SOLITAIRE ENCRYPTION WITH DECKID AND DYNAMIC DECK GENERATION

ABOUT SOLITAIRE

The **Solitaire** cryptographic algorithm was designed by Bruce Schneier to allow field agents to communicate securely without having to rely on electronics or having to carry incriminating tools, at the request of Neal Stephenson for use in his novel *Cryptonomicon*. It was designed to be a manual cryptosystem calculated with an ordinary deck of playing cards. In *Cryptonomicon*, this algorithm was originally called **Pontifex** to hide the fact that it involved playing cards.

REQUIREMENTS

A deck of Cards with Two Jokers

THEORY

A deck of cards provides us with 54! possible combinations of the arrangement of cards in the deck. Using one such deck combination, we move ahead with the KeyStream Generation, which is the heart of the Solitaire Cipher. When implemented with a PassPhrase then prior to generating KeyStream, the cards have to shuffled using the method described for PassPhrase Shuffle.

Since, Solitaire Cipher, is an output-feedback mode stream cipher, the KeyStream Generator will throw out One Card at the end of every KeyStream generation process, and this process is basically shuffling of cards based on the described method.

The reason for generating KeyStream is that for every character of the Plaintext ,a unique Keystream Character is generated and upon the completion of this entire process, the number associated with the alphabets of both the Plaintext and the KeyStream are added to generate a new number , and subsequently the character associated with this newly generated Number becomes the Solitaire Cipher Text.

Let us take a hypothetical example, wherein, a Random Combination of Deck is used which generates the below mentioned KeyStream.

```
Plain Text       : THISWORKS      : Length : 9 characters
Key Stream       : AJUTRNFHD      : Length : 9 characters
Cipher Text      : URDMOGUSW      : Length : 9 characters
```

In order to generate the cipher text, we shall rely on a table which associates the Alphabets to their corresponding number. Ie. A = 1 , B = 2 ….. Y = 25 , Z = 26

IMPLEMENTATION

From a holistic point of view, the basic requirement for implementing this is that both the processes ie. Encryption and Decryption processes should be aware of the following

1: The Deck Combination Order.
2: Passphrase , (If any)

However, when we have to implement this cipher for (non-sensitive) Digital Communications, the paradigm shifts drastically, as the information between the sender and the receiver can be intercepted and the cipher-message can be brute forced, especially when multiple messages with the same passphrase are generated.

The simple reason being, the deck combination never changes, as changing the deck combination would provide incorrect output after decryption. Moreover, the passphrase also needs to be known to both the parties / programs. It is also stated that after sending every message the pass-phrase needs to be changed so as to raise the level of difficulty for those who are intercepting the messages.

Due to this we now have two parameters, which are deemed crucial for a successful implementation, and have to be known to both the parties.

In order to take advantage of Solitaire Cipher and its 54! Combinations, with the additional PassPhrase and reducing the overheads of sending the entire deck of cards to the recipient, we shall instead rely on associating ID Tags with a subset of deck combinations ie. 6! * 7! * 4!.

This particular method would now be dependent on those who are implementing Solitaire Cipher. Moreover, the implementers may choose to randomize the associating tables in case of a breach.

It is also to be noted that, even though this method of Tag-IDs is not fool-proof however, it does provide an edge over the conventional implementation of Solitaire Cipher.

DECKID GENERATION

As we all know, every deck has 13 cards belonging to 4 different suits. We begin by breaking up the 13 cards into two different sets of 6 cards and 7 cards and associating every card with a particular Alphabet. The implementer over here may choose their own set of cards to be a part of which group.

REFERENCE TABLES :

TABLE 1:

Deck	A	2	3	4	5	6	7	8	9	10	J	Q	K
Reference	A	B	C	D	E	F	G	H	I	J	K	L	M
Index	1	2	3	4	5	6	7	8	9	10	11	12	13

First Set : ABCDEF
Second Set: GHIJKLM

Anagrams, will allow us to get all the unique combinations of the strings and we produce the initial Arrays for both the sets.

The First Set of 6 Alphabets will generate 6! Or 720 unique combinations while the Second Set will generate 7! Or 5040 unique combinations

COMBINATIONS 1

Set 1: "ABCDEF", "ABCDFE", ... , "FEDCAB", "FEDCBA"

Set 2: "GHIJKLM", "GHIJKML", …. , "MLKJHIG", "MLKJIGH", "MLKJIHG"

Moreover the, 4 different suits will give rise to 4! Different combinations and these too can be assigned an Alphabet.

TABLE 2 :

Clubs	Diamonds	Hearts	Spades
C	D	H	S
0	13	26	39

COMBINATIONS 2

"HDSC", "HDCS", …., "CSHD", "CSDH"

Furthermore, this combination can be represented by a Character set of 24 randomly chosen alphabets. Alternatively, one may choose to add 64 to the index number and consider it as ASCII representation.

TABLE 3
This is a reference table which will be used to lookup the reference values of numbers between 01 and 99

	A	B	---	Y	a	b	---	x	Y
0	00	01	---	24	25	26	---	48	49
1	50	51	---	74	75	76	---	98	99

In case we have to lookup the no. 24 then its reference cell no. is “Y0” similarly, for 76 the reference cell is “b1”

TABLE 4
This is a reference table which will be used to lookup the reference values of binary between 0000 and 1111

A	B	C	D	E	F	G	H
0000	0001	0010	0011	0100	0101	0110	0111
I	J	K	L	M	N	O	P
1000	1001	1010	1011	1100	1101	1110	1111

In case we have to lookup the “0110” then its corresponding value is “G”.

INITIALIZE

We need 3 random numbers to choose the Deck

STEP 1

Deck Suit - random number between 1 – 24 (Inclusive). This becomes the index of the Suit Combinations which will denote the order in which Suits will be arranged. You may choose randomize the order in which you place the Suits.

This shall also become your SuitID.

Eg: Random Number : 23 ,

Associated Suit : CSHD ,

ASCII Representation : 23+64 = 87 or “W”

SUITID “W”

STEP 2

Suit Cards -

Set 1 : random number between 1-720 (inclusive) and padding the results with 0 in case the random number generator churns out a single or a double digit number. Convert these numbers into String Format with padding as mentioned below.

Eg: 001 , 045 or 199

Set 2: random number between 1-5040 (inclusive) and padding the results with 0 in case the random number generator churns out a single or a double or a triple digit number.

Eg: 0001 ,0045 , 0199 or 3256

Looking up against the Array of individual Sets, we will get the corresponding Alpha Strings.

Eg: 199 , may point to “BECADF” and 3256 accordingly may point to "KJGLIMH"

Lookup the corresponding Indexes of the Alphabets from **Table 1**.

TABLE 4:

```
B = 2, E = 5, C = 3, A = 1, D = 4, F = 6, K = 11, J = 10,
G = 7, L = 12, I = 9, M = 13, H = 8
```

These are the values form the Suit of Cards and the sequence in which they are placed. Concatenating Set1 + Set2 will give us a 7 character string (numbers converted to string). From this string, we shall generate the DeckID. Eg: 1993256 or 0010119

GENERATING DECKID

STEP 1: SUM OF DIGITS. ADD UP ALL THE DIGITS TO GENERATE ONE SINGLE DIGIT NO.

Eg: 1993256 = 1 + 9 + 9 + 3 + 2 + 5 + 6 = 35 = 3 + 5 = 8 (its Even)

Note: In case the Sum of all the digits turn out to be an odd number then subtract 1 from it.

Eg: 2993256 = 2 + 9 + 9 + 3 + 2 + 5 + 6 = 36 = 3 + 6 = 9 (Its odd) 9 – 1 = 8

The Single digit is to be appended to the Original String, hence the number now becomes

Eg: 19932568

STEP 2: GROUPS. CREATE GROUPS OF 2 DIGITS

Eg: 19,93,25,68

Using the reference "**Table 3**" we find out the corresponding cell nos.

19	93	25	68
T0	s1	a0	S1

STEP 3: SEGREGATE
The Cell Reference nos. are separated out as follows:

TsaS 0101

The only possible combinations for 4 digit 0's and 1's is the Binary Table representing nos. from 0 to 15 . We shall use the reference "**Table 4**"

0101 = F

STEP 4: The DeckID

From Previous Step we have "**TsaSF**" and from Initialize Section's Step 1 we have already generated the SuitID ie. "**W**"

DECKID = TsaSFW

GENERATING INITIAL DECK

The basic requirement of Solitaire is a Deck. As of this moment we do know
1: DeckID – this will appended to the Encrypted Text and will always be a 6 character string.
2: The Sequence of Cards of a Suit
3: The Sequence of Suits

We reduce the placement information of 13 individual cards into a 7 Digit Number, 24 Suit combinations into one single character and the DeckID would ultimately turn out to be of "**SIX Characters" which represent a Deck of 52 Cards with 2 Jokers.**

We will have 6! * 7! * 4! Different Combinations, being represented.

STEP 1:
Revisit Step 1 of Initialize Section and get the Associated Suit

Use the Table 4 of Step 2 belonging to Initialize Section

```
B = 2, E = 5, C = 3, A = 1, D = 4, F = 6, K = 11, J = 10,
G = 7, L = 12, I = 9, M = 13, H = 8
```

There are two ways of arranging the Suit Cards . Also note that the weight of the Suit cards is to be applied from **"Table 2"**

1: C:2,5,3,1,4,6,11,10,7,12,9,13,8;S:….;H:….;D:…..

2: C1: 2,S1:5,H1:3,D1:1 …..D13:8

I have preferred the second method

Seq	1	2	3	4	5	6	7	8	9	0	11	12	13
C+0	2	4	7	8									
S+39	5	6	12										
H+26	3	11	9										
D+13	1	10	13										

Upon Completion the Deck would look like
2,44,29,14,4,45,37,23,7,51,35,26,8……..

STEP 2: INSERTING JOKERS
In the Initialize section from Step 2 we generated 2 nos. viz. "199" and "3256" . We again sum up the digits which will denote the position of Jokers in the Deck.

Joker A Position : 199 = 1+9+9 = 19
Joker B Position : 3256 = 3+2+5+6 = 16

Joker A will be inserted from the 0th position in the array while Joker B will be inserted from the 26th position in the array.

Note : The values Joker B may range from 1-31 hence, and the no. of available position is just 26 we shall check whether the value is <= 22 . In case it is, then we simply add 25 to it and insert it in that position, otherwise we add 4 to it and then insert the JokerB in that position.

Eg1: JokerB = 21 , which is <= 22 . the Joker B Position would be 25+21 ie. 46.
Eg2: JokerB = 31 , which is > 22 . the Joker B Position would be 31+4 ie. 35.
Eg3: JokerB = 23 , which is > 22 . the Joker B Position would be 23+4 ie. 27.

Now you have an Initial Deck with Jokers and this Deck can now be used to carry on with the Solitaire Encryption.

Random Deck No	4581337
Random Suit No	7
Before Adding Jokers	MXjsPG,17,32,40,2,18,29,47,13,20,37,51,9,23,30,45,1,15,31,42,8,26,33,50,12,22,36,43,6,14,28,44,3,21,39,46,11,25,35,49,4,19,27,41,5,16,34,52,7,24,38,48,10
Joker A Sum	Number Left : 458 ; Sum:17
Joker B Sum	Number Right: 1337; Sum:14
Joker B Position	39
After Adding Jokers	MXjsPG,17,32,40,2,18,29,47,13,20,37,51,9,23,30,45,1,53,15,31,42,8,26,33,50,12,22,36,43,6,14,28,44,3,21,39,46,11,25,54,35,49,4,19,27,41,5,16,34,52,7,24,38,48,10
Encrypted Text	VIUREZMWODHZRFKWZGRUTXJPHMIKVRGTXECA-MXjsPG

Reversing the DeckID will give you the 7 Digit number , the SuitID , the SuitCards , The Joker Positions and finally the Deck.

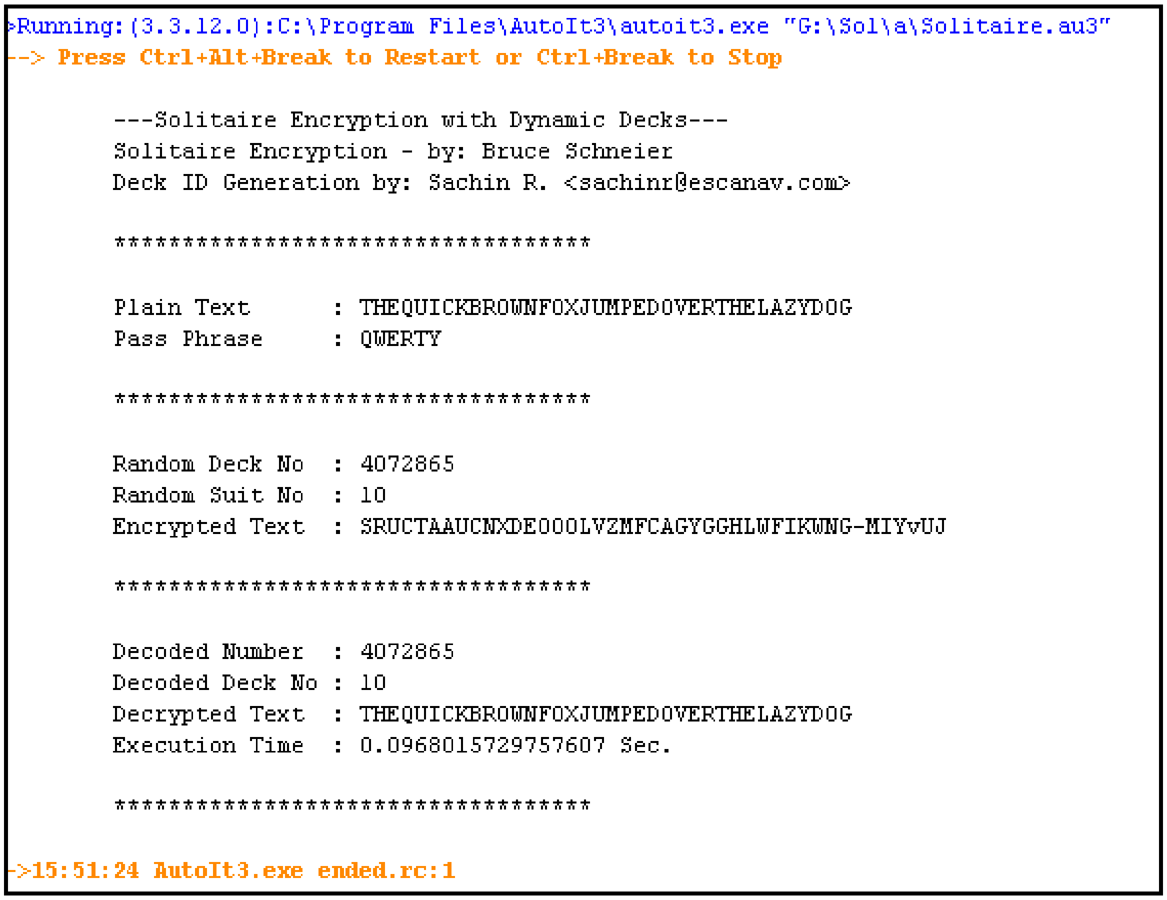
SOME ADDITIONAL THOUGHTS

The DeckID, is not just about the Card and Suit combination, but also defines the position of the Jokers.

The algorithm for DeckID generation is more of a compression algorithm for small strings, whose primary goal is to generate an output string consisting of Printable Characters ie. (Alpha-Numeric). The implementer may choose their favorite algorithm viz, SMAZ or Shoco

Presently, Solitaire with Dynamic Decks has been designed as a proof of concept, wherein the DeckID has been demarcated by a "-" and sticks out like a sore-thumb, however, implementers may choose to mix the DeckID with the Solitaire Encrypted Text.

Note: The values in all the reference tables can be stored in random positions. Hence two different implementations will have two different results.



```
>Running:(3.3.12.0):C:\Program Files\AutoIt3\autoit3.exe "G:\Sol\a\Solitaire.au3"
--> Press Ctrl+Alt+Break to Restart or Ctrl+Break to Stop

        ---Solitaire Encryption with Dynamic Decks---
        Solitaire Encryption - by: Bruce Schneier
        Deck ID Generation by: Sachin R. <sachinr@escanav.com>

        ***********************************

        Plain Text      : THEQUICKBROWNFOXJUMPEDOVERTHELAZYDOG
        Pass Phrase     : QWERTY

        ***********************************

        Random Deck No  : 4072865
        Random Suit No  : 10
        Encrypted Text  : SRUCTAAUCNXDEOOOLVZMFCAGYGGHLWFIKWNG-MIYvUJ

        ***********************************

        Decoded Number  : 4072865
        Decoded Deck No : 10
        Decrypted Text  : THEQUICKBROWNFOXJUMPEDOVERTHELAZYDOG
        Execution Time  : 0.0968015729757607 Sec.

        ***********************************

>15:51:24 AutoIt3.exe ended.rc:1
```

FIG. 1 : SHOWING THE OUTPUT OF THE PROGRAM.