Bruce Schneier

The Solitaire Encryption Algorithm

version 1.2, 5/26/99

Designed by Bruce Schneier
Featured in Neal Stephenson's *Cryptonomicon*

test vectors - Perl - Ada - C (#1) - C (#2) - C++ - C++ GUI - C# - Delphi (#1) - Delphi (#2) - Erlang - Forth (#1) - Forth (#2) - Java - Javascript - K - Palm OS - Pascal - Perl CGI - Python (#1) - Python (#2) - Ruby - TCL
Note: only the Perl implementation has been tested by Counterpane.

This page has been translated into German by Nils Plaumann, into French by Fernandes Gilbert, into Spanish by Jesús Cea Avión, and into Italian by Silvio Coccaro.

Update: Paul Crowley has discovered a bias in Solitaire's random-number generator. His site also has C and Perl implementations of Solitaire.

In Neal Stephenson's novel *Cryptonomicon*, the character Enoch Root describes a cryptosystem code-named "Pontifex" to another character named Randy Waterhouse, and later reveals that the steps of the algorithm are intended to be carried out using a deck of playing cards. These two characters go on to exchange several encrypted messages using this system. The system is called "Solitaire" (in the novel, "Pontifex" is a code name intended to temporarily conceal the fact that it employs a deck of cards) and I designed it to allow field agents to communicate securely without having to rely on electronics or having to carry incriminating tools. An agent might be in a situation where he just does not have access to a computer, or may be prosecuted if he has tools for secret communication. But a deck of cards...what harm is that?

Solitaire gets its security from the inherent randomness in a shuffled deck of cards. By manipulating this deck, a communicant can create a string of "random" letters that he then combines with his message. Of course Solitaire can be simulated on a computer, but it is designed to be implemented by hand.

Solitaire may be low-tech, but its security is intended to be high-tech. I designed Solitaire to be secure even against the most well-funded military adversaries with the biggest computers and the smartest cryptanalysts. Of course, there is no guarantee that someone won't find a clever attack against Solitaire (watch this space for updates), but the algorithm is certainly better than any other pencil-and-paper cipher I've ever seen.

It's not fast, though. It can take an evening to encrypt or decrypt a reasonably long message. In David Kahn's book *Kahn on Codes,* he describes a real pencil-and-paper cipher used by a Soviet spy. Both the Soviet algorithm and Solitaire take about the same amount of time to encrypt a message: most of an evening.

Encrypting with Solitaire

Solitaire is an output-feedback mode stream cipher. Sometimes this is called key-generator (KG in U.S. military speak). The basic idea is that Solitaire generates a stream, often called a ``keystream,'' of numbers between 1 and 26. To encrypt, generate the same number of keystream letters as plaintext letters. Then add them modulo 26 to plaintext letters, one at a time, to create the ciphertext. To decrypt, generate the same keystream and subtract, modulo 26 from the ciphertext to recover the plaintext. (Don't worry, I'll explain "modulo" in a minute.)

For example, to encrypt the first Solitaire message mentioned in Stephenson's novel, "DO NOT USE PC":

1. Split the plaintext message into five-character groups. (There is nothing magical about five-character groups; it's just tradition.) Use X's to fill in the last group. So if the message is "DO NOT USE PC" then the plaintext is:

DONOT USEPC

2. Use Solitaire to generate ten keystream letters. (Details are below.) Assume they are:

KDWUP ONOWT

3. Convert the plaintext message from letters into numbers, A=1, B=2, etc:

4 15 14 15 20 21 19 5 16 3

4. Convert the keystream letters similarly:

11 4 23 21 16 15 14 15 23 20

5. Add the plaintext number stream to the keystream numbers, modulo 26. (All this means is, if the sum is more than 26, subtract 26 from the result.) For example, 1+1=2, 26+1=27, and 27-26=1, so 26+1=1.

15 19 11 10 10 10 7 20 13 23

6. Convert the numbers back to letters.

OSKJJ JGTMW

If you're really good at this, you can learn to add letters in your head, and just add the letters from steps (1) and (2). It just takes practice. It's easy to remember that A+A=B; remembering that T+Q=K is harder.

Decrypting with Solitaire

The basic idea is that the receiver generates the same keystream, and then subtracts the keystream letters from the ciphertext letters.

1. Take the ciphertext message and put it in five-character groups. (It should already be in this form.)

OSKJJ JGTMW

2. Use Solitaire to generate ten keystream letters. If the receiver uses the same key as the sender, the keystream letters will be the same:

KDWUP ONOWT

3. Convert the ciphertext message from letters into numbers:

15 19 11 10 10 10 7 20 13 23

4. Convert the keystream letters similarly:

11 4 23 21 16 15 14 15 23 20

5. Subtract the keystream numbers from the ciphertext numbers, modulo 26. For example, 22-1=21, 1-22=5. (It's easy. If the first number is less than or equal to the second number, add 26 to the first number before subtracting. So 1-22=? becomes 27-22=5.)

4 15 14 15 20 21 19 5 16 3

6. Convert the numbers back to letters.

```
DONOT  USEPC
```

As you can see, decryption is the same as encryption, except that you subtract the keystream from the ciphertext message.

Generating the Keystream Letters

This is the heart of Solitaire. The above descriptions of encryption and decryption work for any output-feedback mode stream cipher. It's the way RC4 works. It's the way OFB mode for DES works. This section is specific to Solitaire, and explains how Solitaire generates those keystream letters.

Solitaire generates its keystream using a deck of cards. You can think of a 54-card deck (remember the two jokers) as a 54-element permutation. There are 54!, or about 2.31 * 10^71, possible different orderings of a deck. Even better, there are 52 cards in a deck (without the jokers), and 26 letters in the alphabet. That kind of coincidence is just too good to pass up.

To be used for Solitaire, a deck needs a full set of 52 cards and two jokers. The jokers must be different in some way. (This is common. The deck I'm looking at as I write this has stars on its jokers: one has a little star and the other has a big star.) Call one joker A and the other B. Generally, there is a graphical element on the jokers that is the same, but different size. Make the "B" joker the one that is "bigger." If it's easier, you can write a big "A" and "B" on the two jokers, but remember that you will have to explain that to the secret police if you ever get caught.

To initialize the deck, take the deck in your hand, face up. Then arrange the cards in the initial configuration that is the key. (I'll talk about the key later, but it's different than the keystream.) Now you're ready to produce a string of keystream letters.

Here's how to produce a single output character. This is Solitaire:

1. Find the A joker. Move it one card down. (That is, swap it with the card beneath it.) If the joker is the bottom card of the deck, move it just below the top card.

2. Find the B joker. Move it two cards down. If the joker is the bottom card of the deck, move it just below the second card. If the joker is one up from the bottom card, move it just below the top card. (Basically, assume the deck is a loop...you get the idea.)

It's important to do these two steps in order. It's tempting to get lazy and just move the jokers as you find them. This is okay, unless they are very close to each other.

So if the deck looks like this before step 1:

```
A 7 2 B 9 4 1
```

at the end of step 2 it should look like:

```
7 A 2 9 4 B 1
```

And if the deck looks like this before step 1:

```
3 A B 8 9 6
```

at the end of step 2 it should look like:

```
3 A 8 B 9 6
```

If you have any doubt, remember to move the A joker before the B joker. And be careful when the jokers are at the bottom of the deck. If the joker is the last card, think of it as the first card before you start counting.

3. Perform a triple cut. That is, swap the cards above the first joker with the cards below the second joker. If the deck used to look like:

```
2 4 6 B 5 8 7 1 A 3 9
```

then after the triple cut operation it will look like:

```
3 9 B 5 8 7 1 A 2 4 6
```

"First" and "second" jokers refer to whatever joker is nearest to, and furthest from, the top of the deck. Ignore the "A" and "B" designations for this step.

Remember that the jokers and the cards between them don't move; the other cards move around them. This is easy to do in your hands. If there are no cards in one of the three sections (either the jokers are adjacent, or one is on top or the bottom), just treat that section as empty and move it anyway. If the deck used to look like:

```
B 5 8 7 1 A 3 9
```

then after the triple cut operation it will look like:

```
3 9 B 5 8 7 1 A
```

A deck that looks like:

```
B 5 8 7 1 A
```

will remain unchanged by this step.

4. Perform a count cut. Look at the bottom card. Convert it into a number from 1 through 53. (Use the bridge order of suits: clubs, diamonds, hearts, and spades. If the card is a club, it is the value shown. If the card is a diamond, it is the value plus 13. If it is a heart, it is the value plus 26. If it is a spade, it is the value plus 39. Either joker is a 53.) Count down from the top card that number. (I generally count 1 through 13 again and again if I have to; it's easier than counting to high numbers sequentially.) Cut after the card that you counted down to, leaving the bottom card on the bottom. If the deck used to look like:

```
7 ... cards .. 4 5
... cards ... 8 9
```

and the ninth card was the 4, the cut would result in:

```
5 ... cards ... 8 7
... cards ... 4 9
```

The reason the last card is left in place is to make the step reversible. This is important for mathematical analysis of its security.

A deck with a joker as the bottom card will remain unchanged by this step.

Be sure not to reverse the order when counting cards off the top. The correct way to count is to pass the cards, one at a time, from one hand to another. Don't make piles on the table.

5. Find the output card. To do this, look at the top card. Convert it into a number from 1 through 53 in the same manner as step 4. Count down that many cards. (Count the top card as number one.) Write the card after the one you counted to on a piece of paper; don't remove it from the deck. (If you hit a joker, don't write anything down and start over again with step 1.) This is the first output card. Note that this step does not modify the state of the deck.

6. Convert the output card to a number. As before, use the bridge suits to order them. From lowest to highest, we have clubs, diamonds, hearts, and spades. Hence, A-clubs through K-clubs is 1 through 13, A-diamonds through K-diamonds is 14 though 26, A-hearts through K-hearts is 1 through 13, and A-spades through K-spades is 14 through 26. (We need 1 through 26, and not 1 through 52, so we can get to letters.)

That's how to use Solitaire to encrypt a single character. You can use it to create as many keystream numbers as you need; just go through the same six steps once for each output character. (Don't rekey the deck). And remember, you'll need one per message character.

I know that there are regional differences in decks of cards, depending on the country. In general, it does not matter what suit ordering you use, or how you convert cards to numbers. What matters is that the sender and the receiver agree on the rules. If you're not consistent you won't be able to communicate.

Keying the Deck

Before you start producing output cards, you have to key the deck. This is probably the most important part of the whole operation, and the one that the entire security of the system hinges upon. Solitaire is only as secure as the key. That is, the easiest way to break Solitaire is to figure out what key the communicants are using. If you don't have a good key, none of the rest of this matters. Here are some suggestions for exchanging a key.

1. Use identically shuffled decks. A random key is the best. One of the communicants can shuffle up a random deck and then create another, identical deck. One goes to the sender and the other to the receiver. Most people are not good shufflers, so shuffle the deck at least six times. And both parties should keep an additional spare deck in the same keyed order, otherwise if you make a mistake you'll never be able to decrypt the message. Also remember that the key is at risk as long as it exists; the secret police could find the deck and copy down its order.

2. Use a bridge ordering. A description of a set of bridge hands that you might see in a newspaper or a bridge book is about a 95-bit key. Agree on a way to take the bridge-hand diagram and convert it into an ordering of the deck. Then agree on a way to put the two jokers into the deck. (One obvious one is to put the A joker after the first card mentioned in the text, and the B joker after the second card mentioned in the text.)

Be warned, though: the secret police can find your bridge column and copy down the order. You can try setting up some repeatable convention for which bridge column to use; for example, "use the bridge column in your hometown newspaper for the day on which you encrypt the message," or something like that. Or use a list of keywords to the *New York Times* website, and use the bridge column for the day of the article that comes up when you search on those words. If the keywords are found or intercepted, they look like a passphrase. And pick your own convention for converting bridge columns into deck orderings; remember that the secret police read Neal Stephenson's books, too.

3. Use a passphrase to order the deck. This method uses the Solitaire algorithm to create an initial deck ordering. Both the sender and receiver share a passphrase. (For example, "SECRET KEY.") Start with the deck in a fixed order; lowest card to highest card, in bridge suits, followed by the A and then the B joker. Perform the Solitaire operation, but instead of Step 5, do another count cut based on the first character of the passphrase (19, in this example). In other words, do step 4 a second time, using 19 as the cut number instead of the last card. Remember to put the top cards just above the bottom card in the deck, as before.

Repeat the five steps of the Solitaire algorithm once for each character of the key. That is, the second time through the Solitaire steps use the second character of the key, the third time through use the third character, etc.

Optional step: (This is NOT used in the examples below.) Use the final two characters to set the positions of the jokers. If the second to last character is a G (a 7), put the A joker after the seventh card. If the last character is a T (a 20), put the B joker after the twentieth card.

Remember, though, that there are only about 1.4 bits of randomness per character in standard English. You're going to want at least an 64-character passphrase to make this secure; I recommend at least 80 characters, just in case. Sorry; you just can't get good security with a shorter key.

Sample Output

Here's some sample data to practice your Solitaire skills with:

Sample 1: Start with an unkeyed deck: A-clubs to K-clubs, A-diamonds to K-diamonds, A-hearts to K-hearts, A-spades to K-spades, A joker, B joker. (You can think of this as 1 ... 52, A, B.)

Here's how to generate the first two outputs. The initial deck is:

1 2 3 4 ... 52 A B

After the first step (moving the A joker):

1 2 3 4 ... 52 B A

After the second step (moving the B joker):

1 B 2 3 4 ... 52 A

After the third step (the triple cut):

B 2 3 4 ... 52 A 1

After the fourth step (the count cut):

2 3 4 ... 52 A B 1

The last card is a 1, which means cut one card. Remember that the 1 stays where it is, so the one card (the B, moves to the bottom of the deck just above the 1.

The fifth step does not change the deck, but produces an output card. The top card is a 2, so count down two cards to the 4. The first Solitaire output is 4. (Of course, you're not supposed to remove this card from the deck. Keep the 4 where it is; just write it down somewhere.)

To produce the second Solitaire output, go through the five steps again.

Step 1:

2 3 4 ... 49 50 51 52 B A 1

Step 2:

2 3 4 ... 49 50 51 52 A 1 B

Step 3:

A 1 B 2 3 4 ... 49 50 51 52

Step 4:

51 A 1 B 2 3 4 ... 49 50 52

The last card is a 52, so count 52 cards down to the 51. Cut the single card, the 51 with the rest of the deck. Remember that the 52 remains unmoved.

Step 5 produces the output card. The first card is a 51. Counting down fifty-one cards gets to the 49, which is the second output card. (Again, don't remove the 49 from the deck.)

The first ten outputs are:

4 49 10 (53) 24 8 51 44 6 4 33

The 53 is skipped, of course. I just put it there for demonstration.

If the plaintext is

```
            AAAAA  AAAAA
```

then the ciphertext is:

```
            EXKYI  ZSGEH
```

Sample 2: Using keying method 3 and the key "FOO," (remember that the optional keying step is not used in these examples) the first fifteen outputs are:

```
            8 19 7 25 20 (53) 9 8 22
            32 43 5 26 17 (53) 38 48
```

If the plaintext is all As, then the ciphertext is:

```
            ITHZU  JIWGR  FARMW
```

Sample 3: Using keying method 3 and the key "CRYPTONOMICON," the message "SOLITAIRE" encrypts to:

```
            KIRAK SFJAN
```

Remember that Xs are used to fill out the last five-character group.

Of course you should use a longer key. These samples are for test purposes only. There are more samples on the website, and you can use the PERL script to create your own.

Real Security, Not Security Through Obscurity

Solitaire is designed to be secure even if the enemy knows how the algorithm works. I have assumed that "Cryptonomicon" will be a best seller, and that copies will be available everywhere. I assume that the NSA and everyone else will study the algorithm and will watch for it. I assume that the only secret is the key.

That's why keeping the key secret is so important. If you have a deck of cards in a safe place, you should assume the enemy will at least entertain the thought that you are using Solitaire. If you have a bridge column in your safe deposit box, you should expect to raise a few eyebrows. If any group is known to be using the algorithm, expect the secret police to maintain a database of bridge columns to use in cracking attempts. Solitaire is strong even if the enemy knows you are using it, and a simple deck of playing cards is still much less incriminating than a software encryption program running on your laptop, but the algorithm is no substitute for street smarts.

Operational Notes

1. The first rule of an output-feedback mode stream cipher, any of them, is that you should never use the same key to encrypt two different messages. Repeat after me: NEVER USE THE SAME KEY TO ENCRYPT TWO DIFFERENT MESSAGES. If you do, you completely break the security of the system. Here's why: if you have two ciphertext streams, A+K and B+K, and you subtract one from the other, you get (A+K)-(B+K) = A+K-B-K = A-B. That's two plaintext streams combined with each other with no key involved, and is very easy to break. Trust me on this one: you might not be able to recover A and B from A-B, but a professional cryptanalyst can. This is vitally important: never use the same key to encrypt two different messages.

2. Keep your messages short. This algorithm is designed to be used with small messages: a couple of thousand characters at most. Use shorthand, abbreviations, and slang in your messages. Don't be chatty. If you have to encrypt a 100,000-word novel, use a computer algorithm.

3. Like all output-feedback stream ciphers, this system has the unfortunate feature of never recovering from a mistake. If you're encrypting a message, and you make a mistake in one of the operations, every letter afterwards will be encrypted wrong. You won't be able to decrypt it, even with the key. And you'll never know. So if you're encrypting a message, go through the encryption process twice to make sure they agree. If you're decrypting, check to make sure the message makes sense as you decrypt it. And if you're keying from a random deck, keep a spare copy of the

ordered deck for this reason.

4. Solitaire is reversible. This means that if you leave the deck lying around after you've encrypted your message, the secret police can find it and work the algorithm backwards using the deck. This process can recover all of the output cards and decrypt a message. It is important that you shuffle the deck completely, six times, after you finish encrypting a message.

5. For maximum security, try to do everything in your hands and head. If the secret police starts breaking down your door, just calmly shuffle the deck. (Don't throw it up in the air; you'd be surprised how much of the deck ordering is maintained during a game of 52-Pickup.) Remember to shuffle the backup deck, if you have one.

6. Be careful about worksheets, if you have to write things down. They will have sensitive information on them.

Burning is probably the best method of data destruction available, but think about the paper. Ungummed, rice cigarette papers seem ideally suited to this role. A colleague did some tests with Club Cabaret Width papers, and they burn completely.

It's not as difficult to write on cigarette papers as you might think. Using a No. 2 pencil with a fine but blunt tip works well. A No. 3 pencil works better, but it is a bit weirder to be carrying. Pens have a number of problems. First the extremely hard tip of the pen is more likely to leave impressions in the surface below the paper. Also, anything with ink has the possibility to bleed through to the surface below.

And good cigarette papers are made to burn cleanly and completely. The Club papers burned best when allowed to burn in the free air. That is, lit and released at about chest level. These papers also have the advantage of having very low volume and could be easily eaten if required.

They are also extremely thin. These three-inch square papers can be folded six times into a 1 cm square which is about 1 mm thick. One paper can comfortably hold 80 characters in 8 rows of two five letter blocks each. It seems quite possible a reasonably careful writer could fit 120 characters.

7. Solitaire can work on computers. Often only one end of the communications has to use the deck of cards; the other is safe enough to use a computer. Use a computer when you can: it's faster and the computer never makes mistakes.

8. Most card games do not include jokers, so carrying a deck around with them may be suspicious. Be prepared with a story.

9. The security of Solitaire does not depend on the secrecy of the method. I assume that the secret police know that you're using it.

Security Analysis

Properties of the Transformation Semigroup of the Solitaire Stream Cipher (B. Pogorelov and M. Pudovkina)
Problems with Bruce Schneier's "Solitaire" (Paul Crowley)

Learning More

I recommend my own book, *Applied Cryptography* (John Wiley & Sons, 1996), as a good place to start. Then read *The Codebreakers*, by David Kahn (Scribner, 1996). After that, there are several books on computer cryptography, and a few others on manual cryptography. It's a fun field; good luck.

test vectors - Perl - Ada - C (#1) - C (#2) - C++ - C++ GUI - C# - Delphi (#1) - Delphi (#2) - Erlang - Forth (#1) - Forth (#2) - Java - Javascript - K - Palm OS - Pascal - Perl CGI - Python (#1) - Python (#2) - Ruby - TCL
Note: only the Perl implementation has been tested by Counterpane.