

Swhich is similar to **Eel** and **Millipede**, but it uses two “keystings” instead of one, and both “roll into” the other, as if they were twisted, hence the name.

```
function swhich!(f)
    f[1] = [f[1][1:end];f[2][1]]
    f[2] = [f[2][2:end];f[1][1]]
    popfirst!(f[1])
end
```

In the encoding function, the leading entries of both keystings are added mod n to the plaintext symbol to get the ciphertext symbol. Then the “tapes” (keystings) are tangled/swhiched. Finally the new first entries entry is modified. The plaintext symbol is added to the top left and the newest ciphertext symbol is added to the bottom left (both, again, mod n). (Note that Julia is 1-indexed.)

```
function encode!(p,f,n)
    c = Int64[]
    for i in eachindex(p)
        push!(c, ( f[1][1] + f[2][1] + p[i] )%n )
        swhich!(f)
        f[1][1] = (f[1][1] + p[i])%n
        f[2][1] = (f[2][1] + c[i])%n
    end
    c
end
```

To move from encoding to encryption, we do the usual reversal between rounds. We also “autospin” and make the length of key the default number of rounds. In this case the autospin is a simple circular shift, but the top spins in one direction and the bottom in another.

```
function encrypt(p,q,n)
    l = length(q)
    for i in 1:l
        f = deepcopy(q)
        f[1] = circshift(f[1],i)
        f[2] = circshift(f[2],l + 1 - i)
        p = encode!(p,f,n)
        p = reverse(p)
    end
    p
end
```