HOMOPHONIC ECA SYMMETRIC CIPHER

The key for this symmetric system is a sequence of bits representing the rule for an elementary cellular automaton in the form popularized by Wolfram.[1] A more serious version of the system might use 256 bits for a neighborhood of size 8, since $2^8 = 256$. One feature of this system is that the size of the key is not fixed. This enlarges the key space. Furthermore, the number of bits in each codeword varies, according to both the key and all of the codewords sent so far. This makes the string of output bits difficult to tokenize (divide into encrypted codewords.)

In the binary version of the system, the sender sends either 0 or 1. Let's assume that the sender decides on 0 as the *first* bit of the plaintext. The sender then generates a string of bits x of length N, where N is the neighborhood size corresponding to the key. Then the CA rule is applied to this random string N times. In other words, let f represent the application of the CA rule encoded by the key. Then the sender examines $f^N(x)$. If the first bit of $f^N(x)$ is 0, then x is the encrypted message. If the first bit is not a 0, then the process is repeated until the first bit *is* a 0. A relatively balanced CA rule should be chosen to begin with, so that there's about an equal chance of succeeding on the first try.

Once an x is found so that $f^N(x)$ has the intended first bit (in our example this was 0), then that x is

[1]For algorithmic simplicity, I start the neighborhood on the square to be updated and then move the right. I use a "cylinder" of bits and wrap around.

sent. But $f^N(x)$ also indicates the length of the next message. To get this length we add the number of $1s$ (mod N) to N. So the next message will have as few as N bits and as many as $2N - 1$ bits. The receiver, who has the key, will be in a position to calculate this length as they calculate the plaintext bit. Attackers without the key will not. Note that the next message will use $f^l(x)$, where l is the length just discussed. So every message determines the "depth" of the calculation used to find the following message/codeword.

Let l_i be the length of codeword i. Then the system defines $l_0 = N$. But then each randomly discovered codeword x_i determines l_{i+1}. So l_i (for $i > 0$) depends both on the key and on all previous randomly generated codewords. Optional: a convention could be established that a codeword of length less than N indicates the end of transmission. This codeword could be entirely random. Because it is too short, the receiver will know that it does not encode a plaintext bit. Such a punctuation mark could also be established as a simple check that the message has not been tampered with. Perhaps the length $N - 1$ (for example) is established as the precise expected length of this punctuated "pseudo-codeword."

This system is homophonic because many codewords work to send the same plaintext bit. The CA rule encodes several different Boolean functions simultaneously, one for each valid codeword size. The sequence l_i indicates which Boolean function is applied, since the applied function must fit the codeword.