## 4.8 Public Key Infrastructure

A *public key infrastructure*, or PKI, is the sum total of everything required to securely use public keys in the real world. It's surprisingly difficult and involved to assemble all of the necessary pieces of a PKI into a working whole. For a discussion of some of the risks inherent in PKI, see [101].

A *digital certificate* (or public key certificate or, for short, certificate) contains a user's name along with the user's public key and this is signed by a *certificate authority*, or CA. For example, Alice's certificate contains[14]

$$M = (\text{“Alice”}, \text{Alice's public key}) \text{ and } S = [M]_{\text{CA}}.$$

To verify this certificate, Bob would compute $\{S\}_{\text{CA}}$ and verify that this matches $M$.

The CA acts as a *trusted third party*, or TTP. By signing the certificate, the CA is vouching for the fact it gave the corresponding private key to Alice. That is, the CA created a public and private key pair and it put the public key in Alice's certificate. Then the CA signed the certificate (using its private key) and it gave the private key to Alice. If you trust the CA, you believe that it actually gave the private key to Alice, and not to anyone else.

A subtle but important point here is that the CA is *not* vouching for the identity of the holder of the certificate. Certificates act as public keys and, consequently, they are public knowledge. So, for example, Trudy could send Alice's public key to Bob and claim to be Alice. Bob must not fall for this trick.

When Bob receives a certificate, he must verify the signature. If the certificate is signed by a CA that Bob trusts, then he uses that CA's public key for verification. On the other hand, if Bob does not trust the CA, then the certificate is useless to him. Anyone can create a certificate and claim to be anyone else. Bob must trust the CA and verify the signature before he can assume the certificate is valid.

But what exactly does it mean for Alice's certificate to be valid? And what useful information does this provide to Bob? Again, by signing the certificate, the CA is vouching for the fact that it gave the private key to Alice, and not to anyone else. In other words, the public key in the certificate is actually Alice's public key, in the sense that Alice—and only Alice—has the corresponding private key.

To finish beating this dead horse, after verifying the signature, Bob trusts that Alice has the corresponding private key. It's critical that Bob does not assume anything more than this. For example, Bob learns nothing about the

[14]This formula is slightly simplified. Actually, we also need to use a hash function when we sign, but we don't yet know about hash functions. We'll give the precise formula for digital signatures in the next chapter. Regardless, this simplified signature illustrates all of the important concepts related to certificates.

sender of the certificate—certificates are public information, so anyone could have sent it to Bob. In later chapters we'll discuss security protocols, where we will see how Bob can use a valid certificate to verify the identity of the sender, but that requires more than simply verifying the signature on the certificate.

In addition to the required public key, a certificate could contain just about any other information that is deemed useful to the participants. However, the more information, the more likely the certificate will become invalid. For example, it might be tempting for a corporation to include the employee's department and phone number in a certificate. But then the inevitable reorganization will invalidate the certificate.

If a CA makes a mistake, the consequences can be dire. For example, VeriSign[15] once issued a signed certificate for Microsoft to someone else [136], that is, VeriSign gave the private key to someone other than Microsoft. That "someone else" could then have acted (electronically, that is) as Microsoft. This particular error was quickly detected, and the certificate was revoked, apparently before any damage was done.

This raises an important PKI issue, namely, *certificate revocation*. Certificates are usually issued with an expiration date. But if a private key is compromised, or it is discovered that a certificate was issued in error, the certificate must be revoked immediately. Most PKI schemes require periodic distribution of certificate revocation lists, or CRLs, which are supposed to be used to filter out compromised certificates. In some situations, this could place a significant burden on users, which could to lead to mistakes and security flaws.

To summarize, any PKI must deal with the following issues:

- Key generation and management
- Certificate authorities (CAs)
- Certificate revocation

Next, we'll briefly discuss a few of the many PKI *trust models* that are used today. The basic issue is deciding who you are willing to trust as a CA. Here, we follow the terminology in [162].

Perhaps the most obvious trust model is the *monopoly model*, where one universally trusted organization is the CA for the known universe. This approach is naturally favored by whoever happens to be the biggest commercial CA at the time (currently, VeriSign). Some have suggested that the government should play the role of the monopoly CA. However, believe it or not, many people don't trust the government.

[15]Today, VeriSign is the largest commercial source for digital certificates [316].

One major drawback to the monopoly model is that it creates a big target for attack. If the monopoly CA is ever compromised, the entire PKI system fails. And if you don't trust the CA, then the system is useless for you.

The *oligarchy model* is one step away from the monopoly model. In this model, there are multiple trusted CAs. In fact, this is the approach that is used today—a Web browser might be configured with 80 or more CA certificates. A security-conscious user such as Alice is free to decide which of the CAs she is willing to trust and which she is not. On the other hand, a more typical user like Bob will trust whatever CAs are configured in the default settings on his browser.

At the opposite extreme from the monopoly model is the *anarchy model.* In this model, anyone can be a CA, and it's up to the users to decide which CAs they want to trust. In fact, this approach is used in PGP, where it goes by the name "web of trust."

The anarchy model can place a significant burden on users. For example, suppose you receive a certificate signed by Frank and you don't know Frank, but you do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should you trust Frank? This is clearly beyond the patience of the average user, who is likely to simply trust everybody or nobody so as to avoid headaches like this.

There are many other PKI trust models, most of which try to provide reasonable flexibility while putting a minimal burden on end users. The fact that there is no generally agreed upon trust model is itself one of the major problems with PKI.

## 4.9 Summary

In this chapter, we've covered most of the most important public key crypto topics. We began with the knapsack, which has been broken, but provides a nice introductory example. We then discussed RSA and Diffie-Hellman in some detail.

We also discussed elliptic curve cryptography (ECC), which promises to play an ever-increasing role in the future. Remember that ECC is not a particular type of cryptosystem, but instead it offers another way to do the math in public key cryptography.

We then considered signing and non-repudiation, which are major benefits of public key cryptography. And we presented the idea of a hybrid cryptosystem, which is the way that public key crypto is used in the real world for confidentiality. We also discussed the critical—and often confused—topic of digital certificates. It is important to realize exactly what a certificate does and does not provide. Finally, we took a very brief look at PKI, which is often a major roadblock to the deployment of public key crypto.