1 Digital Signature Scheme

Digital signature schemes are a 3 PPT algorithms:

1. $\mathsf{KeyGen}(1^k, R) \rightarrow (pk, sk)$
2. $\mathsf{Sign}(pk, sk, D, R) \rightarrow \mathsf{sig}(D)$
3. $\mathsf{Verify}(pk, D, \mathsf{Sig}(D))$

We build a signature scheme that is secure using a 1 way permutation and collision resistant hash function.

1.1 Lamport 1-Time Signature Scheme

Consider the following simple signature scheme using 1-way permutation f to sign bit b.

- Create private keys x_0, x_1 and publish public keys $f(x_0), f(x_1)$.
- To sign bit b, output x_b as the signature for b.
- Verify simply checks that $f(x_b) = b$.

To demonstrate why this is secure, we construct an adversary A that can invert one way permutation f given and an adversary A' that can forge a signature.

1. Challenger C sends $y = f(x), x \leftarrow \$$ to A who needs to generate an x' such that $f(x') = y$.
2. A, given oracle access to A', imitates that challenger C' by sneding A' public keys $(y, f(x'))$ with $x' \leftarrow \$$. A' will now ask for a signature of either y or $f(x')$ by sending bit b. If $b = 1$
3. If $b = 0$, A starts from the beginning and switches the order of the public keys.
4. If $b = 1$, A reveals signature x' and, A' must be able to forge the signature for $b = 1$ with non negligible probability. Since the only valid signature for 1 would be x, A inverts $f(x)$ with non negligible probability.

1.1.1 Extending to sign multiple bits

An easy extension of the above idea to sign a message of length n would be to create $2n$ (pk, sk) pairs using $x_0, .., x_n$ for randomly selected x_i values, and signing each bit in the message by revealing x_{bi}. This signature scheme is secure under the assumption that the adversary sees only one signature. The proof of security follows from the original proof:

1. The adversary A receives $y = f(x)$ for random x.
2. It creates a table of public and private keys with $2n$ entries using random x_{0i}, x_{1i} values.
3. It randomly replaces one $f(x_{0i})$ values with y.
4. If the requested message at index i requests to reveal 0, it fails and retries by selecting a new index and bit value $(f(x_1 i'))$, otherwise it can reveal x_{1b}.
5. If, at index i, A' signs the bit corresponding to $f(x)$, then we sucessfully invert x.

This suceeds with probability $\epsilon/2n$, which is the advantage of A' multiplied by the probability that it doesn't ask for the bit at index i AND that it differs at index i from the original requested message. The probability $1/n$ comes from the fact that at least one bit in the newly signed message must be different from the original requested message.

1.2 Multi-use PK Signature Scheme

We have the following signature scheme that uses a single public key pk_0 than can sign messages of length k under the assumtion that collision resistant hash functions exist.

1. We generate pk_0 by following Lamport's signature scheme to generate a public, private key with $4k$ $(x, f(x))$ pairs.
2. To sign a message of length k, you use the first k bits of the pk to reveal the corresponding bits.
3. For subsequent messages, you generate a new key pk_i, which is verified as being written by the owner of pk_0 by using the second k bits of pk_{i-1} to sign the **hash** of pk_i (which is $4nk$ bits long) to k bits, and now use the first k (pk, sk) pairs of this new key to sign new message m_i

This scheme is secure as one can see it follows from Lamport's signature scheme. Furthermore, to fake a key k_i, one would need to find a collision which can only happen with negligible probability since we use a collision resistant hash function.