to reach a hit in the baby table. To prove that i is less than $\lceil\sqrt{n}/2\rceil$ let us solve for the largest possible power value which is $\varphi - 1$. Then, the discrete log problem becomes as follows:

$$x \equiv \log_g g^{\varphi-1} \quad (17)$$

Equation (17) implies that the algorithm will return the value of $\varphi - 1$ as

$$\varphi - 1 = 2im + 2j \quad (18)$$

It is possible to have $\varphi - 1$ as:

$$\varphi - 1 = 2im + 2j + (\varphi/2) \quad (19)$$

Since we know that $2m = 2\lfloor\sqrt{n}\rfloor$, then the number of iterations i in (18) is:

$$i = \frac{\varphi - 2j - 1 - (\varphi/2)}{2m} < \frac{\varphi - 2j - 1}{2m}$$

$$< \frac{\varphi - 2j - 1}{2\lfloor\sqrt{n}\rfloor} < \frac{\varphi - 2j}{2\lfloor\sqrt{n}\rfloor + 1} \leq \frac{\varphi - 2j}{2\sqrt{n}}$$

$$< \lceil\sqrt{n}/2\rceil \quad (20)$$

While the value of i in (19) is

$$i = \frac{\varphi-2j-1-(\varphi/2)}{2m} < \frac{\varphi-2j-1}{2m} < \lceil\sqrt{n}/2\rceil \quad (21)$$

Therefore, when x is largest, g^x would take less than $\lceil\sqrt{n}/2\rceil$ iterations to hit an entry in the baby step table as proven in (20) and (21).

3. Conclusion

One of the famous algorithms to find a solution to the discrete log problem is the Baby-Step Giant-Step Algorithm. We have shown how Baby-Step Giant-Step Algorithm can be modified to perform better in terms of space complexity or time complexity. The space complexity was reduced by 50%. We have shown an important trade off relationship between space complexity and time complexity in the Baby-Step Giant-Step Algorithm. Furthermore, we have shown how the modified algorithm can run twice as fast as the conventional algorithm with a space complexity of $\sqrt{n}$ where the conventional algorithm uses $\sqrt{n}$ space complexity too. Our modified algorithm can find the solution to the discrete log problem with time complexity of $\sqrt{n}/2$ when it runs with a space complexity of $\sqrt{n}$. This does not solve the discrete log problem but rather modifies the Baby-Step Giant-Step Algorithm by the use of φ function properties in order to decide what elements are useless for the algorithm. Finally yet importantly we have proven that it is possible to improve the Baby-Step Giant-Step Algorithm by means of altering the inputs to the algorithm to reduce either space or time complexities.

References

[1] Agnes Ivy Bimpeh Harrison, “Square-Root Methods for the Discrete Logarithm Problem,” African Institute for Mathematical Sciences (AIMS), 2010.

[2] Pantelimon George Popescu, Sanda Osiceanu, “The discrete logarithm problem in cyclic subgroups of not necessary cyclic groups,” International Conference on Information Technology, IT, 2008.

[3] Whitfeld Diffe and Martin Hellman, “New directions in cryptography,” IEEE Transactions onInformation Theory 22, no. 6, 644-654, 1976.

[4] Jun Zhang, LiQun chen, “An Improved Algorithm For Discrete Logarithm Problem,” IEEE, 2009.

[5] Daniel Shanks, Class number, a theory of factorisation and genera, Symposia in Pure Mathematics (Donald J. lewis, ed.), Lecture Notes in Computer Science, no. 20, pp. 415-440 American Mathematical Society (AMS), 1971.

[6] A. V. Sutherland, “Order computations in generic groups, PhD thesis, " M.I.T., 2007.

[7] Hans Riesel, “Prime numbers and computer methods for factorization,” Birkhäuser, 1994.

[8] Andreas Stein, and Edlyn Teske, “Optimized Baby Step-Giant Step Methods,” Journal of the Ramanujan Mathematical Society 20, 2005.

[9] Chakravarthy Bhagvati, R.Padmavathy, “Methods to solve Discrete Logarithm Problem for Ephemeral Keys,” IEEE, 2009.

[10] Cohen H, “A Course in Computational Algebraic Number Theory,” SpringerVerlag, 1993.

Fahad Alhasoun received bachelor degree in Computer Science (CS) in 2010 from King Fahd University of Petroleum and Minerals (KFUPM) in Saudi Arabia. He is interested in research related to discrete mathematics and computer networks including algorithms designs and optimization.

Dr.Mohammad Abdul Matin, Associate Professor of Electrical and Computer Engineering, in the School of Engineering and Computer Science, University of Denver. He is a Senior Member of IEEE and member of SPIE, OSA, ASEE and Sigma Xi. His research interest is in Optoelectronic Devices (such as Sensors and Photovoltaic) RoF, URoF, Digital, Optical & Bio-Medical Signal & image Processing Engineering Management and Pedagogy in Engineering Education."