Thorium is a specialized binary version of Cesium. 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 then "shredded" as a function of its current diagonal and the processed plaintext bit.

Here's the encoding function:

```
function encode(p,q)
    k = copy(q)
    c = Bool[]
    for i in eachindex(p)
        push!(c,Bool((tr(k) + p[i])%2))
        k = spin(k,p[i])
    end
    c
end
```

It's crucial that the key evolve as a function of the plaintext, or we could have just used a simple masking bit-vector to begin with. Here's the update of the key. Note that Julia gives us a convenient circular shift of rows or columns as needed.

```
spin_row(k,i,p) = circshift(k[i,:], k[i,i] + p + 1)
spin_col(k,i,p) = circshift(k[:,i], k[i,i] + p + 1)
function spin(q,p)
    k = copy(q)
    for j in 1:n[begin]
        if  Bool((j+p)%2)
            k[j,:] = spin_row(k, j, p)
        else
            k[:,j] = spin_col(k, j, p)
        end
    end
    k
end
```

You can see that n spins are performed, alternating between rows or columns. The bit p determines whether rows or columns go first, **and** this bit boosts (or not) every such spin.

For encryption there's also an "autospin" feature that prepares a different evolution of the key for each round. The columns of the key are fed like n plaintext bits to guide this evolution.

```
function auto_spin(q,c)
    k = copy(q)
    for i in 1:n  k = spin(k,k[c,i]) end
    k
end
```