Ternarylogic

Toy Example Diffie-Hellman Key Exchange Customization by Function Modification

Machine cryptography is the execution by a computer of instructions. It is convenient for design and for cryptanalysis that the machine cryptographic instructions can be modeled by mathematical expressions. However, the reverse is not true. That is: cryptographic instructions performed by a computer do not have to be mathematical expressions or represented by mathematical expressions.

The above may not be evident when one represents everything normal decimal form. However, in n-state switching operations with n>2 the distinction between mathematical and computer operations will become clear.

Diffie-Hellman and RSA operations in cryptography are modeled by a mathematical operation called exponentiation, which is generally repeat multiplication. Thus 2^5=2*2*2*2*2. Furthermore, due to associativity, $(g^a)^b=(g^b)^a$ which is (g^{a*b}). Key in this is the operation '*' which in Diffie-Hellman key exchange is the modulo-n multiplication with n being prime.

So what is customization of Diffie-Hellman? Customization of Diffie-Hellman is the replacement of the operation '*', which is practically an n-state computer operation, with another n-state operation '⊛' that has the same important properties (such as associativity) as '*'.

As an example, multiplication modulo-11 will be used, with the traditional multiplication table modulo-11 provided below.

*mg*11=*	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	1	3	5	7	9
3	0	3	6	9	1	4	7	10	2	5	8
4	0	4	8	1	5	9	2	6	10	3	7
5	0	5	10	4	9	3	8	2	7	1	6
6	0	6	1	7	2	8	3	9	4	10	5
7	0	7	3	10	6	2	9	5	1	8	4
8	0	8	5	2	10	7	4	1	9	6	3
9	0	9	7	5	3	1	10	8	6	4	2
10	0	10	9	8	7	6	5	4	3	2	1

Element g=7 is a generating element of group $\mathbb{Z}_{11}$. That is: each element {1,2,3,4,5,6,7,8,9,10} can be formed as an exponentiation of g=7. For instance g^4=*mg11*(7,*mg11*(7,*mg11*(7,7))). Or g^4=*mg11*(7,*mg11*(7,5))=*mg11*(7,3)=10. Select a=6 and b=7. One can determine that g^6= 4, g^7=6. Furthermore $(4)^7$=5 and $(6)^6$=5, which are the Diffie-Hellman values for a common keyword. (using *mg11* as the operation for exponentiation.)

Instead of 11-state function *mg11* ('*') a new function *mg11a* ('⊛') is used which is not a representation of modulo-11 multiplication, and of which the switching table is provided below.

$mg11a = \otimes$	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	10	8	1	7	3	9	5	4	6
3	0	3	8	9	6	1	7	2	4	10	5
4	0	4	1	6	9	8	10	5	3	7	2
5	0	5	7	1	8	10	4	6	2	3	9
6	0	6	3	7	10	4	5	1	9	2	8
7	0	7	9	2	5	6	1	3	10	8	4
8	0	8	5	4	3	2	9	10	1	6	7
9	0	9	4	10	7	3	2	8	6	5	1
10	0	10	6	5	2	9	8	4	7	1	3

Element h=7 is again a generating element of a new group $\mathbb{Z}_{11}$. That is each element {1,2,3,4,5,6,7,8,9,10} can be formed as an exponentiation of h=7. For instance h^4=*mg11a*(7,*mg11a*(7,*mg11a*(7,7))). Or h^4=*mg11a*(7,*mg11a*(7,3))=*mg11a*(7,2)=9. Select again a=6 and b=7. One can determine that h^6= 10, h^7=4. Furthermore $(10)^7$=3 and $(4)^6$=3, which are the Diffie-Hellman values for a common keyword. (exponentiation here is repeat use of 11-state function *mg11a*.)

The above shows that the Diffie-Hellman method is customized by using a modified 11-state switching function. The customized method generates a different result compared to the traditional approach and will be more difficult to attack successfully.

The value of n (n=11) in the example is of course so small that one may try any of 10 potential instances of a practical 11-state keyword. Customization thus does not does not materially improve security. In practice, Diffie-Hellman uses values of n that are greater than 1000 bits and brute force attacks are less likely to succeed. However, Diffie-Hellman is susceptible to deterministic discrete logarithm attacks, which rely on modulo-n multiplication. By modifying the underlying and well-known n-state computer operation, the Diffie-Hellman method is significantly hardened against attacks.

The number of possible customizations is in the order of factorial n (n!). This number (n!) is enormous for even relatively small numbers of n. For n=256 (a representation in 8 bits) the value of 256! is well over 10^{100}.

The trick is to determine a modified switching function *mg11a* that has properties of a switching table that are equivalent to the table of *mg11*. A quick way to do perform a deterministic modification is the Finite Lab-transform (FLT), which is outside scope of this toy example but is described here.

Peter Lablans, January 22, 2020

Contact: ip@ternarylogic.com