KEY AGREEMENTS BASED ON FUNCTION COMPOSITION

Rainer A. Rueppel

Crypto AG
6312 Steinhausen
Switzerland

Abstract:

Two protocols are presented that accomplish the same goal as the original Diffie-Hellman protocol, namely, to establish a common secret key using only public messages. They are based on n -fold composition of some suitable elementary function. The first protocol is shown to fail always when the elementary function is chosen to be linear. This does not preclude its use for a suitable nonlinear elementary function. The second protocol is shown to be equivalent to the Diffie-Hellman protocol when the elementary function is chosen to be linear. Some examples are given to illustrate the use of both protocols. It is still an open problem whether the presented approach allows for an improvement in terms of speed and/or security over the original DH-protocol.

Suppose we are given an autonomous finite-state machine with next-state function F . After one time step an initial state s_0 will be transferred to $s_1 = F(s_0)$. After n time steps we have

$$s_n = F(F(\dots F(s_0)\dots)) = F^n(s_0)$$

where F^n stands for the n -fold application of F to its argument. (Although we do not need the finite-state machine context to derive some results, we use it to illustrate the approach). Now define two functions g and h ,

$$g: y = F^m(x)$$

$$h: y = F^n(x)$$

These two functions f and g will commute, i.e.,

$$h(g(x)) = g(h(x))$$

This commutativity is also the basic requirement in the DH-protocol. Hence, using the number of steps an FSM has taken from a specific starting point as the individual user's secret, we can implement a key agreement as follows:

Key Agreement Protocol 1:

A and B have agreed on a common FSM with next-state function F and a common starting state s_0 .

- (1) A randomly chooses a secret number n_1 and steps its FSM, loaded with s_0 , n_1 times to obtain

$$s^{(1)} = s_{n_1} = F^{n_1}(s_0)$$

A sends $s^{(1)}$ to B.

B randomly chooses a secret number n_2 and steps its FSM, loaded with s_0 , n_2 times to obtain

$$s^{(2)} = s_{n_2} = F^{n_2}(s_0)$$

B sends $s^{(2)}$ to A.

- (2) A loads the received state $s^{(2)}$ into its FSM and steps it n_1 times to obtain

$$s^{(12)} = F^{n_1}(s^{(2)}) = F^{n_1}(F^{n_2}(s_0)) = F^{n_1+n_2}(s_0)$$

B loads the received state $s^{(1)}$ into its FSM and steps it n_2 times to obtain

$$s^{(21)} = F^{n_2}(s^{(1)}) = F^{n_2}(F^{n_1}(s_0)) = F^{n_2+n_1}(s_0)$$

- (3) Since every state has a unique successor the resulting states $s^{(12)}$ and $s^{(21)}$ must be identical and could serve as a common secret between A and B.

So far we did not impose any restriction on F. But, of course, in order not to render the above protocol useless, the next-state function F must possess the following properties:

- (1) to compute $s_n = F^n(s_0)$ must be "easy".
- (2) to infer n from s_0 and s_n must be "hard".
- (3) to compute $s^{(12)}$ from s_0 , $s^{(1)}$, and $s^{(2)}$ must be "hard".

Example 1: Suppose we use a linear next-state function $F(x) = ax \pmod{p}$ and nonzero initial state s_0 . Then computing the nth state directly is easy (using square and multiply),

$$s^{(1)} = a^{n_1} s_0 \pmod{p}$$

$$s^{(2)} = a^{n_2} s_0 \pmod{p}$$

Inferring n from s_0 and s_n corresponds to taking the discrete log $\pmod{p}$. But computing $s^{(12)}$ from s_0 , $s^{(1)}$, and $s^{(2)}$ can be done at almost no cost,

$$s^{(12)} = s_0^{-1} s^{(1)} s^{(2)} = s_0^{-1} a^{n_1} s_0 a^{n_2} s_0 \pmod{p}$$

In fact, the combination of the above protocol with any linear FSM is insecure. Let A be the state transition matrix, i.e.

$$\underline{s}_{j+1} = A \cdot \underline{s}_j$$

Now the following attack will recover $s^{(12)}$ efficiently.

- (1) compute

$$A^{n_1} \text{ from } \underline{s}_0, \underline{s}_{n_1}, \text{ and } A$$

Determine A^{n_2} the same way.

- (2) form the product

$$A^{n_1} \cdot A^{n_2} = A^{n_1+n_2}$$

which gives away $s^{(12)}$.

Example 2: Suppose we use $F(x) = x^e \pmod{p}$ as the next-state function. Then the public messages $s^{(1)}$ and $s^{(2)}$ to be transmitted are

$$s^{(1)} = F^{n_1}(s_0) = (s_0)^{e^{n_1}} \pmod{p}$$

$$s^{(2)} = F^{n_2}(s_0) = (s_0)^{e^{n_2}} \pmod{p}$$

A computes

$$s^{(12)} = F^{n_1}(s^{(2)}) = (s^{(2)})^{e^{n_1}} = (s_0)^{e^{n_2 \cdot n_1}} \pmod{p}$$

which is identical to the outcome of B's computation and may serve as the common secret.

If an attacker is able to efficiently compute discrete log's mod p , he can also efficiently compute $s^{(12)}$,

$$\log_{s_0} s^{(1)} = e^{n_1} \pmod{p}$$

$$\log_{s_0} s^{(2)} = e^{n_2} \pmod{p}$$

and

$$e^{n_1+n_2} = e^{n_1} \cdot e^{n_2}$$

Note that at this point the attacker has not yet succeeded in deriving the individual secrets n_1 and n_2 of parties A and B. To obtain, say n_1 , he will have to take discrete log's mod $p-1$, whose factorization may be difficult to find.

Example 3: (due to C. Thomé and R. Schwarzenberger) Suppose we use a nonlinear feedback shift register with next-state function

$$F:(x,y) \rightarrow (y, xy) \pmod{p}$$

$$\underline{s}_0 = (a,b) \quad a, b \neq 0$$

The n th state of this NLFSR will be

$$\underline{s}_n = (a^{F_{n-1}} b^{F_n}, a^{F_n} b^{F_{n+1}})$$

where F_n denotes the n th Fibonacci number. Consequently, the n th state is efficiently computable. On the other hand, to break this system cannot be harder than to take discrete log's mod p , since then we may express a , b , and the n th state relative to some generator g ,

$$a = g^{e_1} \pmod{p}$$

$$b = g^{e_2} \pmod{p}$$

$$\underline{s}_n = (g^{e_1 F_{n-1} + e_2 F_n}, g^{e_1 F_n + e_2 F_{n+1}})$$

which, after 2 more log p operations, results in a system of linear congruences which can efficiently be solved.

In a slightly more general approach, we may want to allow that the next-state function of the FSM is changed during the execution of the protocol. Let the two functions g and h be defined as above; then it also holds

$$g^n(x) = h^m(x)$$

Therefore, the above protocol could be modified as follows:

Key Agreement Protocol 2:

A and B have agreed on a common FSM with next-state function F and a common starting state s_0 .

- (1) A randomly chooses a secret number n_1 and computes the description of the function

$$g_1(\cdot) = F^{n_1}(\cdot)$$

A sends the function description of g_1 to B.

B acts correspondingly on its secret n_2 .

- (2) A loads the received function description of g_2 as next-state function into its FSM and steps it n_1 times, started at s_0 , to obtain

$$s^{(12)} = g_2^{n_1}(s_0) = \left(F^{n_2}\right)^{n_1}(s_0) = F^{n_1 n_2}(s_0)$$

B acts correspondingly on the received function description of g_1 in order to obtain s^{21} .

- (3) The resulting states s^{12} and s^{21} are identical and could serve as a common secret between A and B.

Here the conditions on the next-state function F are slightly different (as compared to protocol 1):

- (1) to compute $g = F^n$ from F and n must be "easy".
- (2) to infer n from g and F must be "hard".
- (3) to compute $s(n_1 n_2)$ from s_0 , F , g_1 , and g_2 must be hard.

Example 4: Let the next state function be $F(x) = ax \pmod{p}$, and suppose $s_0 = 1$. Then

$$g_1(x) = F^{n_1}(x) = a^{n_1}x = a_1x \pmod{p}$$

$$g_2(x) = F^{n_2}(x) = a^{n_2}x = a_2x \pmod{p}$$

A sends the function description of g_1 , consisting of the coefficient a_1 , to B. B loads $s_0 = 1$ and g_1 as the next-state function into its FSM, and steps it n_2 times to arrive at

$$s^{(12)} = \left(a^{n_1}\right)^{n_2} \pmod{p}$$

A acts accordingly on the received g_2 . (Note that this is the reformulation of the original Diffie-Hellman protocol [1]).

In general, let A be a linear operator on a finite-dimensional vectorspace over a field F . Let $g(x)$ be the minimal polynomial of A , that is, the polynomial of least positive degree k such that $g(A)=0$. The Cayley-Hamilton theorem states that $g(x)$ must divide the

characteristic polynomial of A , and thus, that the degree of $g(x)$ is smaller than or equal to the dimension of the vectorspace A operates on. Applying Euclid we may write

$$x^n = q(x)g(x) + r(x)$$

where the degree of $r(x)$ is smaller than k . Consequently,

$$A^n = r(A)$$

since $g(A)=0$. Thus, any linear map A used in the second protocol leads to the following problem: given two polynomials $r(x)$ and $g(x)$ over F . Find the least positive exponent n such that

$$x^n = r(x) \pmod{g(x)}$$

If $g(x)$ is irreducible, this is the discrete log problem in an extension field. Thus, when used with a linear elementary function F , the second protocol is equivalent to the original Diffie-Hellman protocol.

Example 5: Let the next-state function be $F(x) = x^e \pmod{p}$, with $1 \leq s_0 < p-1$. Then

$$g_1(x) = x^{e^{n_1}} = x^{e_1} \pmod{p}$$

$$g_2(x) = x^{e^{n_2}} = x^{e_2} \pmod{p}$$

A sends the function description of g_1 , consisting of the exponent e_1 , to B. B computes, based on g_1 as next-state function,

$$s^{(12)} = g_1^{n_2}(s_0) = s_0^{e_1^{n_2}} = s_0^{e_1^{e_2}} \pmod{p}$$

A acts accordingly on the received g_2 .

Note that the security of this system rests on the difficulty of taking logarithms modulo $p-1$ whose factorization may be difficult to find.

Summary:

There exist different protocol versions and different elementary functions F that allow for a Diffie-Hellman type key agreement. Protocol 1 (in this paper) is insecure with linear F , but reveals only intermediate states. Protocol 2, when used with linear F , is equivalent to the original Diffie-Hellman key agreement, and reveals whole function descriptions F^n . It is still an open problem whether the presented approach allows for an improvement in terms of speed and/or security over the original DH-protocol.

Acknowledgment:

I wish to thank Kjell-Ove Widman and Jim Massey for their helpful comments.

References:

- [1] W.Diffie, Martin E. Hellman, "New Directions in Cryptography", IEEE Trans. on Information Theory, Vol. IT-22, Nov. 1976.