به نام خدا

گزارش تمرین هشتم – درس رمزنگاری کاربردی

پیاده سازی و تست الگوریتم RC4

بنیامین تفرشیان

محمد مرادی

علی احمدی

در خصوص الگوریتم RC٤ می‌دانیم که RC٤ یک رمز دنباله ای با طول کلید متغیر بوده و عملیات آن روی بایت‌ها انجام می شود. الگوریتم بر مبنای استفاده از یک جایگشت تصادفی است. طبق بررسی ها و مطالعاتی که ما انجام دادیم تحلیل های موجود در مورد این رمز نشان می‌دهد که دوره تناوب رمز با احتمال قریب به یقین بزرگتر از $۱۰^{۱۰۰}$ است . برای تولید هر بایت خروجی بین ۸ تا ۱۶ عمل لازم است.

الگوریتم RC٤ به صورت قابل توجهی ساده است. و دارای یک کلید با طول متغیر ۱ تا ۲۵۶ است. برای شروع یک آرایه ۲۵۶ بایتی S با مؤلفه های S[۰], S[۱],...,S[۲۵۵] مورد استفاده قرار می گیرد. در همه حالات، S شامل جایگشت همه اعداد ۸ بیتی از صفر تا ۲۵۵ است. برای رمزنگاری و رمز گشائی، یک بایت k از میان ۲۵۵ مؤلفه S به صورت سیستماتیک انتخاب می شود. همینطور که هر مقدار k تولید می شود، مؤلفه های S یک بار دیگر جایگشت می یابند.

برای شروع، مقادیر صفر تا ۲۵۵ بصورت صعودی در مؤلفه های S قرار داده می شود، یعنی S[۰] = ۰ و S[۲۵۵] = ۲۵۵ همچنین یک آرایه موقت T هم ایجاد می شود. اگر طول کلید K برابر ۲۵۶ بایت باشد، آنگاه K به T منتقل می شود. در غیر اینصورت برای کلیدی با طول keylen بایت، اولین مؤلفه های T از K کپی شده و سپس K هر چندبار لازم باشد تکرار شده تا T پر شود. این عملیات ابتدائی را در تصویر زیر که مربوط به کد الگوریتم است مشاهده می کنید:

```
12      /* S & K initialization */
13      for(int i = 0 ; i < 256 ; i++)
14      {
15          S[i]=i;
16          T[i]= key[i % keyLen];
17      }
```

سپس از T برای جایگشت آغازین S استفاده می شود. این امر با S[۰] شروع شده و تا S[۲۵۵] ادامه می یابد. هر S[i] با بایت دیگری در S بر اساس روشی که توسط T[i] مشخص می شود، تعویض می گردد:

```
18      /* State Permutation */
19      for(int i = 0 ; i < 256; i++)
20      {
21          j = ( j + S[i] + T[i] ) % 256;
22
23          //Swap S[i] & S[j]
24          tmp = S[j];
25          S[j]= S[i];
26          S[i] = tmp;
27      }
```

چون تنها عمل روی S یک تعویض محل بایت هاست، تنها اثر این امر ایجاد یک جایگشت است S .همچنان شامل تمام اعداد بین صفر تا ۲۵۵ خواهد بود.

تولید دنباله

همین که بردار S با مقادیر اولیه پر شد، دیگر از کلید ورودی استفاده نخواهد شد. تولید دنباله شامل عبور از S[۰] تا S[۲۵۵] بوده و هر مقدار S [i] با بایت دیگری در S، بر حسب قانونی که توسط STATE فعلی S مشخص می شود، جایگزین می گردد. بعد از اینکه به S[۲۵۵] رسیدیم، پردازش با شروع مجدد از S[۰] ادامه می یابد:

```
28      j =0; // reintializing j to reuse it
29      for(int x = 0 ; x< dataLen ; x++)
30      {
31          i = (i+1) % 256; // using %256 to avoid exceed the array limit
32          j = (j + S[i])% 256; // using %256 to avoid exceed the array limit
33
34          //Swap S[i] & S[j]
35          tmp = S[j];
36          S[j]= S[i];
37          S[i] = tmp;
38
39          t = (S[i] + S[j]) % 256;
40
41          result[x]= data[x]^S[t]; // XOR generated S[t] with Byte from the plaintext / cipher and append each Encrypted/Decrypted byte to result array
42      }
43  }
```

برای رمزنگاری، اندازه k را با بایت بعدی متن ساده XOR می کنیم. برای رمز گشائی، اندازه k را با بایت بعدی متن رمز شده XORمی کنیم.