ANALYSIS OF DATA SECURITY IN CLOUD COMPUTING USING MapReduce WITH ECDLP BASED ALGORITHMS

M.Subhashini, Dr.P.Srivaramangai
Ph. D.Research scholar Department of Computer Science, Maruthu pandiyar college of Arts & Science, Thanjavur
Professor, Department of Computer Science, Maruthu pandiyar college of Arts & Science , Thanjavur.

Abstract: Elliptic Curve Cryptography (ECC) is capable of constructing public-key crypto-systems. Specifically, the security of the ECC minimizes to testing the ability to handle and solve the DLP (Discrete Logarithmic Problem) in the group of points of an elliptic curve (ECDLP). ECC based on ECDLP is in the list of recommended algorithms for use by NIST (National Institute of Standards and Technology) and NSA (National Security Agency). Given that ECDLP based cryptosystems are in wide-spread utilization, continuous efforts on monitoring the effectiveness of new attacks or improvements to pre-existing attacks on ECDLP over large prime factor is a significant part in that. This paper aims to provide a secure, effective, and flexible method to improve data security in cloud computing. We present a novel algorithm to solve the ECDLP problems using MapReduce and enhance the security level. Here, the comparative results were analyzed and implemented for ECDLP to assess the security of ECC using different methods such as Pollard's Rho Method, Baby-Step, Giant-Step Algorithm, MapReduce, and Pohlig-Hellman Attack, and also effectually choose security parameters.

Keywords: Elliptic Curve Cryptography (ECC), Elliptic Curve Discrete Logarithmic Problem (ECDLP), Pollard's Rho Method, Baby-Step, Giant-Step Algorithm, Pohlig-Hellman Attack, MapReduce for ECDLP.

I. INTRODUCTION

The security of ECC depends upon a trapdoor function (compute in one direction) so that assumed the discrete logarithm of a random elliptic curve element similar to a publicly known base point infeasible state. This is known as ECDLP (Elliptic Curve Discrete Logarithm Problem) to be computationally infeasible to solve [1]. The difficult part of the DLP is based on the representation of the group of points in ECDLP. The two most popular finite groups are used for discrete logarithm problems are the group of points on an elliptic curve (EC) over a finite field, represented by E ($\mathbb{F}_p$) and the multiplicative group ($\mathbb{Z}/\mathbb{Z}p$)* of integers modulo denoted as a prime factor p. For solving the ECDLP: Pohlig-Hellman algorithm (which reduces the problem to subgroups of prime order), Shanks' baby-step-giant-step method [2], both the rho method and the kangaroo method have parallel versions because of van Oorschot and Wiener, MOV (Menezes-Okamoto-Vanstone) attack using the Weil pairing, Frey-Rueck attack using the Tate pairing, The attacks on anomalous elliptic curves due to Semaev, Satoh-Araki and Smart, Weil descent (for some special finite fields) [3]. The baby-step-giant-step algorithm needs large storage for computation and it is difficult to distribute or parallelize over the internet. The complexity of this method is approximately denoted as O (√n) and also it the giant steps require to the maximum amount for computation. The main complexity of the baby step method wants a huge amount of storage capacity for computation| √n | group elements [4]. The baby-step-giant-step method is to solve the issue because the mathematical operations are compared less than other techniques. **Pollards Rho** algorithm has a parallel running time to the Baby-Step Giant-Step technique. The rho and kangaroo algorithms need less storage space and can be distributed. **Pollards Rho** algorithm exploits the parallel technologies such as Cluster computing, GPGPU, and so on to solve the ECDLP [5]. The Pohlig-Hellman algorithm is an important deterministic algorithm and also computes discrete logarithms in O (√p) operations whereas p represents the largest prime factor of n, which is used for computing discrete logarithms in a multiplicative group whose order is a smooth integer. Pohlig-Hellman algorithm considers as a recursive algorithm that decreases the problem by computing discrete logarithms in the prime order subgroups of (P) [6]. The time complexity of Pohlig-Hellman algorithms is O (√n) for a group of order n. For this algorithm, the expected running time evaluates $o\sqrt{q_i'}$ in that q'$_i$ is the largest prime factor of p − 1 that can select the value of p so that p − 1 considers as a large prime factor.

But, the existing techniques perform and focus on improving a certain part of the process, for example, better random point's generation method and faster elliptic curve points operations. There is a lack of system-level study including scalability with real computation infrastructure. Even if the vast amount of computation/storage resources makes use of processing the parallel collision search, and also it has a big challenge to control them effectively and prevent all potential failures/ errors of the processing procedure, to solve ECDLP using MapReduce. MapReduce is the de-facto industry standard for big data applications and verified as a scalable framework for huge data processing. In this paper, a comparative analysis of these Pollard's methods, Pohlig-Hellman, baby-step-giant-step method, and proposed MapReduce based algorithms are carried out for solving the ECDLP problems.

1.1 Background Study (doubt)

Elliptic curve cryptography is powerful. Calculating the public key from a known private key and base point can be handled easily. On the other hand, extracting the private key from known public key and base point is not an easy task. This is called an Elliptic Curve Discrete Logarithm Problem. The security of elliptic curve cryptography, which is based on the computational hardness of ECDLP, has been extensively studied for decades. The size of the elliptic curve determines the difficulty of the problem. Therefore the elliptic curve E and the base point P have to be chosen carefully.

The discrete log problem is believed to be hard compared to the exponentiation problem, and the elliptic curve discrete logarithm problem is even harder. This is because of its different algebraic structure, it's complex arithmetic rules to "add" two points on an elliptic curve, and the lack of an index calculus method for the elliptic curve domain. The main reason why ECDLP is "more trusted" than DLP or FP is because DLP and FP can be solved in subexponential time with index calculus algorithms. Until recently, no similar result had been obtained for ECDLP: except for exceptional curves and somewhat non-natural families of parameters, the time required to solve was believed to be exponential in the size of the parameters. Schemes and protocols such as the Diffie-Hellman key exchange, Massey-Omura encryption, El-Gamal public-key encryption, and El-Gamal digital signatures and even the Elliptic Curve Digital Signature Algorithm(ECDSA), all use the fact that attempting to solve the ECDLP is a difficult, if not intractable, problem. For example, notice that the security of this system does not rely at all on Alice and Bob finding a secure way to transmit information, but it relies very heavily on Alice and Bob each having private keys that are very, very difficult to retrieve using only their public keys. Eve can only be thwarted if the information that she can intercept is useless. This brings us to the elliptic curve discrete logarithm problem.

Currently, no such algorithm is known so cryptographic systems based on ECC provide a high level of security with relatively small key sizes. However, as we will see, the complicated calculations make ECC somewhat less effective. Note that elliptic curves are not the only mean to create groups where the DLP is hard. The elliptic curve is just the first member of a bigger family of groups, defined as a group structure on the Jacobi surface of specific curves. The next one is called hyperelliptic. But it seems that these constructions do not add as much insecurity as in complexity.

On the other hand, the ECDLP can be efficiently solved on a quantum computer. Of course, it is well known that the integer factorization and the ordinary discrete logarithm problems can also be efficiently solved on a quantum computer [59]. Thus, if large-scale quantum computers are ever built, then all the major families of public-key systems (DL, RSA, ECC) will be insecure.

In cryptography, an attack is a method of solving a problem. Specifically, an attack aims to find a fast method of solving a problem on which an encryption algorithm depends. The known methods of attack on the elliptic curve (EC) discrete log problem that works for all curves are slow, making encryption based on this problem practical. However, several efficient methods for solving the EC discrete log problem for specific types of elliptic curves are known. This means that one should make sure that the curve one chooses for one's encoding does not fall into one of the several classes of curves on which the problem is tractable. When a huge amount of computation/storage resources are used for the existing methods, it is also a big challenge to manage them efficiently and avoid all potential errors/failures of the processing procedure. To fulfill these above gaps, we design an algorithm to solve ECDLP using MapReduce.

1.2 Mapreduce Framework

The MapReduce framework contains in variations namely Map step and reduces step. In Map Step, the master node obtains large problem from the input which splits into smaller sub-problems and then distributes it into worker nodes. A worker node may repeat this and leads to a multi-level tree structure. Worker node processed smaller issues and then it provides back to master node. In Reduce step, the Master node obtains the answers to the subproblems and merges into a predefined method while obtain the output/answer to identify the original problem. The MapReduce framework solves fault-tolerant since every node in the cluster is expected toward periodically report back with status updates and completed work. When a node remains silent for processing longer than the expected interval, a master node formulates note and re-allocates the task to other nodes.

II. ALGORITHMS FOR ECDLP

Algorithms for ECDLP are classified as it follows: generic algorithms applicable to all instances of ECDLP, special algorithms acquire the benefit of a particular instance of the problem. While all special attacks to the ECDLP recognize how to be easily prevented by a suitable option of the parameters that have to focus on generic algorithms. In the variations of generic algorithms includes Pohlig-Hellman, Baby-Step Giant-Step, Pollard's rho method, and MapReduce for ECDLP. In this phase, the following attacks work on arbitrary groups including elliptic curve and these attacks can be applied for solving ECDLP.

a. Pollard's Rho Method

The Pollard concept is to find k which is satisfying Q = [d]P by isolating the group of points on an elliptic curve into three

disjoint sets S_1, S_2 and S_3 in an equal size [7]. Define the original iteration function on a point R as follows:

$$f(R) = \begin{cases} R \oplus P, & if\ R \in S_1, \\ [2]R, & if\ R \in S_2, \\ R \oplus P, & if\ R \in S_3, \end{cases}$$

To begin with a point $R_0 = [a_0]\ P \oplus [b_0]Q$ where a_0 , $b_0 \in [1, n - 1]$ are arbitrarily selected and make a sequence R_i by using this function until the collision takes place. The sequence R_i can be expressed in the term of $a_i]P \oplus [b_i]Q$, where the numbers $a_i, b_i \in [1, n - 1]$ are computed as follows:

$$a_{i+1} = \begin{cases} (a_i + 1) mod\ n, & if\ R_i \in S_1, \\ 2a_i\ mod\ n, & if\ R_i \in S_2, \\ a_i, & if\ R_i \in S_3, \end{cases} \quad \text{And} \quad b_{i+1} = \begin{cases} b_i, & if\ R_i \in S_1, \\ 2b_i\ mod\ n, & if\ R_i \in S_2, \\ (b_i + 1) mod\ n, & if\ R_i \in S_3, \end{cases}$$

Given that the number of point's lies on the elliptic curve which form cyclic group is a finite field, this sequence does not become periodic update after applying this function but will start to repeat. Upon detection of a matching, this is $R_i = R_j \oplus [a_i]P \oplus [b_i]Q = [a_j]P \oplus [b_j]Q$, if gcd $(b_j - b_i, n) = 1$, d can be obtained as follows:

$$d = log_p\ Q = \left(\frac{a_i - a_j}{b_j - b_i}\right) mod$$

b. Baby-Step, Giant-Step Algorithm

This attack utilizes a combination of memory storage and computational power to solve the DLP. Let G be a cyclic group with generator α. Assume that α has order n and set $m = \lceil\sqrt{n}\rceil$. Observe that if $\beta = a^x$, then using the euclidean algorithm x can write as follows: $x = ip + j$, where $0 \leq i, j < p$. Thus algorithm have that $\beta = a^x = a^{ip+j} = a^{ip}a^j$, which implies that $\beta\ (a^{-p})^i = a^j$. To compute the discrete logarithm, this method begins by storing and computing the values (j, a^j) for $0 \leq j \leq p$. After that compute $\beta\ (a^{-p})$ and increase that to the exponent i for $0 \leq j \leq m - 1$ and check these values against the stored values of a^j to find a match. When a match is discovered and then the DLP solved and have $x = ip + j$ as required. The negative aspects of this algorithm lie in the computation and formulation of the table of pairs (j, a^j). At each phase needed to compute power of α and look in the table to see if it returns a match. If this is successful then the DLP has been solved. Unfortunately, one has to store around $O(\sqrt{n})$ group elements, perform around $O(\sqrt{n})$ multiplications to find the correct power of α, and in turn perform $O(\sqrt{n})$ table look-ups [8]. As an outcome of this technique has an expected running time of $O(\sqrt{n})$ and that makes it impractical for cryptographic intentions.

c. Pohlig-Hellman Attack

The Pohlig-Hellman algorithm was devised by Stephan C. Pohlig and Martin E. Hellman in 1978. It has an improved algorithm for computing discrete logarithms over the cyclic field G = GF(p), and their findings collision on ECC[9]. Given that the ECDLP Q = lP, Pohlig-Hellman algorithm considers as a recursive algorithm and that has to reduce the problem by computing discrete logarithms in the prime order subgroups of (P). Each of these smaller subproblems can be solved using methods, such the Pollard's rho algorithm works as follows:
Let n be computes $n = p_1^{e_1} p_2^{e_2} \ldots . p_r^{e_r}$

Calculate $l_i = l\ mod\ p_i^e$ for all $1 \leq i \leq r$.

The CRT (Chinese Remainder Theorem) is used to obtain a unique solution to the following system of congruences:

$$l = l_1\ (mod\ p_1^{e_1}), l = l_2\ (mod\ p_2^{e_2}) \ldots \ldots . l = l_r\ (mod\ p^{e_r})$$

At this point $p_1, p_2, \ldots . , p_r$ is a mutually co-prime as a set of positive integers, i.e. gcd (p_i, p_j). = 1 for all i = j. $l_1, l_2, \ldots . , l_r$ are all positive integers such that $0 \leq l_i \leq p_i$. The unique positive integer l can be analyzed efficiently by using the Extended Euclidean Algorithm to compute the linear congruence's above [10]. Each l_i can be expressed in base p the following way:

$$l_i = z_0 + z_1 p + z_2 p^2 + \cdots + z_{e-1} p^{e-1}$$

Where $z_i \in [0, p - 1]$. Let

$Q_0 = \frac{n}{p_i}$ And $Q_0 = \frac{n}{p_i} Q$

The above equation rewrites the fact that the order of p_0 is p, get $Q_0 = lP_0 = z_0P_0$ Next z_0 is found by computing the ECDLP solution in (P_0). Every $z_{0,\ldots..,} z_{e-1}$ is then computed by solving $Q_i = z_iP_0$ in (P_0) [11, 12].

III. PROPOSED METHODOLOGY

The selection of the iteration function and the distinguishing property affects the performance of the algorithm. Also using specialized libraries to perform multi-precision operation helps in achieving good speedup. This paper discusses the background of elliptic curve DLP in which finite fields are the basic underlying fields on which the cryptosystem is built. The solvability of the elliptic curve cryptosystem reduces to the solvability of the ECDLP problem in that group. Thus different attacks on the ECDLP have been studied. Dealing with the ECDLP problem based on the large underlying finite field requires large computational resources. Using MapReduce based ECDLP algorithm on the CPU cluster is a

better approach to such instances. Solving ECDLP based on the large finite field requires huge resources and special algorithm (or hardware) have to be considered to perform arithmetic‟s on large integers. Making the use of arithmetic libraries helps in such scenarios. The MapReduce based ECDLP algorithm can be modified to run on a hybrid cluster of openMP. Also, selecting proper iteration function and proper distinguished property as per the parallel platform can result in good speedup.

3.1 ALGORITHM

In this section, we will describe proposed methods based on the difficulty of the discrete log problem for elliptic curves.

a. Choosing a Curve

For each of the cryptographic methods depended on the difficulty of the EC discrete log problem, we must begin by choosing an elliptic curve that is not susceptible to the known fast attacks on the discrete log problem. The curve must, therefore, satisfy the following restrictions:

- There exists a large prime p dividing #E ($\mathbb{F}_p$).
- #E ($\mathbb{F}_p$) ≠q (i.e. the curve is not anomalous).
- The order of P does not divide q^k-1 for all k such that $1 \leq k \leq C$, where C is a sufficiently large constant so that it is difficult to solve the discrete logarithm problem in $\mathbb{F}_p{}^x C$.

b. MapReduce for ECDLP

Mapreduce framework performs with data processing works in the variations of Map phase and reduce phase. To fit parallel collision search into MapReduce, in this algorithm also divide the ECDLP solving process into two steps. In MapReduce based approach, both of the two steps are carried out on distributed computation nodes in a parallel manner.

Map Phase: Each Mapper begins from a different point and a distinguished point in a random way to be searched. For mapper i, it runs a random number generator to obtain two random numbers $a_0, b_0 \in \mathbb{Z}_p$ and calculates the start point $X_0 = a_0P + b_0Q$. After that mapper, i start the iteration process to search for a distinguished point. Different approaches have been developed for the iteration process, this algorithm follows Pollard's approach for the iteration process [13]. The group (P) is separated into three subsets S_1, S_2, and S_3 with parallel size, and mapper i run the iteration according to the below equation:

$$\begin{cases} X_{j+1} \leftarrow Q + X_j, a_{j+1} \leftarrow a_j, b_{j+1} \leftarrow b_j + 1, & \text{if } X_j \in S_1 \\ X_{j+1} \leftarrow 2X_j, a_{j+1} \leftarrow 2a_j, b_{j+1} \leftarrow 2b_j, & \text{if } X_j \in S_2 \\ X_{j+1} \leftarrow P + X_j, a_{j+1} \leftarrow a_j + 1, a_{j+1} \leftarrow b_j, & \text{if } X_j \in S_3 \end{cases}$$

The membership judgment has to simplify and use for Hamming weights of points to divide (*P)* into S_1, S_2, and S_3:

$$S_k = \{T \in (p) \,|(hw(T) mod\ 3) = k - 1\}, k = 1,2,3.$$

The sum of the Hamming weights of *T*'s measures two coordinate values using *hw*(*T*), and if *T* = *O*, *hw*(*T*) is set to be 0. At the end of j^{th} iteration (*j* = 1, 2, . . .), *mapper i* checks whether X_{j+1} is a distinguished point.

As a collision search job will be detected with distinguished points, the more distinguished points are collected so that collision search job will be detected. This algorithm phase utilizes the Hamming weight of an elliptic curve point to determine whether it is a distinguished point or not. If the Hamming weight is less than *z*, classify it as a distinguished point. The parameter *z* can be tweaked under available sources. If X_{j+1} is a distinguished point, *mapper i* outputs a key-value pair in the following form

$$\left[h(X_{j+1}), (X_{j+1}, a_{j+1}, b_{j+1})\right],$$

This procedure generates and continuously repeats until a new random start point. Here *hw*(*T*) is a pre-defined function for *key* generation, and (X_{j+1}, a_{j+1}, b_{j+1}) is the *value*. If X_{j+1} is not a distinguished point, *mapper i* continue to the next iteration process accordance to given Equation.

Reduce phase: Reducers were taken responsible for storing generated distinguished points and detection of collisions among these points. To compare with mappers function and their computation burden is much lesser. After receiving a key-value pair [H(X), (X, c, d)], reducer i saves the triple (X, c, d) to the storage system and checks whether there is a pre-existing triple (X', c',d') such that X = X'. If such a collision is detected, reducer i can evaluate the value of *l* as

$l \leftarrow \frac{d-d'}{c'-c} \ mod\ p$, and immediately terminates the whole collision search job.

c. Diffie Hellman Key Exchange

The following series of steps describes the Diffie Hellman Key Exchange

- Alice and Bob publicly agree on E ($\mathbb{F}_p$).), chosen so that the discrete log problem is hard, as described above. They also agree on point P ϵ E ($\mathbb{F}_p$).) of high (usually prime) order.
- Alice chooses a secret a ϵ ℤ, computes aP, and sends it to Bob.
- Bob chooses a secret b ϵ ℤ, computes bP , and sends it to Alice.
- Alice computes a(bP) = abP .
- Bob computes b(aP) = abP .

- Alice and Bob now have the same point abP. They use a publicly agreed on method, such as taking the last 256 bits of the y-coordinate of the point, to extract a key.

d. Message Encryption

Message m is considered as a point P_m with coordinates (x, y) in the elliptic curve P_m. The point P_m is encrypted as a ciphertext C_m and subsequently decrypted.

To encrypt and send a message P_m to Bob, Alice chooses a random positive integer k and produces the ciphertext C_m consisting of the pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

kP_B is a public key of bob.

e. Message Decryption

Bob decrypts the ciphertext by multiplying the first point in the pair by Bob's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG)$$

$$= P_m + k(n_B G) - n_B(kG)$$

$$= P_m$$

Alice has masked the message P_m by adding kP_B to it.

III. EXPERIMENT RESULT

The implementation of these techniques is used in the C++ GNU Multi-precision library (GMP) and using the Big Integer library in Java. It does not easy to efficiently parallelize the Java version using multiple JVMs, so that decided to use only GMP library. The prime number generator is applied in java to generate semi-primes of required bit sizes according to a mathematical formula and it has stored a pre-computed list within text files [14]. To analyze and compare the algorithms GMP library was used for factoring using good curves and performing basic EC operations [15]. OpenMP was mainly used for achieving the computations in a parallel manner. Our programs were run using Amazon EMR (Elastic Map Reduce) platform, which is implemented on the top of Amazon EC2.

The algorithms recognize size of the bits (S_N) for performing N (number of bit) as an input parameter and then randomly pull two prime-numbers p and q that size is about half of N, such that their sizes, S_p and S_q lie in a range $(S_N)/2+\varepsilon < S_p$, $S_q < (S_N)/2 + \varepsilon$ (ε was fixed as 7) and $S_p + S_q = S_N$. The constant term ε is used to limit the range of prime numbers p and q with the intention that they are amply larger and close to each other. To compute the time taken and the number of iterations manipulates as an average over 10-iterations using each technique. Corresponding to each iteration produces a different random semi-prime of parallel size (S_N). The observations for various sizes of S_N between 16 and 130 have to be recorded. For $S_N > 120$, factorization times become significantly large (greater than 15 minutes) and hence they do not appear in results. Table 1 shows the number of cycles per iteration that are needed to solve the ECDLP and using different platforms like Baby Step Giant Step, Pollard Rho, MapReduce and Pohlig Helman for ECDLP.

Table 1: No of cycles per iteration and time

S_N	No of Cycles per Iteration				Time (m) to solve ECDLP			
	Pollard Rho	**BSGS**	**Pohlig Helman**	**MapReduce for ECDLP**	**Pollard Rho**	**BSGS**	**Pohlig Helman**	**MapReduce for ECDLP**
42	179	242	321	19	300	359	429	212
54	305	362	458	25	401	468	512	356
62	397	456	547	29	523	569	599	496
74	501	557	667	3	603	687	712	586
85	607	649	764	35	712	798	832	699
96	721	766	881	42	817	903	978	762
107	808	854	987	45	894	1162	1298	834
118	879	931	1132	46	1009	1358	1467	912

120	1016	1034	1235	52	1137	1498	1598	1013
134	1207	1234	1368	57	1298	1645	1787	1135

The time complexity of Pohlig Helman algorithm is based on the pre-computation of the primitive $p_i^{e\,i}$ -th roots of unity. For each m = p^e, preparation of the table {u_m, j} involves one exponentiation and (p^e−1) multiplications. The entire pre-computation procedure needs $o\left(\sum_{i=1}^{k}\left(log^3\, q + p_i^{e\,i}\, log^2\, p_i\right)\right)$ operations. The main benefit of the algorithm proposed is that, once the pre-computation process is completed, the algorithm uses only one exponentiation and one comparison with the look-up table and directly obtains the value x (mod p^e). BSGS algorithm computes discrete logarithms in a cyclic group G of order n in deterministic time O($\sqrt{n}$) group operations and requires p$\sqrt{n}$ group elements storage. In MapReduce based algorithms, the concept of order of a point is to establish time complexity, since linear combinations of P and Q are elements of (Q). Let n be the order of Q. MapReduce algorithm resolves an ECDLP using n/2 steps on average using constant memory. Pollard Rho algorithm solves the same problem on $\sqrt{\pi\frac{n}{2}} \cong 1.2533\sqrt{n}$ steps on average [9] and requires maximizing memory. Pollard Rho algorithm (sequential) needs MQ $\frac{\sqrt{\pi^5}\,n}{288} \cong 1.2533\sqrt{n}$ steps on average [7] for a random function $\sqrt{n}$ using constant memory; however, each step requires more computation than the MapReduce algorithm, because each step computes - three times. The comparison based on the storage complexity of the comparative algorithms, The BSGS algorithm performs O ($\sqrt{r}$) group operations and requires O($\sqrt{r}$) group elements of storage. The baby-step-giant-step algorithm needs large storage but Rho and Pohlig algorithms require less storage. The basic idea is to reduce the DLP to the problem of collision-finding, and then use low-storage collision-detection techniques. MapReduce based algorithm provides low storage computation to other algorithms by using parallel collision search methods and low Hamming weight DLP. The entire part of the general attacks focuses on the ECDLP and that have expected to run in fully exponential time. The best attacks out of the above are MapReduce based Method with its speedup storage computation.

The observations made from the above experiment are as follows: The spike in Pollard is the most evident, lesser in Pohlig, BSGS, and minimum in MapReduce based ECDLP. This is understandable, since as the size of the field and curve increases, so does the additional time taken by BSGS to generate the prime factors of the order. MapReduce based ECDLP is the most effective since it tries to generate a sequence of points, until $x_i = x_{2i}$. This means that it considers twice as many points in its search for the discrete log solution. This was because the MapReduce based ECDLP involves the computation of an inverse as mentioned earlier, which may not exist, and hence the solution may not be found even if it exists.

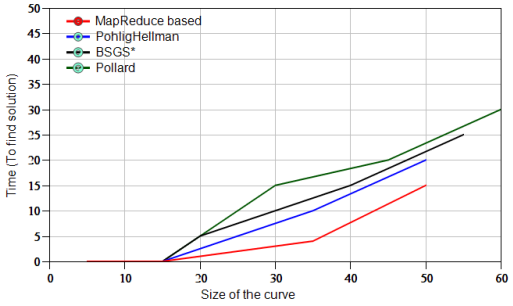

Figure 1: Performance over the different curve

There is no doubt that MapReduce based ECDLP when applicable to same curves performs best. Here, it is quite evident that BSGS and Pollard do worse out of all others. This is again, due to the prime factors generation.

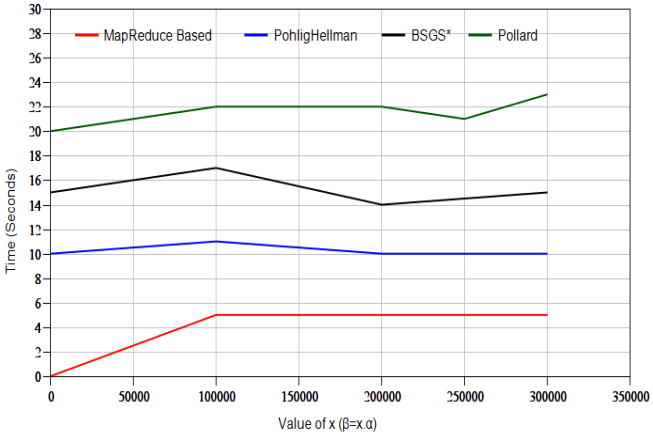

Figure 2: Performance over the same curve

Figure 3 shows how to compare the proposed algorithm with other algorithms. The result shows Mapreducec based ECDLP is about 50 times faster for 100-bit numbers and about 200 faster for 110 to 120-bit numbers. This clearly shows that Mapreduce based ECDLP has a better edge for number sizes between 100 to 150 bits.

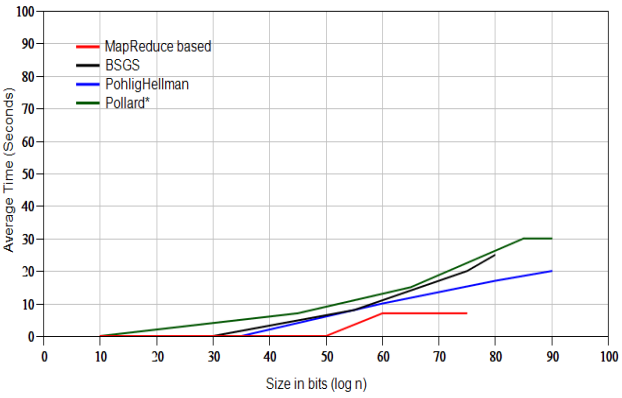

Figure 3: Comparison of using time

The Throughput is the maximum speed achieved by the network. It also refers to the bandwidth consumed by the nodes in the network. Figure 4 represents the throughput plot of MapReduce based ECDLP algorithm compared with BSGS and Pollard. The Throughput of the proposed algorithm provides the maximum throughput level of 350 kb/s. Figure 4. Throughput The Pollard algorithm takes more time to execute and requires larger packet size increases the network delay this leads to a decrease in throughput which is made very close to the performance of the BSGS technique.

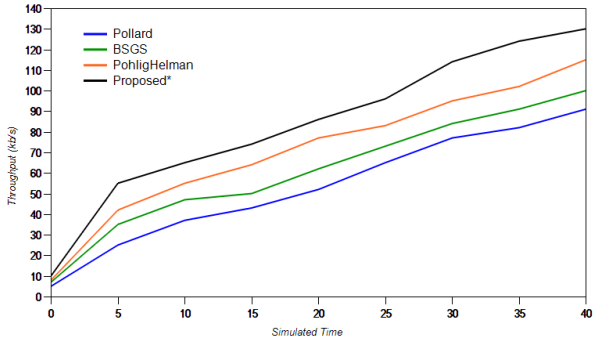

Figure 4: Throughput

IV. CONCLUSION

In this paper, the various techniques analyzed and focused to attack elliptic curve cryptosystems in which the security of ECC depends upon the ECDLP. Most of the pre-existing algorithms perform on improving the scalability of a single node and also handle the ECDLP problem based on the large prime factor under finite field needs large computational resources. By using MapReduce based ECDLP is a better approach to such instances that can scale to huge clusters with thousands of computation nodes. Moreover, proper distinguished property and the proper iteration process select as per the parallel platform for better speedup result. Therefore the proposed algorithm gives better result compare to other algorithms. Hence, concluded that results of this paper will be useful for designers to choose the appropriate curve for each application and the device, based on the performance requirements of the elliptic curve protocol, like ECDH, ECC, or ECIES, in the given application and environment.

V. REFERENCES

[1] M. J. Wiener and R. J. Zuccherato, “Faster Attacks on Elliptic Curve Cryptosystems”, In selected areas in Cryptography- SAC, volume 1556 of LNCS, pages 190-200, Springer, 1999.

[2] Menezes A. J., Okamoto T., Vanstone S. A, "Reducing Elliptic Curve Logarithms to Finite field", IEEE Trans. Info. Theory, 39: 1639-1646, 1993.

[3] P. C. van Oorschot and M. J. Wiener, “Parallel collision search with cryptanalytic applications,” Journal of Cryptology, vol. 12, no. 1, pp. 1–28, 1999.

[4] E. Wenger and P. Wolfger, “Solving the Discrete Logarithm of a 113- bit Koblitz Curve with an FPGA cluster”, SAC, volume 8781 of LNCS, pages 363-379, Springer, 2014.

[5] P. C. van Oorshant and M. J. Wiener, "Parallel Collision Search with Cryptanalytic Applications", Journal of Cryptology, Volume 12 No. 1, pages 1-28, 1999.

[6] M. J. Wiener and R. J. Zuccherato, “Faster Attacks on Elliptic Curve Cryptosystems”, In selected areas in Cryptography- SAC, volume 1556 of LNCS, pages 190-200, Springer, 1999.

[7] R. Gallant, R. Lambert, and S. Vanstone, "Improving the parallelized Pollard lambda search on anomalous binary curves", Mathematics of Computation of the American Mathematical Society, volume 69 No. 232, pages 19-46, 2002.

[8] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone., “Handbook of applied cryptography”, CRC Press Series on Discrete Mathematics and its Applications, CRC Press, Boca Raton, FL, 1997., With a foreword by Ronald L. Rivest.

[9] Pohlig, Hellman: "An Improved Algorithm for Computing Logarithms over GF(p) and it's Cryptographic Signi_cance", 1978, http://www-ee.stanford.edu/ hellman/publications/ 28.pdf

[10] https://en.wikipedia.org/wiki/ Chinese remainder theorem, December 7, 2015

[11] Hankerson, Menezes, Vanstone: "Guide to Elliptic Curve Cryptography", Chapter 4.1.1, 2004, Springer-Verlag New York, Inc.

[12] Novotney: "Weak Curves In Elliptic Curve Cryptography", 2010, http://wstein.org/edu/2010/414/projects/ novotney.pdf

[13] M. Pollard, "Monte Carlo methods for index computation (mod p)," *Mathematics of Computation*, vol. 32, pp. 918–924, 1978.

[14] Primes just less than a power of two: https://primes.utm.edu/lists/2small/0bit.html
[15] EECM-GMP Library: http://eecm.cr.yp.to/