A New Substitution Cipher - Random-X

Falguni Patel, Mohammed Farik

Abstract: Ciphers are the encryption methods to prepare the algorithm for encryption and decryption. The currently known ciphers are not strong enough to protect the data. A new substitution cipher – Random-X, that we introduce in this paper, can be used for password encryption and data encryption. Random-X cipher is a unique substitution cipher which replaces the units of plaintext with triplets of letters. The beauty of this cipher is that the encrypted string of the same plain text is not always same. This makes it strong and difficult to crack. This paper covers the principle, the implementation ideas and testing of Random-X cipher.

Index Terms: Decryption, Encryption, Random-X cipher, Substitution cipher

1 INTRODUCTION
There are various types of authentication processes being used for system authentication commonly known as something you know, are or has. The most common among them is some secret which nobody else knows. This is called a password. The system must store the password in a table or file to authenticate it later. Storing the password in plaintext is a bad idea because tables or files are vulnerable to theft. If someone gets the access to this table or file, he/she can find the password. This increases the necessity of Password Encryption. If the password is stored encrypted then the system will encrypt this password again and then try to match it with existing encrypted value which will never match. The currently available ciphers and encryption methods do not provide high security. Random-X password encryption method discussed in this paper, is difficult to decrypt or crack. In this paper, briefly explain encryption and substitution ciphers, before detailing Random-X cipher, and its encryption method for implementation, testing, and conclusions.

2 ENCRYPTION
Encryption is an interesting piece of technology that works by scrambling data so it is unreadable by unintended parties [2]. The popular ciphers algorithm are RSA (Asymmetric Key Encryption / Public Key Encryption), AES (Symmetric Key Encryption) etc.

2.1 RSA
RSA is a public key encryption algorithm. In this a public key is used to encrypt and a private key is given to decrypt it. Unlike Triple DES, RSA is considered an asymmetric algorithm because of its pair of keys for encryption and decryption. It takes more time and processing power because the result of RSA encryption is a huge mumbo jumbo [2]. This is widely used for Key exchange and Digital signatures.

2.2 AES
AES (Advanced Encryption Standard) is a symmetric encryption algorithm. It supports both hardware and software implementation [4]. AES is a three block ciphers. Each block Encrypts and decrypts the data in block of 128 bits.

3 SUBSTITUTION CIPHERS
Substitution ciphers is a method of encoding where each characters in the plain text is replaced with a character or a symbol or a string of characters and symbols. Monoalphabetic substitution cipher, ADFGVX cipher, Alberti cipher, Caesar Cipher, Enigma ciphers, Four-square cipher, Freemason cipher, Kamasutra cipher, Larrabee cipher, Pollux cipher, Polybius cipher, takes a letter of an alphabet and substitutes it with another letter or number. This can be easily cracked with frequency analysis [6].

3.1 Random-X Cipher
The Random-X Cipher gives different encrypted values for one Input string. Though it gives multiple encrypted values, it does not use any key to determine this. This increases the time and power for attacker to crack the algorithm. With Random-X method you can increase the strength by increasing substitution options, for instance Random-3, Random-5, and et cetera. This is difficult to crack with Brute force attack. This idea is based on a multiple character substitution. For each valid password characters (ideally A-Z, a-z, 0-9, and special characters - ! @ # $ % _ * etc.) ASCII value must be identified first. We do not need to store this table into database. We can always call system built-in functions to get ASCII value during application development.

TABLE 1
ASCII VALUES FOR COMMON CHARACTERS

<table>
<thead>
<tr>
<th>Char</th>
<th>ASCII</th>
<th>Char</th>
<th>ASCII</th>
<th>Char</th>
<th>ASCII</th>
<th>Char</th>
<th>ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>65</td>
<td>a</td>
<td>97</td>
<td>0</td>
<td>48</td>
<td>1</td>
<td>49</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>b</td>
<td>98</td>
<td>1</td>
<td>50</td>
<td>2</td>
<td>51</td>
</tr>
<tr>
<td>C</td>
<td>67</td>
<td>c</td>
<td>99</td>
<td>2</td>
<td>52</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>d</td>
<td>100</td>
<td>3</td>
<td>54</td>
<td>4</td>
<td>55</td>
</tr>
<tr>
<td>E</td>
<td>69</td>
<td>e</td>
<td>101</td>
<td>4</td>
<td>56</td>
<td>5</td>
<td>57</td>
</tr>
<tr>
<td>F</td>
<td>70</td>
<td>f</td>
<td>102</td>
<td>5</td>
<td>58</td>
<td>6</td>
<td>59</td>
</tr>
<tr>
<td>G</td>
<td>71</td>
<td>g</td>
<td>103</td>
<td>6</td>
<td>60</td>
<td>7</td>
<td>61</td>
</tr>
<tr>
<td>H</td>
<td>72</td>
<td>h</td>
<td>104</td>
<td>7</td>
<td>62</td>
<td>8</td>
<td>63</td>
</tr>
<tr>
<td>I</td>
<td>73</td>
<td>i</td>
<td>105</td>
<td>8</td>
<td>64</td>
<td>9</td>
<td>65</td>
</tr>
<tr>
<td>J</td>
<td>74</td>
<td>j</td>
<td>106</td>
<td>9</td>
<td>66</td>
<td>10</td>
<td>67</td>
</tr>
<tr>
<td>K</td>
<td>75</td>
<td>k</td>
<td>107</td>
<td>10</td>
<td>68</td>
<td>11</td>
<td>69</td>
</tr>
<tr>
<td>L</td>
<td>76</td>
<td>l</td>
<td>108</td>
<td>11</td>
<td>70</td>
<td>12</td>
<td>71</td>
</tr>
<tr>
<td>M</td>
<td>77</td>
<td>m</td>
<td>109</td>
<td>12</td>
<td>72</td>
<td>13</td>
<td>73</td>
</tr>
<tr>
<td>N</td>
<td>78</td>
<td>n</td>
<td>110</td>
<td>13</td>
<td>74</td>
<td>14</td>
<td>75</td>
</tr>
<tr>
<td>O</td>
<td>79</td>
<td>o</td>
<td>111</td>
<td>14</td>
<td>76</td>
<td>15</td>
<td>77</td>
</tr>
</tbody>
</table>

 Falguni Patel is a Project Manager - Enterprise Resource Planning (ERP) at Motibhai Group of Companies Ltd - Fiji Islands and a student of Post Graduate Diploma in Information Technology at UniFiji, Sawenti Campus. Ph: + 679 6668941. Email: falguni.divyesh@yahoo.co.in or falguni.divyesh@gmail.com.

Mohammed Farik (Member, IEEE) is a Lecturer in Information Technology at The University of Fiji. E-mail: mohammedf@unicflj.ac.fj
Now assign 3 unique strings which is 3 characters long to each of the ASCII value identified in above Table I. Below Table II shows an example of assignment of A-Z, a-z and 0-9. The same assignment can be done for all valid special characters as per password policy. Please ensure that these strings are unique. We can use spreadsheet functions to get a set of unique strings.

**TABLE 2**  
**ASCII VALUES WITH 3 SUBSTITUTION (RANDOM-3) OPTIONS**

<table>
<thead>
<tr>
<th>Char</th>
<th>ASCII Value</th>
<th>Random-X 1</th>
<th>Random-X 2</th>
<th>Random-X 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>65</td>
<td>7TF</td>
<td>R0J</td>
<td>VO5</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>2GO</td>
<td>T6E</td>
<td>AF9</td>
</tr>
<tr>
<td>C</td>
<td>67</td>
<td>6WM</td>
<td>G2Z</td>
<td>RI3</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>3NH</td>
<td>C0N</td>
<td>TJ0</td>
</tr>
<tr>
<td>E</td>
<td>69</td>
<td>3VU</td>
<td>Z3X</td>
<td>BX8</td>
</tr>
<tr>
<td>F</td>
<td>70</td>
<td>8YC</td>
<td>G1P</td>
<td>TX9</td>
</tr>
<tr>
<td>G</td>
<td>71</td>
<td>1AI</td>
<td>F8O</td>
<td>GM5</td>
</tr>
<tr>
<td>H</td>
<td>72</td>
<td>1ZG</td>
<td>H9R</td>
<td>NS3</td>
</tr>
<tr>
<td>I</td>
<td>73</td>
<td>4CW</td>
<td>P0J</td>
<td>RY5</td>
</tr>
<tr>
<td>J</td>
<td>74</td>
<td>7MO</td>
<td>U6R</td>
<td>YR7</td>
</tr>
<tr>
<td>K</td>
<td>75</td>
<td>5QD</td>
<td>V4X</td>
<td>OX7</td>
</tr>
<tr>
<td>L</td>
<td>76</td>
<td>7XN</td>
<td>D7A</td>
<td>QC7</td>
</tr>
<tr>
<td>M</td>
<td>77</td>
<td>1EW</td>
<td>Y1P</td>
<td>TU8</td>
</tr>
<tr>
<td>N</td>
<td>78</td>
<td>9TK</td>
<td>E1I</td>
<td>JW6</td>
</tr>
<tr>
<td>O</td>
<td>79</td>
<td>5XQ</td>
<td>H0U</td>
<td>AO8</td>
</tr>
<tr>
<td>P</td>
<td>80</td>
<td>1NA</td>
<td>O0G</td>
<td>ZP1</td>
</tr>
<tr>
<td>Q</td>
<td>81</td>
<td>0VK</td>
<td>Z9O</td>
<td>UI3</td>
</tr>
<tr>
<td>R</td>
<td>82</td>
<td>4EK</td>
<td>X7D</td>
<td>ZY4</td>
</tr>
<tr>
<td>S</td>
<td>83</td>
<td>0ZV</td>
<td>Q6R</td>
<td>L0O</td>
</tr>
<tr>
<td>T</td>
<td>84</td>
<td>8BH</td>
<td>Z3K</td>
<td>ZF4</td>
</tr>
<tr>
<td>U</td>
<td>85</td>
<td>7WW</td>
<td>V7I</td>
<td>MD6</td>
</tr>
<tr>
<td>V</td>
<td>86</td>
<td>7XD</td>
<td>K1L</td>
<td>BM0</td>
</tr>
<tr>
<td>W</td>
<td>87</td>
<td>7RQ</td>
<td>O6D</td>
<td>OW6</td>
</tr>
<tr>
<td>X</td>
<td>88</td>
<td>1IS</td>
<td>D6O</td>
<td>TY8</td>
</tr>
<tr>
<td>Y</td>
<td>89</td>
<td>9YO</td>
<td>R4V</td>
<td>OQ1</td>
</tr>
<tr>
<td>Z</td>
<td>90</td>
<td>8MK</td>
<td>S4X</td>
<td>PP5</td>
</tr>
<tr>
<td>A</td>
<td>97</td>
<td>3MR</td>
<td>B3Y</td>
<td>VU3</td>
</tr>
<tr>
<td>B</td>
<td>98</td>
<td>2QF</td>
<td>T4V</td>
<td>EN8</td>
</tr>
<tr>
<td>C</td>
<td>99</td>
<td>9PM</td>
<td>X9P</td>
<td>H8C</td>
</tr>
<tr>
<td>D</td>
<td>100</td>
<td>0WG</td>
<td>U0L</td>
<td>IR9</td>
</tr>
<tr>
<td>E</td>
<td>101</td>
<td>0MN</td>
<td>O4U</td>
<td>WL7</td>
</tr>
<tr>
<td>F</td>
<td>102</td>
<td>9UQ</td>
<td>D9D</td>
<td>UQ2</td>
</tr>
<tr>
<td>G</td>
<td>103</td>
<td>8PD</td>
<td>O2U</td>
<td>BU5</td>
</tr>
<tr>
<td>H</td>
<td>104</td>
<td>3DR</td>
<td>H2D</td>
<td>UT7</td>
</tr>
<tr>
<td>I</td>
<td>105</td>
<td>9EA</td>
<td>Y4H</td>
<td>UK7</td>
</tr>
<tr>
<td>J</td>
<td>106</td>
<td>6RB</td>
<td>D1M</td>
<td>HA7</td>
</tr>
<tr>
<td>K</td>
<td>107</td>
<td>9ST</td>
<td>I3X</td>
<td>KE0</td>
</tr>
<tr>
<td>L</td>
<td>108</td>
<td>9QA</td>
<td>O3K</td>
<td>KO9</td>
</tr>
<tr>
<td>M</td>
<td>109</td>
<td>8EH</td>
<td>H5Z</td>
<td>LY1</td>
</tr>
<tr>
<td>N</td>
<td>110</td>
<td>7RT</td>
<td>A1G</td>
<td>XW9</td>
</tr>
<tr>
<td>O</td>
<td>111</td>
<td>6DC</td>
<td>F0H</td>
<td>VF2</td>
</tr>
<tr>
<td>P</td>
<td>112</td>
<td>5WA</td>
<td>O6E</td>
<td>DM7</td>
</tr>
</tbody>
</table>

Below is the example of how the Password ‘Hello’ is encrypted using this algorithm. For each character of a string, get the ASCII Value (from Table 1), pick a random number between 1 and 3, read the value (from Table 2) and concatenate this. For example, the password Hello is encrypted as:

```
H Random No. 2 = H9R
e Random No. 1 = 0MN
l Random No. 1 = 0MN
m Random No. 3 = KO9
o Random No. 3 = VF2
```

So, the encrypted string is “H9R0MN9QAKO9VF2”. When the same password is encrypted again the string can be as below:

```
H Random No. 1 = 1ZG
e Random No. 2 = 0Q4
l Random No. 3 = KO9
o Random No. 1 = 6DC
```

So, the encrypted string is “1ZG4UKO9O3K6DC”.

This way password “Hello” can have 5 * 3 = 15 different encrypted values. If we use 5 substitution options, the password “Hello” will have 5 * 5 = 25 different encrypted values. Every time when the password is entered, this encryption algorithm will convert it into an encrypted string. The encrypted string will be stored in a database table or file for application validation purpose. The Decryption process is the reversal process of above Encryption process. Fig.1 shows the password authentication process. According to this process the password authentication is done by comparing the decrypted value of the encrypted password.

![Fig. 1. Password Authentication Process](image-url)
4 IMPLEMENTATION OF ENCRYPTION METHOD
To implement this method following four primary functions are required.

4.1 f_encryptChar
The sample function is as shown in Fig.2.

```cpp
using namespace std; // So the program can use cout and endl.
int _tmain(int argc, char** argv)
{
    string sEncrypted_PWD;
    //Looping through the characters of a std: string with an old-fashioned for-loop
    for (std::string::size_type i = 0; i < sPWDEntered.size(); i++)
    {
        //New Character
        char ls_char = sPWDEntered[i];
        int iDec_ascii = int(ls_char);
        string sEncrypted_char = f_encryptChar(lsASCII_Char);
        iEncrypted_PWD.append(sEncrypted_char);
    }
    return 0;
}
```

**Fig. 2. f_encryptChar function**

Arguments: integer (ASCII Value)  
Return: String (Encrypted String)  
Purpose: This function converts the ASCII value of a character to an encrypted string.

4.2 f_encryptPWD
Argument: String (Password Entered)  
Return: String (Encrypted Password)  
Purpose: This function converts the Entered Password string to an Encrypted string and returns Encrypted Password. This function calls f_encryptChar() function while looping through each character of the entered password. The sample figure is as shown in Fig.3.

```cpp
int f_decryptChar(string as.EncryptedString)
{
    int li_Dec_ascii;
    if (as.EncryptedString == "777")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //A
    }
    else if (as.EncryptedString == "123")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //B
    }
    else if (as.EncryptedString == "890")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //C
    }
    else if (as.EncryptedString == "765")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //D
    }
    else if (as.EncryptedString == "987")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //E
    }
    else if (as.EncryptedString == "543")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //F
    }
    else if (as.EncryptedString == "069")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //G
    }
    else if (as.EncryptedString == "321")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //H
    }
    else if (as.EncryptedString == "210")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //I
    }
    else
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //J
    }
}
```

**Fig. 3. f_encryptPWD function**

4.3 f_decryptChar
Arguments: string (Encrypted String parse (3chars))  
Return: Integer (ASCII Value)  
Purpose: This function converts the String parse (3characters) into ASCII Value. The sample function is as shown in Fig. 4.

```cpp
int f_decryptChar(string as_EncryptedString)
{
    int li_Dec_ascii;
    if (as_EncryptedString == "777")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //A
    }
    else if (as_EncryptedString == "123")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //B
    }
    else if (as_EncryptedString == "890")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //C
    }
    else if (as_EncryptedString == "765")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //D
    }
    else if (as_EncryptedString == "987")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //E
    }
    else if (as_EncryptedString == "543")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //F
    }
    else if (as_EncryptedString == "069")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //G
    }
    else if (as_EncryptedString == "321")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //H
    }
    else if (as_EncryptedString == "210")
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //I
    }
    else
    {
        li_Dec_ascii = 65; return li_Dec_ascii; //J
    }
}
```

**Fig. 4. f_decryptChar function**

4.4 f_decryptPWD
Argument: String (Encrypted String)  
Return: String (Original Password)  
Purpose: This function converts the encrypted password into original password. This function loops through each characters of encrypted string, group into a string of 3 characters and calls f_decryptChar function to decrypt the password. The sample is as shown in Fig. 5.

```cpp
int f_decryptPWD(string as_Encrypted_PWD)
{
    string ls_Original_PWD;
    //string length
    if (as_Encrypted_PWD.length() % 3 != 0)
    {
        cout<< "The Encrypted Password is not valid.";
        return 0;
    }
    for (int i = 0; i < as_Encrypted_PWD.length(); i++)
    {
        //new Character
        char li_char = as_Encrypted_PWD[i];
        int li_ascii_char = int(li_char);
        string ls_Encrypted_char = f_decryptChar(liASCII_Char);
        ls_Original_PWD.append(ls_Encrypted_char);
    }
    return ls_Original_PWD;
}
```

**Fig. 4. f_decryptPWD function**

5 TESTING
The C++ program is created to test this algorithm which encrypts and decrypts the password. This program has options to select either Encrypt or Decrypt utility. Sample screen snapshot are as Fig. 5 and Fig. 6.
The method suggested here provides strong password protection. As shown in the Test Result Table 3, for same password, the encrypted password is most of the time different. This provides better security. The security can be enhanced by:

- Having 5 character substitution options – Table 2 shows 3 substitution options for each valid character. We can modify this and create 5 options to substitute. With this change, 8 character password will have $8 \times 5 = 40$ different Encrypted string instead of $8 \times 3 = 24$.
- Another option to increase the strength is by increasing the length of substitute string from 3 characters to 4 characters. i.e. $H = H9R$ can be changed to MIUT or Li0@ etc.

Table III shows the test results for encryption and decryption algorithm on passwords.

### Table 3

<table>
<thead>
<tr>
<th>No</th>
<th>Plaintext</th>
<th>Encrypted Password</th>
<th>Decrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HelloWorld</td>
<td>1ZG0MNKO99QAOFOH06DVF20VXO3KIR9</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>2</td>
<td>HelloWorld</td>
<td>1ZGQ4UKO99QA6DCO6DF20VXO3KU0L</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>3</td>
<td>HelloWorld</td>
<td>H9RWL7KO99QAF0HOW66DCO6VXO3KIR9</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>4</td>
<td>HelloWorld</td>
<td>H9RWL7KO99QAF27RQ6DCP1XKO90WG</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>5</td>
<td>HelloWorld</td>
<td>NS3WL7O3KKO66DCO6DF0HCFQ9QAU0L</td>
<td>HelloWorld</td>
</tr>
<tr>
<td>6</td>
<td>Password</td>
<td>1NA3MRQ9TQV96SM5FP1XIR9</td>
<td>Password</td>
</tr>
<tr>
<td>7</td>
<td>Password</td>
<td>1NA3MRQ9TQV96SM5FP1XIR9</td>
<td>Password</td>
</tr>
<tr>
<td>8</td>
<td>Password</td>
<td>NVU3OQ97V9M5U270VXO3KIR9</td>
<td>Password</td>
</tr>
<tr>
<td>9</td>
<td>Password</td>
<td>NVU3OQ97V9M5U270VXO3KIR9</td>
<td>Password</td>
</tr>
<tr>
<td>10</td>
<td>Password</td>
<td>O0GVU37IM9T5ZU70VXU0L</td>
<td>Password</td>
</tr>
<tr>
<td>11</td>
<td>RedFlower2</td>
<td>4EKOMNU0LTX903KF0HU740MN1XL70VXO3KIR9</td>
<td>RedFlower2</td>
</tr>
<tr>
<td>12</td>
<td>RedFlower2</td>
<td>4EK04UIR98YCO3KF0H6SMQ4UCF07UG</td>
<td>RedFlower2</td>
</tr>
<tr>
<td>13</td>
<td>RedFlower2</td>
<td>ZY40MNULG1PK96DCZ5BQ4UCF07UG</td>
<td>RedFlower2</td>
</tr>
<tr>
<td>14</td>
<td>RedFlower2</td>
<td>X7DWL7IF98YCO3KF0H6SMQ4UCF07UG</td>
<td>RedFlower2</td>
</tr>
<tr>
<td>15</td>
<td>RedFlower2</td>
<td>ZY40MNULG1PK96DCZ5BQ4UCF07UG</td>
<td>RedFlower2</td>
</tr>
</tbody>
</table>

5 Conclusion

The cipher method used in this paper provides an alternative and stronger substitution algorithm for password or message security by encryption and decryption.

### References


