Implementation of 5PM(5ecure Pattern Matching) on Android Platform

Cetin Sahin
cetin@cs.ucsb.edu CS 290G/S13

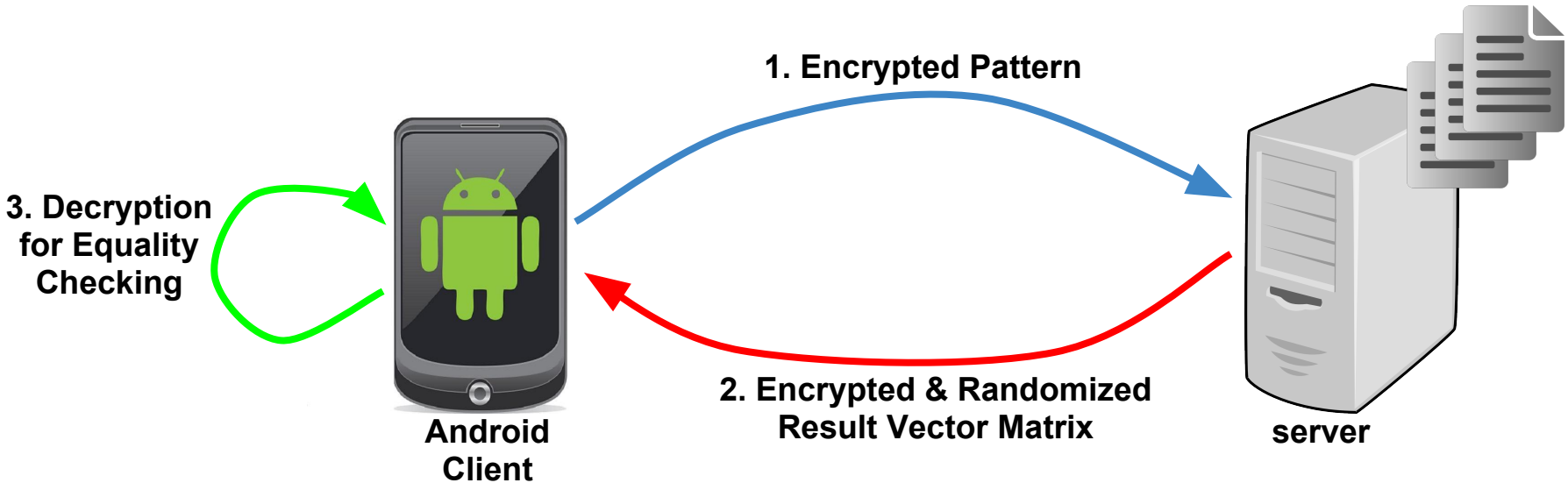
Overview

- Main Objective:
  - Search for a pattern on the server securely
  - The answer at the end -> either YES it is found or NO it is not found
- Algorithm:
  - 5PM HBC model
    - Homomorphic ElGamal Cryptosystem
      - Standard Multiplicative (Mod P) Group
      - Elliptic Curve Additive Group

Overview cont...

- Implementation Platform:
  - Java -> BigInteger Class
  - Client -> Android Device
  - Server -> Glassfish Server with JAX-WS proxy

- System Architecture



ElGamal Based on Mod P

- Based on large primes p and q such that $q|(p-1)$
- Operations are performed in the subgroup G_q of Z_p
- g of G_q is a generator
- private key sk_C element of G_q
- public key $pk_C = g^{skC}$
- $E(m) = (c1, c2)$ such that $c1 = g^r \pmod p$ and $c2 = g^m \cdot pk_C^{\ r} \pmod p$

ElGamal Based on EC

- Over the field of p elements GF(p) where p is a large prime and in generally range of 160 <= |p| <= 256
- The private key is the integer sk_C which is element of Zn
- The public key is a point on the curve pk_C where $pk_C = d \times P$ (P is a base point)
- E(m) = (R1, R2) such that R1 = r x P and R2 = m x P + r x Q

Edwards Curve

$x^2 + y^2 = 1 + d.x^2.y^2$

where d is not equal to 0 or 1

Addition Law:

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + x_2 y_1}{1 + d x_1 x_2 y_1 y_2} , \frac{y_1 y_2 - x_1 x_2}{1 - d x_1 x_2 y_1 y_2} \right)$$

- Neutral Point O = (0, 1)
- Inverse of (x, y) is (-x, y)

Algorithm - The Setup

- Alphabet for a fixed size
- Client has a pattern *p* with a size of *m* -> defined over alphabet
- Client has both public (pk_c) and private key (sk_c)
- Homomorphic Element Addition: $E(a) \oplus E(b) = E(a + b)$
- Homomorphic Scalar-by-Element Product: $\alpha \otimes E(a) = E(\alpha\, a)$
- With given *E(a)* and *E(-m)*, homomorphic equality checking heq(a, m):

$$heq(a, m) = \begin{cases} \text{yes} & \text{if } D(E(a) \oplus E(-m)) = 0 \\ \text{no} & \text{if } D(E(a) \oplus E(-m)) \neq 0 \end{cases}$$

- Server has a text T with a size of n
- Server just has public (pk_c)

Algorithm - How It Works ?

Client

1. Computes Character Delay Vector (CDV) from pattern for each element in the alphabet
2. Encrypt CDV vector either using ModP or Elliptic Curve ElGamal algorithm
3. Also encrypts *-m* as E(-m) and send E(M_{CDV}), E (-m) and public key to the server

Server

4. Transforms text to a matrix
5. Computes Activation Vector E(AV) with the operations Stretch, Cut and Column Sum
6. Checks the entries of E(AV) to find encrypted m. Accomplished by doing h. element addition on E(-m)
7. Randomize vector by scalar-by-element operation
8[optional]. Permute in order not to reveal match indexes

Algorithm - How It Works ?

Client

Server

9. Send E(a AV) where a is random scalar to client

10. Applies decryption operation to the components of E(a AV)

11. If a 0 exist at position i of E(a AV), then the existence of a 0 in E(a AV) indicates a **MATCH**

$$
\begin{aligned}
D(E(\alpha_1\, av_1)) &\stackrel{?}{=} 0 \quad \text{no} \\
D(E(\alpha_2\, av_2)) &\stackrel{?}{=} 0 \quad \text{no} \\
D(E(\alpha_3\, av_3)) &\stackrel{?}{=} 0 \quad \text{yes} \\
D(E(\alpha_4\, av_4)) &\stackrel{?}{=} 0 \quad \text{no} \\
D(E(\alpha_5\, av_5)) &\stackrel{?}{=} 0 \quad \text{no} \\
D(E(\alpha_6\, av_6)) &\stackrel{?}{=} 0 \quad \text{no} \\
D(E(\alpha_7\, av_7)) &\stackrel{?}{=} 0 \quad \text{no} \\
D(E(\alpha_8\, av_8)) &\stackrel{?}{=} 0 \quad \text{no}
\end{aligned}
$$

Implementation and Testing

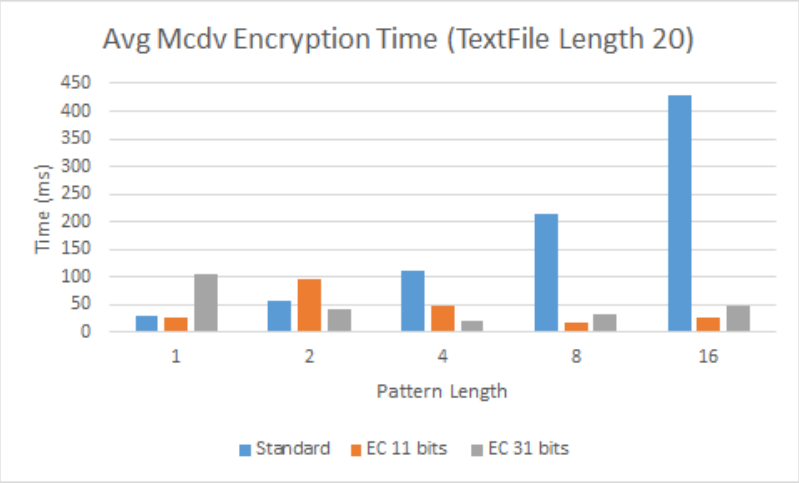
- Java
  - Android SDK -> 4.2.0 min API
  - JAX-WS (XML based communication) Web Service runs on Glassfish
  - BigInteger Class -> Built in modular arithmetic
- Server
  - 4 core @3.00 GHz
  - 12 GB RAM
- Standard Multiplicative (Mod P) Group
  - |p| = 1024 bits
- Elliptic Curve Additive Group
  - Two curves with 11 and 31 bits orders
- Alphabet: {A, C, G, T}

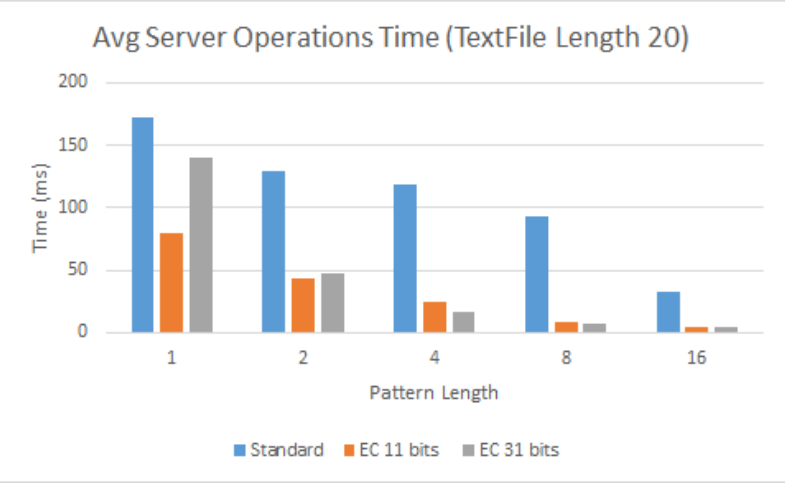
Implementation and Testing

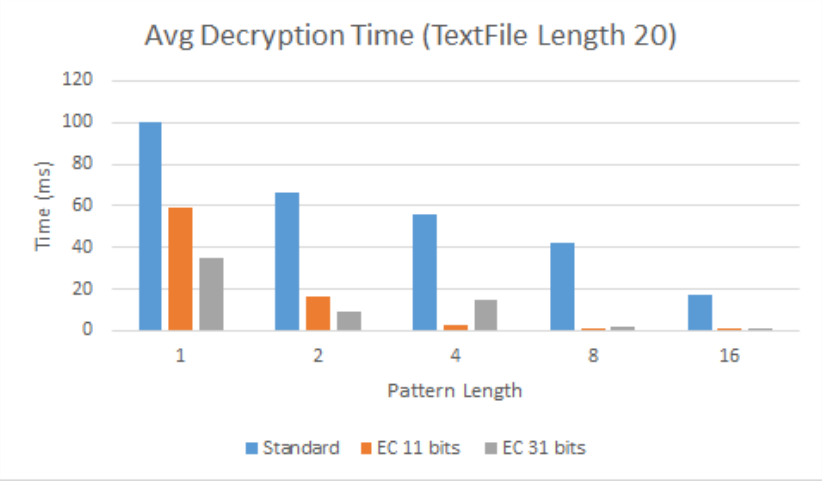
- Text Files in the server
  - File 1 has 20 Character
  - File 2 has 100 Character
  - File 3 has 500 Character
- Every experiment was repeated 10 times and their averages are presented in the following experimental results sections.

Experimental Results

Average Matrix encryption, server operations and decryption times regarding different pattern lengths when text file contains 20 characters

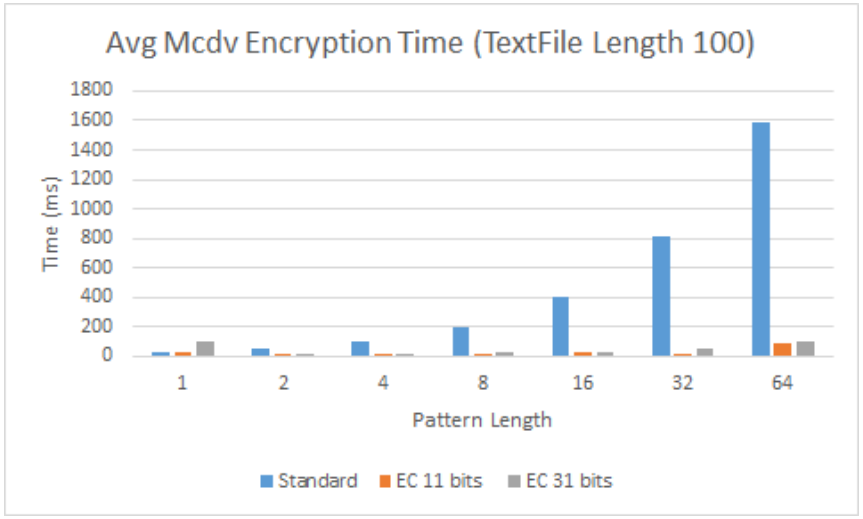


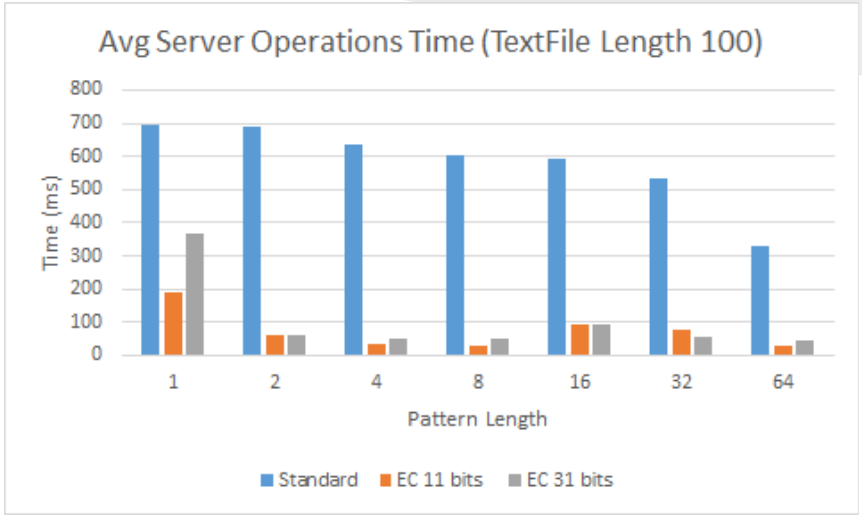


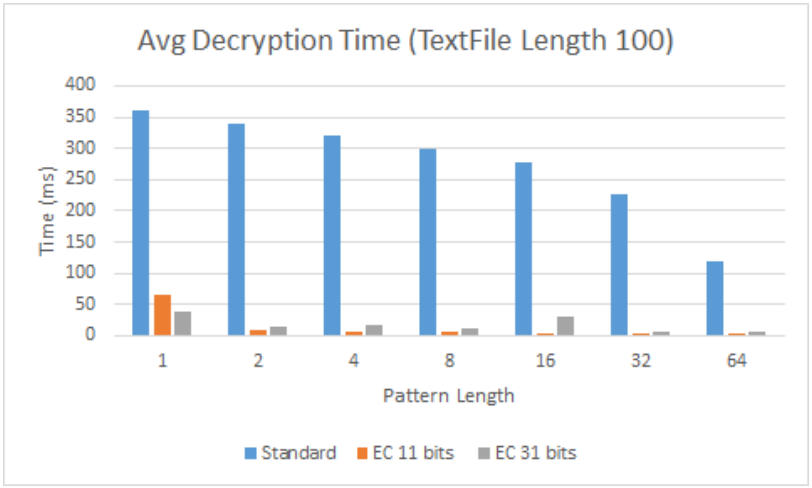


Experimental Results

Average Matrix encryption, server operations and decryption times regarding different pattern lengths when text file contains 100 characters

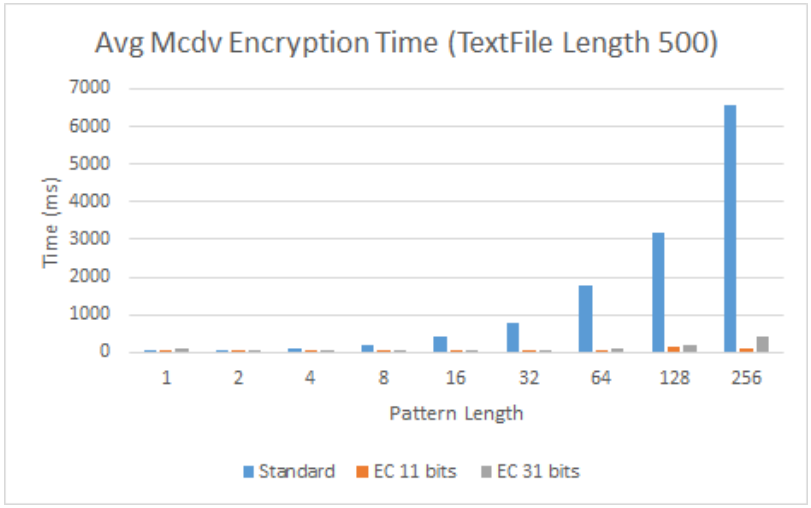


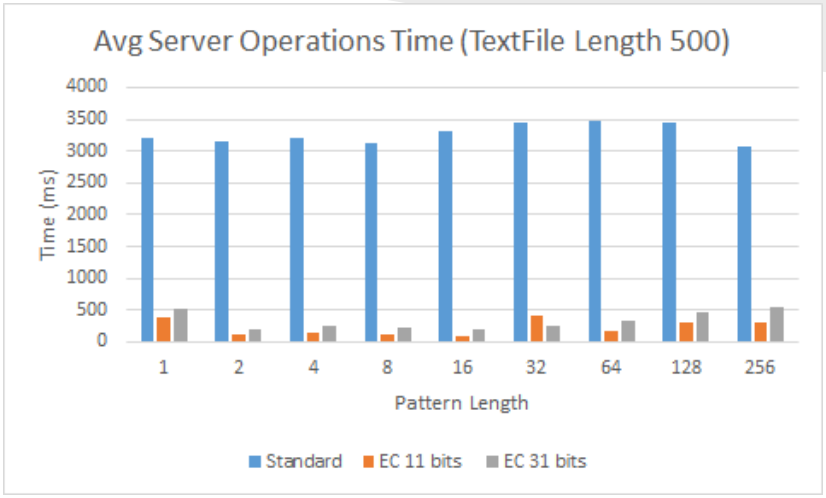


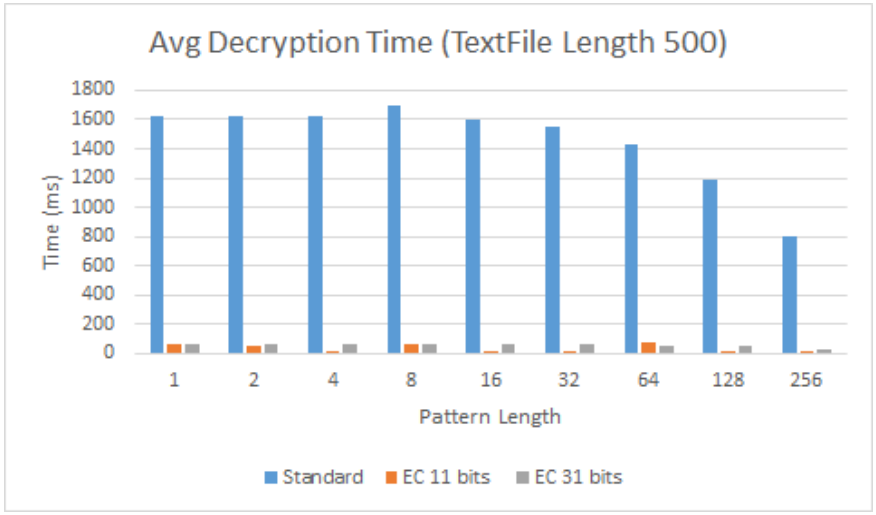


Experimental Results

Average Matrix encryption, server operations and decryption times regarding different pattern lengths when text file contains 500 characters







Analysis

- Encryption time depends on the pattern length m. CDV vector has a dimension |E| x m where |E| is the size of an alphabet.
  - Thus, when m increases the average encryption time for matrix also increases as we can infer from results.
  - Elliptic curve implementations outperform standard mod P group implementation in encryption of matrix, especially with pattern lengths greater than 4.
  - Elliptic curve with 11 bits order has better encryption time regarding elliptic curve with 31 bits order but EC with 31 bits provides stronger security.
  - In order to encrypt matrix with pattern length of 256:
    - Standard Mod P group needs almost 6.5 seconds
    - EC 11 bits needs 111 milliseconds
    - EC 31 bits needs 401 milliseconds
    - EC 31 performs factor of 16 better than standard mod p group

Analysis

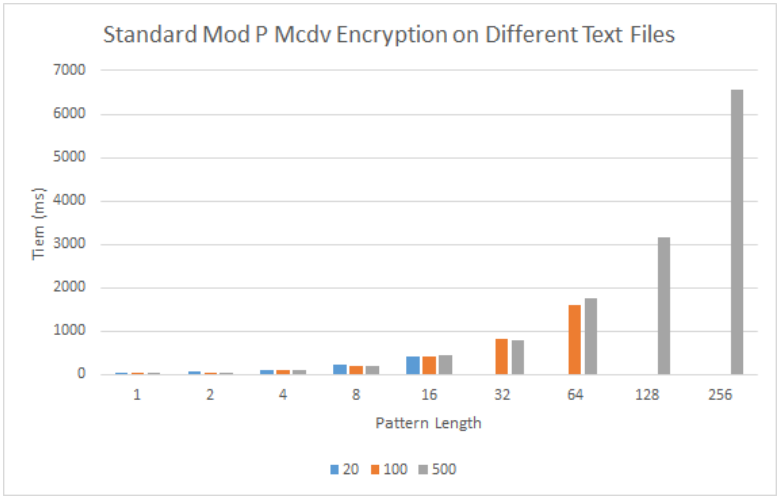
- Server operations contains Stretch, Cut, Column Sum and vector randomization(scalar element multiplication)
- Cut operation reduces the matrix to the dimension of n x (n - m - 1) where n is the number of characters in the text file. Thus, larger pattern size generates smaller matrices to operate.
  - The average time for server operations usually decreases with increasing pattern lengths. Can be observed especially with experiments on text files with lengths of 20 and 100.
  - EC operations again performs a lot better than standard mod p group. We can infer from these results that EC operations complete Column sum(homomorphic element addition) and vector randomization (homomorphic scalar element multiplication) operations in shorter times compared to standard mod p group.
  - As expected, EC with order of 11 bits performs better than EC with 31 bits order but EC with 31 bits provides stronger security.

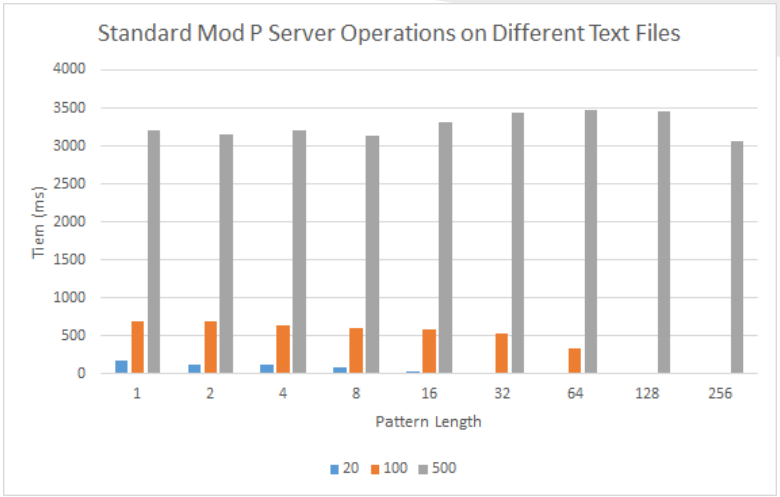
Analysis

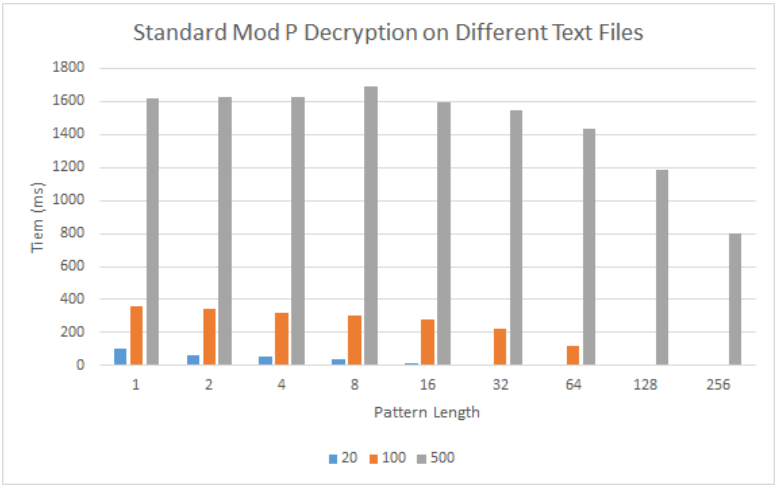
- ColumnSum operation outputs a single row vector with size of (n-m+1). The client receives the result row vector with the same size to decrypt. Thus, larger pattern length produces smaller row vector to decrypt.
  - Decryption time decreases when pattern length increases as we expect.
  - EC operations again perform better than standard mod p group.
  - Especially with the search in short pattern length, the standard mod p operations are so inefficient when the size of text file is large.
  - Same relation between EC with 11 bits and EC with 31 bits holds for decryption also as it does in encryption of matrix and server operations.

Experimental Results

Standard Mod P Group average matrix encryption, server operations and decryption times on different text files with length 20, 100, 500







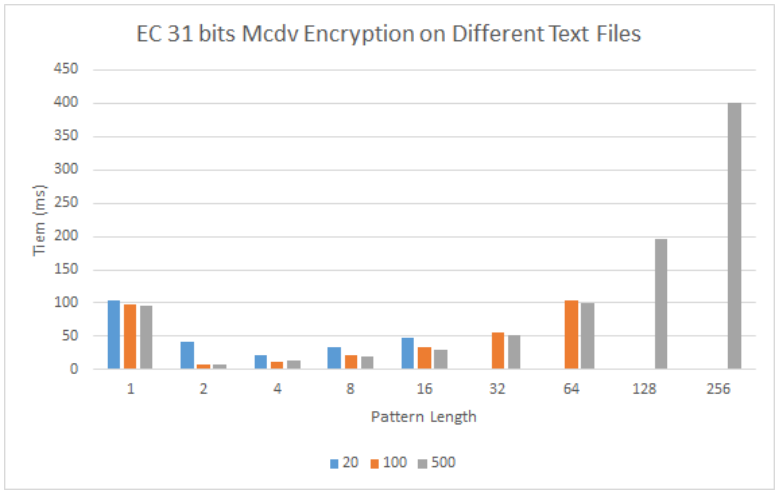
Please note that there is no search query with length greater than 20 for text file with 20 character. Also no search query with length greater than 100 for text file with 100 characters.

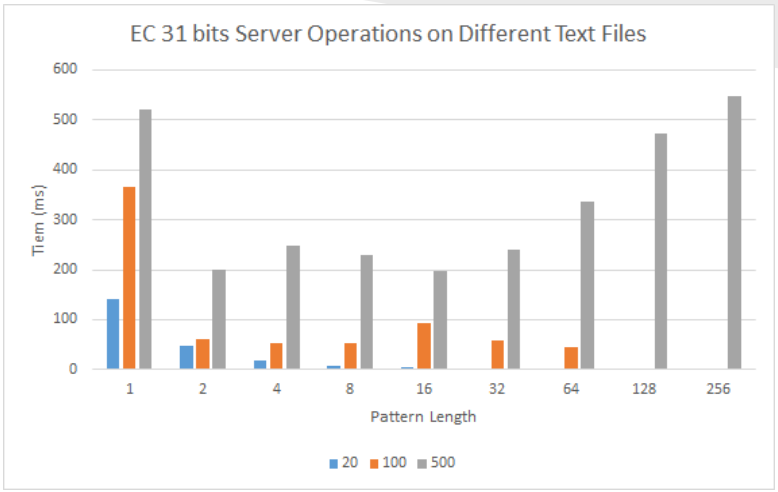
Analysis

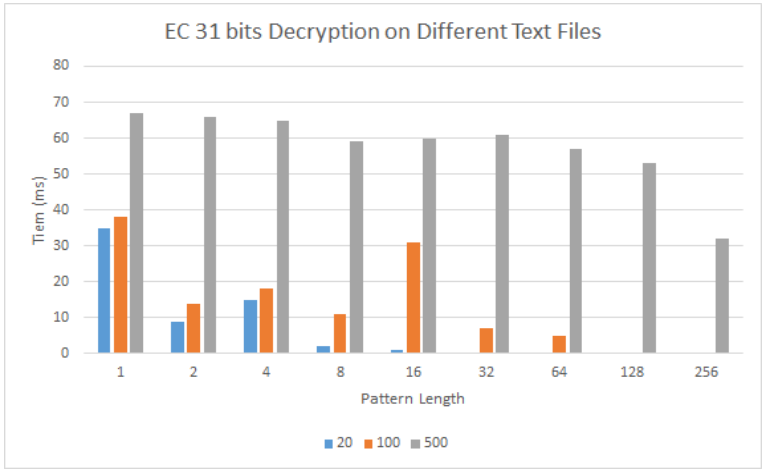
- For standard mod p group
  - Average matrix encryption time linearly increases regarding the increasing pattern length and it is independent from the sizes of text files.
  - When the size of text files are small, the server operations decreases with the increasing pattern lengths. However, this is not true when the size of text file is large (in our case 500).
  - Decryption times decrease as we discussed before as the pattern length increased.
- Standard mod p operations are applicable when you search over a small text files. However, when the size of text file and pattern is large, there is too much non-negligible computational cost.

Experimental Results

Elliptic Curve Group 31-bits average matrix encryption, server operations and decryption times on different text files with length 20, 100, 500







Please note that there is no search query with length greater than 20 for text file with 20 character. Also no search query with length greater than 100 for text file with 100 characters.

Analysis

- For Elliptic Curve group with the order of 31 bits curve
  - Encryption time usually increases with increasing pattern length.
    - When the pattern length is 1, EC group strangely completes the encryption in a longer time than expected. This is possibly caused by random scalar multiplier selection while encrypting data.
  - When the size of text file increases, the average time for server operations also increases. So shorter text file brings less server computation overhead.
  - Shorter text files also result in shorter decryption times. As we expect when the size of text file increases, the resulted vector will have larger size.
- Comparing to the standard mod p group implementation, EC curve groups performs a lot better. The computation times are remarkably reasonable although with a higher order curves we will likely have larger computation times. However, large orders of ECs provides more secure computation environment.

THANKS!