7. The protocols in Cryptosystems

7.0 Introduction

The important sections and points that we are going to go through in this topic are the signature concerned with the elliptic curve, secondly the encryption part that is of public key type and finally the establishment of keys. The various algorithms that come under the discrete logarithmic problem are seen in the first part then their behavior and interlink need to be scrutinized for concluding the elliptic curve system. Also the verification and distribution of the domain parameters and the keys that are concerned in the elliptic curve problem are studied in detail. [22]

7.1 The elliptic curve discrete logarithm problem

As far as the EC logarithmic applications are made tough to break the more will be the security of the applications where all it is applied like the elliptic curve one. The logarithmic problem in $E$ which is that is stated on a finite field $F_q$. Here we take a point $P$ is and element of $E(F_q)$ that is $n$th ordered one and another point $Q$ which is an element of $(P)$

Then the integer $L$ is an element of $[0,n-1]$ so that $Q$ is equal to $LpP$ in which the integer $L$ is the discrete logarithm of $Q$ to the base $P$ and will be shown as

$L = \log_P Q.$

The elements that are to be used in the elliptic curve functions that are to be used in the cryptography need to be selected in a very careful manner as to avoid unnecessary attempts to corrupt and hack the data. The very apt and very appropriate algorithm that is found in discrete logarithmic problem is the extensive search one in which one calculates by incrementing $p$ until
the value of $Q$ is reached. Roughly its half the value of $n$ is the time taken to run in most of the cases but in some it is forced to go up to the whole value. Thus with this search method with the aid and help of the elliptic curve elements if the value of $n$ is very big to represent the large quantity of the calculations and functions involved. If some combination of algorithm is done like the pollard's rho and P-H algorithm those having full exponential running time there is a maximum probability of attack that can occur. In the running time $O(\sqrt{p})$ if the value of $n$ is taken such that it can be divided by $p$ which is considerably very big and it is impossible to compute the root of the $p$ involved by doing so the possibility of attack can be reduced to a considerable amount. So it is believed in today's situation that if all the elements in the elliptic curve problem are selected in such a way that it will be practically impossible and will be very hard and tough to break and hack the data in the case of discrete logarithmic problem. Till the day there in so proof in the field of mathematics that can prove the intractability of discrete logarithmic problem in elliptic curve. So it clearly supports and conveys the real fact that no algorithm in the field of mathematics is available to efficiently solve the discrete logarithmic problem. Also from the theory point of view also till the day there is no proof to show that discrete logarithmic problem is intractable. [70] Till the date only some information related to the discrete logarithmic problem intractability has been found out.

Till the recent years no sub exponential time algorithm of general purpose has been put forward and even with the involvement of so any researchers and their detailed study did anything fruitful in this area of study. the root of $n$'s lower bound of a discrete problem in groups of generic form of which the elements involved are strings of any kind of bit and only a single has the accessibility to the operations that are in groups and will be possible only with a database such as oracle. This proves only the fact that the discrete logarithmic problem is very hard
7.2 Pohlig-Hellman attack

In this algorithm the subgroups of $P$ which are prime group variety in the discrete logarithmic algorithm is formed by the optimized and highly reduced use of the function $l = \log P Q$. So it confirms the fact that the hardness is much more better in the subgroups of prime order than the elliptic curve discrete logarithmic problem in $(P)$. The value and order of $p$ is taken in such a manner that even with the selection of a very large prime number it should be easily divisible and thereby it increased the security of the pohlig hellman algorithm so it should be very careful while choosing the parameters involved in the function. So let us analyze the algorithm put forwarded by Pohlig Hellman. [98] Now we will see the factorization of the prime number $n$ that is $n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \ldots \ldots p_r^{e_r}$. Now Pohlig hellman proposed a method to find out $l_i = l \mod p^{e_i}$ if the value of $l$ lies between 1 and $r$ both lower and upper limit inclusive and then the congruence involved can be found out.

Let $n/p = w$

Then let us compute $P_o = wP$ and $Q_o = wQ$.

Now $Q_o = wQ = lwP = lP_o = e$ have $Q_0 = (n/p)Q = Z_0P_o$.

So in $[p]$ an instance of elliptic curve discrete logarithmic problem is solved.

Thus one gets $z_0 = \log P_0 Q_0$. Then we calculate $Q_1 = (n/p^2)(Q - z_0P) = ZIP_o$.

7.4 Parallelized Pollard’s rho attack
Now let us take a few number of processors for example $N$ and let them be used to run and solve an elliptic curve discrete logarithmic problem. Then one has to keep them running until they stop automatically. No we can see that each of them on detailed analysis stops when an approximately when they reach cubic root of $n/N$ number of operations.

Therefore it is seen that the speed up has been achieved only by square root of $N$. Then another team of researchers Van and Wiener came up with another version of rho algorithm which produces $N$ as a factor that speedup if there exist the same number of processors. The main concept is to allow a sequences of $iX$ elements where $I$ is greater than or equal to 0 which are produced by the processors and are allowed to collide among themselves. All the processors in the above takes the same function $f$ to calculate the values of $Xi$ while they are free to choose individual starting point $Xo$. In this process if at all two processors collide with each other then from that point onwards they will be resembling each other more likely. To find out the collision that happened in the sequences produced by various processors can be found out with the help cycle finding algorithm of Floyd. The following methods helps in finding more suitably the collision in the sequence produced by various microprocessors. [99] To find out the difference and to identify the tests is to find out the notable properties in each points. Thus by a particular combination of bits like zeros or ones can be used for this purpose. Now in $P$ let the proportion of points be represented by theta that are having distinguishable property. So now if at all a microprocessor gets into such distinguishable point the moment it sends that point to the main server that keep it in memory in a list that is sorted. Now whenever the main server again receives the point then it calculates the discrete logarithm and stops the whole of the processors involved. The total number of steps that are to occur before collision is square root of $\pi$ multiplied with $n$ by two and then it is divided with $N$. Next distinguished point will occur after
one by theta steps. So before the collision all the processors of prefixed points will have approximately. So the awaited number of elliptic curve functions done before the collisions of the distinct points happen by individual processor that is recorded is

\[
\frac{1}{M} \sqrt{\frac{\pi n}{2} + \frac{1}{\theta}},
\]

Number of elliptic curve functions happening and all the versions that were designed and produced along rho algorithm had the same efficiency found in all the processors. Here other notable features is that there is no intercommunications between the processors that take part in the process and also have to do only very rare communication with the main server. Also the main servers memory can be kept in limitation by selecting the peculiar properties.

7.5 Speeding Pollard’s rho algorithm using automorphisms

Consider a rho group auto-morphism function that is \( \psi : (P) \rightarrow (P) \) in which p is an element of E to the finite baseq and has an order of n. unlike the point addition the rho can be calculated very efficiently and in a very fast method. Now let us take t as the order which is the least possible integer out of the series collected by this rho algorithm.
It is at the point $x$ the third and the forth processors collide first, but the algorithm states that its at the point $y$ the collision happens.

$\rho$ (R) is equal to R and is applicable to every R elements of P. The equivalence relation on p is stated by

$R_1$ is equivalent to $R_2$ iff $R_1 = \rho_j(R_2)$ for some $j$ elements of $(0,t-1)$ and this forms the equivalent relation. Now the equivalence class that is $(R)$ having a point $R$ element of P is stated as

$R$ is equal to \{ $R$, $\rho(R),\ldots,\rho^{l-1}(R)$\}

The main objective to speedup is to apply on the equivalence classes by enhancing the repeated function $f$. In order to make this possible we consider the a temporary representation for the equivalence class [R] represented by R so that it forms a point in [R] where the integer $x$ is taken as the least valued one. By doing so another repeating function $k$ is represented by $k(R) = f(R)$

Now let us consider an integer lambda element of $(0,n-1)$ so that $\rho(p) = \lambda P$. 

Fig 7.5(a)
Let us take \( a \) and \( b \) two integers so that \( X = aP + bQ \), so that easily calculate integers \( a_1 \) and \( b_1 \) such that \( X_1 = a_1P + b_1Q \).

The search space will be of the size \( n \) divided by \( t \) if the mostly all the equivalence class has the size \( t \) and the approximate time taken for running the modified parallelized rho algorithm is \( 1/M(\sqrt{\phi n} by 2t) + 1/\theta \). Here we have achieved a speedyp approximately root of 2 and which is felt over almost all elliptic curves.[16]

7.6 Multiple logarithms

In the previous section we have gone through the advanced pollards rho algorithm in which the result of a single elliptic curve discrete logarithmic problem instance in \( p \) stores the distinguished points in \( P \) is applied to other instances in the logarithmic problem in \( (P) \). This attribute is of high concern to the security and safety of the ECC devices as the people involved in the communication distribute the parameters and elements involved and choose the public keys on their own. That is \( Q \) is an element of \( (P) \). So here by applying the pollards rho algorithm to find out some keys then its rather easy task to find out the rest in a faster way. Let us consider \( l = \log \) of \( Q \) to the base \( p \). The distinguished points \( X \) out of the three stored values \((c, d, X)\) that are got upon the calculation the integer \( s \) which is equal to \( c + dl \mod n \) satisfies \( X = sP \). In the same way another integer \( r_j = a_j + bj l \mod n \) also goes with \( R_j = r_jP \) where \( j \) lies between \( 1 \) and \( L \) taking into account the upper and the lower limits.

To evaluate \( l' = \log \) of \( Q' \) to the base \( P \) where \( Q' \) is an element of \( P \) and the element of random series \( Y_i \) for each and every microprocessor is calculated with the formula \( Y_0 = c_0'P + d_0'Q' \)
where \( c'_0, d'_0 \in R [0, n - 1] \) using the same repeating function \( f \) as used previously. And for every particular point \( Y \) that is found out in the new series the three point coordinates \( (c', d', Y) \) where \( Y \) is equal to \( c'P + d'Q' \) is given the updating to main server.[80]. Only two new series or among an old one and a new one the collision happens. If the case is of the first type then we can follow the equation

\[
C'P + d'Q' \text{ is equal to } c''P + d''Q'.
\]

In the second case it is \( c'P + d'Q' \) is equal to \( sP \).

Out of the first and the second particularly identified points that was collected in the elliptic curve discrete logarithmic problem the same computations can be used for the third, fourth and so on. The minimum freely chosen walk steps denoted by \( W_k \) before \( j \) instances in logarithmic problem are repeatedly solved as using the following.

\[
W_k \quad \text{(approximately equal to ) } T \quad \sigma \text{ from } i = 0 \text{ to } k-1( \text{upper 2i lower i) / 4 to the power i.)}
\]

Here the \( T \) denotes the minimum free walk steps that is being used to solve single instance.

In a similar way the rest of the instance like \( 3^{rd} \), \( 4^{th} \) and so on are done and it takes only 49 % and 35 % respectively of the time taken to find the first solution. So we have found out that its easy to find out the solution for all the other instances like second third and so on to make it secured and highly protective we will make the first instance itself very hard to break and to find the solution infeasible. [131]
7.7 Index-calculus attacks

In certain groups such as multiplicative one the jacobian one in order to calculate the discrete logarithms the index calculus algorithm is the most apt and feasible one. The class group of conceptual quadratic number field, the F*q multiplicative group in a finite field and the Jacobian in the finite field of a hyper elliptic curves are some of the algorithm that make use of the above. So the index calculus is the best method to explore further algorithms in the elliptic curve discrete logarithmic problem

The root cause for developing index-calculus method

Consider a cyclic group C of nth order formed by alpha. Then next is to find the value of log of beta to the base alpha where beta is and element of C.

1. Selection of the factor base is the first step involved.

   From among the C s few elements are taken and put into a series named S and let it be \{p1,p2,.....pt \} of C then we that the s is the factor base and is selected in such a way that the majority of the important elements are included in S so that C is represented in short. Thus the attributes of the special group C is taken into consideration in making the S.

   Second Calculate the logarithms of the members within S.
First select any integers represented by $k$ from $S$ in such a way that $k$ is an element of $[0, n-1]$ so that the elements selected can be represented by the product of elements within $S$.

$$\alpha^k = \prod_{i=1}^{t} p_i^{c_i}, \text{ where } c_i \geq 0.$$  

Now in this equation finding on either side of the equation the logarithm to the base $\alpha$ it results in a equation of linear type and gives

$K$ is approximately equal to

$$\text{sigma from } i=1 \text{ to } k (C_i \log \text{ of } P_i \text{ to the base } \alpha) \text{ (mod n)}$$

Here the iteration process is carried out till a value more than $t$ an equation equivalent is obtained, so as to find solutions to linear equations that is got as a result can be solved so as to get

Log of $\text{Pi}$ to the base $\alpha$ where the value of $i$ lies between 1 and $t$ taking the upper and lower limits into considerations.

Now compute log of beta to the base $\alpha$ as the next step. Till the products of the members of $S$ that is $\alpha^k \beta$ can be reached we need to select any integer value $k$.

$$\alpha^k \beta = \prod_{i=1}^{t} p_i^{d_i}, \text{ where } d_i \geq 0.$$  

Now we will take log to the base $\alpha$ on either side of the above formula and by doing so we can get the logarithm of beta.

Log of beta to the base $\alpha$ = - $k + \text{sigma} [d_i \log \text{ of } P_i \text{ mod n}]$ where $i$ varies from one to $t$. 
The base factor $S$ selection is very very important. The running of index algorithm depend very much on this selection. [67] There is a relation between the size of the $t$ and the value of $S$. Mostly larger value of $t$ is preferred. This is because as the value increases as the chances of a group element that factors $S$ will be much larger in size. The number of linear equations will come down once the value of $t$ is small. So the element that is in $G$ and which is going to factorise over $S$ will be factor for choosing the value of $t$.

### 7.8 Isomorphism attacks

Consider a finite field $F_q$. Then define an elliptic curve $E$ over it. Next let $P$ be an element of $E(F_q)$ with $n$ as prime order. Now consider a group $f$ with $n$ as the order and it should be prime so that $P$ and $G$ are both cyclic and isomorphic.

The particular notable points in $(P)$ can be easily brought up to the instances in the case of discrete logarithmic problem in $G$ if $P$ and $Q$ are the elements of $(p)$ then

$$\log Q \text{ to the base } P \text{ is equal to } \log \rho(P) \rho(Q).$$

If the sub exponential time algorithms are familiar then ten attacks that are isomorphism converts the elliptic curve DLP to digital logarithmic problem of group $G$. The attacks of this type are of very ideal purpose. They finally conclude in the elliptic curve discrete logarithmic problem solver that perform even faster than rho algorithm of pollard dedicated for the ideal cases of elliptic curves. [17]
7.9 Elliptic curve Diffie-Hellman problem

Consider an elliptic curve which is denoted by $E$ and a finite field which is denoted by $F_q$ over which it is defined then a point $P$ element of $E$ over finite field $F_q$ and is of order $n$ and the points are $A$ and $B$. Where $A = aP$ and $B = bP$ is an element of $(P)$ then we have to evaluate the value of $C$ which is given as product of $a$, $b$ and $P$.

When one can do find the solution in $(P)$ the problem of elliptic curve discrete logarithmic one then it can also be very wisely used in Hiper problem with the first step of finding the value of $a$ from the coordinates $P$ and $A$ then we can calculate with this application in the product of $a$ and $B$ which will yield the value of $C$. [114] hence the above states that the hyper problem in elliptic curves is no longer harder and can be made very simple even than the logarithmic version. Thus the discrete hyper problem in elliptic one is not much harder than the ECDLP. With the support of hypothetical oracle the hyper application in elliptic curve can be solved easily. In some cases like a prime number $m$ is taken and found that every prime factors of $m - 1$ are smaller then in those places the hyper problem is found to be more simple and easier than the logarithmic version of the elliptic curve. The cryptographers like boneh and lipton did extensive study in this field. Their work established the fact that the hyper problem is very hard and strong in some conditions like certain constants like $c$ then for some prime number like $m$ the elliptic curve discrete elliptic curve problem cant be prove in $Lm[12,c]$ and the hyper problem cannot be solved in $Lm[12,c-2]$. 
7.10 Signature schemes

Like the one that is used in hand written documents to prove its authenticity and ownership, the
digital communication also demand and need an authentication or ownership confirmation and
that is done here too as signature. With this not only the ownership but also integration of data
and the confirmation that no information has been changed or no one has hacked and done some
manipulation is confirmed through this process. By clubbing together the particular member and
the public key the signed certificate that is being issued by special authorities that are trusted and
authorized to issue the same. [69]

**The structure of the signature consists of mainly 4 algorithms.**

The first is an algorithm which generates a group of domain elements say D.

Second algorithm deals with the generation of the key which takes D as the input and with this
domain parameters it produces pair of keys Q and d.

The third algorithm is the main in signature schema that it generated the signature an utilizes the
above parameters the private key , the plain text m and with all these produces the signature
sigma

The final algorithm that is associated with the signature process is the verification algorithm that
takes all the inputs from the above three algorithm and finally makes out if one has to accept it
or reject it. [60] The above process as a whole is said to be safe and protected one if there is no
loop hole to forge the content or hack the content, alter the content or duplicate the signature
even with the application of the complicated computations. The real fact is that once the intended user has signed and dispatched the message and in between any one who again who even though is capable of getting any signature for any number of messages the device that already signed the message wont again give the signature as once it has already dispatched it. Thus seven though the evil signature hacker has the capability to get any number of signature here the security aspect is very strong as the last presented ot the signing cannot be obtained.

So as far as the certain applications are concerned even though if the highly capable signature hackers can get any signature at the ease but even with very hard try throught permutation and combinational methods to get the signature they will remain unsuccessful and even in some cases already dispatched case too they will remain in utter failure. So while the designing section is going on they will be looking into all the aspects of the attempts to hack and fail or retrieve the signature. So it will be tested in all the possible environments and in almost all the computational possibilities that can be encountered in the real time surroundings. Then only the signature algorithm will be released into the market taking into account the contemporary adversaries. The design will be a uniformed one whether it is applied for a highly sensible or general message the weight age, the effort put and the security aspects will be the same and no compromise in it is entertained.

7.11 ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the Digital Signature Algorithm’s (DSA) elliptic curve version. In the modern scenario and available algorithm the most trust. Let
us now elaborate the algorithm where we will be using H to denote the cryptographic hash function. Then n is taken as a value which act as the reference for the output bit lengths as it should not go up to the value of n and failing which the outputs will be discarded in the hash function.

Creating ECDSA signature

D denotes domain parameters ie D is equal to \((q, FR, S, a, b, P, n, h)\),

Let small \(d\) denote the key used as private key, \(m\) denote the plain normal text or the message that need to be encrypted and finally let the coordinate pair \(r\) and \(s\) form the signature ie. Signature \((r, s)\).

Steps involved in the signature creation

First select the value of \(k\) element of \(R\) of \((1, n-1)\).

Secondly find the value of product of \(k\) and \(P\) that is \(kP\) is equal to the coordinates \((x_1, y_1)\) and then the x coordinate is changed to integer \(x_1\)

In the third phase we will compute the value of \(r = \text{int}(x_1) \mod n\). Here if the value of \(r = 0\) then go to step 1.

In the next stage find the value of \(e\) by computing \(e\ is\ equal\ to\ \text{H} of\ m\).

Then in the final stage we will come to the calculate the signature \(s\) which is equal to \(k\ inverse\ of\ (e+dr)\ mod\ n\). If \(s = 0\) then go to step 1.

Then once the value of \(s\) is secured return the value of \(r\) and \(s\).
Cross checking ECDSA signature

For the purpose of cross verification let us take the domain parameters which is denoted by $D$ having elements as stated above also let us take $Q$ the public key , $m$ be the plain text or message , signature $(r, s)$.

Whether the signature is to be Accepted or rejected

1. First do the Verification process that $r$ and $s$ are of integer type that comes in the interval with lower limit 1 and upper limit n-1. While verifying if the process does not satisfy the conditions then return the message that the “signature has been rejected

In the second step we will compute the value of e which is $H(m)$.

In the third step we will compute the value of $w$ which is $s−1 \mod n$.

In the forth step we will compute the value of $u1$ that is $ew \mod n$ and then the value of $u2$ which is taken as $rw \mod n$.

In the fifth step we will compute the value of $X$ which is $u1P + u2Q$.

The we will check the condition that if the value of $X = \infty$ then we will return the message that the signature is rejected.
Then in the seventh step we x 1 of X that is the x coordinate to an integer \( \bar{x}_1 \) then we will compute the value of \( v \) which is equal to \( \bar{x}_1 \mod n \).

In the eighth step the verification is again done that is the condition is checked that is If \( v \) is equal to \( r \) on that condition the message is returned that the signature is returned. If the above is not satisfied then naturally the other option that is the signature is rejected is returned.

**Cross verifying that whether signature verification works**

If the signature is indeed dispatched by the genuine user then on the plain text message \( m \) the signature \( s \) and \( r \) together taken and crossverified in the following way that is \( s \) is approximately equal to inverse of \( k \) multiplied with sum of \( e \) and \( dr \) and again multiplied with \( \mod n \) that is \( s \equiv k^{-1}(e+dr) \mod n \). Again arranging we get \( s^{-1}(e+dr) \equiv s^{-1} e + s^{-1} rd \equiv we + wrd \equiv u_1 + u_2d \mod n \).

Thus \( X = u_1 P + u_2 Q = (u_1 + u_2d)P = kP \), and so \( v = r \) as required.
**Points that establish the security of ECDSA**

**First Point**

The security to achieve as per GMR standard the elliptic curve discrete standard algorithm it should not be traceable any way in (P). Then secondly the hash function H should be protected and secured cryptographically. The elliptic curve discrete standard algorithm has proven its worth and very secured one among the generic group category. It strongly holds the H function very secured and application oriented. Though the generic group category never means a fully fool proof one still as far as the elliptic curve discrete standard is concerned it inspires and support the need for the confidence in the security. [130]

**Second Point**: The security aspects and considerations on the hash function is emphasized here. The information even in the form of an image or picture need to be protected by the hash function because the signature can take any form and hence it need to be protected and prevented from forging by E. What it does is that it takes an arbitrary value k an integer type and estimates s as the x coordinate of Q+kP simplified modulo n. After this the E assigns l = r and then it does calculate e = rs mod n. Now if any message or plain text say m was taken by E then e is equal to hash(m) and once its is ok then it implies that the pair r and s is the valid signature for that plain text. The simple conclusion is that if the hash function is not resistant to collision then the hacking or duplicating the signature is not a big issue to E.

**Third Point**
The first step in this is the verification and confirmation that the values of \( r \) and \( s \) in that are associated with the signature are genuine and unaltered. This is first done by the verification an confirmation of the fact that the values of \( r \) and \( s \) lies in the interval 1 and \( n-1 \). During the verification all the aspects and possibilities of attack and the counter measures are taken and the test is a very successful one. The verification is done with the help of ElGamal verification procedure for the signatures is employed.

**Forth Point**

This emphasize on the protection aspects that one need to see and consider for the secrets that one need to convey before the message transmit. One need to consider and plan the same level of security and safety as we give to the secret keys the before the message encryption and transmission the signature generation and the parameters like \( k \) associated with it too need to have that much high level of security. The security of the the whole message transmission in this lies in the real fact that how much one has secured the signature parameter \( k \) that is to what extend it is secured is the security of the whole. If at all the valueo of \( k \) is hacked by the unwanted elements then they can easily apply it on the other texts and easily take the Private or secret key and apply the same on other algorithm to break and leak the matter.so once \( k \) is available to unwanted hands they will make the signature with \( r \) and \( s \) and will apply it on some text using the formula

\[
d \text{ is equal to } r^{-1}(ks - e) \mod n
\]

So in general all these experiences and studies reveal and emphasise on the point that the pre parameters involved before signature formation need to be protected at the most important.
**Fifth Point**

The need to again and again new production of the pre message of the secret k is emphasized and its importance and link in the aspect of security. That is the idea behind this is that if every time one new is produced means there will be no way to get the value in wrong hands and thus we can protect the private key from being recovered or hacked.
8. Proposed Embedded Elliptic Curve Signature Algorithm

8.0 Proposed new Algorithm

• EECDSA CREATION OF DOMAIN ELEMENTS

The domain elements for EECDSA are made of a finite field \( F_p \) of characteristic \( p \) with a carefully selected elliptic curve denoted by \( E \) and stated over it and \( G \) a fundamental point which is an element of \( E_p \) with coordinate elements \( a \) and \( b \) having order as \( n \). The steps followed in the process are

First choose any number \( x \) at free will or pseudo-random which is an integer so that the value of \( x \) lies between one and \( n-1 \) inclusive of the lower and upper bounds.

Secondly we need to calculate the value of \( Q \) which is the product of \( x \) and \( G \).

Thirdly assign the keys to \( x \) that is \( Q \) is taken as the public one and \( X \) as the private key of \( A \)

• EECDSA SIGNATURE GENERATION

Let us see how a plain text \( m \) is signed. Then take the domain parameters and for that chose a member named \( A \). Let the set \([P, E_p[a, b], G, n]\) be the members considered and let the keys that are taken be \( x \) and \( Q \) respectively

1. First choose any two integers \( e_1, e_2 \) in such a way that \( 1 \leq e_1, e_2 \leq n-1 \) and \( e_2 = (e_1+e_3)/2 \).
2. Then calculate \( e_1G = (X_1, Y_1) \) and \( e_2G = (X_2, Y_2) \).
3. Then find out \( r_1 = X_1 \mod n \) and \( r_1 = X_2 \mod n \). If \( r_1 = 0 \) and \( r_1 = 0 \) then leap back to step1.
4. Then calculate the inverse of \( e1 \mod n \).
5. Then calculate (Secure hash algorithm – 1) (m) and change into an integer the string that is available that is to H(m).

6. Then find the value of \( s = e_1^{-1} \left[ H[m] e_1 + X [r1+r1] \right] \) (mod n).

Here if the value of \( s \) is found to be zero then step 1 is repeated.

7. Then we have the signature \( r1 \) and \( s \) that is available for the plain text \( m \).

- CROSS CHECKING EECDSA SIGNATURE

The following is how cross verification the signature \((r_1, s)\) of A on \( m \) is carried out, B gets a copy that proves the ownership with regards to domain elements \([p, E_p[a, b]], G, n]\) of A. Let the public key that is taken into consideration be represented by \( Q \). The below procedures are followed by B:

1. First cross check that \( r_1 \) and \( r_2 \) are numbers of the type integer within \( 1 \) and \( n-1 \) as the interval.

2. Then calculate (secure hash algorithm – 1)(m) then make this to an integer hash(m) with this string.

3. Then find out the value of \( w = s^{-1} \) (mod n).

4. Then calculate \( u_1 = H(m)we_2 \) (mod n) and \( u_2 = [r_1 + r_2]w \) (mod n).

5. Then find out \( X = (X_3, Y_3) = u_1G+u_2Q \).

6. Reject the signature once the value of \( X = O \). Else calculate \( v = x_3 \) (mod n).

7. The signature is accepted iff \( v = r_1 \).

8.1 • PROOF THAT SIGNATURE VERIFICATION WORKS

If A has generated the signature \((r_1, s)\) on a message \( m \), then

\[
s = e_1^{-1} \left( H(m)e_2 + x(r_1 + r_2) \right) \pmod{n}.
\]

Rearranging gives
\( e_1G = \text{inverse of } s(Hash(m)*e_2+x*(r_1+r_2)) * G^* (\text{mod } n) \)

\[ = s^{-1}H(m)e_2G+s^{-1}(r_1+r_2)xG \pmod{n} \]

\[ = H(m)we_2G+ (r_1+r_2)* wQ \pmod{n} \]

\[ = u_1G+u_2Q \pmod{n}. \]

Thus \( u_1G+u_2Q = (u_1+u_2d)G = e_1G, \) and thus we have \( v = r_1 \) which verifies the result.

**EDCSA VS. proposed embedded ECDSA**

<table>
<thead>
<tr>
<th></th>
<th>ECDSA</th>
<th>Proposed embedded ECDSA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key generation</strong></td>
<td>Take ( E_p(a,b), ) and ( x ), where the value of ( x ) lies between 1 and ( n ) the lower bound included.</td>
<td>Take ( E_p(a,b) ) and ( x ), where the value of ( x ) lies between 1 and ( n ) the lower bound included.</td>
</tr>
<tr>
<td></td>
<td>Now consider ( G ) element of ( E_p(a,b) ) having order ( n )</td>
<td>Here also Now consider ( G ) element of ( E_p(a,b) ) having order ( n )</td>
</tr>
<tr>
<td></td>
<td>Now with all these values find ( Q = dG )</td>
<td>Now with all these values find ( Q = dG )</td>
</tr>
<tr>
<td></td>
<td>((E_p(a,b), p, G, n, Q)) be the public key</td>
<td>((E_p(a,b), p, G, n, Q)) be the public key</td>
</tr>
<tr>
<td></td>
<td>And ( x ) be the private key</td>
<td>And ( x ) be the private key</td>
</tr>
<tr>
<td><strong>Signature generation</strong></td>
<td>Now take ( k ) where it lies between one and ( n ) with lower bound included.</td>
<td>Take any three integers ( e_1, e_2 ) and ( e_3 ). ( e_1 ) any two integers where ( e_1 ) is less than or equal to 1 and ( e_3 ) is less than ( n ) and the value of ( e_2 ) is ( (e_1+e_3)/2 )</td>
</tr>
</tbody>
</table>
\[ K^*G = (x_1, y_1), \quad r = x_1 \mod n \]
\[ s = k^{-1}(H(m)+xr) \mod n \]
\( (r, s) \) forms the signature pair of \( m \).

\[ e_1G = (x_1, y_1), \quad r_1 = x_1 \mod n \]
\[ e_2G = (x_2, y_2), \quad r_2 = x_2 \mod n \]
\[ s = e_1^{-1}(H(m)e_2 + x(r_1 + r_2)) \mod n \]
\( (r_1, s) \) forms the signature pair of \( m \).

<table>
<thead>
<tr>
<th>Signature verification</th>
<th>( w = s^{-1} \mod n )</th>
<th>( u_1 = H(m)w \mod n )</th>
<th>( u_2 = rw \mod n )</th>
<th>( u_1G + u_2Q = (x_2, y_2) ), ( v = x_2 \mod n )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( w = s^{-1} \mod n )</td>
<td>( u_1 = H(m)we_2 \mod n )</td>
<td>( u_2 = (r_1 + r_2)w \mod n )</td>
<td>( u_1G + u_2Q = (x_3, y_3) ), ( v = x_3 \mod n )</td>
</tr>
</tbody>
</table>

now since the value of \( v \) and \( r \) a is the same that implies signature is accepted.

8.2 Conclusion

The reason why the proposed Embedded Elliptic curve Digital signature algorithm is the best one than the former ECDSA is explained in the following points:

**Former old ECDSA.** Messages are signed using \( k \) as the secret key need to be produced separately with respect to each other. That is it need to be signed separately with the key \( k \) as secret one or else it will be easy to get away the secret key \( x \). It should be taken into account as
if some freely choosen and secure number that is generated is being taken, so the probability of creating a value *k which repeats* is very less. Now let us evaluate the ways to estimate private keys when there is a repetition of secrets. Let us assume that the same *k value which is secret* was used to generate ECDSA signatures \((r, s_1)\) and \((r, s_2)\) on two different messages \(m_1\) and \(m_2\). Then

\[
s_1 = k^{-1}(H(m_1) + xr) \pmod{n}
\]

\[
s_2 = k^{-1}(H(m_2) + xr) \pmod{n},
\]

where \(H(m_1) = \text{SHA-1}(m_1)\) and \(H(m_2) = \text{SHA-1}(m_2)\). Then

\[
ks_1 = H(m_1) + xr \pmod{n}
\]

\[
ks_2 = H(m_2) + xr \pmod{n}.
\]

Subtraction gives \(k(s_1 - s_2) = H(m_1) - H(m_2) \pmod{n}\). If \(s_1 \neq s_2 \pmod{n}\) having a very high level of possibility and probability so that we can see that the value of \(k\) is \((s_1 - s_2)^{-1}(H(m_1) - H(m_2)) \pmod{n}\). Thus, the value of \(k\) can be determined by the adversary, and then apply this to recover \(x\).

**In the proposed embedded ECDSA**, if we use the same secret \(e_1, e_2\) was used to find out the signatures in ECDSA as \((r_1, s_1)\) and \((r_1, s_2)\) acting upon two plain texts \(m_1\) and \(m_2\). Then

\[
s_1 = e_1^{-1} (H(m_1)e_2 + x(r_1 + r_2)) \pmod{n}
\]

\[
s_2 = e_1^{-1} (H(m_2)e_2 + x(r_1 + r_2)) \pmod{n},
\]

Where \(H(m_1) = \text{SHA-1}(m_1)\) and \(H(m_2) = \text{SHA-1}(m_2)\). Then
\[ e_1s_1 = H(m_1)e_2 + x(r_1 + r_2) \pmod{n} \]

\[ e_1s_2 = H(m_2)e_2 + x(r_1 + r_2) \pmod{n}. \]

Subtraction gives \( e_1(s_1 - s_2) = (H(m_1) - H(m_2))e_2 \pmod{n} \). Even if \( s_1 \neq s_2 \pmod{n} \), so as to get a similar equation related as \( e_1(s_1 - s_2) = (H(m_1) - H(m_2))e_2 \). We cannot find out the value of \( k \) with the aid of above equation and with which we can get the value of \( x \). Thus we can conclude that our new algorithm is more secured and apt for embedded systems.