Fidget: A Secret Decoder Ring Protocl for Alice and Cory

Chris Smith, smith@vex.net
2008 September 30th

Alice and Cory chose to have identical wedding rings created in the form of secret decoder rings. Bruce was requested to provide basic cryptographic design for the rotating bands. Once the rings were constructed, Cory and Bruce announced a contest to create cryptographic protocols using the rings. This paper is an entry in that contest. It explores several factors in the creation of an appropriate protocol, selects a design, outlines the testing that was done on this protocol, and provides additional support materials for those wishing to further explore either this or other protocols for the rings.

Protocol Name

The name Fidget was chosen to reflect several aspects of the rings and their use as secret decoder rings. The rings themselves, with three bands of letters, one fixed and two moving, are a style of ring known as a spinner or fidget ring. Wearers of these rings are often seen idly rolling the moving bands around their rings – fidgeting.

Given that the rings are "secret decoder rings", it is within the spirit of the rings that their use may be non-obvious, such that while the wearer is executing a cryptographic protocol, they may appear to only be fidgeting with their ring. If their use is noticed, the wearer can quickly destroy the state of the rings by fidgeting with them in a random fashion.

Design Parameters

The wearer of one of these rings will always have the ring itself, but may not have any other apparatus. They may be missing paper, pencil, computer, calculator, or even additional common items such as coins. Critically, they may also be missing printed instructions. It was decided that the protocol must be simple enough that it can be readily memorized, and in operation it must not require additional apparatus. Having a pencil and paper might make use of the ring easier, but it should not be a necessity if the wearer has had at least modest experience with the protocol.

After some experimenting, it was further decided to keep the protocol sufficiently simply that it could be easily remembered even when under stress or if infrequently used. Although this may seen to be an overly severe restriction, it allows for use of the rings under a wider variety of conditions.

To assist the wearer, a simple key, value, and result can be memorized. If the wearer is in doubt about the correctness of the protocol, then they can encode the known value under the simple key to see if they get the expected result.

Manipulation of the Ring

First and foremost, it was recognized that the rings are already finished. It is usually the case when creating cryptographic devices that the protcol designer has some influence over the design of the hardware components if they are relevant to the overall use of the system. This is not the case here – the rings are finished, and will not be modified. The protocol must maximize its value within that context. With this point in mind, the first step was to create a model ring. Papercraft was chosen as a medium for its ease of design and construction, along with the sheer ubiquity of printers that effectively serve as manufacturing devices. Once a paper ring, with the appropriate bands, was created, it was worn and played with for several days. It was quickly determined that although many manipulations of the ring and the bands are possible, relatively few of them are easily executed. Some manipulations which are simple in theory are almost impossible to execute accurately using a real ring.

The ring was constructed with a leftmost static band, and two moving bands in the middle and on the right. Throughout this paper, the rings are referred to as “ring 1”, “ring 2”, and “ring 3”, referring to the left, middle, and right rings, respectively.

The final primitive manipulations allowed in the protocol are as follows:

1) Ring Roll – the entire ring is rolled either up or down one position. The band do not move relative to the ring, and thus all three bands appear to move either up or down one position.
2) Ring 3 Roll – ring 3 is rolled either up or down one position. Neither the ring itself, seen as ring 1, nor ring 2, are moved using this manipulation.
3) Ring 2 and 3 Roll – ring 2 and ring 3 together are rolled either up or down one position. The ring itself, seen as ring 1, is not moved using this manipulation.

The following manipulations were rejected, each for a specific reason:

1) Roll ring 1 – because ring 1 is also the carrier for the other two rings, it cannot easily be moved independently. Any attempt to move ring 1 will also move ring 2 and ring 3.
2) Roll ring 2 – because ring 2 is between ring 1 and ring 3, it is difficult to move only ring 2 without also moving ring 3. Any attempt to use this manipulation is likely to cause inadvertant movement of ring 3, disrupting the state of the ring.
3) Roll ring 1 and 2 – although these two rings are side by side and can be easily turned, ring 3 will be moved as well because ring 1 is the carrier for ring 1.

Using the dots as an indication of how to move the rings requires careful analysis of the final protocol. Because the dots are not independent of the letters, corelations can begin to appear almost immediately between certain initial conditions and subsequent states of the rings. Furthermore, as noted by Bruce, the overall entropy of the system is very low. This would likely mean that an attacker could attempt brute force attacks on the system with less effort than it takes to analyse the system. Although there is a possibility that there are valuable solutions involving use of the dots in this way, the decision was made to close off this avenue of exploration, and focus on other possibilities.

Comparing dots to other dots showed more promise, although again the corelation between letters and dots threatened to reduce the output space dramatically. After several abortive attempts, it was realized that it was possible to look at more than one dot at a time – to look at “dot patterns” instead of “dots”.

Before this was explored in detail, a check of all possible ring states and their dot positions was done. This showed a key factor regarding dot patterns on the rings. When Ring 2 and Ring 3 were examined as a pair, it was found that any relative position of Ring 2 and Ring 3 – that is, matching any possible pair of letters – always generated every possible combination of dots. With three dot states possible on a letter, two letters should generate nine dot states. For any relative position of Ring 2 and Ring 3, there were always nine dot states to be found around the pair of rings. This was not the case for Ring1 and Ring 3. Here, certain relative positions resulted in missing dot-pair states. It was realized that attempting to find an arbitrary dot-pair state on the Ring 1 / Ring 2 pair would sometimes be impossible.

Based on all of the above factors, a combined primitive was designed. This is known as a Dot Match Roll. It is executed as follows:

1) The dot pattern on Ring 1 and Ring 2 is noted. This will be the comparison dot pair.
2) A direction (up towards A or down towards Z) is chosen.
3) The Ring Roll manipulation is done once in the chosen direction.
4) The comparison dot pair is looked for on Ring 2 and Ring 3.
5) If the comparison dot pair is not found, repeat from Step 3.

The Dot Match Roll will always adjust the ring at least one step. It will also always complete within one complete rotation (26 steps) of the ring. Any given initial ring state will always generate the same final ring state, although the up and down directions may give different final states. However, Dot Match Roll should not be repeated without a change in the relative position of Ring 2 or Ring 3. Repeated use of Dot Match Roll will eventually lead to a short cycle of patterns being repeated. Figure 1 shows the directed graph for the AAA ring states when repeatedly applying Dot Match Roll.

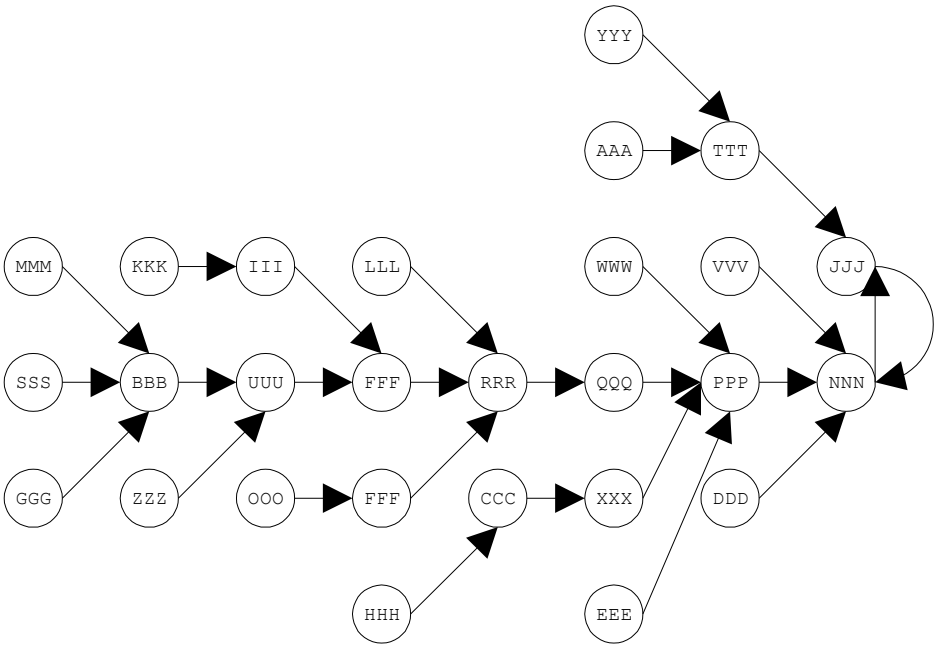

Illustration 1: Directed Graph of Ring AAA Dot Match Roll Up States

Note that repeated use of Dot Match Roll forces the AAA ring to a cycle of JJJ and NNN. Clearly the ring state must be altered by movement of rings 2 or 3 before re-use of Dot Match Roll. A detailed

examination of all possible ring states (all 26^3 states) is left for further analysis.

It was decided to implement only a single function, that of Encode/Decode. This reduces the protocol information that must be memorized by the wearer. Although a hash function would also be valuable, a hash function can also be constructed using the Encode function.
Initially, the idea of only using three character keys was explored, allowing for key entry with nothing more than setting the initial ring state.. Simple simulations showed that it was very difficult to get useful results. It was then decided to explore the idea of a mixing function which brings a new key character into the ring state on each step. This appeared much more promising, so this line of development was followed. This would allow keys of arbitrary length.

Finally, a protcol that allowed for both encode and decode to be possible required that the encode or decode be a final step after the mixing function. If the plaintext character was mixed in too early, it was possible that the limited number of ring states might result in a single final value from more than one plaintext character. This would make decoding extremely difficult.

Combining all of these components and constraints leads to the Fidget protocol.

The Fidget Protocol

Key Schedule

The key consists of a character string of arbitrary length. Although a one character key is possible, keys of several characters will be just as easy to use. Each time a key character is needed, the next key character is taken. When the end of the key string is reached, the wearer will simply revert to the beginning of the key string and continue from there.

Ring Setup

The ring is setup with the first three letters from the key schedule. These key characters are then considered consumed.

Key Mixing Function

The following mixing function blends the next key character into the current ring state. Although this may seen redundant at the start of the protocol, on subsequent steps the ring state will also be dependent on the encoded plaintext.

1) Note the “next key character”.
2) If the “next key character” is in the first half of the alphabet, the ring roll direction will be towards the start of the alphabet. If the “next key character” is in the last half of the alphabest, the ring roll direction will be towards the end of the alphabet.
3) Note the dot pattern on Ring 1 and Ring 2.
4) Turn the whole ring until that dot pattern appears on Ring 2 and Ring 3.
5) Turn Ring 3 to match the “next key character”.
6) Reverse the direction chosen in Step 2.
7) Note the dot pattern on Ring 1 and Ring 2.
8) Turn the whole ring until that dot pattern appears on Ring 2 and Ring 3.
9) The “next key character” has now been mixed.
10) If you are Encoding, turn Ring 2 and Ring 3 together until Ring 2 matches the next plaintext letter. The ciphertext letter is now on Ring 3.
11) If you are Decoding turn Ring 2 and Ring 3 together until Ring 3 matches the next ciphertext letter. The plaintext letter is now on Ring 2.

Sample Encoding

The following steps use the key XYZ and encode the letter A.

```
1) The ring is set to state XYZ.
2) This is the ring state:[.X 'Y .Z]
3) The next key letter is X, direction is towards alphabet end (“up”).
4) The dot pattern is noted (it is down-up).
5) Roll up, giving ['Y 'Z 'A]
6) Roll up, giving [-Z 'A 'B]
7) Roll up, giving ['A 'B 'C]
8) Roll up, giving [-B -C -D]
9) Roll up, giving [.C -D -E]
10)Roll up, giving ['D .E -F]
11)Roll up, giving [-E .F .G]
12)Roll up, giving [.F 'G .H]
13)Roll up, giving ['G 'H .I]
14)Roll up, giving [-H -I 'J]
15)Roll up, giving [.I -J 'K]
16)Roll up, giving ['J .K 'L]
17)The dot pattern has been found.
18)Ring 3 is set to key letter X.
19)This is the ring state: ['J .K -X]
20)The dot pattern is noted (it is up-down).
21)The direction is reversed to be towards alphabet start (“down”).
22)Roll down, giving [.I -J -W]
23)Roll down, giving [-H -I -V]
24)Roll down, giving ['G 'H 'U]
25)Roll down, giving [.F 'G 'T]
26)Roll down, giving [-E .F 'S]
27)Roll down, giving ['D .E .R]
28)Roll down, giving [.C -D .Q]
29)Roll down, giving [-B -C .P]
30)Roll down, giving ['A 'B -O]
31)Roll down, giving [-Z 'A -N]
32)Roll down, giving ['Y 'Z -M]
33)Roll down, giving [.X 'Y 'L]
34)Roll down, giving [-W .X 'K]
35)Roll down, giving ['V .W 'J]
36)Roll down, giving [.U -V .I]
37)Roll down, giving [-T -U .H]
38)Roll down, giving ['S 'T .G]
39)The dot pattern has been found.
40)The next plaintext letter is A.
41)Roll down, giving ['S 'T .G]
42)Roll down, giving ['S 'S -F]
43)Roll down, giving ['S .R -E]
44)Roll down, giving ['S .Q -D]
45)Roll down, giving ['S -P 'C]
46)Roll down, giving ['S -O 'B]
47)Roll down, giving ['S 'N 'A]
48)Roll down, giving ['S 'M .Z]
49)Roll down, giving ['S .L .Y]
50)Roll down, giving ['S .K -X]
51)Roll down, giving ['S -J -W]
52)Roll down, giving ['S -I -V]
53)Roll down, giving ['S 'H 'U]
54)Roll down, giving ['S 'G 'T]
55)Roll down, giving ['S .F 'S]
56)Roll down, giving ['S .E .R]
57)Roll down, giving ['S -D .Q]
58)Roll down, giving ['S -C .P]
59)Roll down, giving ['S 'B -O]
60)Roll down, giving ['S 'A -N]
61)The plaintext letter has been matched on Ring 2.
62)The ciphertext letter is 'N'.
```

This example also constitutes the first check feature to be memorized by ring wearers:

> Key XYZ, encodes message A, to final state SAN.

In order to account for the dependency of the protocol to the key, a second check feature can also be memorized:

Key DOG, encodes the message A, to final state TAQ.

A Complex Example

The following example shows only the initial ring state, next key letter, plaintext, and final ring states after each encoding on each line.

Key of POESY, encoding WELOVEYOU.

```
POE S W QWQ
QWQ Y E FEO
FEO P L ZLV
ZLV O O TOO
TOO E V JVW
JVW S E BEK
BEK Y Y EYN
EYN P O HOA
HOA O U UUG

Encoded result: QOVOWKNAG
```

Protocol Hygiene

The limited key size and set of characters means that long messages will likely be highly vulnerable to automated attacks. Messages should be kept as short as possible. Detailed analysis of the unicity distance would need to account for the varying length of keys.

A detailed analysis of attacks give multiple ciphertexts from the same key has not been done. Although it complicates use of the protocol, it is wise for a pair of ring wearers to have different keys for each sender.

Although the ring is likely to be jostled and played with on a regular basis, wearers should always reset the ring to the base state of AAA following an encode or decode session. This ensures that no remnants of the message or key are left in the ring state.

Simulator

Included with this paper is the sample C code used to test the Fidget protocol. The code is not of great quality, and is in need of serious rework. It is likely to be most valuable as a working example of the protocol for others to re-implement.

Building Your Own Ring

A basic paperstrip ring can be used to construct your own ring. The following strips can be used to build a large demonstration ring around a paper tube.If printed at a smaller scale, they can be used to create a ring that will fit on a finger.