Suitable function

Daniel Nager
daniel.nager@gmail.com

March 3, 2021

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

We will use a substitution 16x16 table, this is one of them:

7	9	13	10	15	2	0	6	3	12	8	4	1	5	14	11
1	15	6	3	9	4	11	13	10	5	14	2	7	12	8	0
3	0	12	1	11	8	9	5	7	13	2	14	10	6	4	15
4	6	15	8	13	1	5	9	14	11	10	7	2	0	3	12
0	3	8	15	10	12	7	14	9	2	13	5	11	4	6	1
10	11	5	7	0	14	15	12	1	6	4	8	3	13	2	9
5	14	10	13	8	11	4	3	6	1	15	0	12	7	9	2
15	1	4	0	7	6	10	2	11	14	5	13	9	8	12	3
12	8	3	6	14	0	2	10	13	7	9	11	5	1	15	4
13	2	7	5	4	9	8	1	12	3	0	15	6	10	11	14
6	4	1	12	2	15	14	7	5	10	11	9	13	3	0	8
9	7	2	11	1	13	3	4	0	8	12	6	15	14	5	10
11	10	14	9	3	5	1	8	15	4	6	12	0	2	13	7
14	5	11	2	12	10	6	0	4	15	1	3	8	9	7	13
8	12	0	4	5	3	13	11	2	9	7	10	14	15	1	6
2	13	9	14	6	7	12	15	8	0	3	1	4	11	10	5

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 $f(a, b) \neq f(b, a)$ – non-commutativity 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 $[0, 15]$ to meet the size required. For 256 bits we need $N = 64$. This list can be interpreted as a 64 digit base-16 number.

Next we define a mixing procedure of elements of this kind, t and k, N-element lists of numbers in the integer range $[0, 15]$.

But before, we define a deterministic sequence obtained by a prng, for example, that returns always the same sequence of numbers in the interval $[0, N - 1]$. frist_seq gets the first element of the sequence and next_seq the next one.

The mixing procedure is:

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

    //one-to-one mixing of k and t
    for i in 0..N-1
        r[i] <- f(t[i],k[i])
    end for
    i <- first_seq
    for M number of applications of f -- 4096 for example
        // accumulative mixing of r with itself
        j <- next_seq
        r[j]<-f(r[j],r[i])
        i <- j
    end for
    //one-to-one mixing of k and r
    for i in 0..N-1
        r[i] <- f(r[i],k[i])
    end for

return r
```

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.

At this point an stop should be made to talk about differential and linear cryptoanalysis. If we take each row and each column as a 16 4-bit lenght substitution table, i turns out that the probability of a linear equation holding is at most 14/16 and more or less is the same for differential characteristics. As we're doing 4096 iteration on the s-table, despite 14/16 is a high probability, $log_2((14/16)^4096)$ is by far lesser than 2^{-128}. So there's no attack feasible here.

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 sent value $(CQ)(KC)$, and Bob does the same and computes $(CK)(QC)$.

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

For the signature the procedure is the following:

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

Bob needs to verify if Alice has signed H. Computes $(HQ)(CK)$ and $(HC)(QK)$. Both values must be equal due to restricted commutativity if (HQ) 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:

$$(QCCK)(KH_1H_2Q) = (QK)(CH_1)(CH_2)(KQ)$$

In this formula, C is 128 bit public constant 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 $(QCCK)$, (QK) and (KQ).

In order to sing a value represented by H_1 and H_2, the Alice computes $S = (KH_1H_2Q)$.

To verify the signature Bob computes (CH_1) and (CH_2), and checks for the initial equality to hold, as Bob has all the elements needed. If the equality holds then is a valid signature from Alice.

Daniel Nager - daniel.nager@gmail.com