20 ELECTRONIC DIGITAL SIGNATURE

The final cryptographic primitive which we'll define here is the *digital signature*. The basic idea is that a signature on a message can be created by only one person, but checked by anyone. It can thus perform the sort of function in the electronic world that ordinary signatures do in the world of paper. Applications include signing software updates, so that a PC can tell that an update to Windows was really produced by Microsoft rather than by a villain.

Signature schemes can be *deterministic* or *randomized*: in the first, computing a signature on a message will always give the same result and in the second, it will give a different result. (The latter is more like handwritten signatures; no two are ever alike but the bank has a means of deciding whether a given specimen is genuine or forged). Also, signature schemes may or may not support *message recovery*. If they do, then given the signature, anyone can recover the message on which it was generated; if they don't, then the verifier needs to know or guess the message before he can perform the verification. (There are further, more specialised, signature schemes such as blind signatures and threshold signatures but I'll postpone discussion of them for now.)

Formally, a signature scheme, like public key encryption scheme, has a keypair generation function which given a random input R will return two keys, σR (the private signing key) and VR (the public signature verification key) with the properties that

1. Given the public signature verification key VR, it is infeasible to compute the private signing key σR;
2. There is a digital signature function which given a message M and a private signature key σR, will produce a signature $Sig_{\sigma R}(M)$; and
3. There is a signature verification function which, given the signature $Sig_{\sigma R}(M)$ and the public signature verification key VR will output TRUE if the signature was computed correctly with σR and otherwise output FALSE.

We can model a simple digital signature algorithm as a random function that reduces any input message to a one-way hash value of fixed length, followed by a special kind of block cipher in which the elf will perform the operation in one direction, known as *signature*, for only one principal, while in the other direction, it will perform verification for anybody.

Signature verification can take two forms. In the basic scheme, the elf (or the signature verification algorithm) only outputs TRUE or FALSE depending on whether the signature is good. But in a scheme with *message recovery*, anyone can input a signature and get back the message corresponding to it. In our elf model, this means that if the elf has seen the signature before, it will give the message corresponding to it on the scroll, otherwise it will give a random value (and record the input and the random output as a signature and message pair). This is sometimes desirable: when sending short messages over a low bandwidth channel, it can save space if only the signature has to be sent rather than the signature plus the message. An example is in the machine-printed postage stamps, or *indicia*, being brought into use in many countries: the stamp may consist of a 2-d barcode with a digital signature made by the postal meter and which contains information such as the value, the date and the sender's and recipient's post codes.

However, in the general case we do not need message recovery, as the message to be signed may be of arbitrary length and so we will first pass it through a hash function and then sign the hash value. As hash functions are one-way, the resulting compound signature scheme does not have message recovery — although if the underlying signature scheme does, then the hash of the message can be recovered from the signature.