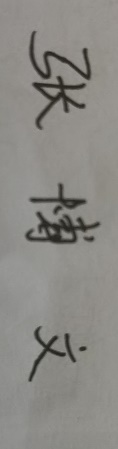
**计算机科学与技术学院**

**2021－2022 学年第 1 学期**

**题目：** **信息伪装和图像置乱**



**姓名： 张博文**

**学号： 200340164**

**交报告时间： 2022/11/7**

**教师： 付宇**

**成绩：**

目 录（必须要有）

[题目：信息伪装和图像置乱](#_Toc28195)

[题目名称：信息伪装和图像置乱](#_Toc21091)

[1 系统概述](#_Toc10329)

1.背景

2 [实验环境](#_Toc13923)

[3 系统设计与实现](#_Toc9989)

1.第一部分

1. encode.m

2. decode.m

3. encode\_srcipt.m

4. decode\_srcipt.m

2.第二部分

1.arnoldImg

2.rearnold

3.messForArnold

[4 系统使用说明](#_Toc25860)

[4 项目分析总结](#_Toc1449)

[参考文献](#_Toc5611)

题目名称： 信息伪装和图像置乱

# 系统概述

信息伪装就是将秘密信息隐藏于另一非机密的文件内容之中，其形式可以是任何一种数字媒体，如图像、声音、视频等等。信息伪装技术不同于传统的加密技术，密码仅仅隐藏了信息的内容，而信息伪装不但隐藏了信息的内容而且隐藏了信息的存在。

本实验的第一部分参考了lsb的算法，对实验二进行升级，实现rgb的信息伪装，而不是灰度图的伪装

随着多媒体技术、信息存储技术和网络的的飞速发展，以及网络带宽限制的放松，越来越多的数字图像得以在网络上传输，并逐步成为人们获取信息的主要手段。网络上传输的数字图像有些无关紧要，有些却至关重要，这其中有可能涉及到个人隐私、公司利益、军事机密、国家安全，其价值无法衡量。另一方面，Internet 网络的日益普及使得任何人都有可能接触并搜集到网络中的图像信息，而不管它是善意的还是恶意的、合法的还是非法的，从而使得在网络上传输的图像安全倍受关注,字图像的安全已经成为信息安全领域中重要的研究分支，而置乱技术在数字图像加密技术中起着不可忽视的作用。

# 实验环境

win11，matlab2018R

# 系统设计与实现

**第一部分**

**encode.m**

%% 图像嵌入

function [final\_img] = encode(img1,img2)

% 忽略十进制的十位

% 142->140

img1 = idivide(img1,10) \* 10;

%% 裁剪尺寸使其一致

[h,w]=size(img1);

img2=imresize(img2,[h,w]);

img2=img2 \* (9/255);

% imtool(img2);

%% 图像相加

final\_img =img1+img2;

**decode.m**

% 解函数

function [img1,img2] = decode(image)

img1=idivide(image,10)\*10;

img2=image - img1;

img2=img2\*(255/9);

end

嵌入脚本

encode\_srcipt.m

img1 = imread('3.png');

img2 = imread('2.png');

subplot(2,3,1);

imshow(img1);

title('载体');

subplot(2,3,2);

imshow(img2);

title('隐藏');

% 提取rgb的三层矩阵

img1\_r = img1(:,:,1);

img1\_g = img1(:,:,2);

img1\_b = img1(:,:,3);

img2\_r = img2(:,:,1);

img2\_g = img2(:,:,2);

img2\_b = img2(:,:,3);

% 对三层矩阵进行处理

final\_img\_r=encode(img1\_r,img2\_r);

final\_img\_g=encode(img1\_g,img2\_g);

final\_img\_b=encode(img1\_b,img2\_b);

% 合成图像

final\_img= cat(3,final\_img\_r,final\_img\_g,final\_img\_b);

imwrite(final\_img,"output.png");

subplot(2,3,3);

imshow(final\_img);

title('隐藏后');

% img1 = rgb2gray(img1);

% img2 = rgb2gray(img2);

%

% final\_img=encode(img1,img2);

% imshow(final\_img);

提取脚本

decode\_srcipt.m

image = imread('output.png');

% imgae = final\_img;

image\_r=image(:,:,1);

image\_g=image(:,:,2);

image\_b=image(:,:,3);

[img1\_r,img2\_r] = decode(image\_r);

[img1\_g,img2\_g] = decode(image\_g);

[img1\_b,img2\_b] = decode(image\_b);

img1 =cat(3,img1\_r,img1\_g,img1\_b);

img2 =cat(3,img2\_r,img2\_g,img2\_b);

subplot(2,3,4);

imshow(img2);

title('提取的信息')

% figure;

% imshow(img1);

% figure;

% imshow(img2);

**第二部分（图像置乱）**

**一维置乱**

**messe1.m**

% 算法1.把图像的数据重塑成一维的形式，对其进行置乱；

clear

close

clc

G=imread('1.jpg');

G=rgb2gray(G)

subplot(1,3,1);

imshow(G);

title('原图');

A=G(:);

A1=G(end: -1:1);%乱

subplot(1,3,2);

A1=reshape(A1,512,512)

% A1=imresize(A,0.5);

imshow(A1);

title('一维置乱'); %title一下

A2=A1(end:-1:1);

A2G=reshape(A2,512,512);

subplot(1,3,3) ;

imshow(A2G)

title('置乱恢复'); %title一下

二维置乱

mess2.m

% 算法2.把图像看成二维的形式，采用二维坐标的形式对其进行置乱；

clear

close

clc

G=imread('1.jpg');

G=rgb2gray(G);

%缩小尺寸

G=imresize(G,0.5);

subplot(1,3,1);

imshow(G);

title('原图') %title一下原图

Gadd=fix(256\*rand(256,256));

for i=1:256

for j=1:256

G1(i,j)=0.1\*G(i,j)+0.9\*Gadd(i,j);

end

end

subplot(1,3,2);

imshow(G1);

title('置乱后的图像') ;

for i=1:256

for j=1:256

G2(i,j)=(G1(i,j)-0.9\*Gadd(i,j))./0.1;

end

end

subplot(1,3,3)

imshow(G2); %显示图像

title('恢复后的图像');

1、二维图像直接行置乱与列置乱  
（1）随机置乱

行/列随机：借助于混沌系统产生长度为M/N的[随机数](https://so.csdn.net/so/search?q=%E9%9A%8F%E6%9C%BA%E6%95%B0&spm=1001.2101.3001.7020)向量X/Y，然后将图像矩阵的第i行/第j列与第Xi行/第Yj列互换。

实现：  
这里使用的超混沌Lorenz系统

mess3.m

clc,clear;

P=imread('1.jpg');P=rgb2gray(P);% 读入图像 转化为灰度图像

iptsetpref('imshowborder','tight');% 设置图像显示无边框

figure(1);imshow(P);% 显示

[M,N]=size(P);P=double(P);% 获得图像行列数M N

n=M+N;% 设定计算的混沌序列长度

h=0.002;t=800;% 以下

a=10;b=8/3;c=28;r=-1;

x0=1.1;y0=2.2;z0=3.3;w0=4.4;

s=zeros(1,n);

for i=1:n+t

K11=a\*(y0-x0)+w0;K12=a\*(y0-(x0+K11\*h/2))+w0;

K13=a\*(y0-(x0+K12\*h/2))+w0;K14=a\*(y0-(x0+h\*K13))+w0;

x1=x0+(K11+K12+K13+K14)\*h/6;

K21=c\*x1-y0-x1\*z0;K22=c\*x1-(y0+K21\*h/2)-x1\*z0;

K23=c\*x1-(y0+K22\*h/2)-x1\*z0;K24=c\*x1-(y0+h\*K23)-x1\*z0;

y1=y0+(K21+K22+K23+K24)\*h/6;

K31=x1\*y1-b\*z0;K32=x1\*y1-b\*(z0+K31\*h/2);

K33=x1\*y1-b\*(z0+K32\*h/2);K34=x1\*y1-b\*(z0+h\*K33);

z1=z0+(K31+K32+K33+K34)\*h/6;

K41=-y1\*z1+r\*w0;K42=-y1\*z1+r\*(w0+K41\*h/2);

K43=-y1\*z1+r\*(w0+K42\*h/2);K44=-y1\*z1+r\*(w0+h\*K43);

w1=w0+(K41+K42+K43+K44)\*h/6;

x0=x1;y0=y1;z0=z1;w0=w1;

if i>t

s(i-t)=x1;

if mod((i-t),3000)==0% 每3000次迭代后加扰动

x0=x0+h\*sin(y0);

end

end

end% 以上 生成混沌序列

X=mod(floor((s(1:M)+100)\*10^10),M)+1;% 生成行置乱向量

Y=mod(floor((s(M+1:M+N)+100)\*10^10),N)+1;% 生成列置乱向量

A=P;

for i=1:M

t=A(i,:);A(i,:)=A(X(i),:);A(X(i),:)=t;

end

figure(2);imshow(uint8(A));% 行置乱后的图像

B=A;

for j=1:N

t=B(:,j);B(:,j)=B(:,Y(j));B(:,Y(j))=t;

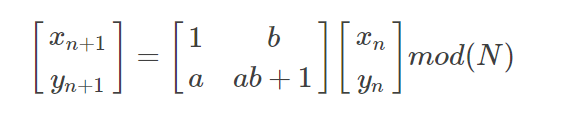
end

figure(3);imshow(uint8(B));% 列置乱后的图像

基于使用函数Arnold变换置乱

Arnold变换是俄国数学家VladimirI.Arnold提出的一种变换

一幅M\*N的数字图像的二维Arnold 变换定义为:



其中表示变换前灰度图中像素的位置，表示变换之后的像素位置，为参数，表示当前变换的次数，为图像的长或宽（由于该算法只适用于长宽相等的图像，所以我们不讨论M不等于N的情况），mod为模运算。

%img 灰度图像 a,b为参数 n为变换次数

function arnoldImg = arnold(img,a,b,n)

img=rgb2gray(img);

[h,w] = size(img);

N=h;

arnoldImg = zeros(h,w);

for i=1:n

for y=1:h

for x=1:w

%防止取余过程中出现错误，先把坐标系变换成从0 到 N-1

xx=mod((x-1)+b\*(y-1),N)+1;

yy=mod(a\*(x-1)+(a\*b+1)\*(y-1),N)+1;

arnoldImg(yy,xx)=img(y,x);

end

end

img=arnoldImg;

end

arnoldImg = uint8(arnoldImg);

end

基于函数Arnold变换复原

复原时用变换矩阵的逆矩阵即可

图表

描述已自动生成

function img = rearnold(arnoldImg,a,b,n)

[h,w] = size(arnoldImg);

img = zeros(h,w);

N = h;

for i=1:n

for y=1:h

for x=1:w

xx=mod((a\*b+1)\*(x-1)-b\*(y-1),N)+1;

yy=mod(-a\*(x-1)+(y-1),N)+1 ;

img(yy,xx)=arnoldImg(y,x);

end

end

arnoldImg=img;

end

img = uint8(img);

end

直接置乱不复原

messForArnold.m

G=imread('1.jpg');

w0=double(G)/255;

[m,n]=size(w0);

w1=w0;

subplot(1,5,1);

imshow(w1,[]);

title('原图');

for k=1:1

for x=1:m

for y=1:n

x1=x+y;

y1=x+2\*y;

if x1>m

x1=mod(x1,m);

end

if y1>n

y1=mod(y1,n);

end

if x1==0

x1=m;

end

if y1 == 0

y1=n;

end

w1(x1,y1)=w0(x,y);

end

end

w0=w1;

end

subplot(1,5,2);

imshow(w1,[]);

title('变换一次');

for k=1:15

for x=1:m

for y=1:n

x1=x+y;

y1=x+2\*y;

if x1>m

x1=mod(x1,m);

end

if y1>n

y1=mod(y1,n);

end

if x1==0

x1=m;

end

if y1 == 0

y1=n;

end

w1(x1,y1)=w0(x,y);

end

end

w0=w1;

end

subplot(1,5,3);

imshow(w1,[]);

title('变换15次');

for k=1:95

for x=1:m

for y=1:n

x1=x+y;

y1=x+2\*y;

if x1>m

x1=mod(x1,m);

end

if y1>n

y1=mod(y1,n);

end

if x1==0

x1=m;

end

if y1 == 0

y1=n;

end

w1(x1,y1)=w0(x,y);

end

end

w0=w1;

end

subplot(1,5,4);

imshow(w1,[]);

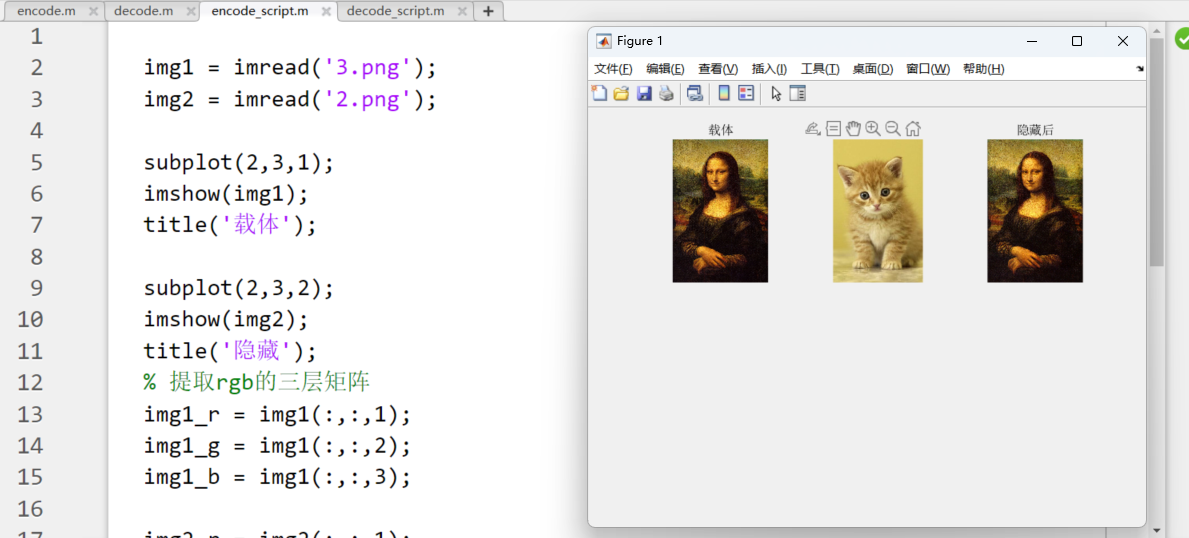
title('变换95次');

% imwrite(mat2gray(w1),'no2.jpg');

# 系统使用说明

第一部分

运行嵌入脚本



运行解脚本

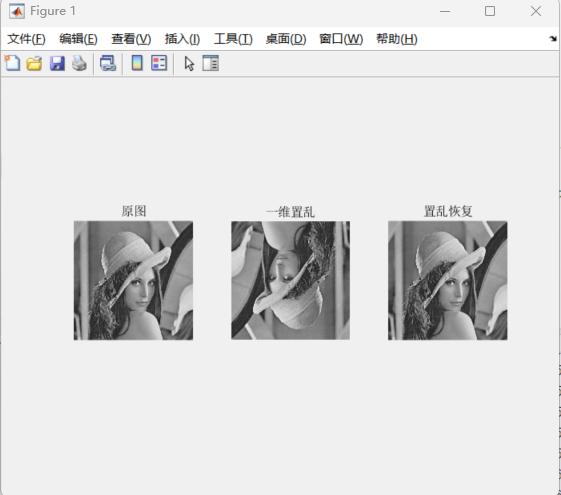
图形用户界面, 日程表

描述已自动生成

第二部分

置乱

一维置乱

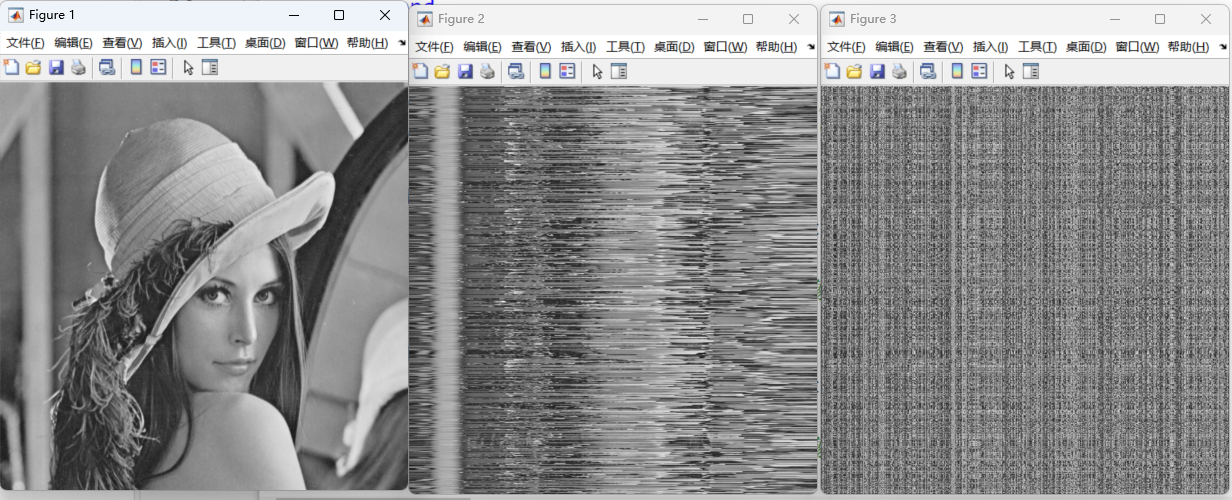


二维置乱

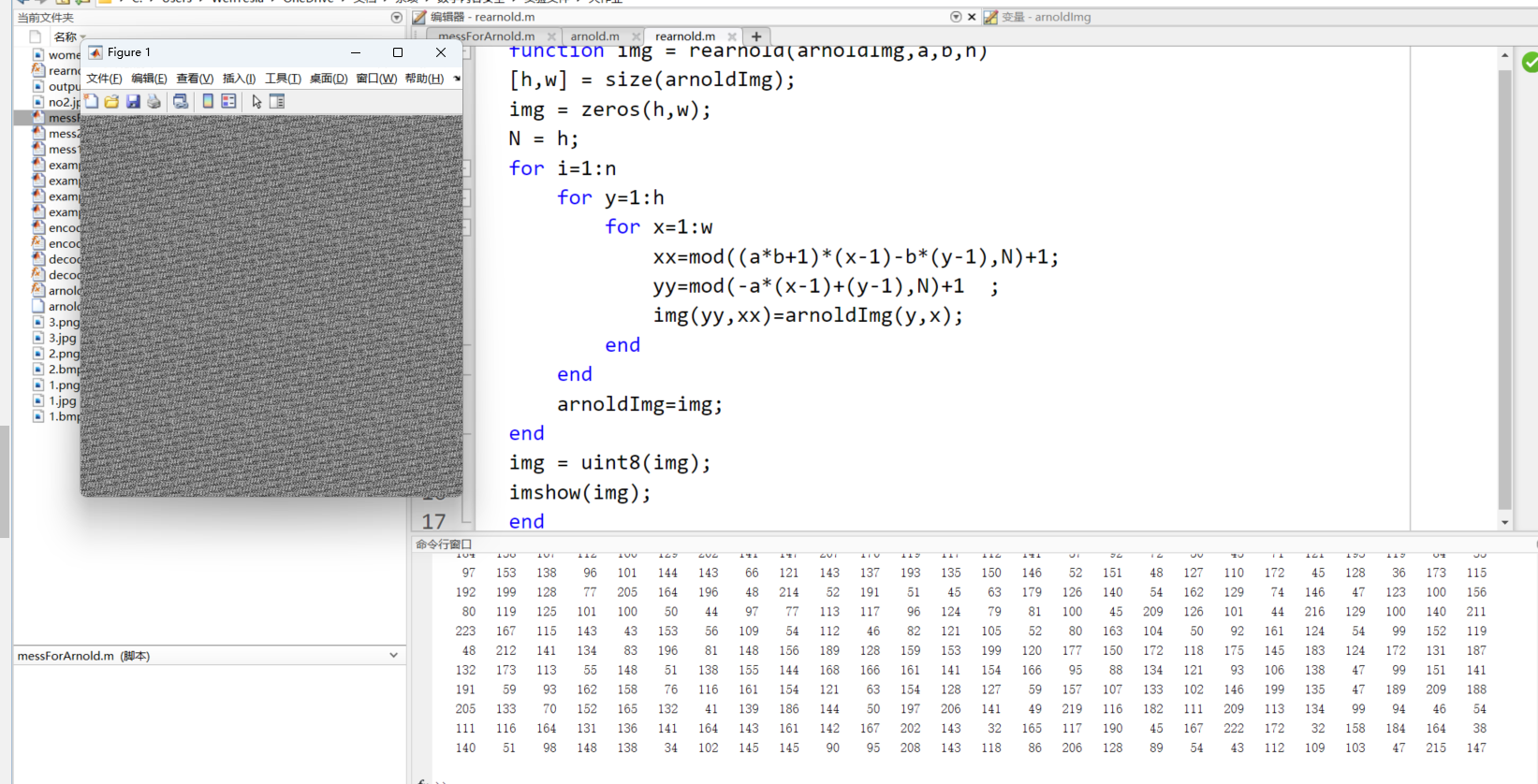
QR 代码

低可信度描述已自动生成

行/列随机



参数 8 2 15



还原

8，2，15

社交网站的手机截图

中度可信度描述已自动生成

如果参数不一致 8 5 15

图形用户界面, 应用程序

描述已自动生成

则解密错误

也可直接运行mess4.m

clear;

%

x=imread("1.jpg");

subplot(2,2,1);

% figure;

imshow(x);

title('原图');

x1=x;

img1 = arnold(x,8,2,15);

subplot(2,2,2);

% figure;

imshow(img1);

title('置乱后');

img2 = rearnold(img1,8,2,15);

subplot(2,2,3);

% figure;

imshow(img2);

title('正确解密');

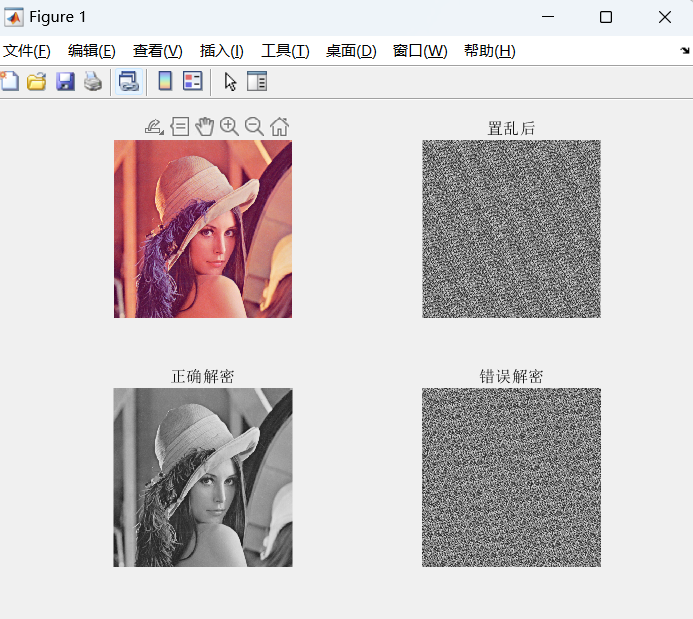
img3=rearnold(img1,8,3,15);

subplot(2,2,4);

% figure;

imshow(img3);

title('错误解密');



即可

# 项目分析总结

分析项目开发过程中遇到的问题及解决办法。

图像置乱

第一种方法方法原理简单，容易实现，运算量小，但同时存在以下部分不足:

(1）置乱后的图像不够加密，置乱度不够高

(2）方程变换过于简单，容易找到逆运算，安全性不高;

第二种方法

经典的arnold变换有很多局限。比如说只能对长宽相等的图像进行处理。一般可以通过对图像进行分块填充等客服这个限制。但除此之外，还有一些其他缺点，比如它必须通过多次的置乱来达到一个比较好的效果，且变换具有周期性，密钥空间不足等。

参考文献

必须要有。

[基于混沌理论的高分辨率数字图像加密算法及实现 - 中国知网 (cnki.net)](https://kns.cnki.net/kcms/detail/detail.aspx?dbcode=CMFD&dbname=CMFD2012&filename=1012345821.nh&uniplatform=NZKPT&v=JFbkSWl-AxPaytNKk6_jUA7LtVnsRqCCzv8eypSqGy1whdtwJtEN38uhtoI3sqEW)