实验二 基于 Arnold 变换的图像置乱的仿真、一点推理和改进实践

201811123001-陈安婕

1.1 简述

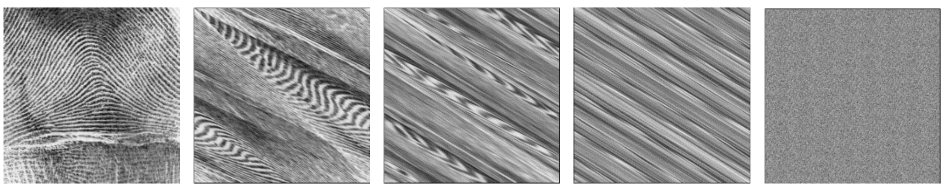
基于 Arnold 变换的图像置乱由俄国数学家 Vladimir I. Arnold 提出。将数字图像视为数字矩阵，通过变换式（1）改变每个图像像素的位置。其中 n 表示当前变换的次数；N 表示图像的长或宽；（x, y）表示像素坐标；a, b 为正整数，且通常取 a=b=1。

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & b \\ a & ab+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} (\mathrm{mod}\ \mathrm{N}) \qquad (1)$$

1.2 仿真结果

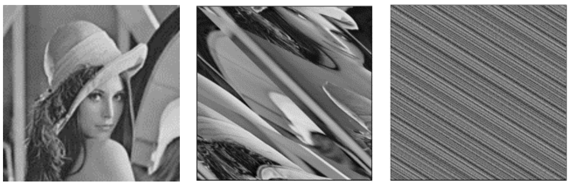
实验环境：python == 3.6.7;　cv2 == 3.4.1;　numpy == 1.16.0.

根据 a=b=1 时的式(1)：

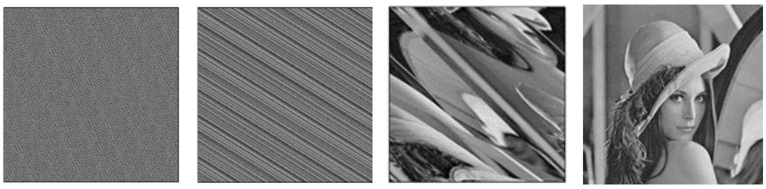


a　　b　　c　　d　　e

Fig.1. 加密仿真结果举例。(a)是 8bit 512*512 fingerprint.bmp 原图；(b)-(e)分别是置乱 1、2、3、50 次得到的图片；代码见文件 Arnold_encypt.py。



a b c



d e f g

Fig.2. 解密仿真结果举例。(a)是 8bit 512*512 lena.bmp 原图；(b)是 Arnold 置乱 1 次得到的图片；(c)是 Arnold 置乱 5 次得到的图片；(d)是待解密的经过了 10 次 Arnold 置乱的图片；(e)-(f)是分别经过 5 次、9 次逆变换的结果；(g)是经过 10 次逆变换的结果。代码见文件 Arnold_decrypt.py。

2. 关于“基于 Arnold 变换的图像置乱是否可以通过计算变换矩阵直接从原图像得到置乱 n 次后的图像”的问题的一点数学推导和仿真实验

2.1 数学推导

变换式为:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} (\mathrm{mod\ N}) \qquad (2)$$

所以,

$$x_{n+1} \equiv x_n + y_n (mod\ N)$$
$$y_{n+1} \equiv x_n + 2y_n\ (mod\ N)$$

n=1 时,

$$x_2 = x_1 + y_1 - k_1 N;$$

$$y_2 = x_1 + 2y_1 - k_1' N$$

n=2 时,

$$x_3 = x_2 + y_2 - k_2 N = 2x_1 + 3y_1 - (k_1 + k_1' + k_2)N;$$
$$y_3 = x_2 + 2y_2 - k_2' N = 3x_1 + 5y_1 - (k_1 + 2k_1' + k_2')N$$

n=3 时，

$$x_4 = x_3 + y_3 - k_3 N = 5x_1 + 8y_1 - (2k_1 + 3k_1' + k_2 + k_2' + k_3)N;$$
$$y_4 = x_3 + 2y_3 - k_3' N = 8x_1 + 13y_1 - (3k_1 + 5k_1' + k_2 + 2k_2' + k_3')N$$

显然，k_n, k_n' 都是大于等于 0 的整数

综上：

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} (\mathrm{mod}\ \mathrm{N})$$

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} (\mathrm{mod}\ \mathrm{N})$$

$$\begin{bmatrix} x_4 \\ y_4 \end{bmatrix} = \begin{bmatrix} 5 & 8 \\ 8 & 13 \end{bmatrix} \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} (\mathrm{mod}\ \mathrm{N})$$

……

递推得：

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^n \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} (\mathrm{mod}\ \mathrm{N}) \qquad (3)$$

由此可见：若要得到置乱 n 次的图像，可先计算$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^n$，然后利用上式直接从原始图像的位置坐标$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$得到。

2.2 仿真结果

要得到 n 次置乱结果，首先通过传统的算法计算出 n 次置乱后的像素矩阵，再通过计算变换矩阵，即$\begin{bmatrix} 1 & b \\ a & ab+1 \end{bmatrix}$（a=b=1）的 n 次方，直接得到 n 次置乱后的像素矩阵。分别在 19 张不同的 512*512 8bit 灰度图像上，1 次、3 次、10 次置乱得到 19*3 个两两一组的矩阵并比较，仿真结果验证了 2.1 的推导，两个像素矩阵完全一样。灰度图像见文件夹 images，代码见文件 vs2ways.py。

3. 关于“基于 Arnold 变换的图像置乱的周期与图像长或宽的关系的一点推导和仿真实验。

3.1 数学推导

由周期的性质可知:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} \equiv \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^n \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} (\text{mod N}) \tag{4}$$

求得$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^n =$

$$\begin{bmatrix} \frac{(\sqrt{5}+3)^n(-1+\sqrt{5})+(-\sqrt{5}+3)^n(1+\sqrt{5})}{\sqrt{5}\times 2^{n+1}} & \frac{(\sqrt{5}+3)^n-(-\sqrt{5}+3)^n}{\sqrt{5}\times 2^n} \\ \frac{(\sqrt{5}+3)^n-(-\sqrt{5}+3)^n}{\sqrt{5}\times 2^n} & \frac{(\sqrt{5}+3)^n(1+\sqrt{5})+(-\sqrt{5}+3)^n(-1+\sqrt{5})}{\sqrt{5}\times 2^{n+1}} \end{bmatrix} \tag{5}$$

（求解具体过程略）

所以不妨设$\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^n = \begin{bmatrix} a & Y \\ Y & b \end{bmatrix}$, 其中 n 是周期. 所以,

$$\begin{bmatrix} a & Y \\ Y & b \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \equiv \begin{bmatrix} ax_1 + Yy_1 \\ Yx_1 + by_1 \end{bmatrix} \equiv \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} (mod\ N) \tag{6}$$

所以,

$$x_1 \equiv ax_1 + Yy_1 (mod\ N)$$
$$y_1 \equiv Yx_1 + by_1 (mod\ N)$$

所以,

$$Y \equiv 0 (mod\ N)$$
$$a \equiv b \equiv 1 (mod\ N)$$

所以,

$$Y = \frac{(\sqrt{5}+3)^n - (-\sqrt{5}+3)^n}{\sqrt{5} \times 2^n} = kN$$

即

$$\frac{(\sqrt{5}+3)^T-(-\sqrt{5}+3)^T}{\sqrt{5}\times 2^T} = kN,\ k\epsilon z^+ \tag{7}$$

(7)化简后是因变量周期 T 与自变量 N 的关系。

3.2 仿真结果

将 8bit 512*512 lena.bmp 图像的长或宽(N)分别缩放至$[4, 256]$区间，批量生成不同 N*N 的图片，其中 N∈ $[4, 256]$。对于每个 N，一次次地进行 Arnold 置乱，并将每次的置乱结果的像素矩阵和原始图片的像素矩阵进行比较。若两个矩阵完全一样，则找到了这个个 N 的最小周期；若两个矩阵不完全一样，则继续下一次的 Arnold 置乱。

当$N \in [4, 256]$时，得到$T \in$ {3, 10, 12, 8, 6, 12, 30, 5, 12, 14, 24, 20, 12, 18, 12, 9, 30, 8, 15, 24, 12, 50, 42, 36, 24, 7, 60, 15, 24, 20, 18, 40, 12, 38, 9, 28, 30, 20, 24, 44, 15, 60, 24, 16, 12, 56, 150, 36, 42, 54, 36, 10, 24, 36, 21, 29, 60, 30, 15, 24, 48, 70, 60, 68, 18, 24, 120, 35, 12, 74, 114, 100, 9, 40, 84, 39, 60, 108, 60, 84, 24, 90, 132, 28, 30, 22, 60, 56, 24, 60, 48, 90, 24, 98, 168, 60, 150, 25, 36, 104, 42, 40, 54, 36, 36, 54, 30, 76, 24, 38, 36, 120, 21, 84, 87, 72, 60, 55, 30, 20, 15, 250, 24, 128, 96, 44, 210, 65, 60, 72, 204, 180, 18, 138, 24, 23, 120, 16, 105, 70, 12, 70, 222, 56, 114, 74, 300, 25, 18, 36, 120, 30, 84, 158, 39, 108, 120, 24, 108, 164, 60, 20, 84, 168, 24, 182, 90, 36, 132, 174, 84, 200, 60, 116, 66, 89, 60, 45, 168, 60, 24, 190, 60, 90, 48, 72, 90, 95, 48, 194, 294, 140, 168, 198, 60, 11, 150, 68, 75, 56, 36, 20, 312, 24, 84, 45, 120, 21, 54, 140, 36, 220, 36, 120, 54, 148, 30, 126, 228, 224, 24, 300, 114, 228, 36, 57, 120, 40, 42, 26, 84, 80, 87, 156, 72, 119, 60, 120, 165, 324, 30, 280, 60, 126, 30, 84, 750, 125, 24, 120, 384, 180, 192}（一一对应）验证了数学推导的结果，绘图如下：

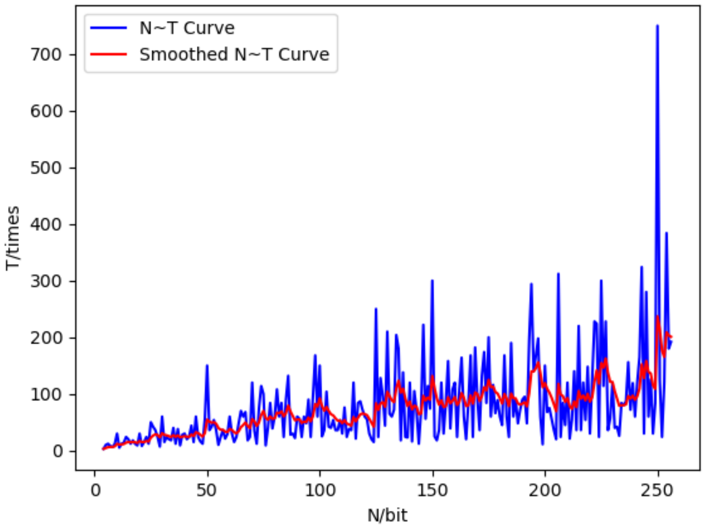


Fig.3. 将 512*512 的 lena 图缩小至 n*n，其中 n∈ [4,256]；代码见文件 find_cycle.py。

4.1 基于 Arnold 变换的图像置乱的优缺点

优点：较好地破坏了像素之间的相关性；保持了文件格式的一致性；计算量小，实时性较好。

缺点：加密时要求图像长宽相等，适用性小；图像的灰度直方图等像素统计信息加密前后保持不变，泄露了原图像部分信息；变换具有周期，若有经过 k 次变换后的图像，再经过 T-k 次变换图像复原，破译难度小。

鉴于以上优缺点，提出如下两点改进方法。

4.2 改进方法及其仿真实验

4.2.1 Arnold 置乱长宽不等的图像，可将该图像填充约定的花纹或者像素，以弥补适用范围小的不足，还可以一定程度上掩盖灰度直方图等统计信息。

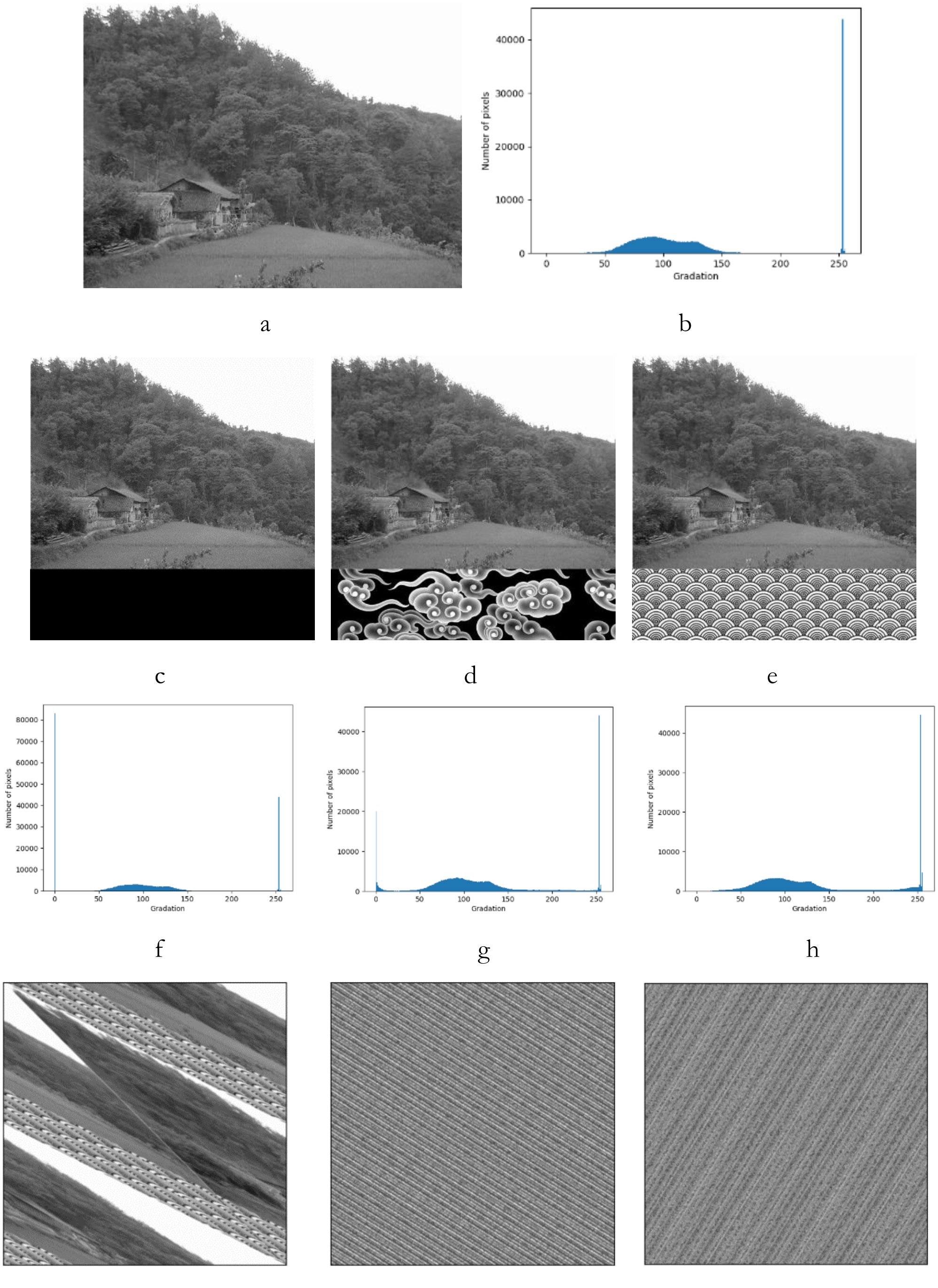


Fig.4. (a)是 hometown.bmp 原图; (b)是(a)对应的灰度直方图; (c)-(e)是用不同花纹填充的矩形图像; (f)-(h)分别是(c)-(e)对应的灰度直方图; (i)-(k)是(e)置乱 1 次、5 次、20 次得到的图像。代码见 filling.py。

从图 4(a)(i)(j)(k)可以看出，填充花纹后的矩形图像可以进行 Arnold 置乱，并且对比图 4(b)和(c)-(e)，花纹的填充一定程度上掩盖了原矩形图像的灰度直方图的统计信息。

4.2.2 每个比特面分别进行不等次的 Arnold 置乱。这样既可以增加密钥长度，使密钥空间变大，破译难度变大，以更好地达到安全性拓展依赖于密钥的保密性的目的；又可以改变灰度直方图等统计信息。

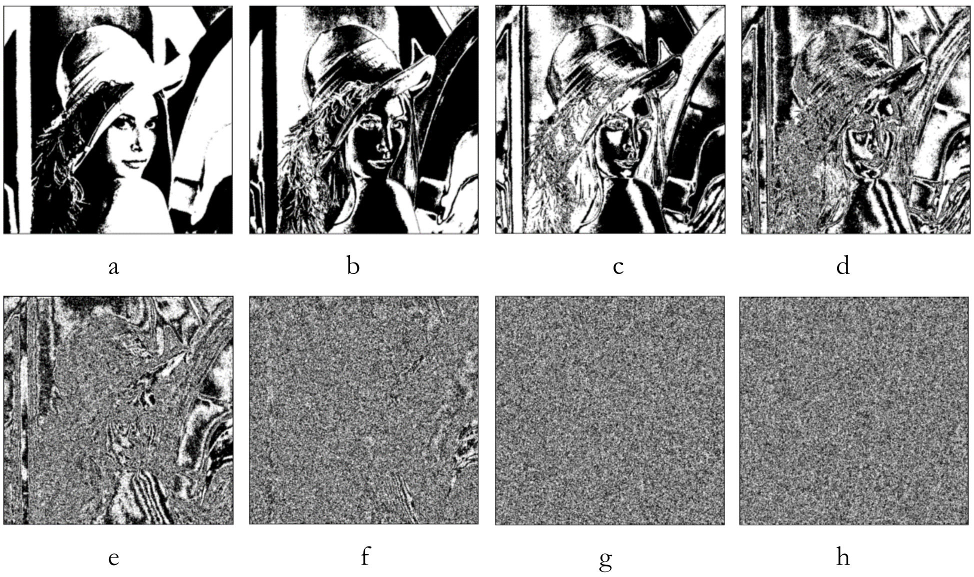


Fig.5. 位平面举例。(a)-(h)是 lena.bmp 分别为位平面 7 至位平面 0 的图像。代码见文件 bitmap.py。

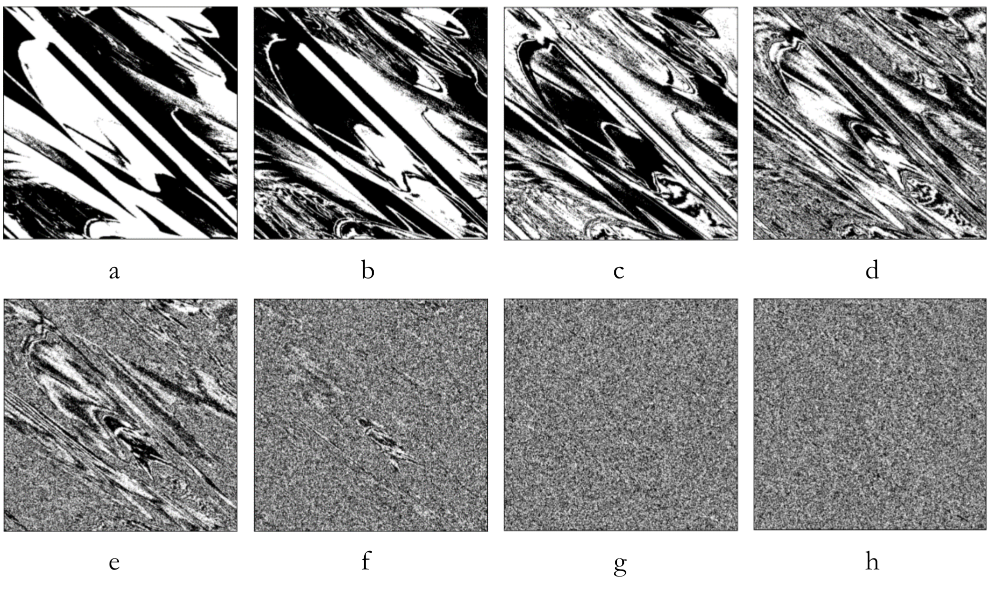
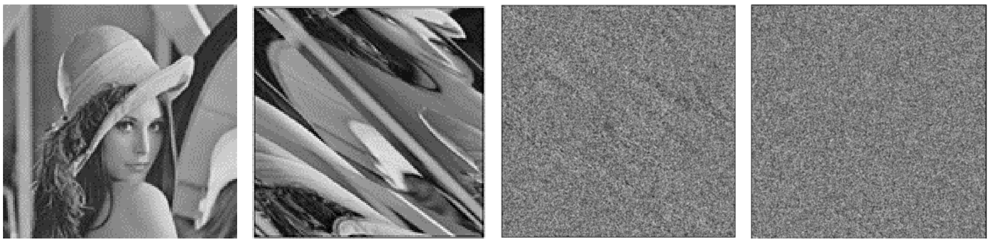


Fig.6. (a)-(h)lena.bmp 位平面 7-位平面 1 分别 Arnold 置乱 1 次得到的图像。代码见文件 bitmap.py。

a b c d

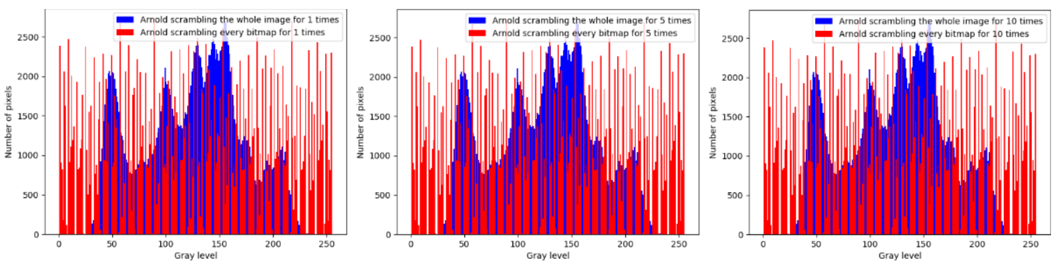
Fig.7. 每个位图 Arnold 置乱解密举例。(a)是 lena.bmp 每个位平面 Arnold 置乱 1 次所得到的图像; (b)是根据(a)解密得到的图像; (c)是 pirate.bmp 用位平面 0-7 分别用{8, 9, 14, 23, 15, 7, 1, 3}置乱得到的图像；(d)是从(c)解密出来的图像。代码见文 bitmap.py。



a b c d

Fig.8. (a)是 8bit 512*512 的 lena.bmp 原图; (b)是(a)Arnold 置乱 1 次所得到的图像; (c)(d)分别是(a)每个位平面 Arnold 置乱 1、5 次所得到的图像。代码见文件 bitmap.py。

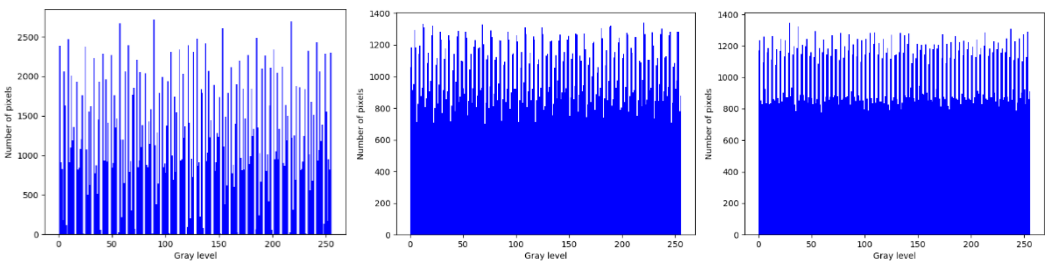
这里值得注意的是，图像的每个比特面分别置乱 n 次会和整幅图像进行同样 n 次的 Arnold 置乱得到的结果相同吗？我开始预测会，但是仿真实验的结果跟我预测的恰恰相反，如图 8 所示。*这是为什么呢？*



a b c

Fig.9. 通过比较得知这三幅图片完全一样;(a)(b)(c)分别是 lena.bmp 置乱 1、5、10 次和 lena.bmp 每个位图置乱 1、5、10 次所得到的图像的灰度直方图。代码见 bitmap.py。

更值得注意的是，通过对比同一张图像 8 个比特面置乱不同的相同次，发现得到的灰度直方图都是一样的，但是得到的图像并不相同；例如将 lena.bmp 的 8 个比特面都置乱 1 次，和将 lena.bmp 的 8 个比特面都置乱 5 次得到的图像的灰度直方图是一样的，如图 9(a)(b)和图 8(c)(d)所示。*这是为啥?* 更进一步，对比每个比特面置乱 k 次然后整幅图像置乱 b 次得到的图像和每个比特面置乱(k+b)次得到的图像，两者并不相同。



a b c

Fig.10. (a)是 lena.bmp 每个位平面置乱 5 次得到的图片的灰度直方图; (b)(c)是 lena.bmp 位平面 0-7 分别置乱{1, 2, 3, 4, 5, 6, 7, 8}、{8, 9, 14, 23, 15, 7, 1, 3}得到的图片的灰度直方图。代码见 bitmap.py。

通过对比同一张图像 8 个比特面都置乱相同次，和置乱不同次，可以得出它们的灰度直方图不相同，如图 10 对比(a)和(b)(c)，并且可以观察到，比特面置乱不同次比相同次的灰度直方图的“块现象”更明显。

这些说明，这种改进方法只能比较有限地掩盖原图像的灰度直方图的信息，但是能增加密钥长度。

5. 参考文献：曹老师课件