

Suitable function

Daniel Nager
daniel.nager@gmail.com

August 2, 2020

This file and referenced files are on the address <https://github.com/danielnager/xifrat/>

We will use a substitution table, for example the following 13×13 table:

5	3	0	12	11	4	9	10	8	1	6	7	2
3	7	2	11	9	10	5	6	0	12	8	4	1
0	2	3	10	6	12	8	11	5	4	9	1	7
12	11	10	0	2	5	1	3	4	8	7	9	6
11	9	6	2	1	3	12	7	10	0	4	5	8
4	10	12	5	3	8	7	0	1	9	2	6	11
9	5	8	1	12	7	11	4	6	2	10	3	0
10	6	11	3	7	0	4	2	12	5	1	8	9
8	0	5	4	10	1	6	12	9	7	11	2	3
1	12	4	8	0	9	2	5	7	6	3	11	10
6	8	9	7	4	2	10	1	11	3	12	0	5
7	4	1	9	5	6	3	8	2	11	0	10	12
2	1	7	6	8	11	0	9	3	10	5	12	4

to define a function $c = f(a, b)$, where c is the element in the a -th row and b -th column.

The following two properties hold:

$f(f(a, b), c) \neq f(a, f(b, c))$ – non-associativity in general

$f(f(a, b), f(c, d)) = f(f(a, c), f(b, d))$ – restricted commutativity

Next we define a list of N integers in the range $[0, 12]$ to meet the size required. For 256 bits we need $256/\log_2(13) = 69$ approximately. So let's set $N = 69$. This list can be interpreted as a 69 digit base-13 number.

Next we define a mixing procedure of elements of this kind, t and k , N -element lists of numbers in the integer range $[0, 12]$.

The mixing procedure is:

```
function m(t,k) returns t

    for M number of rounds -- 64 for example
        //one-to-one mixing of k and t
        for i in 0..N-1
            t[i]<-f(t[i],k[i])
        end for
        // accumulative mixing of t with itself, t[-1]=t[N-1]
        for i in 0..N-1
            t[i]<-f(t[i],t[i-1])
        end for
    end for

return t
```

The function m is neither associative nor commutative, and meets the restricted commutativity property:

$$m(m(a, b), m(c, d)) = m(m(a, c), m(b, d))$$

With this a Secret agreement and a Digital signature can be done as explained in the document:

<https://github.com/danielnager/xifrat/blob/raw/cryptosystem.pdf>

The computationally hard problem proposed is:

in $c = m(t, k)$, knowing c and t , find k .

Now lets define the secret agreement and the digital signature using the mixing function m . To put it more clear we will use the following notation:

$m(a, b)$ is written as (ab)

$m(m(a, b), m(c, d))$ is written as $(ab)(cd)$

$m(m(a, b), c \dots)$ is written as $(abc \dots)$

For the secret agreement the procedure is the following:

Both Alice and Bob agree on some constant C . Alice chooses a random key K , and Bob does the same choosing a random key Q . Alice sends to Bob (CK) , Bob sends to Alice (CQ) . Alice computes using Bob's sent value $(CQ)(KC)$, and Bob does the same and computes $(CK)(QC)$.

By the property of restricted commutativity $(CQ)(KC) = (CK)(QC)$

For the signature the procedure is the following:

Alice, the signer, chooses a public value C and two random keys K, Q . Its credentials are C , (CKK) and (QK) . To sign a value, H , Alice computes $S = (HQQ)$.

Bob needs to verify if Alice has signed H . Computes $(HQQ)(CKK)$ and $(HC)(QK)(QK)$. Both values must be equal due to restricted commutativity if (HQQ) is a valid signature from Alice.

In order to do smaller signatures, of 128 bits in this case, there's an approach that must be carefully tested.

We apply the following equality:

$$\begin{aligned} (CQK)(DKQ)(H_1H_2C)(H_2H_1D)(CQK)(DKQ) \\ = \\ (CDH_1H_2CD)(QKH_2H_1QK)(KQCDKQ) \end{aligned}$$

In this formula, C and D are two 128 bit public constants provided by Alice, the signer, K and Q are two 128 bit keys known only by the signer, and H_1 and H_2 is a 256 bits value to be signed split in two halves.

The credentials of Alice are (CQK) , (DKQ) and $(KQCDKQ)$.

In order to sign a value represented by H_1 and H_2 , the Alice computes $S = (KQH_2H_1QK)$.

To verify the signature Bob computes (CDH_1H_2D) , (H_1H_2C) and (H_2H_1D) , and checks for the initial equality to hold, as Bob has all the elements needed.

Daniel Nager - daniel.nager@gmail.com