14.2: Diffie-Hellman Key Agreement

Key agreement refers to the problem of establishing a private channel using public communication. Suppose Alice & Bob have never spoken before and have no shared secrets. By exchanging public messages (i.e., that can be seen by any external observer), they would like to establish a secret that is known only to the two of them.

The Diffie-Hellman protocol is such a key-agreement protocol, and it was the first published instance of public-key cryptography:

Construction 14.3: Diffie-Hellman

Both parties agree (publicly) on a cyclic group \( \mathbb{G} \) with generator \( g \). Let \( n = |\mathbb{G}| \). All exponentiations are with respect to the group operation in \( \mathbb{G} \).

1. Alice chooses \( a \leftarrow \mathbb{Z}_n \). She sends \( A = g^a \) to Bob.
2. Bob chooses \( b \leftarrow \mathbb{Z}_n \). He sends \( B = g^b \) to Alice.
3. Bob locally outputs \( K := A^b \). Alice locally outputs \( K := B^a \).

By substituting and applying standard rules of exponents, we see that both parties output a common value, namely \( K = g^{ab} \in \mathbb{G} \).
Defining Security for Key Agreement

Executing a key agreement protocol leaves two artifacts behind. First, we have the collection of messages that are exchanged between the two parties. We call this collection a transcript. We envision two parties executing a key agreement protocol in the presence of an eavesdropper, and hence we imagine that the transcript is public. Second, we have the key that is output by the parties, which is private.

To define security of key agreement, we would like to require that the transcript leaks no (useful) information to the eavesdropper about the key. There are a few ways to approach the definition:

- We could require that it is hard to compute the key given the transcript. However, this turns out to be a rather weak definition. For example, it does not rule out the possibility that an eavesdropper could guess the first half of the bits of the key.
- We could require that the key is pseudorandom given the transcript. This is a better definition, and the one we use. To formalize this idea, we define two libraries. In both libraries the adversary / calling program can obtain the transcript of an execution of the key agreement protocol. In one library the adversary obtains the key the resulted from the protocol execution, while in the other library the adversary obtains a totally unrelated key (chosen uniformly from the set $\Sigma$.? of possible keys).

Definition \(\PageIndex{1}\)

Let $\Sigma$ be a key-agreement protocol. We write $\Sigma$.? for the keyspace of the protocol (i.e., the set of possible keys it produces). We write $(t,K) \leftarrow \text{EXEC PROT}(\Sigma)$ to denote the process of executing the protocol between two honest parties, where $t$ denotes the resulting transcript, and $K$ is resulting key. Note that this process is randomized, and that $K$ is presumably correlated to $t$.

We say that $\Sigma$ is secure if $\mathcal{L}^\Sigma_{\text{ka-real}} \equiv \mathcal{L}^\Sigma_{\text{ka-rand}}$, where:

\[
\begin{align*}
\hat{L}^\Sigma_{\text{ka-real}}(\text{QUERY}()) &= (t,K) \leftarrow \text{EXEC PROT}(\Sigma) \\
&\text{return } (t,K)
\end{align*}
\]

\[
\begin{align*}
\hat{L}^\Sigma_{\text{ka-rand}}(\text{QUERY}()) &= (t,K') \leftarrow \Sigma.K \\
&\text{return } (t,K')
\end{align*}
\]