19335118_梁冠轩_信息安全技术实验报告

RSA加密和解密

RSA

选择一个1024位的明文M，选择两个1024位的素数p和q，计算n = p * q，ϕ(n) = (p - 1) * (q - 1)

选择一个素数e，使得gcd(e, ϕ(n)) = 1，计算e对于ϕ(n)的模反元素d

通过这几步，我们得到了e，d，n，将e和n作为公钥进行加密，得到密文，将d和n作为私钥进行解密，得到明文

```
#开始时间
start_time = time.time()

#1024位的明文
M =
0x0000190510a41630679425c803846ebf88239c321688a8c37de2fbaf1312362b3a95952e9f7373
be722427520455275973871 8c888310218850f01b2a56ff0b604e4f79eb7376464d4207a06494b91
7e05f2aaa90a98226c1f1e83870160d74aa1ec92e3939fa0e39aaeb694e81d749f39317f5b7f2a65
142246647c0667359ea7a72a021a122e780375a4c308dd794ef407eaeee963be10417e11c54408dc
fbaf765ed0af344956240284eddc3b03f94d4addf34a0defec143e27505f77f3557d1e6cfc45af52
7202e9ec42571eab2f0aefb17ba0cb599416fd4d8a6599b2c38e51b4dc3695414d5f6eb00eb0a546
3dc1e07cbded15b80eedba9c8d6509f886
#选择两个1024位的素数作为p和q
p =
350745980337414028701224405911527637769597477228258122620176098338762820320362 34
158131883762850216623997088950455302388093413922032430390440287678109018554632 70
540846560340550179318956082923804072990491521773857196596448044384441308745348 00
268238968710827701411011809597879102690237634573664840339787506833 07
q =
177868678153255329271515065918260540535007072753615173407369983397392818131430 40
689777487431627959782814367993005835619267696380928422133194738867972693497703 75
674712937221839028700230507667426391350063881598276567020144498897316559267686 21
644334320354221155537925493671542876775525110735235963228602004248263
n = p * q   #计算n
phi_n = (p - 1) * (q - 1)   #计算n的欧拉函数phi_n
e = 65537    #随机选择一个素数e
```

```
d = modulaInverse(e, phi_n)     #计算e对于phi_n的模反元素d
C = fastPower(M, e, n)      #将e和n作为公钥加密，得到密文
M_ = fastPower(C, d, n)     #将d和n作为私钥解密，得到明文

#结束时间
end_time = time.time()
print("明文：", M)
print("RSA解密：", M_)
print("RSA运行时间：", end_time - start_time)
```

CRT

使用中国剩余定理CRT，对RSA的解密操作进行优化

计算dp = d mod (p - 1)，dq = d mod (q - 1)，计算q对于p的模反元素qinv

```
#利用中国剩余定理
dp = d % (p - 1)
dq = d % (q - 1)
qInv = modulaInverse(q, p)
```

将p，q，d，dp，dq，qinv作为私钥进行解密，得到明文

```
Cp = C % p
Cq = C % q
M1 = fastPower(Cp, dp, p)
M2 = fastPower(Cq, dq, q)
h = (qInv * ((M1 - M2) % p)) % p
M_ = M2 + h * q
```

实验结果

```
PS D:\vscode workspace\python\RSA> python RSA.py
明文：  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA解密：  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA运行时间：  0.0438847541809082
PS D:\vscode workspace\python\RSA> python RSA.py
明文：  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA解密：  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA运行时间：  0.041861534118652344
```

```
PS D:\vscode workspace\python\RSA> python RSA_CRT.py
明文:  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA_CRT解密:  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA_CRT运行时间:  0.013961315155029297
PS D:\vscode workspace\python\RSA> python RSA_CRT.py
明文:  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA_CRT解密:  48194202737366350680607458407164432312230756179896576059919921731402029837705369385669723987792141527321920039106395254345795478692704367560026963885306840827674995762568410155020628004724781719140893916525395623956313969179133678955461998977235929192581674957072503879717103420743408139178423810815073812891780134165805611469232371797424761382411461657068685861659822392861888741664391197395860130003606123736694375460725049649534716181560121305891791047434163096302108846304012244669026214589805710978341581381785115248537651895641993593319279459469152611621110777185994128994939812565319773517189188223105158
RSA_CRT运行时间:  0.012973785400390625
```

可以看到，使用中国剩余定理CRT优化后的RSA运行速度快了大约0.03秒，没有使用CRT的RSA是使用CRT所花费时间的四倍，中国剩余定理可以十分显著地提升RSA的解密速度，效率更高。

DWT扩频数字图像水印

图像扩频

读取图像，生成一个与图像大小相同的随机均匀矩阵，用于异或处理

```
function img = img_expand(img, seed)
[row, col] = size(img);
%根据seed，获取随机矩阵
rng(seed);
ramdom_matrix = randi(1000, row, col);
```

遍历图像每个像素，对像素进行扩频处理，将像素值转换为二进制，然后把每一位扩展为两位数据

```
function result = num_expand(num)
bin_num = dec2bin(num);
%将一位扩展为两位
for i = 1 : length(bin_num)
    if bin_num(i) == '1'
        str((i - 1) * 2 + 1) = '1';
        str((i - 1) * 2 + 2) = '1';
    else
        str((i - 1) * 2 + 1) = '0';
        str((i - 1) * 2 + 2) = '0';
    end
end
result = bin2dec(str);
```

每位像素先乘10000，然后进行扩频，扩频后进行异或，再除以10000，得到的结果赋值给该像素

```
for i = 1 : row
    for j = 1 : col
        tmp = img(i, j) * 10000
        tmp = num_expand(tmp);
        tmp = bitxor(tmp, ramdom_matrix(i, j));
        img(i, j) = tmp / 10000;
    end
end
```

逆图像扩频

读取图像，生成一个与图像大小相同的随机均匀矩阵，用于异或处理

```
function img = img_unexpand(img, seed)
[row, col] = size(img);
%根据seed，获取随机矩阵
rng(seed);
ramdom_matrix = randi(1000, row, col);
```

遍历图像每个像素，对像素进行逆扩频处理，将像素值转换为二进制，然后把每两位变为一位数据

```
function result = num_unexpand(num)
bin_num = dec2bin(num);
%将两位提取为一位
for i = 1 : floor(length(bin_num) / 2)
    if bin_num(2 * i - 1) == '1' && bin_num(2 * i) == '1'
        str(i) = '1';
    elseif bin_num(2 * i - 1) == '1' && bin_num(2 * i) == '0'
        str(i) = '1';
    elseif bin_num(2 * i - 1) == '0' && bin_num(2 * i) == '1'
        str(i) = '0';
    elseif bin_num(2 * i - 1) == '0' && bin_num(2 * i) == '0'
        str(i) = '0';
    end
end
result = bin2dec(str);
```

每一位像素先乘10000，进行异或，然后进行逆扩频，再除以10000，得到的结果赋值给该像素

```
%对图片的每一个像素进行逆扩频
for i = 1 : row
    for j = 1 : col
        tmp = img(i, j) * 10000;
        tmp = bitxor(int32(tmp), ramdom_matrix(i, j));
        tmp = num_unexpand(tmp);
        img(i, j) = tmp / 10000;
    end
end
```

DWT嵌入水印图像

读取图像，对图像进行正方化，读取水印，对水印进行扩频处理

```
%读取图像，对图像进行正方化
img1 = imread(img);
img1 = double(img1) / 255;
img1 = img1(:, :, 1);
[row, col] = size(img1);
width = max(row, col);
new_img = zeros(width, width);
if row <= col
   new_img(1 : row, :) = img1;
else
   new_img(:, 1 : col) = img1;
end

%读取水印
watermark1 = imread(watermark);
watermark1 = double(watermark1) / 255;
%对水印进行扩频处理
watermark1 = img_expand(watermark1, seed);
new_watermark = zeros(width, width);
new_watermark(1 : 101, 1 : 101) = watermark1;
```

对图像与水印分别使用db6小波函数进行9级小波分解，提取低频系数，再对低频系数进行单值分解

```
%对图像进行小波分解，提取低频系数
[C, S] = wavedec2(new_img, 9, 'db6');
CA = appcoef2(C, S, 'db6', 9);
%对水印进行小波分解，提取低频系数
[C1, S1] = wavedec2(new_watermark, 9, 'db6');
CA1 = appcoef2(C1, S1, 'db6', 9);
%对图像和水印的低频系数进行单值分解
[U, sigma, V] = svd(CA);
[U1, sigma1, V1] = svd(CA1);
```

将水印嵌入到图形中，利用逆小波分解实现重构，得到嵌入水印图形

```
CA_tilda = reshape(U1 * sigma * V1, 1, S(1, 1) * S(1, 2));
C(1, 1 : S(1, 1) * S(1, 2)) = CA_tilda;
watermarkImg = waverec2(C, S, 'db6');
watermarkImg(:, :, 1) = watermarkImg(1 : row, 1 : col);
imwrite(watermarkImg, 'watermarkImg.bmp');
```

DWT提取水印图像

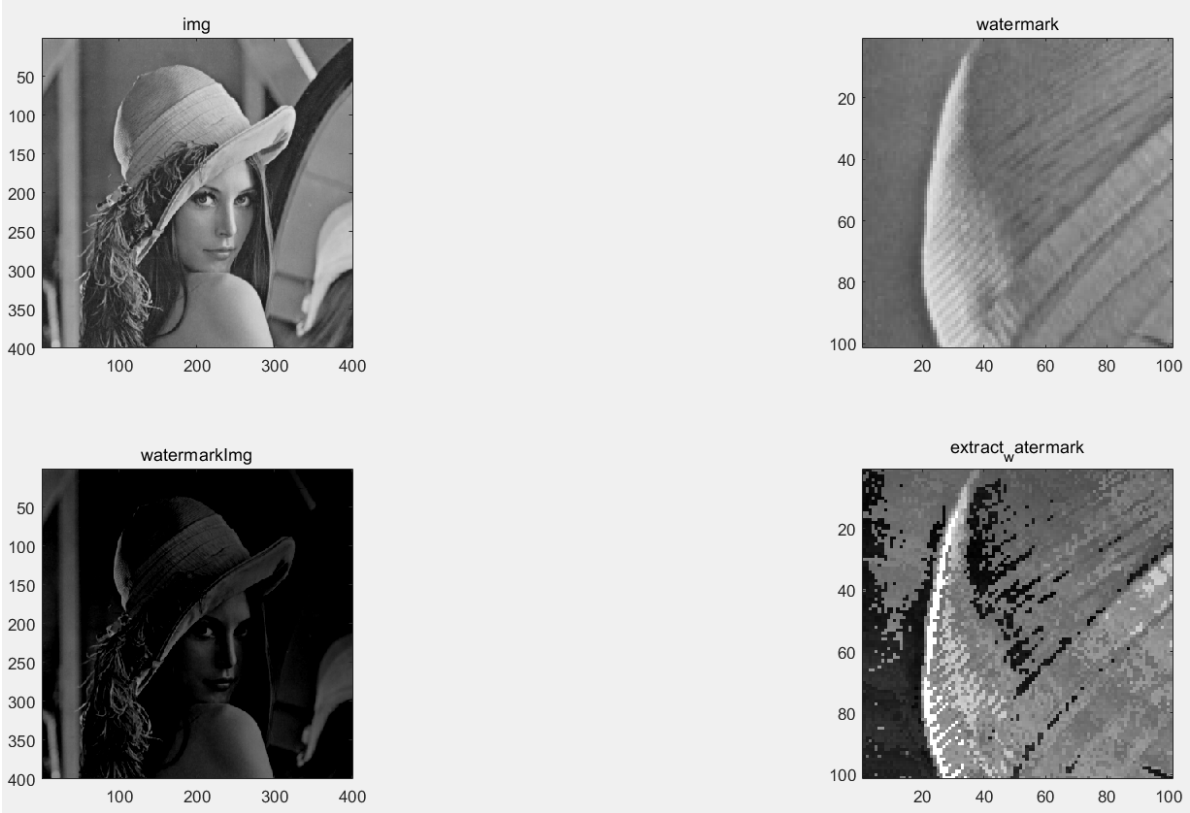
对嵌入水印图形使用db6小波函数进行9级小波分解，提取低频系数，再对低频系数进行单值分解

```
%对嵌入图像进行小波分解，提取低频系数
[C, S] = wavedec2(watermarkImg, 9, 'db6');
CA = appcoef2(C, S, 'db6', 9);
%对低频系数单值分解
[U, sigma, V] = svd(CA);
```

再使用逆小波分解实现重构，得到水印图形

```
%从嵌入图像中提取水印，完成重构
CA_tilda = reshape(U * sigma1 * V, 1, S1(1, 1) * S1(1, 2));
C1(1, 1 : S1(1, 1) * S1(1, 2)) = CA_tilda;
extract_watermark = waverec2(C1, S1, 'db6');
extract_watermark = img_unexpand(abs(extract_watermark), seed);
extract_watermark = extract_watermark(1 : 101, 1 : 101);
imwrite(extract_watermark, 'extract_watermark.bmp');
```

实验结果



提取水印对比原来水印的错误率为0.23907

```
error rates: 0.23907
```

对水印图形进行扩频处理，能增强水印的安全性，即使提取出来了水印，不对水印进行正确的逆处理，也无法得到正确的信息。扩频算法的选择，也决定了提取水印的正确率，本次实验只使用了一位扩频为两位的做法，这种做法会导致误差过大，因为只有四种情况，采用更多位的扩频可能会提升正确率。