**Thorium-M** is a specialized binary version of Cesium, and a simpler variant of the Thorium idea. 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.

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)
    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 "m" version of Thorium is also gentler and less expensive. It uses bits from the key (column by column) to update itself as if by plaintext bits, one column at a time.

```
function autospin(q,c)
    k = copy(q)
    for i in 1:n Bool(q[i,c]) ? k[i,:] = spin_row(k,i) : k[:,i] = spin_col(k,i) end
    k
end
```

Note that all shifts are by one unit, so the plaintext determines only the order of the shifts ( such as "row, row, col" versus "col, row, col" and so on.)

The point of **autospin** is to meet each round with a different version of the key. Decryption requires using these keystates in reverse order.

```
function encrypt(p, q, r)
    for i in 1:r
        k = autospin(q, mod1(i,n))
        p = encode(p,k)
        p = reverse(p)
    end
    p
end
function decrypt(p, q, r)
    for i in 1:r
        k = autospin(q,mod1(r + 1 - i,n))
        p = reverse(p)
        p = decode(p,k)
    end
    p
end
```

The "official" (ideal) version sets the rounds equal to $n$. This gives us a well defined function $f_k$ with only $k$ for a parameter. Likewise we have $f_k^{-1}$.