Network Security

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

Week - 05
Lecture - 26
Public Key Infrastructure: Part 1

Hello, in earlier lectures, we discussed public key cryptography, but in order to use public key cryptography in a practical scenario, we require what is known as public key infrastructure. We will discuss this in the present lecture and in the next lecture. So, suppose there are two users, Alice and Bob, who don't know each other, and Alice wants to send an encrypted message to Bob. The question is, how does Alice get Bob's public key? So, one obvious solution that may occur to us is the following:

Bob just puts his public key on his website, and Alice connects to Bob's website and downloads his public key from the website. But is this a secure way of distributing a public key from a user Bob to a user Alice? So, this has a shortcoming. It is vulnerable to an attack such as the following: This attack is illustrated in this figure here.

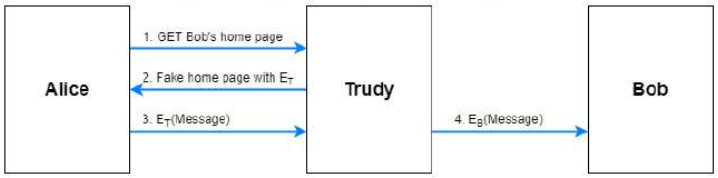
So, Alice is the user who wants to obtain Bob's public key. Alice sends a request to Bob's webpage asking for Bob's homepage, which includes his public key. So, this is the message from Alice: "GET Bob's homepage". But this message of Alice is intercepted by an intruder, Trudy, who controls a compromised router on the path from Alice to Bob. So, this message, "GET Bob's homepage", is intercepted by Trudy.

And instead of sending Bob's actual webpage, Trudy sends a fake homepage to Alice. This fake homepage contains a fake public key. So, this fake homepage that Trudy sends to Alice may be a copy of Bob's webpage with Bob's public key replaced with Trudy's public key. And Trudy sends this fake webpage to Alice. We denote Bob's true public key by E_B, and Trudy's public key is E_T.

So, Trudy sends a fake webpage to Alice, which contains E_T, that is Trudy's public key. Now Alice thinks that the public key that she has obtained is that of Bob. She then tries to send an encrypted message to Bob, encrypted with the public key that she has obtained, which is E_T. So, when Alice encrypts her message with this public key E_T, Trudy is able to

decrypt it because Trudy has the private key corresponding to E_T, and Trudy decrypts and reads the message. After Trudy decrypts the message, she obtains Bob's actual public key, that is E_B, and then re-encrypts the message using E_B and sends E_B (message) to Bob.

- Want a mechanism using which Alice can check whether a public key that is supposed to be Bob's is indeed that of Bob
- **Public key infrastructure (PKI)**: components (e.g., organizations, policies, procedures) used to securely distribute public keys



So, this is the message encrypted using Bob's public key and sent to Bob. But in this process, confidentiality has been lost because Trudy was able to read the message that Alice sent to Bob. So, in this attack, the confidentiality of the message is lost. So, that defeats the purpose. This happens because an intruder, Trudy, is able to replace Bob's true public key with a fake public key, that is the public key of Trudy.

So, this is one illustration of the fact that it is difficult to check in general whether a public key is indeed that of the user it is claimed to be. So, we want a mechanism using which Alice can check whether a public key that is supposed to be user Bob's is indeed that of Bob or someone else. Public key infrastructure refers to different components that are used to securely distribute public keys. These components include organizations, policies, procedures, and so on. We'll discuss public key infrastructure in this lecture and the next lecture.

So, first we discuss one approach to public key distribution based on a key distribution center or KDC. Suppose a key distribution center, or KDC, is available online for 24 hours every day for securely distributing public keys on demand. So, when a user, Alice, wants to obtain Bob's public key, she connects securely to the KDC and obtains Bob's public key from the KDC. The public key of the KDC is known to everyone. For example, it may be preloaded in browsers.

So, everyone is able to securely communicate with the KDC. If a user Alice wants to know the current public key of Bob, she just connects to the KDC and requests it. And if some user wants to change their public key, then they have to just inform the KDC. And the new key is stored at the KDC. But this has some disadvantages.

Each time some user wants to obtain the public key of some other user, they have to connect to the KDC. So, this approach is not scalable. Every user in the network has to connect to the KDC. So, in a large network such as the internet, there is a performance bottleneck. So, this approach is not scalable.

Every user needs to connect to the same point, that is, the KDC, to obtain the public key of any user. The KDC can become a performance bottleneck since all users connect to the KDC whenever they want to obtain a public key. So, this is a shortcoming of this approach. Then another shortcoming is that if the KDC fails, for example, it crashes, then users are not able to communicate securely. So, the KDC has to be online 24/7, and whenever the KDC is down for some reason, users are not able to obtain public keys of other users.

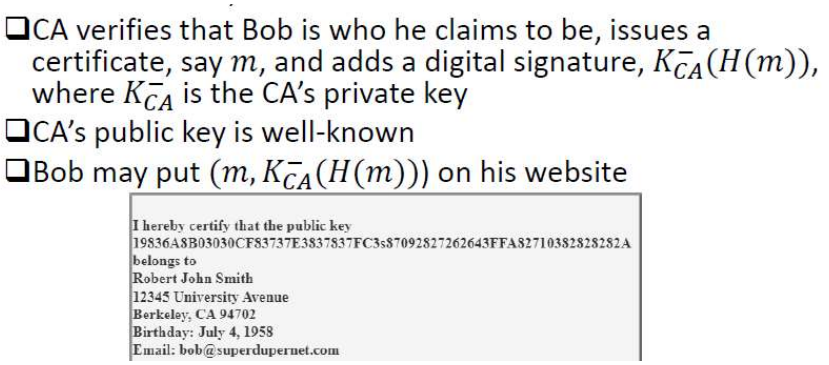
So, these are the disadvantages of this approach, and these disadvantages stem from the fact that there is an entity, the KDC, which needs to remain online all the time, and whenever any user wants to obtain a public key, they need to connect to that entity, the KDC, which is online. So, because of these disadvantages, people have developed a different solution for public key distribution. This alternative solution does not require any organization, such as a KDC, to be online. It is based on an alternative concept called certificates. There are organizations that issue certificates stating that public keys belong to specific persons, organizations, and so on.

So, a certificate is a digital document. It is issued to a person. For example, a certificate might be issued to Bob. That certificate states that Bob has a specific public key. Bob can pass the certificate to any user who wants to communicate with him.

This certificate has built-in security features. A user, Alice, does not need to connect to an entity like the KDC to obtain a public key. The certificate itself has inbuilt security mechanisms. We will discuss the concept of a certificate in detail. A certification authority is an organization that issues a certificate that a public key belongs to a specific person, organization, and so on.

So, a certificate is a digital document that binds a public key to a specific person or organization. And certification authority is an organization which issues a certificate. As an example, suppose Bob wants a certificate for his public key. To obtain the certificate, Bob goes to a certification authority with his public key along with some documents to prove that he is indeed Bob. These documents might be his passport, driver’s license, and so on.

So, Bob goes with these documents to the CA, who can issue a certificate. The CA performs the verification. The CA verifies that Bob is indeed who he claims to be. And then, if the verification is successful, the CA issues a certificate to Bob. Let's call this certificate m. And for security, the CA adds a digital signature, $K_{CA}^{-}(H(m))$, where K_{CA}^{-} is the certification authority's private key, and H(m) is a cryptographic hash function.



So, the digitally signed certificate is m, $K_{CA}^{-}(H(m))$. This is the same mechanism for creating a digital signature, which we discussed in an earlier lecture. For creating a digital signature on a document m, one has to find out $K_{CA}^{-}(H(m))$, where K_{CA}^{-} is the private key of the signer. So, the certificate is created, and it says that Bob has so and so public key, and then the corresponding digital signature is created, so that someone can check that this certificate was indeed created by the certification authority. So, we should note that this verification process has to be reliable; that is, the CA should reliably check that Bob is indeed who he claims to be, and only then issue a certificate. So, if the CA does not do this verification properly—for example, if the CA issues a certificate to anyone who pays them for getting a certificate—in that case, this certificate can be obtained by fraudulent users.

So, this certificate concept assumes that the certification authority is an honest and competent organization which issues certificates only to genuine parties. This shows the certificate. So, this is just for illustration. We'll see the actual format of a certificate later. But this certificate says that I hereby certify that the public key such and such belongs to Robert John Smith.

This is the official name of Bob, his address, and his email address. So, this certificate binds Bob's public key, which is this, to his name and address. And this part of the certificate is $K_{CA}^{-}(H(m))$. It is the SHA-1 hash of this certificate signed with the CA's private key. This first part is m, which is the certificate that binds the public key to the

name and address of Bob. And this is the digital signature part, the $K_{CA}^{-}(H(m))$. This is an example of a certificate.

Now, how does some entity, say Alice, verify whether this certificate is genuine or not? The CA's public key is well known. For example, it may be preloaded in browsers. So, Alice knows the public key of the CA, that is, K_{CA}^{+}. So, Alice is able to apply K_{CA}^{+} to this digital signature and obtain H(m), then check whether H(m) is equal to the hash of the message m. If yes, then the verification is successful.

Now, once Bob obtains his certificate from the CA, Bob can just put $\left(m, K_{CA}^{-}(H(m))\right)$ on his website. And the certificate itself has inbuilt security mechanisms; that is, to be precise, the security mechanism is this, $K_{CA}^{-}(H(m))$, which is the digital signature. So, anyone can download this certificate, which includes the digital signature, and find out the public key of Bob. So, now let's go back to the attack that we discussed earlier, which is summarized in these bullets. Alice wants to obtain the public key of Bob, and the public key of Bob is posted on his website, but an interceptor, Trudy, intercepts Alice's request and sends a fake homepage with Bob's public key replaced with her own public key.

Now, let's see whether a certificate can defend against this attack. Assuming that Alice knows the CA's public key, does this mechanism of a certificate prevent this attack, which is described here? It does defend against this attack. Let's think of a few ways that Trudy might tamper with Bob's public key. Suppose when Trudy intercepts Alice's get request, she sends the fake homepage of Bob with her own certificate signed by the CA.

So, then in this case, Trudy replaces Bob's certificate with her own certificate. So, will this attack work? So, in this case, this attack fails because Alice can just read the certificate. This certificate says that the public key of so and so entity is so and so. So, when Alice reads the certificate, she sees that Trudy's name is on the certificate.

The certificate belongs to Trudy. Since the certificate binds the public key to the name and address of the user, Alice can read the certificate and see that Trudy's name is in it. So, she can just refuse to accept the public key. So, Alice knows that it is Trudy's public key and not Bob's public key, so this attack fails. Trudy is not able to replace the certificate with her own certificate.

So, this first attack fails. Let's discuss another possible attack. Suppose Trudy modifies Bob's certificate by replacing his public key with her own public key to get m'. So, in Bob's certificate, the certificate says the public key of Bob is so and so. Trudy modifies

this certificate to make it say, the modified certificate says the public key of Bob is so and so, where that public key is actually Trudy's public key.

So, the modified message is m', and Trudy sends $\left(m', K_{CA}^{-}\left(H(m)\right)\right)$ in place of $\left(m, K_{CA}^{-}\left(H(m)\right)\right)$, where m is the actual certificate. So, will this attack work? Will this succeed in confusing Alice? So, this attack also fails because when Alice applies K_{CA}^{+} to the digital signature, then $K_{CA}^{+}\left(K_{CA}^{-}\left(H(m)\right)\right) = H(m) \neq H(m')$. So, this verification fails.

2) Suppose Trudy modifies Bob's certificate by replacing his public key with her own to get m'; sends $(m', K_{CA}^{-}(H(m)))$ in place of $(m, K_{CA}^{-}(H(m)))$
   - ❑ attack fails since $K_{CA}^{+}(K_{CA}^{-}(H(m))) \neq H(m')$

So, hence Alice is able to find out that the certificate has been tampered with. Now notice that the crucial fact here is that a cryptographic hash function has the property that if you have a message m with a certain hash H(m), then it's computationally infeasible to find another message m' which has the same hash H(m). So, hence, it's difficult for Trudy to replace the message m with another message m' with the same hash as the original message has. So, hence this verification of the digital signature fails and Alice is able to detect that the certificate has been tampered with. So, any tampering in the certificate can be detected by checking the digital signature on the certificate.

So, because of these reasons, a certificate is a reliable mechanism for checking whether a public key is indeed that of the user who the certificate is claimed to be of. Now recall that the signed certificate is $\left(m, K_{CA}^{-}\left(H(m)\right)\right)$,. The user needs to trust the CA's public key for this scheme to work. So, the signature is $K_{CA}^{-}\left(H(m)\right)$, and the verifier must know the corresponding public key, K_{CA}^{+}, which is the public key of the CA. So, one way is that software such as browsers, for example Firefox and Chrome, are preloaded with a set of trusted CA certificates. The CA certificate contains the CA's name and public key.

So, using these trusted CA certificates, users are able to obtain the public keys of certain reliable CAs. Then, they can trust certificates issued by those CAs. CA certificates can also be added or removed by a user. So, for example, if I don't trust a particular certification authority, then I can remove the certificate of that CA from my browser. If I trust a certain CA, then I can add their certificate to my browser, and so on and so forth.

Some examples of popular certification authorities are Symantec, Comodo, GoDaddy, and GlobalSign. So, there are a large number of such certification authorities who provide certification services on the internet. Now it would be problematic if different certificates

were in different formats. Browsers would need to understand all the formats in order to parse a certificate. So, we require standardization.

We require a standard format for a certificate. ITU X.509 is a widely used standard that specifies the format of certificates. This table shows some of the fields in a certificate. The first column has the field name, and the second column has the description. Some of the fields are shown here.

One is the version, that is the version number of the X.509 specification, then the serial number. The CA issues a large number of certificates typically, and this serial number is an identifier for a certificate. One certificate might be numbered N, another, the next one will be numbered N+1, and so on. This is the serial number of the certificate. Then there is the signature, which specifies the algorithm used by the CA to sign the certificate.

For example, it may be SHA1 hash function along with the public key algorithm RSA and so on and so forth. Next, we have the issuer name. That is the identity of the CA that issues this certificate in a particular format known as distinguished name format or DN format. We'll see examples of the issuer name in distinguished name format. Then the validity period of the certificate, that is, when the certificate is valid from and up to what time is it valid.

So, the start and end of the period of validity of the certificate. This is another field in the certificate. Then the subject name. The subject is the entity to whom the public key is issued. So, the subject name is the identity of the entity whose public key is associated with this certificate.

Again, the subject name is in the same format as the issuer name in distinguished name format. And then finally, we have the subject public key. This has the subject's public key as well as an indication of the public key algorithm and algorithm parameters to be used with this key. So, if, for example, the algorithm is RSA, then the public key is (N, E). It has the numbers (N, E). So the subject's public key has (N, E) as well as the indication of the public key algorithm, which is RSA in this example. This shows an example of a certificate issued to the website www.freesoft.org by the certification authority Thawte Consulting.

So, this certificate has the following information. One is the version number, then the serial number of the certificate, then the signature algorithm is MD5 with RSA encryption. So, MD5 is the cryptographic hash function, and RSA is the public key algorithm. So, recall

that the signature, the digital signature, is $K_{CA}^{-}(H(m))$. So, that function H is MD5, and the algorithm used for creating the digital signature is RSA.

This is the issuer name in distinguished name format. So, you can check this issuer name. The issuer is Thawte Consulting, and these are some other details about Thawte Consulting, the address, and so on. Then the validity is from July 9th, 4:04 p.m., 1998, to July 9th, 4:04 p.m., 1999. So, the validity is for one year.

And it is issued to this subject, freesoft.org. So, this is the subject name in distinguished name format. And this is the public key information. The public key algorithm is RSA encryption. The public key is given here.

So, recall that the public key is (N, E), where N is a large integer. So, this is the value of N. N is called the modulus. So, this is the value of N in hex format. And this is the exponent, that is E. So in (N, E), the second part E is this exponent that is given here. The signature algorithm is MD5 with RSA encryption.

So, this is the digital signature of the certificate. So, this is used by the verifier to check whether there has been any tampering in this certificate. So, this is an example of a certificate. Now, to verify this certificate, we need the certificate of the CA Thawte Consulting. So, this certificate is issued by Thawte Consulting to freesoft.org, but how do we get the public key of the CA Thawte Consulting itself?

So, this example shows the certificate of the CA Thawte Consulting. So, here the other fields are similar. This is a newer certificate; the version is three, the serial number is one, the signature algorithm is MD5 with RSA encryption, and the issuer is Thawte Consulting. Note that the subject is also Thawte Consulting. So, this certificate is self-signed, that is, it is signed by the issuer itself. This certificate is issued by Thawte Consulting to itself.

So, such certificates are preloaded in web browsers. Also, note that the validity is for a much later period than the previous certificate had. It is from August 1st, 1996, to December 31st, 2020. The previous certificate was for one year, whereas this is for 24 years. So, by looking at this certificate, one is able to obtain the public key of Thawte Consulting, and such certificates are preloaded in web browsers.

So, when one installs an operating system along with a browser, such certificates are included in the browser. So, from such certificates, one is able to obtain the public keys of certain CAs, such as Thawte Consulting, and once the public keys of these CAs are obtained, we can verify the public keys of other entities, namely other people and

organizations, using this public key. This is a so-called root certification authority whose certificates are preloaded in web browsers. So, we started our discussion of public key infrastructure in this lecture. We first attempted to provide public key infrastructure using a key distribution center.

Then, we discussed its disadvantages. And then we studied the concept of a certificate and certification authorities. We discussed the format of a certificate. We'll continue with our discussion of public key infrastructure in the next lecture. Thank you.