Note: Section 3.3 is based on the first edition of "Cryptography Theory and Practice", which was published in 1995, and it has not been updated for this course.

3.3 Diffie-Hellman Key Predistribution

In this section, we describe a key predistribution scheme that is a modification of the well-known Diffie-Hellman key exchange protocol that we will discuss a bit later, in Section 4.1. We call this the **Diffie-Hellman Key Predistribution Scheme**. The scheme is computationally secure provided a problem related to the **Discrete Logarithm** problem is intractible.

We will describe the scheme over $\mathbb{Z}_p$, where p is prime, though it can be implemented in any finite group in which the **Discrete Logarithm** problem is intractible. We will assume that α is a primitive element of $\mathbb{Z}_p$, and that the values p and α are publicly known to everyone in the network.

In this scheme, $\mathrm{ID}(\mathrm{U})$ will denote certain identification information for each user U in the network, e.g. his or her name, e-mail address, telephone number, or other relevant information. Also, each user U has a secret exponent a_{U} (where $0 \leq a_{\mathrm{U}} \leq p-2$), and a corresponding public value

$$b_{\mathrm{U}} = \alpha^{a_{\mathrm{U}}} \bmod p.$$

The TA will have a signature scheme with a (public) verification algorithm ver_{TA} and a secret signing algorithm sig_{TA}. Finally, we will implicitly assume that all information is hashed, using a public hash function, before it is signed. To make the procedures easier to read, we will not include the necessary hashing in the description of the protocols.

Certain information pertaining to a user U will be authenticated by means of a certificate which is issued and signed by the TA. Each user U will have a certificate

$$\mathbf{C}(\mathrm{U}) = (\mathrm{ID}(\mathrm{U}), b_{\mathrm{U}}, sig_{\mathrm{TA}}(\mathrm{ID}(\mathrm{U}), b_{\mathrm{U}})),$$

where b_{U} is formed as described above (note that the TA does not need to know the value of a_{U}). A certificate for a user U will be issued when U joins the network. Certificates can be stored in a public database, or each user can store his or her own certificate. The signature of the TA on a certificate allows anyone in the network to verify the information it contains.

It is very easy for U and V to compute the common key

$$K_{\mathrm{U,V}} = \alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p,$$

as shown in Figure 3.2.

We illustrate the algorithm with a small example.

Example 3.2 Suppose $p = 25307$ and $\alpha = 2$ are publicly known (p is prime and α is a primitive root modulo p). Suppose U chooses $a_{\mathrm{U}} = 3578$. Then she computes

$$\begin{aligned} b_{\mathrm{U}} &= \alpha^{a_{\mathrm{U}}} \bmod p \\ &= 2^{3578} \bmod 25307 \\ &= 6113, \end{aligned}$$

which is placed on her certificate. Suppose V chooses $a_{\mathrm{V}} = 19956$. Then he computes

$$\begin{aligned} b_{\mathrm{V}} &= \alpha^{a_{\mathrm{V}}} \bmod p \\ &= 2^{19956} \bmod 25307 \\ &= 7984, \end{aligned}$$

which is placed on his certificate.

Now U can compute the key

$$\begin{aligned} K_{\mathrm{U,V}} &= {b_{\mathrm{V}}}^{a_{\mathrm{U}}} \bmod p \\ &= 7984^{3578} \bmod 25307 \\ &= 3694, \end{aligned}$$

and V can compute the same key

$$\begin{aligned} K_{\mathrm{U,V}} &= {b_{\mathrm{U}}}^{a_{\mathrm{V}}} \bmod p \\ &= 6113^{19956} \bmod 25307 \\ &= 3694. \end{aligned}$$

$\square$

Let us think about the security of this scheme in the presence of a passive or active adversary. The signature of the TA on users' certificates effectively prevents W from altering any information on someone else's certificate. Hence we need only worry about passive attacks. So the pertinent question is: Can a user W compute $K_{\mathrm{U,V}}$ if W $\neq$ U, V? In other words, given $\alpha^{a_{\mathrm{U}}} \bmod p$ and $\alpha^{a_{\mathrm{V}}} \bmod p$ (but not a_{U} nor a_{V}), is it feasible to compute $\alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p$? This problem is called the **Diffie-Hellman** problem, and it is formally defined (using an equivalent but slightly different presentation) in Problem 3.1.

Problem 3.1:	Computational Diffie-Hellman
Instance:	A multiplicative group $(G, \cdot)$, an element $\alpha \in G$ having order n, and two elements $\beta, \gamma \in \langle \alpha \rangle$.
Question:	Find $\delta \in \langle \alpha \rangle$ such that $\log_\alpha \delta \equiv \log_\alpha \beta \times \log_\alpha \gamma \pmod{n}$. (Equivalently, given α^b and α^c, find α^{bc}.)

It is clear that **Diffie-Hellman Key Predistribution** is secure against a passive adversary if and only if the **Diffie-Hellman** problem is intractible.

If W could determine a_{U} from b_{U}, or if he could determine a_{V} from b_{V}, then he could compute $K_{\mathrm{U,V}}$ exactly as U (or V) does. But both these computations are instances of the **Discrete Log** problem. So, provided that the **Discrete Log** problem in $\mathbb{Z}_p$ is intractible, **Diffie-Hellman Key Predistribution** is secure against this particular type of attack. However, it is an unproven conjecture that any algorithm that solves the **Diffie-Hellman** problem could also be used to solve the **Discrete Log** problem. (This is very similar to the situation with **RSA**, where it is conjectured, but not proved, that breaking **RSA** is polynomially equivalent to factoring.)

By the remarks made above, the **Diffie-Hellman** problem is no more difficult than the **Discrete Log** problem. Although we cannot say precisely how difficult this problem is, we can relate its security to that of another cryptosystem we have already studied, namely the **ElGamal Cryptosystem**.

THEOREM 3.2 *Breaking the* **ElGamal Cryptosystem** *is equivalent to solving the* **Diffie-Hellman** *problem.*

PROOF First we recall how **ElGamal** encryption and decryption work. The key is $K = (p, \alpha, a, \beta)$, where $\beta = \alpha^a \bmod p$ (a is secret and p, α and β are public). For a (secret) random number $k \in \mathbb{Z}_{p-1}$,

$$e_K(x, k) = (y_1, y_2),$$

where

$$y_1 = \alpha^k \bmod p$$

and

$$y_2 = x\beta^k \bmod p.$$

For $y_1, y_2 \in \mathbb{Z}_p{}^*$,

$$d_K(y_1, y_2) = y_2({y_1}^a)^{-1} \bmod p.$$

Suppose we have an algorithm **A** to solve the **Diffie-Hellman** problem, and we are given an **ElGamal** encryption (y_1, y_2). We will apply the algorithm **A** with inputs p, α, y_1 and β. Then we obtain the value

$$\begin{aligned}\mathbf{A}(p, \alpha, y_1, \beta) &= \mathbf{A}(p, \alpha, \alpha^k, \alpha^a)\\ &= \alpha^{k\,a} \bmod p\\ &= \beta^k \bmod p.\end{aligned}$$

Then the decryption of (y_1, y_2) can easily be computed as

$$x = y_2(\beta^k)^{-1} \bmod p.$$

Conversely, suppose we have an algorithm **B** that performs **ElGamal** decryption. That is, **B** takes as inputs p, α, β, y_1 and y_2, and computes the quantity

$$x = y_2({y_1}^{\log_\alpha \beta})^{-1} \bmod p.$$

Now, given inputs p, α, β and γ for the **Diffie-Hellman** problem, it is easy to see that

$$\begin{aligned}\mathbf{B}(p, \alpha, \beta, \gamma, 1)^{-1} &= 1((\gamma^{\log_\alpha \beta})^{-1})^{-1} \bmod p\\ &= \gamma^{\log_\alpha \beta} \bmod p,\end{aligned}$$

as desired. ∎

FIGURE 3.2
Diffie-Hellman key predistribution

1. A prime p and a primitive element $\alpha \in \mathbb{Z}_p{}^*$ are made public.
2. V computes

$$K_{\mathrm{U,V}} = \alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p = b_{\mathrm{U}}{}^{a_{\mathrm{V}}} \bmod p,$$

   using the public value b_{U} from U's certificate, together with his own secret value a_{V}.
3. U computes

$$K_{\mathrm{U,V}} = \alpha^{a_{\mathrm{U}} a_{\mathrm{V}}} \bmod p = b_{\mathrm{V}}{}^{a_{\mathrm{U}}} \bmod p,$$

   using the public value b_{V} from V's certificate, together with her own secret value a_{U}.