Randomness in Computing

➢ RC4+

Instructor : Dr. Baha' A. Alsaify

Course: NES452 – Cryptography and Network Security

Date: 29/4/2016

Group Number: 6

Prepared by: Abrar Al-Taj - 20120175038
Alaa Abu-Hantash - 20120175015

CONTENTS

INTRODUCTION

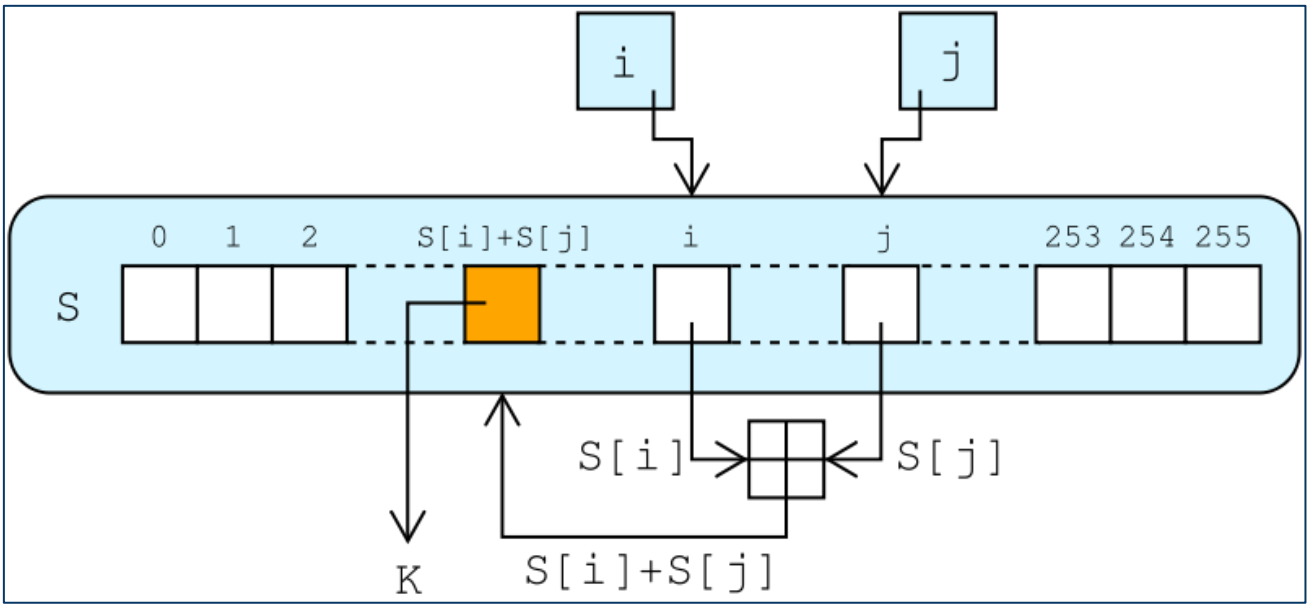
RC4 is a variable-key size stream cipher and it's one of the most used software-based stream ciphers in the world. The cipher is included in popular Internet protocols such as SSL (Secure Sockets Layer) and WEP (for wireless network security). The cipher is fairly simplistic when compared to competing algorithms of the same strength and boasts one of the fastest speeds of the same family of algorithms.

How Does RC4 Work?

RC4 creates a pseudorandom stream of bits that is also referred to as a key stream. Similar to other stream ciphers, the key stream can be leveraged for encryption operations by combining it with plaintext using the XOR operation. Decryption works similarly through the involution of the cipher text. In order to create the key stream, the RC4 cipher will use a secret internal state comprised of two items:

- Two, eight bit pointers that are represented by the letters "I" and "j".
- A permutation of all 256 possible bytes that is connoted by the letter "S."

The RC4 permutation is initialized using a variable length key. This key will typically range between 40 and 256 bits that use the KSA (key scheduling algorithm). Once the permutation is created, the stream of bits will be created using the PRGA (pseudo-random generation algorithm).



RC4 Memory Requirements

The design and implementation of RC4 avoids the use of LSFRs (linear feedback shift registers) for conducting operations since they are not as efficient when implemented in hardware as compared to software. Since RC4 only uses byte manipulations, it only needs 256 bytes of computer memory for the algorithm's state array, k bytes of memory for the RC4 key, and a small amount of memory for the integer values for the I and j array indices. The modulo 256 operation required by the algorithm can be accomplished with a bitwise AND and 255 to have a smaller memory footprint as well.

What are the Variants of RC4?

The predominant weakness of RC4 is the insufficient key schedule since the first bytes of the output will provide information about the key. The issue can be corrected through discarding the first bytes of the output stream in the RC4-dropN variant of the algorithm, with the "N" connoting the total number of bytes to be discarded. N is typically a multiple of 256 with 768 or 1024 bytes being common options.

1. RC4 A
2. VMPC
3. **RC4+: which we search about it.**

❖RC4+

RC4+ stream cipher was proposed by Maitra et. al. at Indocrypt 2008. It was claimed by the authors that this new stream cipher is designed to overcome all the weaknesses reported on the alleged RC4 stream cipher. In the design specifications of RC4+, the authors make use of an 8-bit design parameter called pad which is fixed to the value 0xAA. The first Distinguishing Attack on RC4+ based on the bias of its first output byte was shown by Banik et. al. in Indocrypt 2013. It was also mentioned that the distinguishing attack would still hold if the pad used in RC4+ is fixed to any even 8-bit constant other than 0xAA.

RC4+ is another modified RC4 algorithm. RC4+ makes use of a very complex, three-phase key scheduler. It takes up to three times as long as RC4 to function, and includes a more complex output function that makes four more lookups in the S array than the original RC4 algorithm does for a single output byte. Overall, RC4+ takes approximately 1.7 times as long as RC4 to operate.

CODING

Link of the code: http://ideone.com/98GQCc

```c
#include <stdio.h>
#include <string.h>
#include <time.h>
int s[256], t[256], keyLength, key[256];
void swap(int *a, int *b) {
    int tmp = *a; *a = *b;  *b = tmp;
}
int main() {
    puts("Enter the key length :");
    scanf("%d", &keyLength);
    if (keyLength > 256 || keyLength < 1) {
        puts("key must be between 1 and 256 bytes");
        return 0;
    }
    time_t tt; srand((unsigned) time(&tt));
    int i, j;
    for (i = 0; i < keyLength; i++)
        key[i] = rand() % 256; // Each value must be 8 bits
    for (i = 0; i < 256; i++) {
        s[i] = i;
        t[i] = key[i % keyLength];
    }
    for (i = j = 0; i < 256; i++) {
        j = (j + s[i] + t[i]) % 256;
        swap(&s[i], &s[j]);
    }
    int Result_Key, t, cnt, a, nub;
    i = 0, j = 0;
    puts("Enter the number of keys you want to generate :");
    scanf("%d", &nub);
    for (cnt = 0; cnt < nub; cnt++) {
        i = (i + 1) % 265;
        a = s[i];
        j = (j + s[i]) % 265;
        swap(&s[i], &s[j]);
        t = (s[((i << 5) ^ (j >> 3)) % 256] + s[((j << 5) ^ (i >> 3)) % 256])
                % 265;
        Result_Key = ((s[(a + s[i]) % 256] + s[(t ^ (0xAA)) % 256])
                ^ s[(j + s[i]) % 256]) % 265;
        printf("key %d: %ld \n", cnt + 1, Result_Key);
    }
    return 0;
}

```

RC4+ has a three-layer key scheduling algorithm.

- The physical structure of the RC4+ stream cipher is same as RC4. The first layer of Key Scheduling Algorithm (KSA) of the RC4+ stream cipher is similar to KSA of RC4. RC4+ consists of a permutation S of N = 256 elements. The elements of S come from the integer ring Z_{256}. It also consists of two index pointers i and j. Each of these pointers has the size of 1 byte.
- In the second layer of KSA, the permutation S is further scrambled.
- Finally the third layer is PRGA.
    - The PRGA routine of RC4+ has a slight deviation from the simplistic PRGA structure of RC4. The designers proposed a bit different PRGA in order to protect the cipher.
    - To protect the cipher design against the well known aforementioned attacks, the designers choose to make the output keystream byte functions of a few other locations of the permutation array S.
    - The equation below the exact details of the PRGA routine of RC4+ where and represents the left and right bitwise shifts respectively

$$(S_0[i \gg 3 \oplus j \ll 5] + S_0[i \ll 5 \oplus j \gg 3]) \oplus p.$$

    - Note that the pad p used in the PRGA by the designers of the RC4+ encryption scheme is 0xAA
- The main Functions in RC4+ :
    - Addition.
    - Bitwise XOR.
    - Right bitwise shift.
    - Left bitwise shift.

SETS OF PSEUDORANDOM NUMBERS

- Each set have a different seed.

Set#1		Set#2		Set#3	
Nub 1: 187	Nub 51: 243	Nub 1: 187	Nub 51: 243	Nub 1: 251	Nub 51: 203
Nub 2: 126	Nub 52: 200	Nub 2: 126	Nub 52: 200	Nub 2: 212	Nub 52: 59
Nub 3: 156	Nub 53: 143	Nub 3: 156	Nub 53: 143	Nub 3: 162	Nub 53: 19
Nub 4: 89	Nub 54: 179	Nub 4: 89	Nub 54: 179	Nub 4: 105	Nub 54: 250
Nub 5: 228	Nub 55: 113	Nub 5: 228	Nub 55: 113	Nub 5: 25	Nub 55: 34
Nub 6: 73	Nub 56: 217	Nub 6: 73	Nub 56: 217	Nub 6: 184	Nub 56: 87
Nub 7: 128	Nub 57: 99	Nub 7: 128	Nub 57: 99	Nub 7: 109	Nub 57: 156
Nub 8: 133	Nub 58: 99	Nub 8: 133	Nub 58: 99	Nub 8: 248	Nub 58: 189
Nub 9: 94	Nub 59: 109	Nub 9: 94	Nub 59: 109	Nub 9: 7	Nub 59: 64
Nub 10: 74	Nub 60: 43	Nub 10: 74	Nub 60: 43	Nub 10: 66	Nub 60: 63
Nub 11: 237	Nub 61: 103	Nub 11: 237	Nub 61: 103	Nub 11: 125	Nub 61: 28
Nub 12: 18	Nub 62: 3	Nub 12: 18	Nub 62: 3	Nub 12: 172	Nub 62: 141
Nub 13: 43	Nub 63: 155	Nub 13: 43	Nub 63: 155	Nub 13: 252	Nub 63: 226
Nub 14: 248	Nub 64: 249	Nub 14: 248	Nub 64: 249	Nub 14: 39	Nub 64: 173
Nub 15: 96	Nub 65: 194	Nub 15: 96	Nub 65: 194	Nub 15: 49	Nub 65: 253
Nub 16: 148	Nub 66: 234	Nub 16: 148	Nub 66: 234	Nub 16: 69	Nub 66: 131
Nub 17: 221	Nub 67: 101	Nub 17: 221	Nub 67: 101	Nub 17: 89	Nub 67: 207
Nub 18: 243	Nub 68: 105	Nub 18: 243	Nub 68: 105	Nub 18: 148	Nub 68: 38
Nub 19: 35	Nub 69: 15	Nub 19: 35	Nub 69: 15	Nub 19: 50	Nub 69: 233
Nub 20: 29	Nub 70: 190	Nub 20: 29	Nub 70: 190	Nub 20: 75	Nub 70: 209
Nub 21: 90	Nub 71: 49	Nub 21: 90	Nub 71: 49	Nub 21: 85	Nub 71: 86
Nub 22: 61	Nub 72: 141	Nub 22: 61	Nub 72: 141	Nub 22: 225	Nub 72: 177
Nub 23: 148	Nub 73: 155	Nub 23: 148	Nub 73: 155	Nub 23: 174	Nub 73: 81
Nub 24: 184	Nub 74: 97	Nub 24: 184	Nub 74: 97	Nub 24: 220	Nub 74: 130
Nub 25: 83	Nub 75: 120	Nub 25: 83	Nub 75: 120	Nub 25: 193	Nub 75: 223
Nub 26: 141	Nub 76: 8	Nub 26: 141	Nub 76: 8	Nub 26: 150	Nub 76: 120
Nub 27: 28	Nub 77: 225	Nub 27: 28	Nub 77: 225	Nub 27: 155	Nub 77: 83
Nub 28: 32	Nub 78: 157	Nub 28: 32	Nub 78: 157	Nub 28: 108	Nub 78: 222
Nub 29: 179	Nub 79: 157	Nub 29: 179	Nub 79: 157	Nub 29: 144	Nub 79: 247
Nub 30: 216	Nub 80: 92	Nub 30: 216	Nub 80: 92	Nub 30: 27	Nub 80: 73
Nub 31: 234	Nub 81: 211	Nub 31: 234	Nub 81: 211	Nub 31: 77	Nub 81: 15
Nub 32: 21	Nub 82: 21	Nub 32: 21	Nub 82: 21	Nub 32: 163	Nub 82: 196
Nub 33: 180	Nub 83: 252	Nub 33: 180	Nub 83: 252	Nub 33: 114	Nub 83: 142
Nub 34: 135	Nub 84: 114	Nub 34: 135	Nub 84: 114	Nub 34: 239	Nub 84: 199
Nub 35: 92	Nub 85: 190	Nub 35: 92	Nub 85: 190	Nub 35: 17	Nub 85: 191
Nub 36: 72	Nub 86: 57	Nub 36: 72	Nub 86: 57	Nub 36: 8	Nub 86: 18
Nub 37: 86	Nub 87: 186	Nub 37: 86	Nub 87: 186	Nub 37: 168	Nub 87: 244

Nub 38: 29	Nub 88: 15	Nub 38: 29	Nub 88: 15	Nub 38: 122	Nub 88: 88
Nub 39: 124	Nub 89: 53	Nub 39: 124	Nub 89: 53	Nub 39: 9	Nub 89: 237
Nub 40: 145	Nub 90: 206	Nub 40: 145	Nub 90: 206	Nub 40: 48	Nub 90: 180
Nub 41: 176	Nub 91: 168	Nub 41: 176	Nub 91: 168	Nub 41: 61	Nub 91: 151
Nub 42: 85	Nub 92: 230	Nub 42: 85	Nub 92: 230	Nub 42: 127	Nub 92: 197
Nub 43: 204	Nub 93: 223	Nub 43: 204	Nub 93: 223	Nub 43: 161	Nub 93: 55
Nub 44: 209	Nub 94: 211	Nub 44: 209	Nub 94: 211	Nub 44: 57	Nub 94: 249
Nub 45: 53	Nub 95: 202	Nub 45: 53	Nub 95: 202	Nub 45: 166	Nub 95: 13
Nub 46: 229	Nub 96: 147	Nub 46: 229	Nub 96: 147	Nub 46: 218	Nub 96: 181
Nub 47: 14	Nub 97: 127	Nub 47: 14	Nub 97: 127	Nub 47: 126	Nub 97: 106
Nub 48: 25	Nub 98: 174	Nub 48: 25	Nub 98: 174	Nub 48: 1	Nub 98: 211
Nub 49: 246	Nub 99: 177	Nub 49: 246	Nub 99: 177	Nub 49: 111	Nub 99: 229
Nub 50: 134	Nub 100: 229	Nub 50: 134	Nub 100: 229	Nub 50: 176	Nub 100: 210

TESTING

1. **Frequency Test**
   - This test to know how many times each number repeats.
   - If there is a number that repeats large number of times from other number in the same set, we can know that this set does not imply the randomness.

Results of testing sets with frequency test

Set#1

Frequency of 130 = 1	Frequency of 16 = 1	Frequency of 14 = 2
Frequency of 151 = 1	Frequency of 205 = 1	Frequency of 10 = 1
Frequency of 86 = 2	Frequency of 44 = 1	Frequency of 229 = 2
Frequency of 197 = 1	Frequency of 213 = 1	Frequency of 252 = 1
Frequency of 169 = 1	Frequency of 112 = 1	Frequency of 219 = 1
Frequency of 140 = 1	Frequency of 63 = 1	Frequency of 191 = 2
Frequency of 250 = 1	Frequency of 236 = 1	Frequency of 215 = 1
Frequency of 190 = 1	Frequency of 239 = 1	Frequency of 156 = 1
Frequency of 127 = 1	Frequency of 129 = 1	Frequency of 40 = 1
Frequency of 32 = 1	Frequency of 171 = 1	Frequency of 36 = 1
Frequency of 26 = 1	Frequency of 81 = 1	Frequency of 93 = 1
Frequency of 97 = 1	Frequency of 18 = 2	Frequency of 225 = 1
Frequency of 135 = 1	Frequency of 2 = 1	Frequency of 177 = 1
Frequency of 263 = 1	Frequency of 134 = 2	Frequency of 167 = 2
Frequency of 59 = 1	Frequency of 235 = 1	Frequency of 1 = 1
Frequency of 139 = 1	Frequency of 88 = 1	Frequency of 195 = 2
Frequency of 96 = 1	Frequency of 100 = 1	Frequency of 165 = 1
Frequency of 104 = 1	Frequency of 227 = 1	Frequency of 116 = 1
Frequency of 27 = 1	Frequency of 147 = 2	Frequency of 98 = 2
Frequency of 35 = 1	Frequency of 85 = 1	Frequency of 70 = 1
Frequency of 75 = 1	Frequency of 248 = 1	Frequency of 111 = 1
Frequency of 74 = 1	Frequency of 30 = 4	Frequency of 83 = 1
Frequency of 31 = 1	Frequency of 144 = 1	Frequency of 37 = 1
Frequency of 204 = 1	Frequency of 222 = 1	Frequency of 162 = 1
Frequency of 189 = 1	Frequency of 160 = 1	Frequency of 0 = 1
Frequency of 80 = 1		
Frequency of 19 = 1		
Frequency of 103 = 1		
Frequency of 4 = 1		

Set#2		
Frequency of 109 = 1	Frequency of 32 = 1	Frequency of 187 = 1
Frequency of 103 = 1	Frequency of 179 = 2	Frequency of 126 = 1
Frequency of 3 = 1	Frequency of 216 = 1	Frequency of 156 = 1
Frequency of 155 = 2	Frequency of 234 = 2	Frequency of 89 = 1
Frequency of 249 = 1	Frequency of 21 = 2	Frequency of 228 = 1
Frequency of 194 = 1	Frequency of 180 = 1	Frequency of 73 = 1
Frequency of 101 = 1	Frequency of 135 = 1	Frequency of 128 = 1
Frequency of 105 = 1	Frequency of 92 = 2	Frequency of 133 = 1
Frequency of 15 = 2	Frequency of 72 = 1	Frequency of 94 = 1
Frequency of 190 = 2	Frequency of 86 = 1	Frequency of 74 = 1
Frequency of 49 = 1	Frequency of 124 = 1	Frequency of 237 = 1
Frequency of 97 = 1	Frequency of 145 = 1	Frequency of 18 = 1
Frequency of 120 = 1	Frequency of 176 = 1	Frequency of 43 = 2
Frequency of 8 = 1	Frequency of 85 = 1	Frequency of 248 = 1
Frequency of 225 = 1	Frequency of 204 = 1	Frequency of 96 = 1
Frequency of 157 = 2	Frequency of 209 = 1	Frequency of 148 = 2
Frequency of 211 = 2	Frequency of 53 = 2	Frequency of 221 = 1
Frequency of 252 = 1	Frequency of 229 = 2	Frequency of 243 = 2
Frequency of 114 = 1	Frequency of 14 = 1	Frequency of 35 = 1
Frequency of 57 = 1	Frequency of 25 = 1	Frequency of 29 = 2
Frequency of 186 = 1	Frequency of 246 = 1	Frequency of 90 = 1
Frequency of 206 = 1	Frequency of 134 = 1	Frequency of 61 = 1
Frequency of 168 = 1	Frequency of 200 = 1	Frequency of 184 = 1
Frequency of 230 = 1	Frequency of 143 = 1	Frequency of 83 = 1
Frequency of 223 = 1	Frequency of 113 = 1	Frequency of 141 = 2
Frequency of 202 = 1	Frequency of 217 = 1	Frequency of 28 = 1
Frequency of 147 = 1	Frequency of 99 = 2	
Frequency of 127 = 1		
Frequency of 174 = 1		
Frequency of 177 = 1		

Set #3		
Frequency of 251 = 1	Frequency of 168 = 1	Frequency of 233 = 1
Frequency of 212 = 1	Frequency of 122 = 1	Frequency of 209 = 1
Frequency of 162 = 1	Frequency of 9 = 1	Frequency of 86 = 1
Frequency of 105 = 1	Frequency of 48 = 1	Frequency of 177 = 1
Frequency of 25 = 1	Frequency of 61 = 1	Frequency of 81 = 1
Frequency of 184 = 1	Frequency of 127 = 1	Frequency of 130 = 1
Frequency of 109 = 1	Frequency of 161 = 1	Frequency of 223 = 1
Frequency of 248 = 1	Frequency of 57 = 1	Frequency of 120 = 1
Frequency of 7 = 1	Frequency of 166 = 1	Frequency of 83 = 1
Frequency of 66 = 1	Frequency of 218 = 1	Frequency of 222 = 1
Frequency of 125 = 1	Frequency of 126 = 1	Frequency of 247 = 1
Frequency of 172 = 1	Frequency of 1 = 1	Frequency of 73 = 1
Frequency of 252 = 1	Frequency of 111 = 1	Frequency of 15 = 1
Frequency of 39 = 1	Frequency of 176 = 1	Frequency of 196 = 1
Frequency of 49 = 1	Frequency of 203 = 1	Frequency of 142 = 1
Frequency of 69 = 1	Frequency of 59 = 1	Frequency of 199 = 1
Frequency of 89 = 1	Frequency of 19 = 1	Frequency of 191 = 1
Frequency of 148 = 1	Frequency of 250 = 1	Frequency of 18 = 1
Frequency of 50 = 1	Frequency of 34 = 1	Frequency of 244 = 1
Frequency of 75 = 1	Frequency of 87 = 1	Frequency of 88 = 1
Frequency of 85 = 1	Frequency of 156 = 1	Frequency of 237 = 1
Frequency of 225 = 1	Frequency of 189 = 1	Frequency of 180 = 1
Frequency of 174 = 1	Frequency of 64 = 1	Frequency of 151 = 1
Frequency of 220 = 1	Frequency of 63 = 1	Frequency of 197 = 1
Frequency of 193 = 1	Frequency of 28 = 1	Frequency of 55 = 1
Frequency of 150 = 1	Frequency of 141 = 1	Frequency of 249 = 1
Frequency of 155 = 1	Frequency of 226 = 1	Frequency of 13 = 1
Frequency of 108 = 1	Frequency of 173 = 1	Frequency of 181 = 1
Frequency of 144 = 1	Frequency of 253 = 1	Frequency of 106 = 1
Frequency of 27 = 1	Frequency of 131 = 1	Frequency of 211 = 1
Frequency of 77 = 1	Frequency of 207 = 1	Frequency of 229 = 1
Frequency of 163 = 1	Frequency of 38 = 1	Frequency of 210 = 1
Frequency of 114 = 1		
Frequency of 239 = 1		
Frequency of 17 = 1		
Frequency of 8 = 1		

Link for the code: http://ideone.com/8bbt9R

```c
#include <stdio.h>

int arr[100000000] = {}, res[100000000] = {}, count;

int main() {
    int n, cnt = 0, x, i;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &x);
        if (arr[x] == 0)
            res[cnt++] = x;
        arr[x]++;
    }
    for (i = 0; i < cnt; i++)
        printf("Frequency of %d  = %d\n", res[i], arr[res[i]]);
    return 0;
}
```

2. **Runstest in Matlab**
   - This test is defined in Matlab for randomness
   - This is a test of the hypothesis that the values in set come in a random order, against the alternative that the ordering is not random.
   - The test is based on the number of runs of consecutive values above or below the mean of set.
   - The Output Arguments of the test:
     1. H: Hypothesis test result:
        - Hypothesis test result, returned as a logical value.
        - If h = 1, then runstest rejects the null hypothesis at the Alpha significance level and if h = 0, then runstest fails to reject the null hypothesis at the Alpha (5%) significance level ("sequence is random").
        - The result in runstest is based on the number of runs of consecutive values above or below the mean of x. Too few runs indicate a tendency for high and low values to cluster. Too many runs indicate a tendency for high and low values to alternate.
        - runstest uses a test statistic which is the difference between the number of runs and its mean, divided by its standard deviation. The test

statistic is approximately normally distributed when the null hypothesis is true.

2. p-value:
   - scalar value in the range [0,1]
   - p-value of the test, returned as a scalar value in the range [0,1].
   - For typical analysis, using the standard a = 0.05 cutoff, a widely used interpretation is:
   - A small p-value (≤ 0.05) indicates strong evidence against the null hypothesis, so it is rejected.
   - A large p-value (> 0.05) indicates weak evidence against the null hypothesis (fail to reject).
   - p-values very close to the cutoff (~ 0.05) are considered to be marginal (need attention).
3. Stats - Test data:
   - Test data, returned as a structure with the following fields.
     - nruns : The number of runs
     - n1: The number of values above v
     - n0: The number of values below v
     - z: The test statistic (which is the difference between the number of runs and its mean, divided by its standard deviation. The test statistic is approximately normally distributed when the null hypothesis is true).

Results of testing sets with runstest

Set#1	Set#2	Set #3

`>> [H,P,Stats] = runstest(Set_1)`
 	
`H =`
 	
`0`
 	
`P =`
 	
`0.2287`
 	
`Stats =`
 	
`nruns: 44`
`n1: 48`
`n0: 51`
`z: -1.2042`
	`>> [H,P,Stats] = runstest(Set_2)`
	
	`H =`
	
	`0`
	
	`P =`
	
	`1`
	
	`Stats =`
	
	`nruns: 51`
	`n1: 51`
	`n0: 49`
	`z: -0.0965`
		`>> [H,P,Stats] = runstest(Set_3)`
		
		`H =`
		
		`0`
		
		`P =`
		
		`0.6848`
		
		`Stats =`
		
		`nruns: 53`
		`n1: 50`
		`n0: 49`
		`z: 0.4051`

ANALYZING

FREQUENCY TEST

- From the results of the test, we can notice that all the numbers in set1 have a frequency value in the range [1, 4]. Almost All the numbers have a frequency value equal to 1, 10% of the numbers have a frequency value equal 2 and one number which equal to 30 has a frequency value equal to 4.
- For the set2 the frequency values either 1 or 2. Almost All the numbers have a frequency value equal to 1, 17% of the numbers have a frequency value equal 2.
- For the set3 all the numbers have a frequency value equal to 1.
- **These results can lead us to know that the numbers in this set are almost uniform distributed for the randomness.**

RUNSTEST

- The value of H = 0 means that this test fails to reject the null hypothesis and this indicated that the sequence is random.
- Using the standard a = 5%, A large p-value (> 0.05) **indicates weak evidence against the null hypothesis (fail to reject).**
    - For the set1 P = 0.2287 > 0.05
    - For the set2 P = 1 > 0.05
    - For the set3 P = 0.6848 > 0.05

CONCLUSION

RC4+ only requires byte manipulations and hence it is ideal for software implementation. Its simplistic design allows faster encryption in software.

As we have seen through the tests that the percentage of the randomness in the sets is very high.

However, these results were not 100% random. The reason of that (the numbers are not randomness 100%) is:

1. Different seed.
2. The size of the sets is too small (to have an exact results we must generate an infinite numbers).

At the end, we can say that RC4+ can generate random numbers.

REFERENCES

- https://en.wikipedia.org/wiki/RC4
- https://eprint.iacr.org/2014/900.pdf
- http://www.mathworks.com/help/stats/runstest.html#bubrl2u-2
- https://en.wikipedia.org/wiki/P-value