Thorium-S is a specialized binary version of Cesium, and another variant of the Thorium idea. Like the others, it uses $n \times n$ matrix of bits as a key. The trace of the matrix is added (mod 2) to the plaintext bit to get the ciphertext bit. The key matrix is adjusted as a function of the plaintext bit. In **Thorium-M**, only one row or column is adjusted per bit of plaintext, so Thorium-m is faster, without any obvious loss of security. **Thorium-S** acts just like **Thorium-M** in this regard:

Here's the encoding function:

```
spin_row(k,i) = circshift(k[i,:], k[i,i]  + 1)
spin_col(k,i) = circshift(k[:,i], k[i,i]  + 1)
function encode(p,q)
    k = copy(q)
    n = size(q)[begin]
    c = Bool[]
    for i in eachindex(p)
        push!(c,Bool((tr(k) + p[i])%2))
        if  Bool(p[i])
            k[mod1(i,n),:] = spin_row(k,mod1(i,n))
        else
            k[:,mod1(i,n)] = spin_col(k,mod1(i,n))
        end
    end
    c
end
```

The autospin function of the "S" version is where it diverges from other versions of Thorium. The S version uses the key to encode {itself row by row. Its "story" is an array of iterative self-modification of the key.

```
function self(q)
    n = size(q)[begin]
    k = deepcopy(q)
    for i in 1:n k[i,:] = encode(q[i,:],q) end
    k
end
function story(q)
    n = size(q)[begin]
    k = deepcopy(q)
    s = []
    for i in 1:n
        push!(s,k)
        k = self(k)
    end
    s
end
```

You can see that the story is "told" forward for encryption and backwards for decryption. In version S, the number of rounds is fixed as the size of the key. It's inefficient to generate the story twice, but this simulates the real-world situation. The receiver of the message will not have access to the story, but only the key.

```
function encrypt(p, q)
    n = size(q)[begin]
    s = story(q)
    for i in 1:n
        p = encode(p,s[i])
        p = reverse(p)
    end
    p
end
function decrypt(c, q)
    n = size(q)[begin]
    s = story(q)
    for i in 1:n
        c = reverse(c)
        c = decode(c,s[n + 1 - i])
    end
    c
end
```

Because the number of rounds is fixed by the key, we have a well defined function f_k with only k for a parameter. Likewise we have f_k^{-1}. Users will probably want to check the story of a potential key, just to make sure it stays well-mixed (about half of each kind of bit.)