Network Security

Professor Gaurav S. Kasbekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay

Week - 05
Lecture - 27
Public Key Infrastructure: Part 2

Hello, we were discussing public infrastructure in the previous lecture; we will now continue the discussion. So, we first introduce some terminology in order to discuss public infrastructure in more detail. If Alice signs a certificate vouching for Bob's name and public key, then Alice is said to be the issuer of the certificate and Bob is the subject. That is, Alice issues a certificate to Bob, then in that case Alice is the issuer and Bob is the subject of the certificate. If some user Carol is checking whether a certificate issued to Bob is valid or not, then Carol is the verifier of the certificate.

So, we'll use these terms—issuer, subject, and verifier—in the following discussion. Another term we require is this: trust anchor. A trust anchor is a public key that the verifier has decided, through some means, is trusted to sign certificates. So, in the previous lecture, for example, the certificate of thawte Consulting was preloaded in a web browser. So, a user who has the certificate of thawte Consulting and trusts it.

We use thawte Consulting as a trust anchor. So, thawte Consulting was a trust anchor for those users. So, in general, a trust anchor is a public key that the verifier has decided, through some means, is trusted to sign certificates. Some examples are as follows: Suppose Carol's browser is preloaded with a certificate of the CA Symantec.

Then Symantec is a trust anchor for Carol. That is, Carol trusts certificates that are issued by the CA Symantec. Another example is this. Suppose Carol met Alice in person and Alice provided her public key to Carol; then Alice is a trust anchor for Carol. Since Carol trusts certificates issued using Alice's public key.

So, a public key may be a trust anchor for a particular user in different ways. Some examples are shown here. So, in any case, a trust anchor is a public key which a verifier trusts. So, we'll use this terminology in the following discussion. We now discuss different models for public key infrastructure and the pros and cons of each model.

One simple model is the monopoly model, where the entire world chooses one organization which is universally trusted by all organizations, countries, people, and so on to be the single CA for the world. And the public key of that monopoly organization is embedded in all software and hardware as the PKI trust anchor. So, this monopoly organization is the only one which issues any certificates and everyone trusts certificates issued by this organization. Everyone needs to obtain certificates from this CA. But this monopoly model has clearly several disadvantages, which are as follows:

One disadvantage is that there is no one universally trusted organization. For example, consider users in three nations: India, China, and the US. So, users in India may not trust a CA from China or the US, and vice versa. There isn't a universally trusted organization. So, different groups of people trust different CAs.

So, there isn't any organization which can serve as a common CA for everyone in the world. Another disadvantage of the monopoly model is this: since all hardware and software would come pre-configured with the monopoly organization's public key, it would be infeasible to change the key in case it were compromised. So, all hardware and software are pre-configured with the monopoly organization's public key, and what happens if the corresponding private key is leaked? In that case, the public key would have to be changed in all software and hardware in the world, which would be infeasible. So, hence, this is another disadvantage.

It's infeasible to change the key in case this monopoly organization's private key is leaked. It would be expensive to have a remote organization certify a person's public key. So, for example, assume that the monopoly organization is located in the US, and some user in India wants to obtain a certificate for their public key. In that case, how does the CA in the US reliably check the Indian person's credentials? The public key of the subject, who is from India, must be securely communicated to the monopoly organization.

So, this is another challenge. So, if we have only one organization that issues certificates to everyone in the world, then reliably checking the person's credentials is difficult, and also reliably communicating the public key of the subject to whom the certificate is to be issued. Securely communicating this public key to the monopoly organization is also challenging. So, this is another disadvantage of the monopoly model. Then another disadvantage is that once enough hardware and software were deployed, which was embedded with the monopoly organization's public key, then it would be difficult for the world to switch to another CA.

The monopoly organization could then take advantage of this fact and could charge very high rates to issue certificates. So, this is a common disadvantage in a monopoly. So, the monopoly organization can charge very high rates for issuing certificates. So, this problem wouldn't be there if there were multiple competing CAs. Another disadvantage is that if the monopoly organization had an incompetent or corrupt employee, then the security of the entire world would be compromised because the employee may be tricked or bribed into issuing bogus certificates or disclosing the CA's private key.

So, if the CA's private key were leaked, then the security of the entire world would be compromised. So, the security would rest very critically on the security of the private key, and even one corrupt or incompetent employee may compromise the corresponding private key, and hence this is not a very secure solution. In this way, the monopoly model has several disadvantages, so it is not used in practice. An improvement upon the monopoly model is, we still have a monopoly that issues certificates, but now additionally we have registration authorities, or RA. It is similar to the monopoly model except that the single certification authority, which is the monopoly CA, it chooses some other organizations called registration authorities, or RAs, to securely check identities and vouch for the public keys of a subject.

But the certificates are still issued by the CA, which is the monopoly CA. Each RA securely communicates with the CA. The CA issues a certificate to a subject, which is vouched for by the RA since the CA trusts the RA. So, as an example, suppose there is a monopoly CA which is located in the US. There may be one registration authority in India, and the registration authority in India checks the identities and vouches for the public keys of subjects located in India.

And then the monopoly CA located in the US issues certificates to those subjects based in India. So, this makes it easier to verify the identities of subjects to whom public keys are issued. This model has the following advantages over the monopoly model. It is more convenient and secure to obtain certificates because there are more places to go to for getting a certificate. So, in our example, the monopoly CA was located in the US, but there were many registration authorities, possibly in different countries and states, and so on.

And one could just visit a local registration authority and get their credentials verified. Subsequently, the monopoly CA would issue a certificate to the subject. But the other disadvantages of the monopoly model also apply to this model. So, we discussed several other disadvantages as well for the monopoly model. So, all of them apply to this model.

So, you can verify for yourself that those other disadvantages are still applicable to the monopoly and registration authorities model. So, this model is also not used in practice. Another model is delegated CAs. So, here one CA issues certificates to other CAs, vouching for their public keys and vouching for their trustworthiness as CAs. So, there is one CA, which is a root CA, and it issues certificates to some other CAs, vouching for their public keys as well as for their trustworthiness as CAs.

And then these CAs to whom the certificates have been issued, they can act as certificate-issuing authorities, and they can vouch for the public keys and trustworthiness of other CAs, and so on and so forth. Users obtain certificates from one of the delegated CAs instead of having to go to the delegating CA, and there can be multiple levels of delegation. So, there can be one root CA which issues certificates to some CAs, and then those CAs to whom certificates were issued, they issue certificates to yet other CAs, and so on and so forth. So, we can have multiple levels of delegation in this manner. So, there's a difference between delegated CAs and registration authorities, which we discussed on the previous slide.

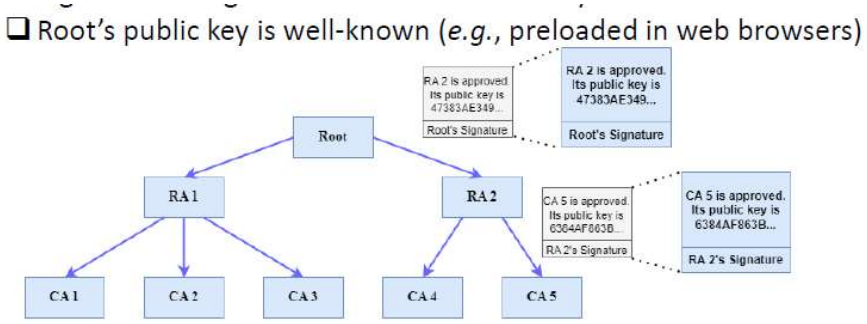
Delegated CAs issue certificates. RAs don't issue any certificates. RAs just check the credentials of the subject, and they say that so and so subject is legitimate, so you can issue a certificate to so and so subject. But RAs themselves do not issue any certificates. In contrast, delegated CAs issue certificates.

As an example, suppose Alice is verifying Bob's certificate. Then, in the latter case, Alice checks only one certificate. That is, when there are registration authorities along with the monopoly CA, in that case, Alice checks only one certificate, which is issued by the monopoly organization. But in the case where there are delegated CAs, Alice checks a chain of certificates. It checks whether the delegated CA is legitimate using the certificate issued to the delegated CA.

And then it checks whether the issuer is legitimate by checking their certificate issued to them by their parent in this hierarchy, and so on and so forth. So, an example will make this clearer. We'll see an example of a situation where a chain of certificates is verified by a verifier to check someone's certificate. Here's the example. So, there is a top-level CA called the root CA, which certifies some second-level CAs, which are regional authorities.

So, these are the regional authorities. There are two regional authorities, RA1 and RA2, in this example. So, this root issues certificates to RA1 and RA2, binding their public keys to

their names and addresses. So, this is a certificate issued by the root to RA2. And similarly, the root issues a certificate to RA1.



One important thing is that we should not confuse regional authority with registration authority, which we discussed in the previous model. These arrays are certification authorities, so they are delegated certification authorities. But they operate over a specific region, so they are named as regional authorities. An example is as follows: Suppose Alice wants to communicate with Bob and receives a certificate containing his public key signed by CA5.

Now, these RAs, RA1 and RA2, have issued certificates to some other delegated CAs lower down in the hierarchy. So, RA1 has issued certificates to CA1, CA2, and CA3, which are delegated CAs. And RA2 has issued certificates to CA4 and CA5, which are yet other delegated CAs. Now, when Alice wants to communicate with Bob, she receives a certificate containing his public key signed by this CA5. So, in this case, Alice uses a chain of certificates to verify the legitimacy of Bob.

Specifically, Alice verifies that Bob is legitimate using the certificate issued by CA5 to Bob. Then Alice verifies that CA5 is legitimate using the certificate issued to CA5 by RA2. That certificate is shown here. This is the certificate issued to CA5 by RA2, and this is used by Alice to check the legitimacy of CA5. But then Alice has to check the legitimacy of RA2.

That verification is done by using the certificate that is issued to RA2 by the Root. So, Alice verifies that RA2 is legitimate using the certificate issued to RA2 by the root, and that certificate is shown here. Finally, how does Alice verify that the root is legitimate? The root's public key is well known. For example, it may be preloaded in web browsers.

So, that way Alice is able to verify the legitimacy of the Root. So, in this model of delegated CAs, a verifier is able to verify the certificates of a large number of users using this chain of certificates. So, as in this example where Alice could verify the legitimacy of Bob by taking a chain of certificates starting from the root, then RA2, then CA5, and then finally Bob's certificate. So, using such a chain of certificates, one can verify the legitimacy of a particular user. So, this is a very convenient architecture for providing certificates.

Next, we have the Oligarchy model. This is a model commonly used in web browsers, such as Firefox and Chrome. In this model, the web browser comes pre-configured with certificates containing the public keys of multiple CAs or roots. So, in this model, we have such a hierarchy, but there isn't one such hierarchy; there are many such hierarchies, each hierarchy with its own root. So, there are multiple root CAs, and there is a separate hierarchy rooted at each root CA.

So, the certificate signed by any of these roots or a chain of certificates rooted at any one of these CAs is accepted. It is also possible for a user to add or delete CAs or roots to or from the pre-configured list. So, if a user Alice trusts one CA, that is, trusts one root CA which is not in the list, then she can add that root CA to the list. Conversely, if there is a particular root CA in the list but Alice does not trust that root CA, then Alice can remove that CA's name from the list. This has several advantages over the monopoly model.

There need not be an organization or root CA that is trusted by all organizations and individuals. So, if a user does not trust certain root CAs, then it will not trust certificates issued by that root CA or a chain of certificates rooted at that root CA. But they can always use certificates issued by other root CAs. So, there need not be any one organization or root CA that is trusted by all organizations and individuals. It is also easier for an individual or organization to obtain a certificate.

Here, there are a large number of CAs which can issue certificates. So, it's easier for an individual or organization to obtain a certificate. Another advantage is that there isn't one monopoly CA. Instead, the organizations chosen as roots will be in competition with each other. So, there will not be monopoly pricing.

So, one of the root CAs cannot charge very high rates for issuing certificates because, in that case, users will opt for other root CAs. But this also has some disadvantages. One disadvantage is that the set of CAs or roots whose public keys are included in the browser is selected by the vendor of the browser, not by the user. And the vendor may sometimes

use inappropriate criteria for the selection. For example, the vendor might include all CAs as roots who pay for being included.

A vendor of a particular browser includes a list of CAs, which are roots, but all of the root CAs may not be legitimate. Some CAs who just paid the vendor high rates, they may be issued certificates by the vendor if the vendor is not doing this verification in a correct manner. So, depending on who the vendor is, the list of root CAs may not always be reliable. Another disadvantage is that browsers today come pre-configured with public keys of more than 80 roots. Even for a knowledgeable user, it is impractical to examine this entire list and see if it has been tampered with, requires modification, and so on.

So, this oligarchy model can become unwieldy because there can be a large number of root CAs, and it's difficult for a user to check whether all the root CAs are reliable or not. So, to examine this list, which may contain more than 80 roots, it's difficult even for knowledgeable users. So, that's another disadvantage of the oligarchy model. We now discuss another model, that is the anarchy model. Here, each user is responsible for configuring some trust anchors.

And the user may use any process for configuring their trust anchors. One example is the public keys of people he or she has met who have handed them a business card with a hash value of the public key and sent an email containing a public key; that hash value may be the trust anchor for a particular person. So, for example, Alice might meet Bob, and Bob provides his public key by email, and for Alice to verify whether the public key has been tampered with, Bob physically hands over a card with the hash value of the public key, and then Alice can verify the hash of the public key, whether it matches the hash value on the card. So, using this process, Alice may configure Bob as a trust anchor. Anyone can potentially sign certificates for anyone else, hence the name anarchy model.

Some organizations, such as MIT, volunteer to maintain a certificate database into which anyone can deposit certificates. So, this is a database into which there are a lot of certificates, and one can use this database to find users and their public keys. So, to obtain the public key of someone, say Alice, you can search through the public database to see if you can find a path that is a chain of certificates from one of your trust anchors to Alice. For example, suppose you want to obtain the public key of Alice, but you don't know the public key of Alice, and you want to obtain the public key of Alice, but then you search this database and you find that Bob has issued a certificate to Alice, and Carol has issued a certificate to Bob, and Carol is a trust anchor for you. So, in this case, through this chain

of certificates, Carol providing a certificate to Bob and Bob providing a certificate to Alice, you can obtain the public key of Alice.

So, this is one example of the anarchy model. But this has some shortcomings. One is it has scaling problems. This public database would get extremely large if deployed on an internet scale. For example, if every user donated just 10 certificates on average, then the database would consist of billions of certificates, and it would be difficult to search the database and obtain a chain of certificates for a particular person whose public key we require.

It may be impractical to search through the database and construct paths. Some malicious individuals may add bogus certificates, making it difficult to know whether to trust the path. So, there are billions of certificates in this database, and a lot of them are bogus certificates added by malicious individuals. So, it is difficult to know whether to trust a particular path. So, these are the disadvantages of the anarchy model.

Next, we study the concept of certificate revocation. So, we have seen how to issue a certificate to a particular entity, but if that entity is no longer trusted, in that case, we should be able to revoke that certificate. So, we now study the process of certificate revocation. So, if someone realizes that their private key has been stolen, or an employee who knew an organization's private key left the organization, in that case, we require some way to revoke their certificate. So, if Alice realizes that her private key has been stolen, then the certificate which binds Alice to that public key should be revoked.

Similarly, if there was an employee in an organization who was trusted to sign documents on behalf of their organization, if that person left the organization, then the certificate binding their public key to their name, that certificate should be revoked because the employee should no longer be able to sign documents on behalf of the organization. So, that particular certificate should be revoked. Certificates have expiration dates on them. So, we discussed the format of certificates in the previous lecture. So, we discussed that certificates have expiration dates on them.

But the validity period of a certificate is usually several months. So, it's insecure to wait until the certificate expires. So, for example, if a private key is stolen now and the certificate will only expire after, say, eight months. In that case, the insecure certificate will be in the network for eight months. So, it is insecure to wait for such a long time until the certificate expires.

So, we require another mechanism for revoking certificates, which is faster. We shouldn't have to wait until the expiration date of the certificate. A mechanism for a CA to revoke certificates that it earlier issued is the following: The CA periodically issues a so-called Certificate Revocation List (CRL). This is a signed and timestamped list of all certificates that have been revoked.

So, if Alice wants to verify Bob's certificate, she checks whether Bob's certificate is included in the latest CRL. If yes, then Alice does not trust that certificate because it has been revoked. So, this certification revocation list is a list of certificates that have been revoked, and it has a timestamp on it, which indicates from what date and time the certificate has been revoked. So, by consulting this certification revocation list, we can find out which certificates are no longer valid. Now, assuming that CRLs are periodically issued, why do certificates have expiration dates at all?

So, when a certificate is issued, why doesn't the issuer issue it with permanent validity? So, in case it needs to be revoked, we can always include that certificate in a certificate revocation list. So, why have expiration dates on certificates? So, the reason is to keep the sizes of CRLs small since expired certificates need not be included in CRLs. If there are no expiration dates on certificates, then the CRL would need to include all the certificates that are invalid up to now.

So, the list would grow with time and it would be very large after a certain point. So, since certificates have expiration dates on them, expired certificates need not be included in the CRLs. So, that keeps the sizes of CRLs small. So, for this reason, there are expiration dates on certificates. To make the concept of certificate revocation lists more efficient, we have the notion of delta CRLs, which is as follows:

It is intended to make CRL distribution more efficient. Here's an example. Suppose we want revocation to take effect within one hour. Then a CA would have to post the certificate revocation list every hour. Every verifier would need to download the latest CRL every hour.

And as an example, suppose the CRL suddenly became very large because the company laid off some 10,000 people, and all of them had a certificate. All of these 10,000 people are no longer authorized to sign documents on behalf of that company. So, all their certificates need to be revoked. So, the certificates of all these 10,000 people are included in the CRL. Every verifier would have to download a huge CRL because every CRL would need to contain the names of the certificates of these 10,000 people.

So, every hour, the CRL is issued with the same information. The certificates corresponding to these 10,000 people were revoked at a particular point in time. But then every hour, the CRL would have to contain the names of these 10,000 people until these certificates expired. So, delta CRLs is a notion that can be used to make this process more efficient. In this example, full CRLs, which contain the list of all the revoked certificates, are posted much less frequently than once every hour.

For example, full CRLs may be issued once every day or once every week, and so on. But another kind of CRL, called a delta CRL, is posted once every hour. It lists the changes from the last full CRL. Here is an example of a delta CRL. Delta CRL says the following is a list of all the certificates that have been revoked since Feb 7th, 10 a.m., which is the most recent full CRL.

So, the most recent full CRL was issued at 10 a.m. on Feb. 7th, and periodically after that, we issue delta CRLs, which provide a list of certificates that have been revoked since that time. Delta CRLs would typically be very short, often containing no certificates. So, if no certificates have been revoked since the last full CRL, then the delta CRL would be empty. The CA does not need to post the full CRL, which is often very large, frequently. It only needs to be posted, say, once every week or once every day, and so on.

And the delta CRLs can be posted more frequently, such as once every hour. Instead, CAs can post a full CRL in place of a delta CRL when the delta CRL gets sufficiently large. Every verifier has to download the latest full CRL and the latest delta CRL. The full CRL will contain a list of revoked certificates, and the latest delta CRL indicates the revoked certificates after the full CRL was issued. So, by combining these, the verifier gets the list of all revoked certificates and can use this information to find out which certificates are valid and which certificates have been revoked.

So, this process of delta CRLs can be used to make certificate revocation more efficient. This concludes our discussion of public key infrastructure. We discussed the notion of certificates, and we discussed different public infrastructure models, including the monopoly model, delegated CAs model, oligarchy model, anarchy model, and so on. We discussed the pros and cons of different models, and we discussed the concept of certificate revocation, including certificate revocation lists and delta CRLs. Thank you.