**Network Security**

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

**Week - 03**

**Lecture - 18**

**Message Integrity, Cryptographic Hash Functions and Digital Signatures: Part 3**

Hello, recall that in the previous few classes, we discussed message integrity and cryptographic hash functions. We now discuss another related concept, that is, digital signatures. So, digital signatures are similar to manual signatures, which we are all familiar with. Recall that manual signatures are extensively used on various documents, such as checks, credit card receipts, legal documents, letters, and so on. So, a person makes a manual signature to indicate that he/she created a document or agrees with or acknowledges its contents.

A digital signature is used to achieve the same objectives for documents in digital form. So, a digital signature is the digital counterpart of a manual signature. So, what properties must a digital signature satisfy? Similar to a manual signature, a digital signature must be verifiable and nonforgeable. So, what do we mean by verifiability and nonforgeability?

By verifiability, we mean that it must be possible to prove that a person's signature on a document is indeed that person's signature. So, for example, if Alice creates her digital signature, then it should be possible for Bob to verify that this is indeed Alice's digital signature and not someone else's. So, verifiability means that it should be possible for a third person to verify that a particular digital signature is that of the person who claims it to be their digital signature. And nonforgeability means that no one should be able to create a person's digital signature except the person himself or herself. For example, only Alice should be able to create Alice's digital signature; some other person, say Trudy, should not be able to forge Alice's digital signature.

So, this is nonforgeability. So, notice that a manual signature must also satisfy these properties: verifiability and nonforgeability. And likewise, a digital signature should also satisfy these properties. So, let's discuss one attempt to create a digital signature. Recall that we discussed message authentication codes (MACs) in the context of message integrity.

Let's try a similar technique to create a digital signature. To sign a message m, Bob appends a field that is similar to a message authentication code to the message. That is, first Bob concatenates m and s, where s is a secret bit string that only Bob knows, to get this (m, s) string, and then computes its cryptographic hash function to get H(m,s). The signed document is then the original message along with H(m,s). So, (m, H(m,s)) is the signed document. Does this scheme achieve the objectives of a digital signature?

- concatenates $m$ and $s$, where $s$ is a secret bit string *that only Bob knows*, to get $(m, s)$; computes $H(m, s)$
- $(m, H(m, s))$ is the signed document

So, this scheme does not achieve the objectives of a digital signature. In particular, to create this digital signature, the creator needs to know the secret s, which only Bob knows. So, no one other than Bob can create this digital signature of Bob. So, the signature is nonforgeable. So, it satisfies one property of the digital signature.

But it does not satisfy the property of verifiability. Because some other party, say Alice, cannot verify whether this signature is indeed that of Bob. To verify that this signature is indeed that of Bob, the secret s is required, but only Bob knows that secret s. So, it is not possible for some third party, say Alice, to verify that this is indeed Bob's digital signature. So, in summary, this scheme achieves the property of nonforgeability but it does not achieve the property of verifiability. So, hence it's not a valid digital signature.

Let's try to modify this version to create another scheme for creating a digital signature. Now, we have the same scheme except that another user, say Alice, knows the secret s. So, Bob uses the same scheme as above to create a digital signature, but now the secret s is known to Bob as well as Alice. So, does this modified version achieve the objectives of a digital signature? So, in this case, we have already mentioned that to verify this signature, one needs to know the secret s. Only Bob and Alice know the secret s, so this signature is verifiable only by Alice.

But we require that the signature should be verifiable by anyone. So, that property is not satisfied by this scheme. Here only Alice is able to verify the signature, whereas we want the scheme to be verifiable by anyone. So, this verifiability property fails. And also since Alice knows the secret s, Alice can also create this digital signature, (m, H(m,s)).

Someone other than Bob can create this digital signature. So, the signature is forgeable. Alice can forge the digital signature of Bob. Hence, this scheme achieves neither verifiability nor nonforgeability. So, this scheme also fails.

We want an alternative scheme for implementing a digital signature. So, we discussed two different schemes for creating a digital signature. The first scheme operates as follows. Recall that if $K_B^+$ denotes Bob's public key and $K_B^-$ denotes Bob's private key, then $K_B^+\left(K_B^-(m)\right) = m$. Recall that for RSA encryption, we first apply the public key to the message m to find out $K_B^+(m)$. So, for RSA encryption, suppose Alice wants to encrypt and send a message to Bob, then she finds out $K_B^+(m)$.

So, this is the message encrypted using Bob's public key. And then to decrypt this message, Bob computes $K_B^-\left(K_B^+(m)\right)$, and we showed that this is the same as m. So, Bob can recover the original message by decrypting it using his private key. Now, during our discussion of RSA, we discussed that if you apply these operations in the reverse order, then we still get back the original message m. That is, if we start from the message m, and apply the encryption algorithm, but with the private key in place of the public key, then we get $K_B^-(m)$, and then if we apply the decryption algorithm, but with the public key in place of the private key, then we get $K_B^+\left(K_B^-(m)\right)$, and this is the same as m. So, if these operations are applied in the reverse order, then still we get the original message m. So, we discussed this property, and this was an exercise for you to show during our lecture on RSA.

So, we now use this property to create a digital signature. To sign a message m, Bob computes $K_B^-(m)$ and appends it to m. So, what is $K_B^-(m)$? $K_B^-(m)$ is obtained by taking the message m and applying the encryption algorithm with Bob's private key to it. So, $K_B^-(m)$ is the digital signature, and it is appended to the message m. So, $\left(m, K_B^-(m)\right)$ is the digitally signed message. This shows the scheme.

- Recall: if $K_B^+$ (respectively, $K_B^-$) denotes Bob's public key (respectively, private key), then:
  - $K_B^+(K_B^-(m)) = m$
- To sign a message $m$, Bob computes $K_B^-(m)$ and appends it to $m$
  - $(m, K_B^-(m))$ is the digitally signed message

This is the original message m. And then, we apply the encryption algorithm along with Bob's private key $K_B^-$ to get the signature $K_B^-(m)$ and this $K_B^-(m)$ along with the message

m, that is a digitally signed message. So, is this signature $K_B^-(m)$ verifiable and nonforgeable? Our claim is that yes, it is verifiable as well as nonforgeable. So, it is verifiable because any third person can use Bob's public key, which is well known, to compute $K_B^+\left(K_B^-(m)\right)$, which we know is the same as m.

So, since $K_B^+\left(K_B^-(m)\right) = m$, which is the same as the plaintext message, hence the third person is able to verify that this is indeed Bob's digital signature. So, the message along with the digital signature is this, $(m, K_B^-(m))$. So, a third person who wants to verify this message applies Bob's public key to it, and the result is $K_B^+\left(K_B^-(m)\right)$, and if that is the same as m, then the verification is successful. Notice that suppose this is m', and this is $K_B^-(m)$, then the verification will fail because this $K_B^-(m)$ is not the digital signature of m'. So, this is the verification step, and the digital signature can be verified by anyone, so hence the verifiability property is satisfied.

Also, is it forgeable? Now, to create this digital signature $K_B^-(m)$, we know that we require Bob's private key, but only Bob knows his private key. Since the knowledge of the private key is required to compute $K_B^-(m)$, it is nonforgeable. Only someone with the knowledge of Bob's private key is able to compute this digital signature. So, since the private key is known only to Bob, this signature is nonforgeable.

We have obviously assumed that Bob has not shared the private key $K_B^-$ with anyone, and it has not been stolen from him. Under these assumptions, the nonforgeability property is satisfied. Now, this scheme has a shortcoming. We discussed that RSA encryption and decryption are computationally expensive operations. So, under this scheme, one needs to calculate $K_B^-(m)$, that is, the encryption algorithm needs to be run for the entire message m.

So, public key encryption/decryption is time-consuming; hence, this scheme is computationally expensive when m is long because we require to compute $K_B^-(m)$. We want a more computationally efficient scheme for creating a digital signature. So, to create a more computationally efficient scheme, we use a cryptographic hash function, which we discussed in the previous lecture. So, this more efficient scheme is as follows. To sign a message m, Bob first computes its hash H(m) and then encrypts it with his private key to get $K_B^-\left(H(m)\right)$, and then appends $K_B^-\left(H(m)\right)$ to m.

So, $\left(m, K_B^-\left(H(m)\right)\right)$ is the digitally signed message. This shows the scheme.

This is the original message m. Bob applies a hash function to it to get the hash value, H(m), and then applies the encryption algorithm along with Bob's private key to this hash. The result is the signed hash, $K_B^-\big(H(m)\big)$, and then the original message m along with the signed hash, $K_B^-\big(H(m)\big)$, are appended, and that's the package that is sent to Alice. So, that is the message along with the digital signature. And this figure on the right shows the operations that are performed by someone who wants to verify the digital signature. So, the message along with the digital signature is this, $\big(m, K_B^-\big(H(m)\big)\big)$.

The verifier takes the message part from it, that is m. And then applies the hash function to it to get the hash value, H(m), and then the verifier takes the other part of the package, that is $K_B^-\big(H(m)\big)$, and then applies the encryption algorithm along with Bob's public key to it to get H(m). Then it compares this H(m) with the hash of the message m. If they are the same, then the verification is successful.

- To sign a message $m$, Bob computes its hash $H(m)$, encrypts it with his private key to get $K_B^-(H(m))$ and appends $K_B^-(H(m))$ to $m$
  - $(m, K_B^-(H(m)))$ is the digitally signed message
- Scheme 2 also works and is computationally more efficient
- Consider the alternative scheme, where $c(m)$ is a checksum and $(m, K_B^-(c(m)))$ is digitally signed message. Is this a secure digital signature scheme?

So, this is the scheme for creating a digital signature. This also works, and it is computationally more efficient. So, let's see why this scheme creates a digital signature successfully. So, it is verifiable using this scheme. To verify it, someone only needs to know Bob's public key, which is well known, so it is verifiable.

And to create this digital signature, one needs to know Bob's private key, which is only known to Bob; hence, it is nonforgeable. Only someone with the knowledge of Bob's private key is able to create it, so hence, only Bob is able to create this digital signature; hence, it is nonforgeable. Hence, this scheme also works; it is verifiable and nonforgeable. And it is computationally more efficient than the previous scheme because, to create the digital signature, Bob only needs to find $K_B^-\big(H(m)\big)$. Recall that, regardless of the length of the input message m, the hash value is a fixed-length message of just a few bits, typically 128 or 256 bits or of that order.

So, H(m) is a short string regardless of the length of the message. Hence, it's computationally easy to compute $K_B^-\big(H(m)\big)$. So, it is computationally more efficient. Now, consider the alternative scheme, where c(m) is a checksum function instead of a hash

function, and suppose we use the same scheme with H(m) replaced with c(m). So, now the message m is combined with the, with $K_B^-(c(m))$, and that is a digitally signed message. So, is this a secure digital signature scheme?

So, my claim is that this is not a secure digital signature scheme because, as we have seen, for a message m, for a checksum function, we can easily find another message m', which has the same checksum as the message m. So, we can find another message m' such that c(m)=c(m'). So, once we find such a message, we can replace this quantity with $\left(m', K_B^-(c(m))\right)$. The verification will succeed because c(m) is the same as c(m'). So, it's easy to modify this message. Someone can easily modify the document and someone is able to create a digital signature of a message m' successfully. Hence, this scheme is not a secure digital signature scheme.

So, it's forgeable because someone is able to forge the digital signature of the message m' provided that c(m)=c(m'). Hence, this is not a secure digital signature scheme. Now, a digital signature is mainly used to indicate that he/she created a document or to acknowledge its contents or to agree with its contents and so on. But a digital signature can also be used to achieve the property of message integrity, as we'll see. Recall that in scheme 1, $(m, K_B^-(m))$ is the digitally signed message, and in scheme 2, $\left(m, K_B^-(H(m))\right)$ is the digitally signed message.

- in scheme 1, $(m, K_B^-(m))$ is the digitally signed message
- in scheme 2, $(m, K_B^-(H(m)))$ is the digitally signed message

- Which of these schemes, if any, achieves message integrity?
  - Both; due to verifiability, the fact that $K_B^+(K_B^-(m)) \neq m'$ for $m \neq m'$ and computational infeasibility of finding $m' \neq m$ such that $H(m') = H(m)$

So, which of these schemes, if any, achieves message integrity? My claim is that both of these schemes achieve message integrity. That is because of verifiability, the fact that $K_B^+(K_B^-(m)) \neq m'$ when $m \neq m'$, and the computational infeasibility of finding a message m' that has the same hash as an original message m. So, using these properties, it is left as a simple exercise for you to show that both these schemes achieve message integrity. So, if one wants to send the message m with integrity, then one just has to send $(m, K_B^-(m))$,

or one can send $\left(m, K_B^-\left(H(m)\right)\right)$, and then the receiver can verify whether the sender is indeed who they claim to be and whether the message was modified during transit.

So, both these schemes achieve the property of message integrity. Now, recall that the message integrity of a message m can be achieved using a message authentication code as well as using a digital signature. So, what are the pros and cons of these techniques, MAC and digital signature? The pros and cons are as follows. Recall that to create a digital signature one requires encryption, which is time-consuming.

So, that is a disadvantage of a digital signature for creating a message authentication code. But a MAC does not require encryption. So, recall that the message authentication code of a message m was this. H(m,s) is the message authentication code. So, to create this one does not require any encryption.

So, whereas a digital signature requires encryption. To create $K_B^-\left(H(m)\right)$, one needs to encrypt H(m) using the private key. So, that's one trade-off. So, the digital signature requires encryption, but the MAC doesn't. So, that's an advantage of a message authentication code.

On the other side, we have that a MAC requires the sender and the receiver to have a shared secret, which is the authentication key s. The digital signature does not require the sender and receiver to have any shared secret. So, to use a message authentication code, the sender and receiver initially have to agree on a shared secret, which will serve as the authentication key s. Subsequently, they can use a message authentication code to achieve message integrity. So, to compare a digital signature and a message authentication code, we note that a message authentication code can only be used to achieve the property of message integrity. So, that is in the context where one sender, Alice, wants to send a message to one receiver, Bob. So then, they can have a secret shared key s, and then use a message authentication code.

If they want to send a lot of messages to each other, then it is efficient to use a message authentication code because the authentication key has to be created and shared only once at the beginning of the communication. Subsequently, they can send any number of messages using the same authentication key. So, in this case, message authentication code is more efficient than digital signature. But digital signature is more general than a message authentication code. A digital signature of a message can be verified by anyone, not just someone who has a shared authentication key with the creator of the digital signature.

So, a digital signature is a more powerful scheme than a message authentication code. So, a digital signature not only achieves message integrity but also is more general and can be used to sign a message, satisfying the properties of verifiability and nonforgeability. So, these are the pros and cons between the message authentication code and a digital signature. Thank you.