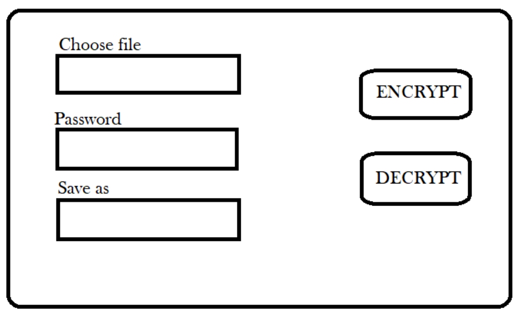
CSCI 462–INTRODUCTION TO CRYPTOGRAPHY PROGRAMMING PROJECT

M. POLAK

Implement a cipher based on family of graphs $D(n, q)$ to learn about cipher encryption and decryption.

1. Overview

(1) Implement a `symmetric-key algorithm` based on special class of graphs
(2) The program shall encrypt/decrypt .txt files
(3) Up to 3 extra points can be earned if the program will work with other type of files (pdf, jpg ...)
(4) The program shall have `user friendly` interface



(5) Perform an attack against the cipher based on a letter frequency count. Encrypt long enough massage. Check if there is any relation between frequency of letters in plaintext and ciphertext.
(6) Check how many percentage of ciphertext will change if you change 5%, 10% of plaintext. Check it for passwords of different length (3,6,9,12,15 characters). Make plots of your results.
(7) Check how many percentage of ciphertext will change if you change 1,2,3 characters of password. Check it for passwords of different length (3,6,9,12,15 characters). Make a plot of your results.

2. Description

`Family of graphs` $D(n, q)$

- Introduced in 1994 by Ustimenko, Lazebnik and Woldar
- Family of expander graphs of `large girth`
- n is a integer ≥ 2
- q is prime power ($q = p^k$, k- positive integer)
- Set of vertices $V = P \cup L$ is a set of n-dimensional vectors over $\mathbb{F}_q$

$$(\bar{p}) = (p_1, p_2, p_3, p_4, ..., p_n) \in P,$$

$$[\bar{l}] = [l_1, l_2, l_3, l_4, ..., l_n] \in L.$$

- $|V| = 2q^n$
- $[\bar{l}]I(\bar{p})$ if and only if

$$\begin{cases} l_2 - p_2 = l_1 p_1 \\ l_3 - p_3 = l_2 p_1 \\ l_4 - p_4 = l_1 p_2 \\ l_i - p_i = l_1 p_{i-2} \\ l_{i+1} - p_{i+1} = l_{i-1} p_1 \\ l_{i+2} - p_{i+2} = l_i p_1 \\ l_{i+3} - p_{i+3} = l_1 p_{i+1} \end{cases}$$

- $E = \{$all pairs $((\bar{p}), [\bar{l}])$ for which $(\bar{p})I[\bar{l}]\}$
- $|E| = q^{n+1}$

3. Encryption/Decryption

(1) **Central map.** Let $(v_1, v_2, v_3, \dots, v_n) \in D(n,q)$ (or $[v_1, v_2, v_3, \dots, v_n] \in D(n,q)$) and $N_t(v)$ be the operator of taking neighbor of vertex v, where the first coordinate is $f(v_1, t)$:

$$N_t(v_1, v_2, v_3, \dots, v_n) \to [f(v_1, t), *, *, \dots, *],$$
$$N_t[v_1, v_2, v_3, \dots, v_n] \to (f(v_1, t), *, *, \dots, *).$$

All asts can be determined `uniquely` using relations between coordinates of vertices in graph $D(n,q)$.
`Multivariate map` $F = N_{t_1, t_2, \dots, t_k} = N_{t_1} \circ N_{t_2} \circ N_{t_3} \cdots \circ N_{t_k}$
   - F is invertible
   - $F : \mathbb{F}_q^n \to \mathbb{F}_q^n$

$$x_1 \to f_1(x_1, x_2, \dots, x_n),$$
$$x_2 \to f_2(x_1, x_2, \dots, x_n),$$
$$\vdots$$
$$x_n \to f_n(x_1, x_2, \dots, x_n).$$

   - $\deg f_i \in \{2, 3\}$

(2) **Extension of the algorithm.** Let L_1 and L_2 be a sparse, affine transformation of the vector space $\mathbb{F}_q^n$

$$L_1 = \bar{x} \longrightarrow \bar{x}A + b,$$
$$L_2 = \bar{x} \longrightarrow \bar{x}C + d,$$

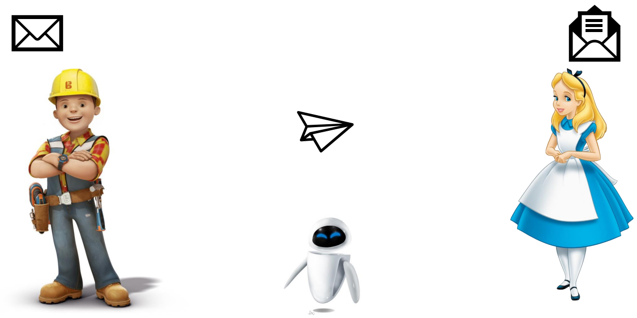
where $A = \left[a_{i,j}\right]$ and $C = \left[c_{i,j}\right]$ are $n \times n$ matrix with $a_{i,j}, c_{i,j} \in \mathbb{F}_q$, $|A| \neq 0$ and $|C| \neq 0$. L_1 and L_2 are invertible.

(3) **Plaintext and ciphertext** are n dimensional vectors over $\mathbb{F}_q$

(4) **Secret Key** K

$$K = (L_1, L_2, t = (t_1, t_2, \dots, t_k))$$

(5) **Encryption/Decryption**



$L_1 \circ N_{t_1,t_2,\dots,t_k} \circ L_2$??? $L_2^{-1} \circ N_{-t_k,-t_{k-1},\dots,-t_1} \circ L_1^{-1}$

Encryption

(a) $L_1(\textit{plaintext}) = x$
(b) $F(x) = y$
(c) $L_2(y) = \textit{ciphertext}$

Decryption

(a) $L_2^{-1}(\textit{ciphertext}) = y$
(b) $F^{-1}(y) = x$
(c) $L_1^{-1}(x) = \textit{plaintext}$

(6) More details can be find here

4. Some implementation aspects

(1) Use modular arithmetic instead of polynomial arithmetic ($q = p^1$; for example $q = 127$ for ASCII/(DEL))
(2) $L_2 = L_1^{-1}$
(3) Let $L_1(x)$ be given by matrix A of the form

$$\begin{bmatrix} \mathbf{1} & 0 & 0 & \dots & 0 & 0 \\ 0/1 & \mathbf{1} & 0 & \dots & 0 & 0 \\ \textit{zeros} & 0/1 & \mathbf{1} & \dots & 0 & 0 \\ \textit{and} & \vdots & \vdots & \ddots & \vdots & 0 \\ \textit{ones} & 0/1 & 0/1 & \dots & \mathbf{1} & 0 \\ 0/1 & 0/1 & 0/1 & \dots & 0/1 & \mathbf{1} \end{bmatrix}$$

(4) $f(v_1, t) = v_1 + t$
(5) Encryption

CENTRAL MAP F
prime=127
n =length of the plaintext
p =plaintext
t =password
$r = 0$
for $i = 1$ to $i =$(password length)
if $r = 0$
$l(1) = (p(1) + t(i)) \mod \text{prime}$
$l(2) = (p(2) + p(1) * l(1)) \mod \text{prime}$

```
l(3) = (p(3) + p(1) * l(2))  mod  prime
for j = 4 to n
  if (j  mod 4) = 3 or (j  mod 4) = 2
    l(j) = (p(j) + p(1) * l(j - 2))  mod  prime
  else
    l(j) = (p(j) + p(j - 2) * l(1))  mod  prime
  end
end
r = 1
else
p(1) = (l(1) + t(i))  mod  prime
p(2) = (l(2) - p(1) * l(1))  mod  prime
p(3) = (l(3) - p(1) * l(2))  mod  prime
for j = 4 to n
  if (j  mod 4) = 3 or (j  mod 4) = 2
    p(j) = (l(j) - p(1) * l(j - 2))  mod  prime
  else
    p(j) = (l(j) - p(j - 2) * l(1))  mod  prime
  end
end
r = 0
end
end
```

(6) Decryption

CENTRAL MAP F^{-1}

```
prime=127
n =length of the ciphertext
c =ciphertext
t =password
if (password length) is even
  r = 0
  p = c
else
  r = 1
  l = c
end

for i =(password length) to i = 1
if r = 0
  l(1) = (p(1) - t(i))  mod  prime
  l(2) = (p(2) + p(1) * l(1))  mod  prime
  l(3) = (p(3) + p(1) * l(2))  mod  prime
  for j = 4 to n
    if (j  mod 4) = 3 or (j  mod 4) = 2
      l(j) = (p(j) + p(1) * l(j - 2))  mod  prime
    else
```

```
            l(j) = (p(j) + p(j − 2) * l(1))  mod  prime
         end
      end
      r = 1
   else
      p(1) = (l(1) − t(i))  mod  prime
      p(2) = (l(2) − p(1) * l(1))  mod  prime
      p(3) = (l(3) − p(1) * l(2))  mod  prime
      for j = 4 to n
         if (j  mod 4) = 3 or (j  mod 4) = 2
            p(j) = (l(j) − p(1) * l(j − 2))  mod  prime
         else
            p(j) = (l(j) − p(j − 2) * l(1))  mod  prime
         end
      end
      r = 0
      end
   end
```

5. Multivariate cryptography

The basic objects of `multivariate cryptography` are systems of multivariate, non-linear (usually quadratic) polynomials.

MQ-Problem

Given m multivariate quadratic polynomials $p_i(x)$ in n variables find a vector x that is a solution of the system of equations $p_i(x) = 0$.

MQ-problem is NP`-hard` when coefficients are random.

MQ-problem is efficiently solved only under special conditions

- Underdefined MQ-problem ($n >> m$) polynomial time algorithms by Kipnis et al. (if char$\mathbb{F}_q$ even) and Hiroyuki Miura & Yasufumi Hashimoto & Tsuyoshi Takagi
- Overdefined MQ-problem ($m >> n$) Gröbner basis algorithms are efficient

`System of equations related to` $D(n,q)$ `graphs is cubic and` $n = m$.

MC-Problem

Given m multivariate cubic polynomials $p_i(x)$ in n variables find a vector x that is a solution of the system of equations $p_i(x) = 0$.

Department of Computer Science, Rochester Institute of Technology
E-mail address: mkp@cs.rit.edu