Elliptic Curve Final Report
Netanel Belgazal, Edward Galfrin
Supervision: Barukh Ziv

Introduction to Elliptic Curve Cryptology

**Definition:** An elliptic curve $E$ over a filed $K$ with characteristic $\text{char}(k) \neq 2,3$ is defined as follows:

$$E : y^2 = x^3 + Ax + B$$

Where $A, B \in K$ s.t. the discriminant isn’t equal to zero.

1) **Group Law:**
A set of points who satisfied the equation together with an infinity ($\infty$) point, who serves as the identity of the group, form an abelian group. We first introduce the addition operation geometrically. Let $P = (x_1, y_1), Q = (x_2, y_2)$ be two different points on $E(K)$.

Let’s draw a line that’s connecting the two points. This line intersects $E(K)$ on a third point. The point $R$ is the reflection of this point on the $x$-axis.

On the second case, when we want to add a point to itself, we call this action - doubling, we take the tangent line to the point. This line intersects $E(K)$ in a second point, the point $R$ is then the reflection of this point about the $x$-axis.
In the algebraic form:

Identity: \( P + \infty = \infty + P = P \) for all \( P \in E(K) \)

Negatives: if \( P = (x, y) \in E(K) \Rightarrow (x, y) + (x, -y) = \infty \).

The negative point of \( P \) is: \( -P = (x, -y) \in E(K) \).

Point addition: let \( P = (x_1, y_1) \in E(K) \) and \( Q = (x_2, y_2) \in E(K) \), where \( (XZ^{-c}, YZ^{-d}) \)

then \( R = P + Q = (x_3, y_3) \)

\[
x_3 = \left( \frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \left( \frac{y_2 - y_1}{x_2 - x_1} \right) (x_1 - x_3) - y_1
\]

Point doubling: let \( P = (x_1, y_1) \in E(K) \) where \( P \neq -P \) then \( R = 2P = (x_3, y_3) \)

\[
x_3 = \left( \frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1 \quad \text{and} \quad y_3 = \left( \frac{3x_1^2 + a}{2y_1} \right) (x_1 - x_3) - y_1
\]

a) **Group Order:**

We are focusing from now on, on elliptic curve over prime fields \( \mathbb{F}_q \), where \( q \) is prime.

The group order is defined by the number of points who satisfies the elliptic curve equation.

We mark the number of point on the elliptic curve as \( \# E(\mathbb{F}_q) \). We will see later that the cryptographic strength of the elliptic curve is determined by the order of the curve.

**Theorem** (Hasse) Let \( E \) be an elliptic curve defined over \( \mathbb{F}_q \). Then:

\[
q + 1 - 2\sqrt{q} \leq \# E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}
\]
2) **Point Representation**

The formulas for adding two points or double a point on the elliptic curve were presented above require a field inversion and several field multiplication, then it may be advantageous to represent points using projective coordinates.

We are looking at a point on the two dimensional plane as a ray in the three dimensional plane.

We define an equivalence class of points. Every point is in the form \((X : Y : Z)\). If \(Z \neq 0\) it corresponds to the point on the curve, \((XZ^{-c}, YZ^{-d})\), for some constants \(c,d\).

We are using two types of point presentation. The first type is the Jacobian presentation, in which \(c=2,d=3\). The projective point \((X : Y : Z), Z \neq 0\), corresponds to the affine point \((X/Z^2, Y/Z^3)\) In this presentation the point at infinity corresponds to \((1,1,0)\), while the negative of \((X:Y:Z)\) is \((X:-Y:Z)\).

The second type of point representation we are using is the standard projective presentation, in which \(c=1,d=1\). The projective point \((X : Y : Z), Z \neq 0\), corresponds to the affine point \((X/Z, Y/Z)\) In this presentation the point at infinity corresponds to \((0,1,0)\), while the negative of \((X:Y:Z)\) is \((X:-Y:Z)\).

3) **Point Multiplication**

Point multiplication or in its other name - scalar multiplication is a method for computing \(kP\) where \(k\) is an integer (scalar) and \(P\) is a point on an elliptic curve \(E\) defined over a field \(F_q\).

In other words, we are doubling \(P\) to itself \(k\) times, \(P + \ldots + P\). This operation dominates the execution time of the elliptic curve cryptographic schemes. The trivial way to compute \(kP\) is by adding \(P\) to itself \(k\) times. The complexity of this algorithm is \(O(K)\).

There exist a better algorithm who know by the name repeated-square-and-multiply. The complexity of this algorithm is \(O(\log(k))\).

We first find the Non-adjacent form (NAF) presentation of \(K\).

**Definition:** A non-adjacent form (NAF) of a positive integer \(k\) is an expression

\[
k = \sum_{i=0}^{l-1} k_i 2^i\]

where \(k_i \in \{0, \pm 1\}, k_{i-1} \neq 0\). And no two consecutive digits \(k_i\) are nonzero.

The length of the NAF is \(l\).
Theorem:
- K has a unique NAF denoted NAF(K).
- NAF(K) has the fewest nonzero digits of any signed digit representation of k.
- The length of NAF(K) is at most one more than the length of the binary representation of k.
- If the length of NAF(K) is l, then \( \frac{2^l}{3} < K < \frac{2^{l+1}}{3} \)
- The average density of nonzero digits among all NAF’s of length l is approximately 1/3.

NAF(k) can be efficiently computed. The digits of NAF(K) are generated by repeatedly dividing k by 2.

After finding the NAF(K) we are computing kP by the following algorithm:
1. \( Q \leftarrow \infty \)
2. For I form l-1 to 0 do
   (i) \( Q \leftarrow 2Q \)
   (ii) If \( k_i = 1 \) then \( Q \leftarrow Q + P \)
   (iii) If \( k_i = -1 \) then \( Q \leftarrow Q - P \)
3. Return (Q)

Let write K in his binary form: \( k_{t-1}, ..., k_1, k_0 \)

4) **Elliptic Curve Cryptography**

Public Key Cryptology is a method to encrypt a message which involves asymmetric key algorithms.

In this cryptography scheme, there are two keys. The public key, which is published to everyone, while the private key is kept in secret.

The public key cryptography method is based on computational hard problems. One of the common problems which used those days is the discrete logarithm. If \( g \) and \( h \) are elements of a finite cyclic group \( G \) then a solution \( x \), to the equation \( g^x = h \) is called a discrete logarithm to the base \( g \) of \( h \).

**Definition:** The elliptic curve discrete logarithm problem (ECDLP) is: given an elliptic curve \( E \) defined over a finite field \( F_q \), a point \( P \in E(F_q) \) of order \( n \), and a point \( Q \in \langle P \rangle \), find the integer \( l \in [0..n-1] \) such that \( Q = lP \). The integer \( l \) is called the discrete logarithm of \( Q \) to the base \( P \), donated \( l = \log_P Q \).

The most naive method to find \( l \) is by exhaustive search, where the attacker computes the sequence of point \( P, 2P, 3P, ... \) until he finds \( Q \). The running time of this algorithm is approximately \( n \) steps in the worst case, and \( n/2 \) steps in average. The best known attack on the ECDLP is the combination of Pohling-Hellman algorithm and Pollard’s rho algorithm, which is running in an exponential running time of \( O(\sqrt{q}) \). Therefore to avoid that attack we
need to choose $P$ such that $n > 2^{160}$. When we choose such an $n$, it will be inefficient to modern computers.

The most common finite field which is used those days is $\mathbb{Z}_p$ (like in the RSA algorithm).

There are few known algorithm that can solve the DLP in $\mathbb{Z}_p$ in sub exponential time. By this fact and the fact that the elliptic curve group is much more complicated that the ones in $\mathbb{Z}_p$ field makes the elliptic curve more effective for cryptographic algorithms.

<table>
<thead>
<tr>
<th>Security (Bits)</th>
<th>Symmetric encryption algorithm</th>
<th>Minimum Size (Bits) of Public Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DSA/DH</td>
</tr>
<tr>
<td>80</td>
<td>Skipjack</td>
<td>1024</td>
</tr>
<tr>
<td>112</td>
<td>3DES</td>
<td>2048</td>
</tr>
<tr>
<td>128</td>
<td>AES-128</td>
<td>3072</td>
</tr>
<tr>
<td>192</td>
<td>AES-192</td>
<td>7680</td>
</tr>
<tr>
<td>256</td>
<td>AES-256</td>
<td>15360</td>
</tr>
</tbody>
</table>

5) **Menezes-Vanstone cryptosystem**

Menezes-Vanstone cryptosystem is a variant of the ElGamal cryptosystem. In this variant the elliptic curve is used to "mask" the massage, instead of the ElGamal cryptosystem, were you need to encode the massage as a point on the elliptic curve.

The algorithm:

Suppose Alice wants to send a massage to Bob.

Bob prepare the following thinks:

1. Randomly chooses a prime $p$
2. Create an elliptic curve over the field $\mathbb{F}_q$.
3. Randomly choose a point $\alpha$ on the elliptic curve.
4. Randomly choose a private key $a$.
5. Computes $\beta = a * \alpha$

Bob then sends to Alice $(E, \alpha, \beta)$. 

Alice who wants to send a message $X = (x_1, x_2) \in F_q \times F_q$ to Bob does:

1. Randomly select $r$.
2. Computes $d = r \alpha$
3. $C_x = p_1 r \beta x$ - in other words, take the $x$ coordination of $r \beta$ and multiply it by $p_1$
4. $C_y = p_2 r \beta y$ - in other words, take the $y$ coordination of $r \beta$ and multiply it by $p_2$

Alice sends to Bob $d, C_x, C_y$.

When Bob wants to decrypt the message he computes:

1. Temp = $ad$
2. $P_x = C_x \div temp.x$ - in other words, take the $C_x$ and divide it by temp $x$ coordination.
3. $P_y = C_y \div temp.x$ - in other words, take the $C_y$ and divide it by temp $y$ coordination.

Why it works?

$P_x = C_x \div temp.x = p_1 r \beta x \div ad.x = p_1 r \beta x \div ad.x$

$= p_1 r \beta x \div ar \alpha x = p_1 r \beta x \div r \beta x = p_1$

$x^3 + ax + b \mod p$

Remark: $r \alpha$ must contain a nonzero component.

6) **How to select a point on the curve?**

A point on the elliptic curve $E$ must satisfies the equation. We randomly choose a value $x \in F_q$ compute $x^3 + ax + b \mod p$ and take a square root modulo $p$ of the value $x^3 + ax + b \mod p$.

Furthermore a corollary from the encryption protocol, we would like that the order of $P$ will be a very large prime number. Assuming our elliptic curve is strong, of order $k \cdot p$, we want to find a point $P$ with order divided by $p$. In order to find such a point, we randomly find a point as described above and check if $kP \neq \infty$, if so we return the point (otherwise find randomly choose another point).
7) **Massage Encoding**

One more question we need to answer is how to encode out massages. We are treating out text as ASCII text, where each character is represent as one byte code. We are splitting our massage to blocks. Each block is defined by the curve’s modulus and the size of byte. For example if p is 257 bit, then the size of block is \( \left\lfloor \frac{257}{8} \right\rfloor = 32 \).

Moreover we need that the number of block will be even, so if we receive odd number of block, we are adding a dummy block at the end. Each two block are treated now as \( F_q \times F_q \).

**Elliptic Curve Generation**

When using elliptic curve cryptology, we should use a strong elliptic curve. The strength of the elliptic curve is determent by the order of the curve. In order to get a sufficient level of security the order of the curve should be \( k \times q \) where \( n \) is small number (usually less than 10) and \( q \) is a big prime number (greater than \( 2^{160} \)).

There are mainly two methods to determinate the number of points in the elliptic curve. Those methods are known as Complex Multiplication and Point counting.

**Point Counting:**

In this method we first build an elliptic curve by randomly generate \( a, b \in F_q \).

After receiving the curve we use counting algorithms to count the number of points on the curve, if the number of point on the curve generate a strong elliptic curve we use it, otherwise we search for another curve. The main problem is that finding the number of point on the curve is a usually hard task. The known algorithms today for point counting are slow and inefficient. So despite their mainly advantage, generate a totally random curve, we are not using this approach in our project.

**Complex Multiplication:**

Another way is to generate an elliptic curve by using a method called complex multiplication.

In this method, we first select the desired order of the curve and then generate it. In this method we can build a cryptographically strong elliptic curve fast and efficiently. The disadvantage of this approach is that the curves which are generated by this method are not totally random. Up to this day, no major weaknesses are known for using this approach.

For an elliptic curve \( E \) over the complex field, we can consider an endomorphism (homomorphism from the curve to itself) of \( E \) given by rational functions.
Those homeomorphisms form a ring. For example, if $\phi, \sigma$ are endomorphism, then also $\phi + \sigma$ is an endomorphism that send a point $P$ to $\phi(p) + \sigma(p)$. Another endomorphism that forms from $\phi, \sigma$ is $\phi \ast \sigma$ that send a point $P$ to $\phi(\sigma(p))$. For an integer $n$, a trivial endomorphism, $\phi_n$, is an endomorphism that send a point $P$ on $E$ to $nP$.

The trivial endomorphism is isomorphic to the ring $\mathbb{Z}$. If there exists a non-trivial endomorphism on $E$, then $E$ is said to have a Complex Multiplication and the endomorphism is isomorphic to an order in an imaginary quadratic number field. We are using the following fact that if $E$ is an elliptic curve over the rational, with a complex multiplication by an order in $\mathbb{Q}(\sqrt{D})$ where $D$ is called the discriminant.

We can find quite easily now the order of $E$ over $F_q$.

We search for solutions to the diophantic equation $(4p = s^2 + |D| t^2)$ the equation is calculate using the Cornacchia-Smith algorithm. If such $s, t$ doesn’t exists, we then select another $D$. On the other hand, if such a pair exists, then the curve order is $p + 1 \pm s$. One of the orders is belong to the curve and another to its twisted curve.

**The Cornacchia Algorithm**

The Cornacchia algorithm (which in fact was discovered by H. Smith), is an efficient and elegant algorithm to solve the well-known Diophantine problem:

Given an integer $d > 0$, and an odd prime $p$, what are the $x, y$ (if they exists) that solves the equation $x^2 + dy^2 = p$.

1. Notice: $D$ must be a square of mod-p (i.e. $D = (\%)^2 (\mod p)$)

The suggested algorithm:

1. find $r$ - a root mod p of $d$
2. by using the Euclid’s algorithm on $(p,r) = x$, until we get $x < \sqrt{p}$.
3. calculate: $y = \sqrt{\frac{p-r^2}{|D|}}$
   - if $\sqrt{\frac{p-r^2}{|D|}}$ has a result, return $(x, y)$
   - else, return “no such $(x, y)$”

Our problem is a bit different: for a negative discriminant $D$, such that $D = 0, 1 (\mod 4)$ what are the $x, y$ (if they exists) that solves the equation $x^2 + 4py^2 = 4p$, so the original algorithm needs a minor adjustments, but it keeps its efficiency and elegance.
The J-Invariant

For an elliptic curve, defined by the equation $y^2 = x^3 + ax + b$, the j-invariant of the equation is defined as:

$$j = j(E) = 1728 \frac{4a^3}{4a^3 + 27b^2}$$

There are two special cases for the j-invariant,

- J=0, in this case the elliptic curve equation is written as: $y^2 = x^3 + b$
- J=1728, in this case the elliptic curve equation is written as $y^2 = x^3 + ax$.

If we find the value of the j-invariant where $j \neq 0,1728$, we can easily find the value of the curves parameters according to the following equations:

$$a = \frac{-3j}{j - 1728}, b = \frac{2j}{j - 1728}$$

So, we can rewrite the curve equation as:

$$y^2 = x^3 + \frac{-3j}{j - 1728} x + \frac{2j}{j - 1728}$$

Additional fact is that if two elliptic curves over field $F_q$ have the same j-invariant, then they are twisted to each other.

We are now facing additional problem. The problem is how to find the J-invariant.

Hilbert Class Polynomial:

For a given D, we can calculate the Hilbert class polynomial, $H_D(x)$. The Hilbert class polynomial, is the minimal polynomial of the set of j-invariants of elliptic curve with discriminant D.

When we find a root modulo our prime p, we can easily find the elliptic curve.

In this project we consider two way of using the Hilbert class polynomial.

The first one was to take the Hilbert class polynomial from pre-generated list.

The second way was to calculate the Hilbert class polynomial on the fly.

When calculating the Hilbert class polynomial in a numeric way, we need to work with arbitrary precision on the complex field.

We have implemented the second way, calculating the Hilbert class polynomial on the fly.
**Elliptic curve generation algorithms:**

1. Randomly find a prime \( p \) with the proper size.
2. Chose a negative Discriminant \( D \) \((-500 \leq D \leq -4)\)
3. Find \( s, t \) such that \( 4p = s^2 + |D| t^2 \). If no solution can be found go to step2. If all the discriminants aren’t suitable go to step1.
4. Check if \( u_1 = p + 1 + s \) or \( u_2 = P + 1 - s \) are cryptographically strong. If not go to step 2.
5. Calculate Hilbert class polynomial \( H_D(x) \)
6. Obtain a root \( j \) of \( H_D \mod p \)
7. Set \( c = j(1728)^{-1} \mod p, r = -3c \mod p, s = 2c \mod p \).
8. Return the curve and its twist. \( \{(a, b)\} = \{(r, s), (rg^2 \mod p, sg^3 \mod p)\} \) where \( g \) is a QNR.

**Implementation:**

We’ve implemented our project in C++ in Linux environment. We only used libraries for basic operations on big numbers. The elliptic curve arithmetic and polynomial operations we’ve implemented ourselves. The libraries we used:

- **GMP**: GNU MP is a portable library written in C for arbitrary precision arithmetic on integers, rational numbers, and floating-point numbers. It aims to provide the fastest possible arithmetic for all applications that need higher precision than is directly supported by the basic C types.
- **MPC**: MPC is a portable library written in C for arbitrary precision arithmetic on complex numbers providing correct rounding. Ultimately, it should implement a multiprecision equivalent of the C99 standard.

**We Implemented:**

Prime field arithmetic:

We implemented a wrapper class for elements in \( F_q \). In this wrapper we implemented arithmetic and comparison action on \( F_q \) elements. We also implemented the Jacobian coordination class. In the Jacobian coordination we implemented the arithmetic and comparison actions.

In this section we implemented the following algorithms:

- Finding square root modulo prime \( p \).
- Cornacchias algorithm, find \( s, t \) such that \( s^2 + |D| t^2 = 4p \)
Polynomials:

We implemented two classes of polynomials. The first class is a class of complex numbers with high precision polynomials. The second class is polynomials over $F_q$.

In this section we implemented the following algorithms:

- Generation Hilbert Class Polynomial for a given discriminant and prime $p$.
- Finding the gcd of two polynomials.
- Finds root of a polynomial over $F_q$.
- Calculate modular exponent of two polynomials.

Elliptic curve:

We implemented the curve operations:

- Efficient point adding using Jacobian coordination.
- Efficient scalar multiplication using NAF.
- Find a random point on the elliptic curve

Cryptosystem:

We implemented:

- Decryption ciphertext using Menezes-Vanstone variant of the ElGamal cryptosystem.
- Key generation
How to use:

The main Library is the EllipticCurve.h

This library is contains the arithmetic function for handling the elliptic curve.

In this library is providing an option to generate a random elliptic curve,
or to build an elliptic curve for a given p.

Moreover in this library you can generate a random point on the elliptic curve.

Elliptic curve generation:

for generation an elliptic curve, you can use one of the following functions:

EllipticCurve (mpz_class _p):
in this function, you should enter the characteristic of the desire prime field.
this function return print an error massage , if she wasn't able and generated an elliptic curve to this prime.
in the generation procces we use the CM method, for discriminant >-500, so not for every prime exist a curve with the desire cryptography strength.

EllipticCurve (int numBit):
in this function, the user enter the bit size of the desire curve.
the function will generate a curve with log(p) = numBits.

Crypthsystem:
The next important library is the "ElGamal.h

This library is contains the encryption and decryption functions.
The constactor of the main function in MenezesVanstone(EllipticCurve * _ecc);
the user sould pass to the function the elliptic curve that was generated before.
before anyone can use the function for encryption/decryption, one needs to pass another parameters to the MenezesVanstone class.
The encryption function is:

```cpp
void encrypt(string massage)
```

This function will encrypt the given massage and will print the ciphertext.

before using this function, the person who wants to encrypt should call to the
next two functions:

```cpp
void setPoint(JacPoint _alpha):
the user should enter a point on the elliptic curve (the user can generate a point using the
randomPoint function in the ellipticCurve.h)
```

```cpp
void setPrivateKey(mpz_class _a):
the user should enter the private key for encryption.
```

```cpp
void decrypt(string massage):
This function will decrypt the given massage and will print the plaintext.
before using this function, the person who wants to encrypt should call to the
void setPrivateKey(mpz_class _a); function.
```
Results

Time to generate an elliptic curve (average of 15 measures on a standard mac laptop):

Our goal in the project was to generate a cryptographically strong elliptic curve in a few seconds. A sufficient strength is about 160 bits, and we’ve generated a 500 bits EC (3 times more the basic security) in less than 2 seconds.

This graph shows that we’ve achieved this goal.

Time to encrypt the file:

The second goal was to encrypt a file of size of 3KB in a few seconds. In this graph can be seen that that goal was also achieved, and the encryption of such a file took about 3 seconds.

we’ve also tried to see if the encryption is efficient enough to encrypt larger files, and to our opinion its practical only for
highly sensitive files due to its relatively long encryption time, and the fact that the strength of the EC doesn’t effect the encryption time much.

Time to decrypt a file:

In this graph we can see that the decryption of the file is also not affected by the strength of the EC, and its time is much less than the encryption, to our opinion, because it has much less scalar multiplication than the encryption.

Size of the encrypted file:
As can be seen, the size of the encrypted files is relatively small (less than other encrypted files, like the original ElGamal).
Further work:

- Polynomial creation – an improvement to the polynomial creation can be achieved by using Webber polynomials instead on Hilbert class polynomials due to its much smaller coefficients.
- Parallelization – in many modern computers we have multiple processors, and more common that each processor has several cores. We can use this advantage to do some work simultaneously:
  - In the EC generation, we can check several discriminants at the same time
  - During the encryption and decryption of the file (which takes the most of the time) – we can compute each block separately in another core/processor, and by that to speed up the process.
- GPUs – use the advantages of the modern GPUs (graphical processors) that specialize in parallel computations to gain speed.

Challenge 1:

We were given messages encrypted using a prime field for a given p. Some information on the curve was given. The elliptic curve was generated using Complex Multiplication with \( D > 500 \). The order of the curve is prime number. In addition the parameter a of the curve was -3.

Another curve was generated with a different p, using Complex Multiplication with \( D > 500 \). The order of the curve is also prime, and the a,b parameters of the curve are equal to the private key.

Additional aspect of this challenge is to find the order of the two curves.

Solution:

We used exhaustive search on suitable D(from -500 to -4) , to find the one, who can generate an elliptic curve with a prime order.

After an Exhaustive search on D, we conclude that only one D can generate a curve with a prime order.

We build the proper Hilbert class polynomial and generate an elliptic curve. We then search an isomorphic elliptic curve to the curve we found, which is a parameter is -3.

We used again exhaustive search on D, we found out that only one D is generate a curve with a prime order. We build the proper Hilbert class polynomial and generate an elliptic curve. We then search an isomorphic elliptic curve to the curve we found, which the a and b parameters are equal.

We encrypted the given massages.
First curve parameters:

Discriminant:
D = -451

Prime:
2370257299711480755942704365095122473222823626703179169027911245520826768235
039195925317588249498990422369192621540829926367172647995744676982145926549

A:
2370257299711480755942704365095122473222823626703179169027911245520826768235
039195925317588249498990422369192621540829926367172647995744676982145926546

B:
2085926647631780925378822176218658334804477556342460670095340146892687685191
074508996470038682990100350685491647027517433260581956509969489975598058607

Order:
2370257299711480755942704365095122473222823626703179169027911245520826768233
70082452536556247541533521753344324976736159552258240608676832449747241389

Second curve parameters:

Discriminant:
D = -211

Prime:
2251082047094495627787291281132641724643896341307407676477183997847357440663
919350206909871585400675856451349537641424907059152913165110634561643

A:
158868263088313511019727355032607505643500363143438077152125272227169765658
195684326572444222016709719729017755769186770503640424309461528000525

B:
158868263088313511019727355032607505643500363143438077152125272227169765658
195684326572444222016709719729017755769186770503640424309461528000525

Order:
2251082047094495627787291281132641724643896341307407676477183997847357442757
538436521783653765917762113873506979695861344665806760697023314395293
The encrypted massage:

A natural nuclear fission reactor is a uranium deposit where analysis of isotope ratios has shown that self-sustaining nuclear chain reactions have occurred. The existence of this phenomenon was discovered in 1972 at Oklo in Gabon, Africa, by French physicist Francis Perrin. The conditions under which a natural nuclear reactor could exist had been predicted in 1956 by Paul Kuroda. The conditions found were very similar to what was predicted.

Oklo is the only known location for this in the world and consists of 16 sites at which self-sustaining nuclear fission reactions took place approximately 2 billion years ago, and ran for a few hundred thousand years, averaging 100 kW of power output during that time.

In May 1972 at the Pierrelatte uranium enrichment facility in France, routine mass spectrometry comparing UF-6 samples from the Oklo Mine, located in Gabon, Central Africa, showed a discrepancy in the amount of the U-235 isotope. Normally the concentration is 0.720%; these samples had only 0.717% - a significant difference. This discrepancy required explanation, as all uranium handling facilities must meticulously account for all fissionable isotopes to assure that none are diverted for weapons purposes. Thus the French Commissariat a l'Energie atomique (CEA) began an investigation. A series of measurements of the relative abundances of the two most significant isotopes of the uranium mined at Oklo showed anomalous results compared to those obtained for uranium from other mines. Further investigations into this uranium deposit discovered uranium ore with a U-235 to U-238 ratio as low as 0.440%. Subsequent examination of other isotopes showed similar anomalies, such as Neodymium and Ruthenium.

This loss in U-235 is exactly what happens in a nuclear reactor. A possible explanation therefore was that the uranium ore had operated as a natural fission reactor. Other observations led to the same conclusion, and on September 25, 1972, the CEA announced their finding that self-sustaining nuclear chain reactions had occurred on Earth about 2 billion years ago. Later, other natural nuclear fission reactors were discovered in the region.

The natural nuclear reactor formed when a uranium-rich mineral deposit became inundated with groundwater that acted as a neutron moderator, and a nuclear chain reaction took place. The heat generated from the nuclear fission caused the groundwater to boil away, which slowed or stopped the reaction. After cooling of the mineral deposit, short-lived fission product poisons decayed, the water returned and the reaction started again. These fission reactions were sustained for hundreds of thousands of years, until a chain reaction could no longer be supported.

Fission of uranium normally produces five known isotopes of the fission-product gas xenon; all five have been found trapped in the remnants of the natural reactor, in varying concentrations. The concentrations of xenon isotopes, found trapped in mineral formations 2 billion years later, make it possible to calculate the specific time intervals of reactor operation: approximately 30 minutes of criticality followed by 2 hours and 30 minutes of cooling down to complete a 3-hour cycle.
A key factor that made the reaction possible was that, at the time the reactor went critical, the fissile isotope U-235 made up about 3% of the natural uranium, which is comparable to the amount used in some of today’s reactors. (The remaining 97% was non-fissile U-238.) Because U-235 has a shorter half life than U-238, and thus decays more rapidly, the current abundance of U-235 in natural uranium is about 0.7%. A natural nuclear reactor is therefore no longer possible on Earth without heavy water.

Challenge 2:

Generate a strong elliptic curve with a size of 200 bit.

Show his parameters (a, b, order) and find a point on the curve.

Solution:

The curve parameters are:

Prime:
1403292601815539771887597189458782069382115161558663452453551

A:
207667927199199628760720303806471733541759242514528188107825

B:
797083116410893428751251257101540223893570612696090176230484

Order:
140329260181553977188759718945747445150748331713362233351364

Point:

X coordination
491197511675322521211131104767772721600217237403491731212115

Y coordination
1014215867018909959318696179517537027141936371238239612277946
REFERENCES

- Guide To Elliptic Curve Cryptography - Hankerson, Menezes, Vanston
- Prime Numbers A Computational Perspective - Crandall, Pomerance
- A Course in Computational Algebraic Number Theory - Cohen
- Counting Points on Elliptic Curves Over Zp Sundriyal
- On the Efficient Generation of Elliptic Curves over Prime Fields - Konstantinou, Stamatiou, Zaroliagis
- A Weakness of the Menezes-Vanstone Cryptosystem - Kiefer
- Generating Elliptic Curve of Prime Order - Savas, Schmidt, Koc
- Generating Elliptic Curves over finite fields by Po-Yi Huang, Ming-Luan Hsieh
- www.wikipedia.org