LASEC
SECURITY AND CRYPTOGRAPHY LABORATORY

EPFL / I & C
CH-1015 Lausanne
Phone: ++41 (0) 21 693 76 03
Fax: ++41 (0) 21 693 68 79
URL: http://lasecwww.epfl.ch

Cryptography & Security: Final Exam Solutions

Implementation of the Diffie-Hellman Protocol

1. Given a secure channel, both ends (say, Alice and Bob) can perform a Diffie-Hellman key-exchange protocol to finally obtain a common secret key. Subsequently, Alice and Bob can use the secret key to encrypt/decrypt the rest of the communication (in order to protect the confidentiality) or to authenticate it using a Message Authentication Code (MAC).

2. The protocol is summarized on Figure 1. On the figure, the notation $x \in_R [1, p-1]$ means that x is is pick in $[1, p-1]$ uniformly at random. Both X and Y are transmitted on the secure channel.

3. If the channel is no longer secure, the protocol can be subject to a *man-in-the-middle attack*. Because Bob cannot make sure that the X he receives indeed comes from Alice, a malicious adversary (Eve) can intercept Alice's messages and send some other message to Bob. Basically, Eve can intercept all messages coming from both Alice and

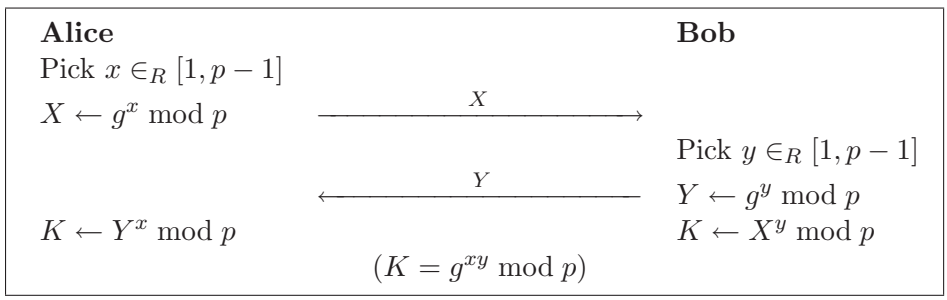

Figure 1: The Diffie-Hellman key agreement protocol between Alice and Bob

Bob, and perform two Diffie-Hellman key exchange protocols in parallel: one with Alice (in which Eve pretends to be Bob) and one with Bob (in which Eve pretends to be Alice). At the end, Eve shares a "secret" key with Alice and another one with Bob. Both Alice and Bob believe that they share a common secret key.

4. We denote by $X\|Y'$ the transcript of Alice and $X'\|Y$ the one of bob, where X' and Y' may be different from X and Y respectively (for example, if a man-in-the-middle attack occurred). Assuming that the hash function H is collision resistant, we have

$$H(X\|Y') = H(X'\|Y) \Rightarrow X\|Y' = X'\|Y \Rightarrow X = X' \text{ and } Y = Y'.$$

The last condition is sufficient to make sure that Alice and Bob share the *same secret* key at the end of the protocol.

5. Both Alice and Bob have to make sure that $H(X\|Y') = H(X'\|Y)$, so that they will send $H(X\|Y')$ and $H(X'\|Y)$ respectively. Assuming that they use a 160 bits hash function, 320 bits must be transmitted in total. Note that the standard Diffie-Hellman protocol would require around $2 \cdot \log p \approx 2048$ bits in the case where p is a prime of 1024 bits.

6. Truncating all digests down to 20 bits is equivalent to consider that H is a 20 bits hash function. The complexity of a second preimage attack is roughly 2^{20} hash computations, which is a trivial computation for a standard PC.

7. As in question 3, assume Eve intercepts all messages between Alice and Bob that are not transmitted over the secure channel. Eve can intercept X and send $X' \leftarrow g^{x'}$ to Bob. Once Eve has intercepted Y, she can look for $Y' \leftarrow g^{y'}$ such that

$$H(X\|Y') = H(X'\|Y),$$

which exactly corresponds to a second preimage attack on H. If the previous equation holds, the attack succeeds as both Alice and Bob rely on it to check whether the protocol is correct. Nevertheless, in this case Eve shares a "secret" key $K_1 = g^{xy'}$ with Alice and a "secret" key $K_2 = g^{x'y}$ with Bob.

8. Using $k = 80$ is enough for a generic hash function to resist second preimage attacks. In this case, the previous attack does not apply.

9. Assume that Eve replaces the value of g that is sent by Alice to Bob by $g' = 1$. The resulting protocol is represented on Figure 2. At the end of the protocol, Alice and Bob share the same key $K_2 = K_1 = 1$ provided that $X^y \bmod p' = 1$, i.e., provided that

$$X^y - 1 \equiv 0 \pmod{p'}. \tag{1}$$

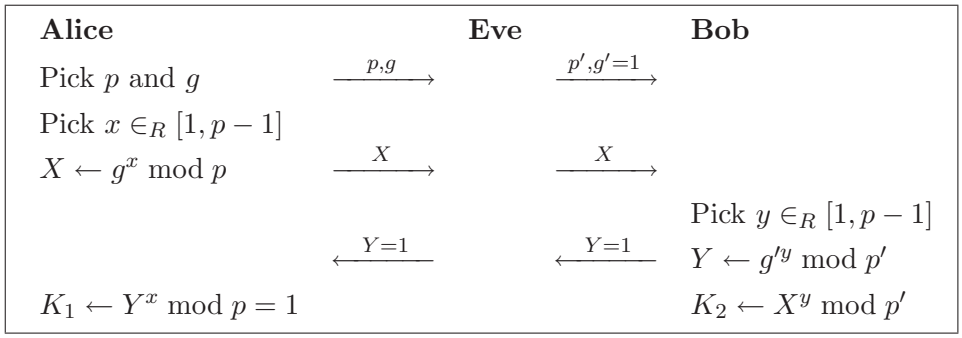

Figure 2: The Diffie-Hellman key agreement protocol without authenticated parameters

Noting that $X^y - 1 = (X-1)(X^{y-1} + X^{y-2} + \cdots + 1)$, Eve can choose $p' = X - 1$ to make sure that (1) holds. In this case Alice and Bob share the same key $K_1 = K_2 = 1$ which can obviously be computed by Eve.

10. Bob can for example generate what is called a *strong* prime number p i.e., a prime such that $q = \frac{p-1}{2}$ is also a prime. This can be done by generating q at random, checking whether it is prime with the Miller-Rabbin primality testing algorithm and, if q is prime, checking whether $p = 2q + 1$ is also prime. Assuming that the primality of q and p are independent events, the complexity of this generation is $O(\ell^5)$, where ℓ is the bit length of p. In such a case, the order of the group $\mathbf{Z}_p^*$ is $\varphi(p) = p - 1 = 2q$. Any element of this group is either of order 2 (which can be checked easily), of order q, or a generator of the group (as, according to Lagrange's Theorem, the order of an element divides the group order). Therefore, in this case it is sufficient to check that the order of g is different from two (i.e., it is sufficient to check that $g^2 \neq 1$) to make sure that its order is large.

A Provably Secure Hash Function

1. The prime number p can be generated by picking ℓ bit integers, and submitting these numbers to the Miller-Rabbin primality test until it succeeds. Given that the number of random numbers that are necessary to obtain a prime is $O(\ell)$ and that the complexity of each Miller-Rabbin primality test is $O(\ell^3)$, the complexity of generating p (and q) is $O(\ell^4)$.

2. To compute $h(x) = g^x$, one typically use a *square-and-multiply* algorithm (also called *exponentiation from left to right*). The complexity is $O(\ell^2 \cdot \log(x))$.

3. Typically, the size of g^x is roughly of the same size as n, which is of size 2ℓ. The complexity of a first preimage attack is $O(2^{2\ell})$ and the complexity of a collision search is $O(2^{\ell})$.

4. According to the Chinese Remainder Theorem (CRT), if p and q are coprimes, the mapping

$$\begin{array}{rccc} f: & \mathbf{Z}_n & \longrightarrow & \mathbf{Z}_p \times \mathbf{Z}_q \\ & x & \longmapsto & (x \bmod p, x \bmod q) \end{array}$$

is an isomorphism. We also have that for all $(x_1, x_2) \in \mathbf{Z}_p \times \mathbf{Z}_q$,

$$f^{-1}(x_1, x_2) = (x_1 \cdot q \cdot (q^{-1} \bmod p) + x_2 \cdot p \cdot (p^{-1} \bmod q)) \bmod n.$$

From this, we deduce that

$$g = (g_1 \cdot q \cdot (q^{-1} \bmod p) + g_2 \cdot p \cdot (p^{-1} \bmod q)) \bmod n.$$

5. The order λ of g is the least positive integer such that $g^\lambda = 1$ in $\mathbf{Z}_n^*$. According to the CRT (and using the notations of the previous question), we know that

$$g^\lambda = 1 \Leftrightarrow f(g^\lambda) = f(1) \Leftrightarrow (g_1^\lambda, g_2^\lambda) = (1, 1).$$

Therefore, λ can be considered as the smallest positive integer such that $g_1^\lambda = 1$ in $\mathbf{Z}_p^*$ and such that $g_2^\lambda = 1$ in $\mathbf{Z}_q^*$. As the order of g_1 and g_2 are $p-1$ and $q-1$ respectively, we have

$$\begin{cases} g_1^\lambda \equiv 1 \pmod{p} \\ g_2^\lambda \equiv 1 \pmod{q} \end{cases} \Leftrightarrow \begin{cases} (p-1) | \lambda \\ (q-1) | \lambda \end{cases}$$

so that the least positive integer satisfying the previous conditions is

$$\lambda = \mathrm{lcm}(p-1, q-1).$$

6. It is easy to see that $h(0) = h(\lambda)$. More generally for two distinct integers k and k' we have $h(k \cdot \lambda) = h(k' \cdot \lambda)$.

7. Assume we know $x \neq y$ such that $h(x) = h(y)$. We have

$$g^x \equiv g^y \pmod{n} \Leftrightarrow g^{x-y} \equiv 1 \pmod{n} \Leftrightarrow \lambda | (x-y).$$

Therfore, $x - y$ is a multiple of λ.

8. It is know that the knowledge of a multiple of λ allows to factoring n (see the lecture notes). The algorithm which is used is very similar to the Miller-Rabbin primality test. Therefore, from the previous question, finding a collision on h is sufficient to factoring n. Conversely, we know from question 5 that the factorization of n allows to compute λ, which is sufficient to computes collisions on h (from question 6). We conclude that finding a collision on h is equivalent to factoring n.

9. One can take for example $\ell = 1024$. The security of the hash function would then be comparable to the security of RSA-2048.