Cyber Security Assignment

Outline

> **Keccak-f[25]**
>
> > θ step mapping
> >
> > π step mapping
> >
> > χ step mapping
> >
> > ι step mapping
>
> **ElGamal**
>
> > Key generation
> >
> > Encryption
> >
> > Decryption
> >
> > Multiplication over encrypted data

Keccak-f[25]

1. θ step mapping
   - C[x]

```
function iniC(){
    for(i=0;i<5;i++){
        C[i] = A[i][0] ^ A[i][1] ^ A[i][2] ^ A[i][3] ^ A[i][4]
    }
}
```

   - D[x]

```
function iniD() {
    for (i = 1; i < 4; i++) {
        D[i] = C[i - 1] ^ C[i + 1];
    }
    D[0] = C[4] ^ C[1];
    D[4] = C[3] ^ C[0];
}
```

- A[x,y] = A[x,y] XOR D[x]

2. π step mapping
    - Read the input grid and set the value to new grid

```
for(i=0;i<5;i++){
        for(j=0;j<5;j++){
            let a =
Number(document.getElementById("θo".concat(Number(i).toString()).con
cat(Number(j).toString()))).value);
            G[j][(2*i+3*j)%5] = a;
        }
    }
```

    - The new grid as the output grid

```
for(m=0;m<5;m++){
        for(n=0;n<5;n++){

 document.getElementById("πo".concat(Number(m).toString()).concat(Nu
mber(n).toString()))).value = G[m][n];
        }
    }
```

3. χ step mapping
    - Move the array

```
function move(arr1, k) {
    let i,j;
    for(i=0;i<5;i++){
        for(j=0;j<5;j++){
            let a =
Number(document.getElementById("πο".concat(Number((i+k)%5).toString(
)).concat(Number(j).toString()))).value);
            arr1[i][j] = a;
        }
    }
    return arr1;
}
```

- Inverse the array

```
function inverse(arr2) {
    let i, j;
    for (i = 0; i < 5; i++) {
        for (j = 0; j < 5; j++) {
            arr2[i][j] = arr2[i][j] === 1 ? 0 : 1;
        }
    }
    // return arr;
    return arr2;
}
```

4. ι step mapping
    - A[0,0]=A[0,0] XOR RC[i]

```
let b = Number(document.getElementById('round').innerText);
    let rc = RC[b][2];
    let rcInt = parseInt(rc, 16).toString(2)[0];
    M[0][0] = M[0][0] ^ parseInt(rcInt);
```

ElGamal

1. Key generation
    - Private key

```
function generatePrivateKey() {
    let privateKey;
    let q = Number(document.getElementById('q').value);
    privateKey = Math.floor((q-2)*Math.random())+1;
    document.getElementById('privatekey').value = privateKey
}
```

- Public key

```
function generatePublicKey() {
    let publicKey;
    let q,g,y,x,p;
    p = Number(document.getElementById('p').value);
    q = Number(document.getElementById('q').value);
    g = Number(document.getElementById('g').value);
    x = Number(document.getElementById('privatekey').value);
    y = fastExponentiation(g,x);
    // publicKey = "
{".concat(q).concat(",").concat(g).concat(",").concat(y).concat("}")
;
    publicKey = "{"+p+","+g+","+y+"}";
    document.getElementById('publickey').value = y;
    document.getElementById('allpublickey').innerHTML = publicKey;
}
```

2. Encryption

```
function encryptMessage() {
    let k = Number(document.getElementById('k').value);
    let plaintext = document.getElementById('message').value;
    let y = Number(document.getElementById('publickey').value);
    let g = Number(document.getElementById('g').value);
    let p = Number(document.getElementById('p').value);
    let K = fastExponentiation(y,k);
    let C1 = fastExponentiation(g,k);
    let C2 = (K*plaintext)%p;
    document.getElementById('c1').value = C1;
    document.getElementById('c2').value = C2;
    document.getElementById('ciphertext').value = "("+C1+","+C2+")";
}
```

3. Decryption

```
function decryptMessage() {
    let c1 = Number(document.getElementById('c1').value);
    let c2 = Number(document.getElementById('c2').value);
    let x = Number(document.getElementById('privatekey').value);
    let p = Number(document.getElementById('p').value);
    let m = fastExponentiation(c1,p-x-1);
    let M = (c2*m)%p;
    document.getElementById('plaintext').value = M;
}
```

4. Multiplication over encrypted data

- Multiply plaintext

```
let multi = (n1*n2*n3*n4*n5)%p;
   document.getElementById('multinput').value = multi;
```

- Get c1

```
let c1 = fastExponentiation(g,k1+k2+k3+k4+k5);
    document.getElementById('c1o').value = c1;
```

- Get c2

```
let c2 = ((n1*n2*n3*n4*n5)%p *
fastExponentiation(y,k1+k2+k3+k4+k5))%p;
    document.getElementById('c2o').value = c2;
```

- Get c

```
let cc =  "("+c1+","+c2+")";
    document.getElementById('co').value = cc;
```

- Decryption

```
let ec1 = fastExponentiation(c1,p-1-x);
    let m = (ec1*c2)%p;
    document.getElementById('Decypt-Result').value = m;
```