REPORT

ON

ENCRYPTED GMAIL

BY

2017A7PS0057P **HARPINDER JOT SINGH**

2017A7PS0080P **VISHAL MITTAL**

As a part of the course

Cryptography(BITS F463)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
(Sem-I, 2019-20)

Submitted to:
Dr. Ashutosh Bhatia

Objective

To build a web app named **securemails** which will help users to send emails using Gmail but in encrypted format and the receiver decrypts it back using our web app. We implemented the secure key exchange mechanism through asymmetric cryptography (Elliptic Curve) and message encryption by Advanced Encryption Standard (AES) using Stanford Javascript Crypto Library (SJCL).

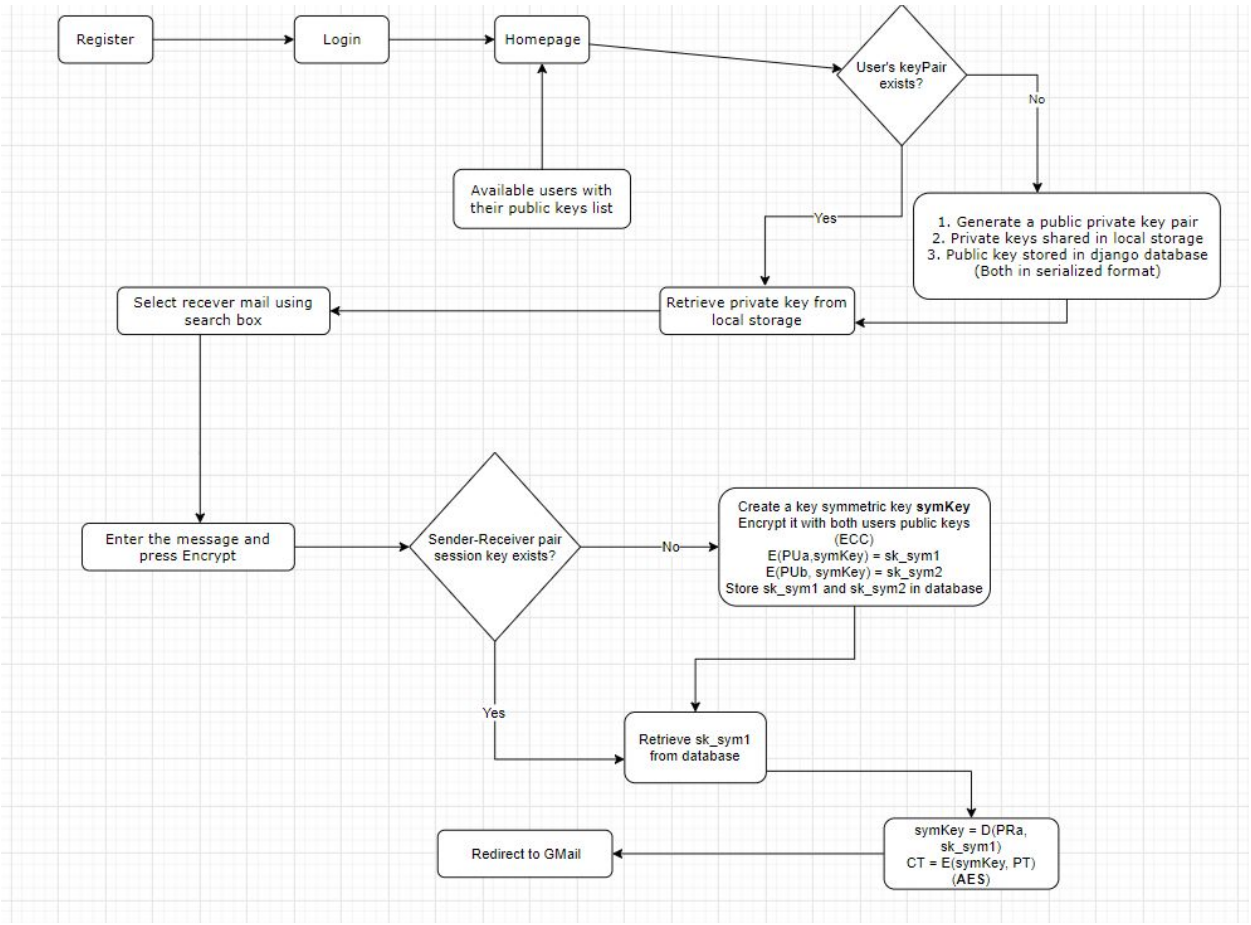
Motivation:

While sending emails on Gmail we cannot be sure whether the information remains confidential since Google can still access the mails. We wanted to make sure that the content is only accessible to the desired recipient and no one else has access to the message shared.

Methodology

1. Built UI on the top of the basic UI available here
2. Used SJCL library provided by Stanford for encryption and decryption purposes.
3. Used Django as the backend framework and it's native SQLite database for storing metadata related to encryption
4. Achieved secure key exchange through asymmetric cryptography (ElGamel through Elliptic Curve).

Flowchart:



Implementation:

1. User registers on the portal.
2. User logins into his account and is redirected to the home page.
3. The home page contains a text box for typing in the message to be sent and a list of users already registered on the portal navigable through a search box.
4. If the user's public-private key pair does not already exist, a new public-private key pair is generated. Private key is stored in local storage while the public key is stored in the Django database - both in serialized form.

5. Otherwise, the private key of the user is retrieved from the local storage and public key from Django database
6. User selects the recipient from the list as receivers using search box.
7. User enters the message and presses the 'Encrypt' button.
8. If sender-receiver pair session key does not already exist, a symmetric key **`symKey`** is created.

   It is encrypted with both users' public keys (through Elliptic Curve Cryptography)

   ```
   E(PUa, symKey) = sk_sym1
   E(PUb, symKey) = sk_sym2
   ```

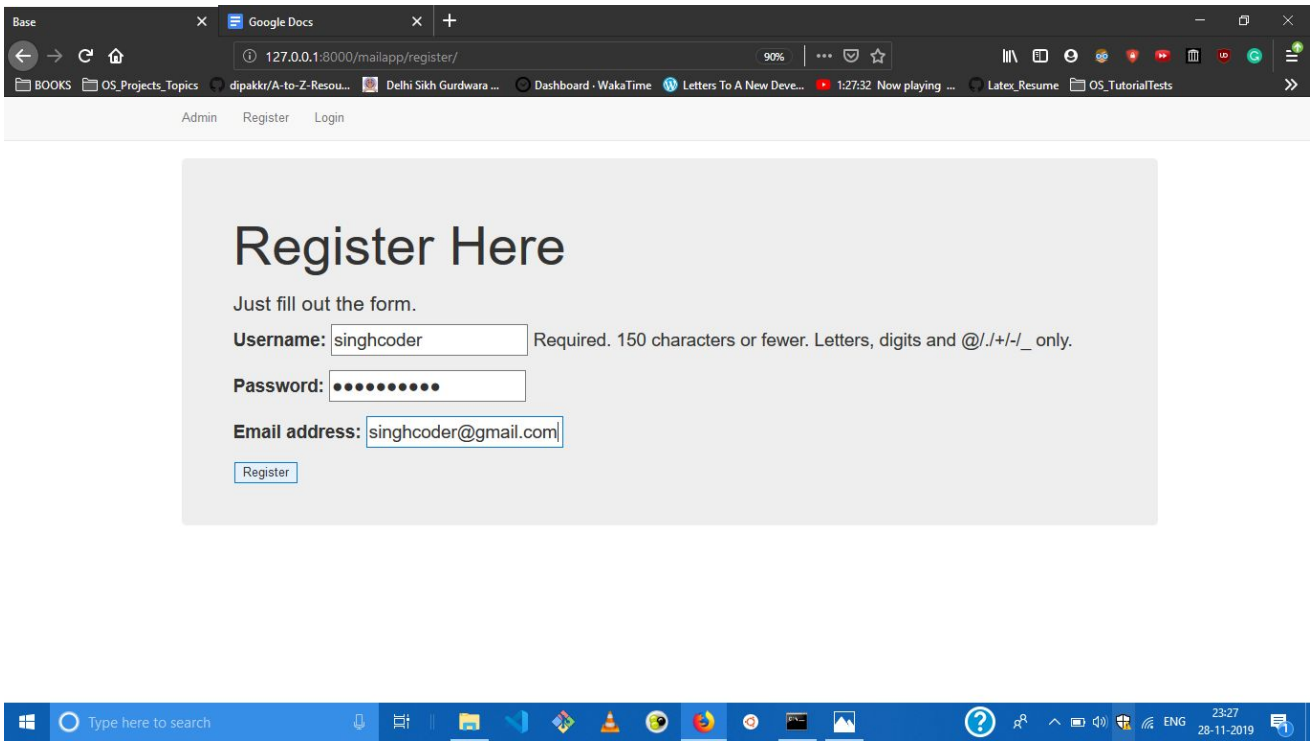
   sk_sym1 and sk_sym2 are stored in the database.
9. Otherwise, retrieve sk_sym1 (Encrypted public key of sender) from the database.
10. The symmetric key for encrypting the message (session key) is deciphered using

    ```
    symKey = D(PRa, sk_sym1)
    CT = E(symKey, PT)        (AES)
    ```
11. User is redirected to Gmail to send the encrypted message.
12. Receiver logins into the portal and enters in browser the link he receives on GMail and he gets the message decrypted.

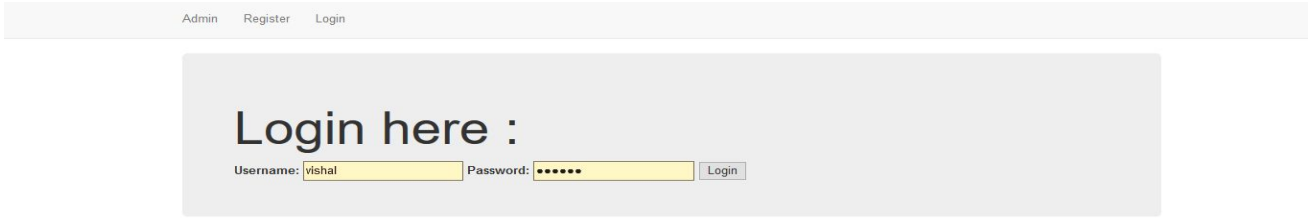
Outcome:

The web application is hosted here [https://securemails.pythonanywhere.com]

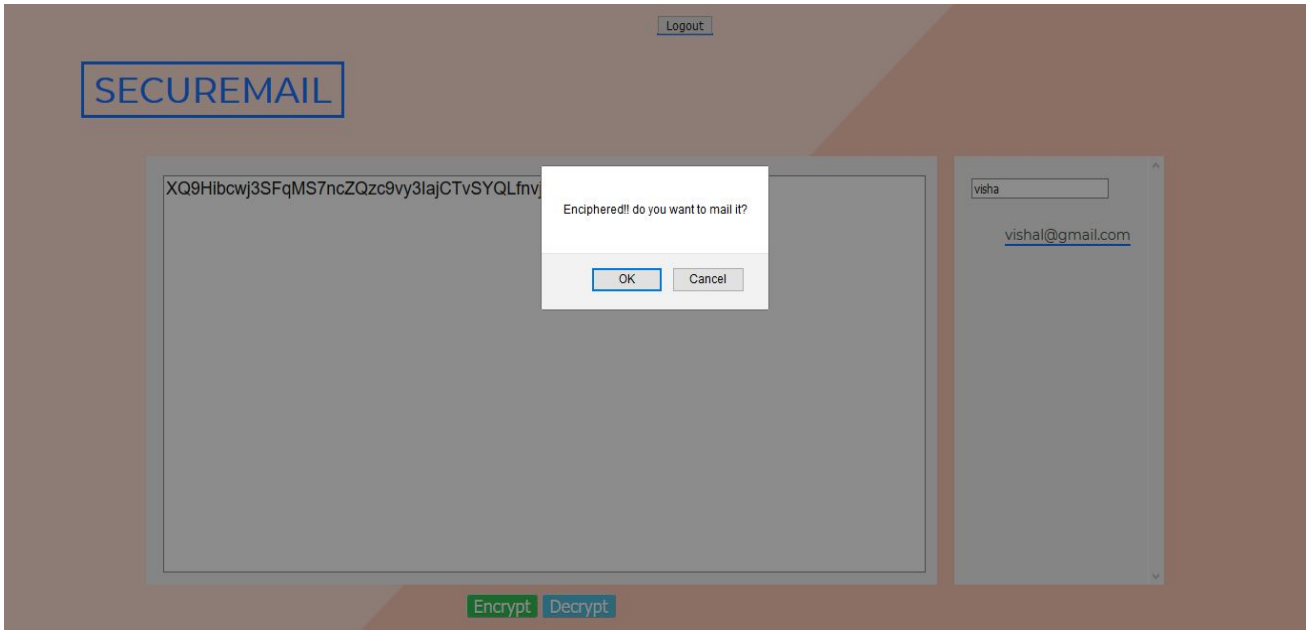
1. Register page



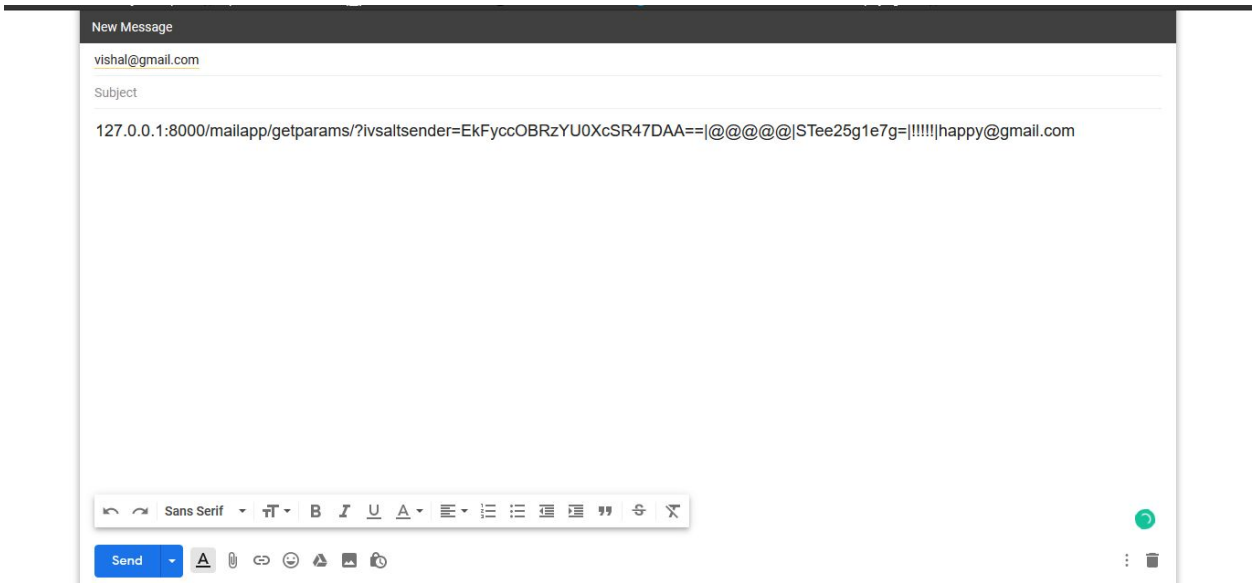
2. Login page



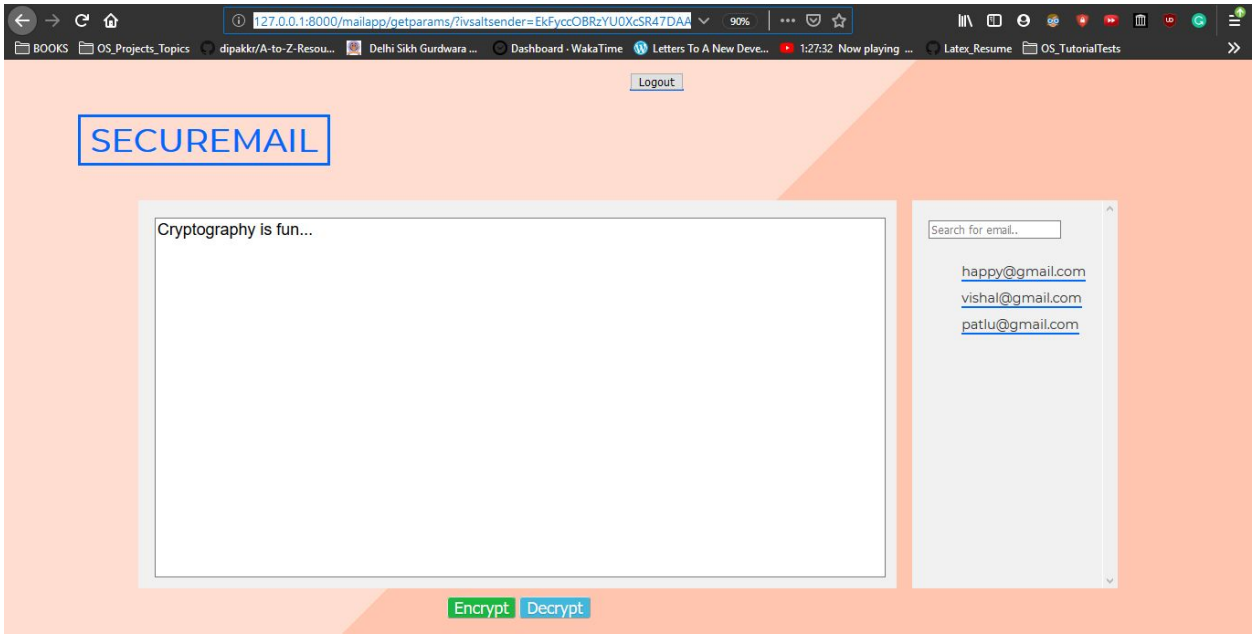
3. Encryption of message on the sender's side



4. Sender is redirected to Gmail



5. Decryption of message on the receiver's side



Tech stack used

HTML, CSS, JS : For frontend

Python, Django: For backend

SJCL (Stanford Javascript Crypto Library): For encrypting and decrypting the messages and for key exchange

References

- http://bitwiseshiftleft.github.io/sjcl/doc/
- https://github.com/bitwiseshiftleft/sjcl
- https://bitwiseshiftleft.github.io/sjcl/demo/
- https://courses.csail.mit.edu/6.857/2016/files/37.pdf
- https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

Elliptic Curve Cryptography

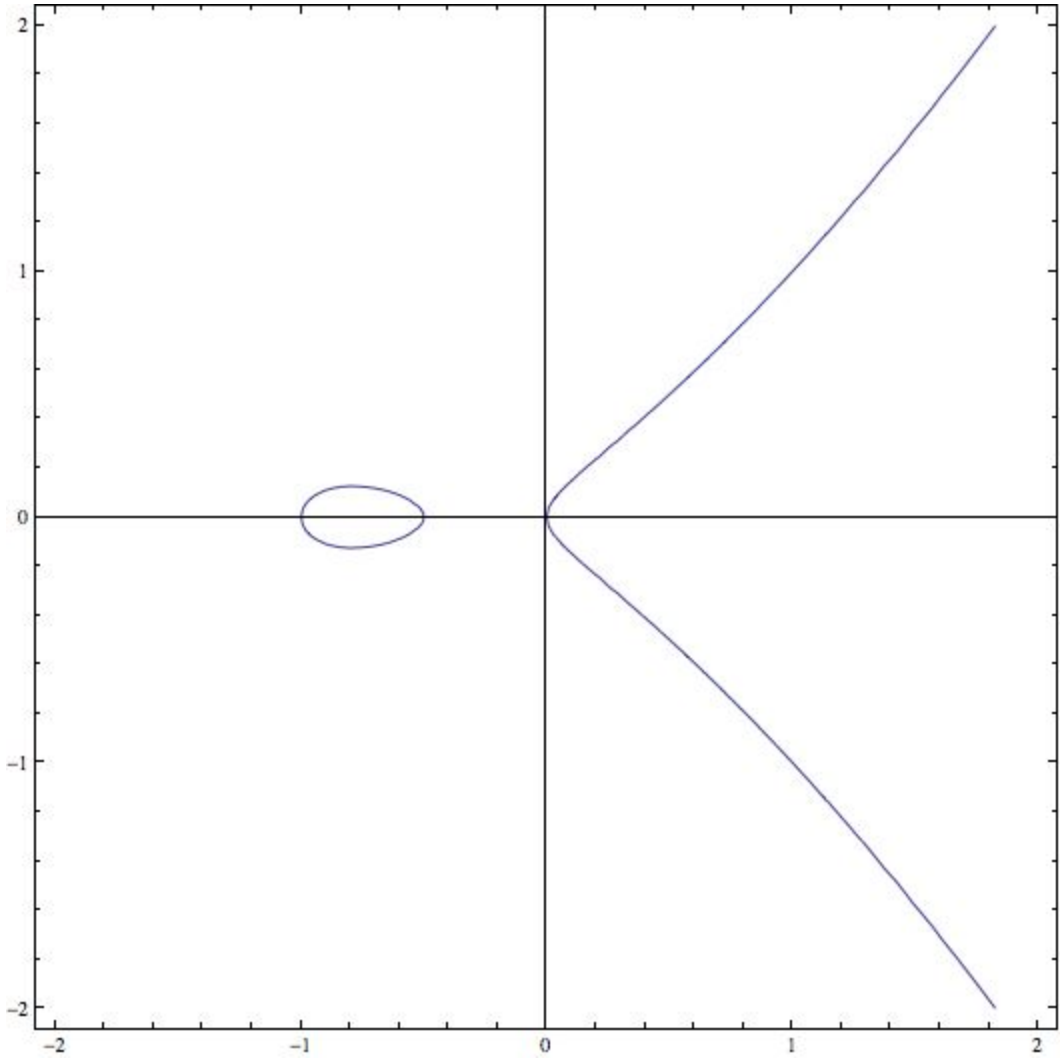
An elliptic curve is as the set of all solutions to a specific kind of polynomial equation in two real variables, x, y. Specifically, the equation has the form:

$$y^2 = x^3 + ax + b$$

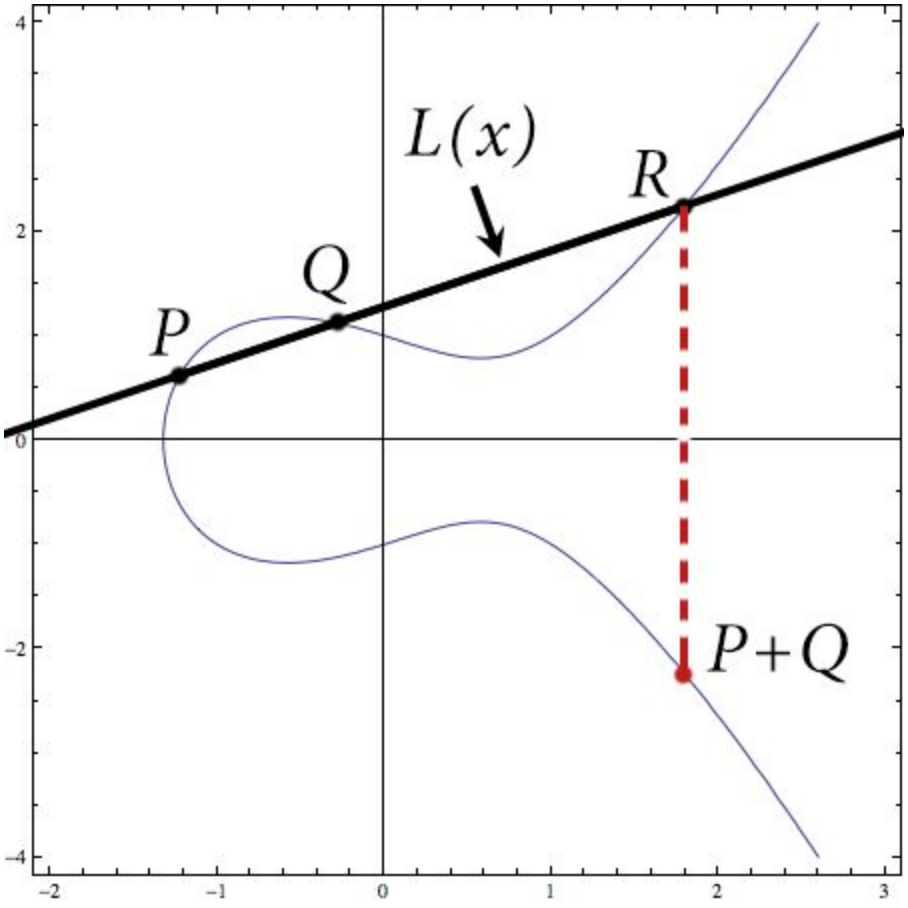
Where a, b are real numbers such that

$$-16(4a^3 + 27b^2) \neq 0$$

$$y^2 = \frac{x(x+1)(2x+1)}{6}$$



The Algorithm to Add Points


L(x)
R
Q
P
P+Q
4
2
0
−2
−4
−2
−1
0
1
2
3