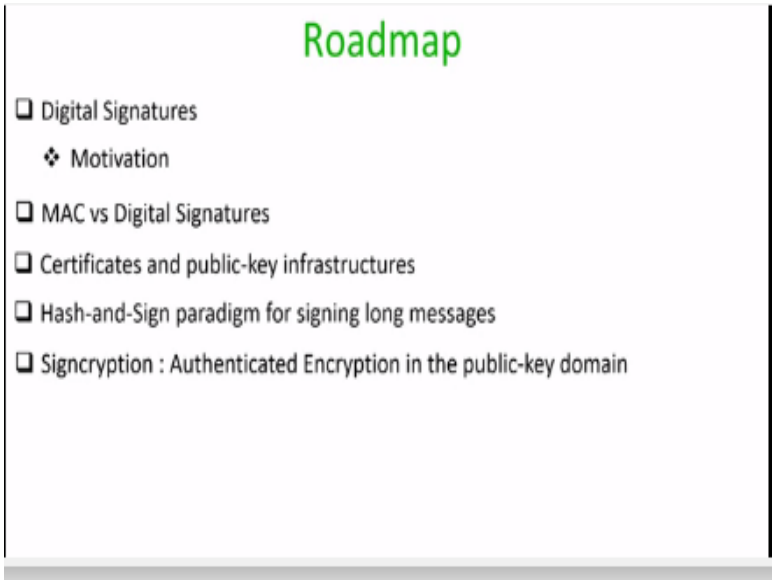
Foundations of Cryptography
Dr. Ashish Choudhury
Department of Computer Science
International Institute of Information Technology - Bangalore

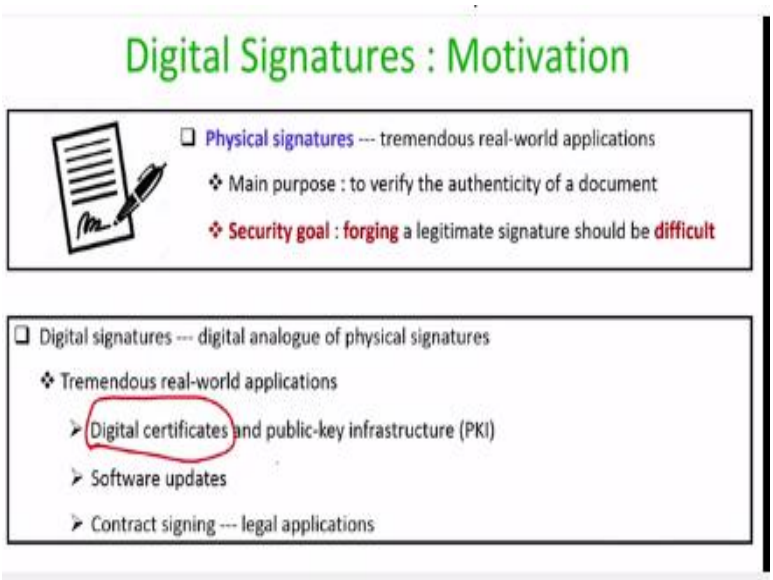
Lecture – 51
Digital Signatures

(Refer Slide Time: 00:31)



Hello everyone, welcome to this lecture, so the plan for this lecture is as follows and this lecture we will introduce a very popularly used cryptographic primitive in the public key setting namely the digital signatures and we will see the motivation for digital signatures, we will see the comparison between message authentication codes and digital signatures. We will discuss certificates and public key infrastructures; hash and sign paradigm for signing long messages and then finally, we will discuss on a very high level how we can achieve authenticated encryption in the public key domain using signcryption schemes.

(Refer Slide Time: 01:03)



So, let us start with the motivation for digital signatures and before going into that let us first try to understand the importance of physical signatures which we use in the real world, so all of us know that physical signatures have got tremendous applications and a main purpose is to verify the authenticity of a document. For example, if I take the digital copy of my Aadhar card, it signed by the; it is digitally; sorry, if I take since I am considering physical signature, if I consider for example, a signed check which I have given to you, then if you submit it to the bank, the bank can verify the authenticity of the signed check by verifying my signature and the main security goal for the physical signature is that even if there is a person who has seen me signing several documents, it should be very difficult for that person to forge a legitimate signature on those document which I had never sign in the past.

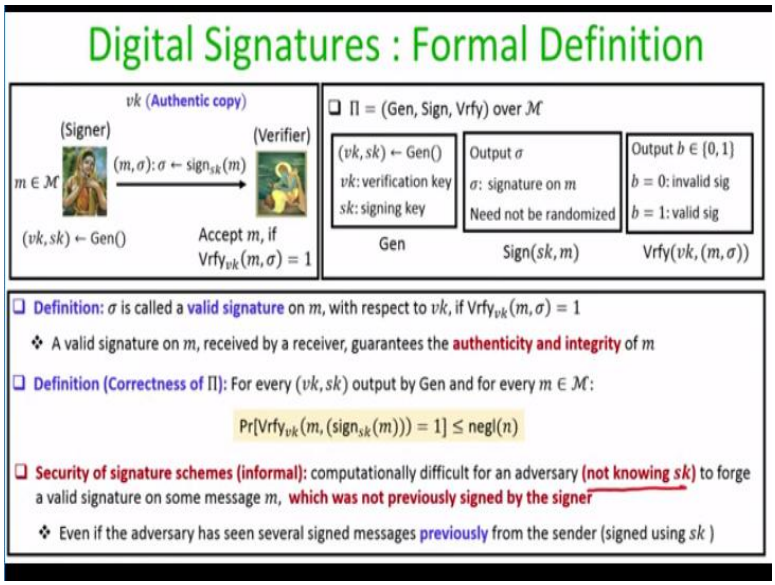
So, the main goal here is basically to verify the authenticity of a document. Now digital signatures they can be viewed as digital analogue of physical signatures and it has got tremendous applications in the context of real world scenario. So, for example they are used extensively for digital certificates and public key infrastructure alright, so when I say digital certificate you can imagine for instance, the digital Aadhar card.

So, you do not need to carry your Aadhar card in the physical format, now you have the digitally signed Aadhar card and if you submit it at some at whatever place you are supposed to produce your Aadhar card, the corresponding entity can verify the digital signature and either accept or

reject your digitally signed Aadhar card, we will see what exactly public key infrastructure is and how digital signatures are used in the public key infrastructure.

We also use digital signatures for software updates, so whenever we buy a software for the first time, the verification key of the vendor who is providing me the software comes along with the purchase and I can store the verification key of the vendor with me and later on if the vendor is giving me some updates for the same software I can verify whether the updates are coming from the legitimate source or not by verifying a signature on the updated software coming from the vendor, we also use digital signatures for contract signing where we have legal applications.

(Refer Slide Time: 03:29)



So, now let us understand the formal definition of digital signatures, so it is a triplet of algorithm; a key generation algorithm, a signature algorithm or a signing algorithm and a verification algorithm and it supports messages from some publicly known plaintext space, so the key generation algorithm outputs 2 keys; since we are in the public key setting; one key will be the verification key which will be available in the public domain.

And other key will be sk, which is the signing key and it will be available only with the signer. So the way this key generation algorithm is executed is as follows. So, if I am a signer; to set up my key, I run the key generation algorithm and I make my verification key available in the

public domain. So we assume that the verification key of the so-called sender, an authenticated copy of the so-called sender will be available in the public domain.

How exactly it will be ensured that indeed the so-called verification key belongs to the intended sender will be ensured through public key infrastructure but for the moment, assume that we have mechanisms to ensure that indeed the legitimate or authentic copy of the intended signer is available in the public domain. Now, to sign a message what the signer does is it runs the signing algorithm and it outputs a signature.

So, the signing algorithm takes the message and the signing key and it need not be randomised, so if signer has the message m, it runs a signing algorithm and along with the message, it outputs a signature. Now, the signature along with the message is verified by running the verification algorithm which takes the (message, signature) and the verification key and it outputs either 0 or 1, 0 means reject the message because it is an invalid signature. 1 means it accepts the message because it is a valid signature. Now you call a signature σ to be a valid signature on the message m with respect to the verification key vk if the verification of the message, signature with respect to the verification function and the verification key vk gives me the output 1 and if there is a valid signature which is received by a receiver which is verified with respect to the verification key vk, then it guarantees the authenticity and integrity of m.

Because if I am a receiver and if I receive a (message, signature) which is a valid signature with respect to the verification key of a designated signer, then it guarantees to me that indeed the messages coming from the intended sign because until and unless forgery is possible, it is very difficult for a third party to act or intend as the signer and produce a signed message on the behalf of the signer.

So, now we go to the properties of the signature scheme that we require. So, the first property is the trivial correctness requirement which requires you that for every pair of key obtained by the key generation algorithm and for every message if the signer has signed some message using the signing key sk and later the signature is verified with the same message m using the verification

algorithm and the corresponding verification key vk, then the output should be 1 except with some negligible probability.

Ideally, we expect that there should be absolutely no error in the signature verification algorithm if everything has happened correctly but the reason we are leaving a scope of negligible error in this definition is that in the instantiation of the key generation algorithm, it might be the case that the public key and a secret key are not consistent with each and what I mean by consistent means, example if I assume that if I were running the; if I am using a RSA key generation algorithm and my generation algorithm is basically to pick primes p and q.

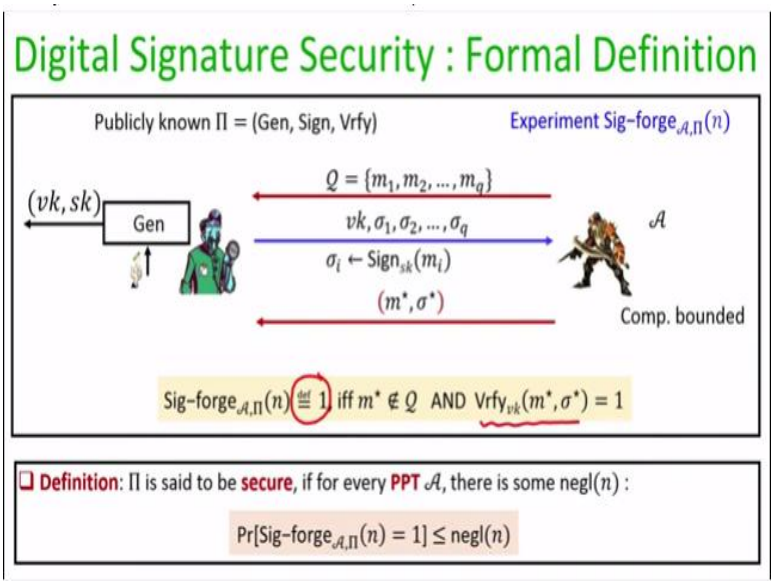
And set the modulus to be the product of those 2 primes, then if I choose a randomised algorithm to pick my prime numbers p and q, then it may so happen that I may end up picking composite values of p and q without even knowing that I have actually picked composite values p and q, in that sense even though my key generation algorithm has given me incorrect parameters which can happen only with this negligible probability, I will not be aware of the fact.

And if I am using such kind of composites p and q and then definitely the signing algorithm will not produce a signature which will be accepted by the verification algorithm and that is why we are giving a scope of negligible error in the correctness requirement of signature scheme, whereas if you are generating your parameters by running deterministic algorithms, in the sense that you are picking the say, value of the primes p and q using deterministic algorithms where there are absolutely no error in picking up in terms; no error in terms of picking primes or composites, then the resultant signature scheme will have no error in the correctness property, now comes the important property of the signature schemes namely the security and on a very high level, we require the same guarantee; security guarantee that we expect from physical signatures. So, remember as I said earlier for the physical signatures, the security guarantee that I require is that even if I have signed multiple documents and given to you, it should be very difficult for you to produce my signature on a message which I never signed earlier.

We expect the same thing to hold in the context of digital signatures namely an entity who knows only the verification key but does not know my signing key but has seen me sign; signing

several messages in the past using the signing key sk, for such an entity it should be computationally difficult to forge my signature on a document which I had never signed.

(Refer Slide Time: 09:26)



So, let us now formalise the informal requirement that we had discussed in the last slide for the signatures by this experiment which we call as $\text{Sig-forge}_{\mathcal{A},\Pi}(n)$ played between a computationally bounded adversary and a challenger and at the very high level, the game is something similar which we had used to model the security requirement of the forgery requirement against the message authentication code.

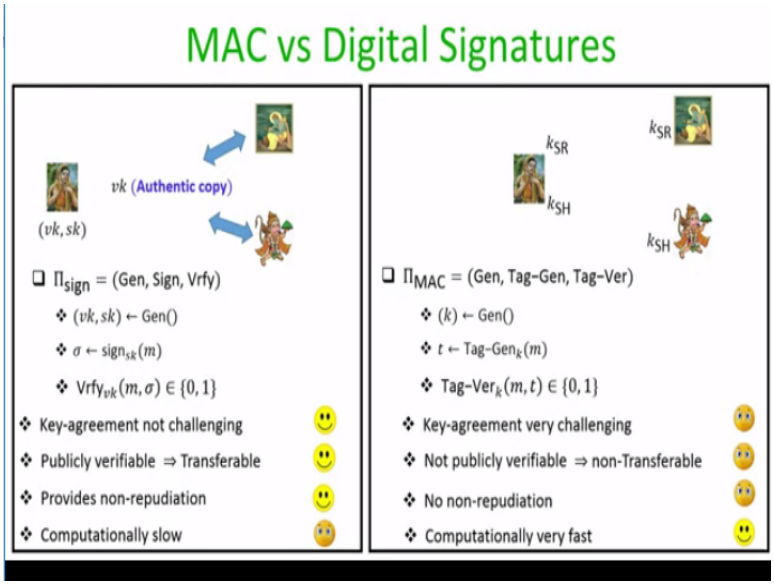
So, we have a training phase and we have an output phase; in the training phase to model the fact that the real world adversary would have seen a signer signing several messages in the past, we allow the adversary the chance to submit several messages of its choice and see the signature on those messages. To respond to those training phase messages, the challenger runs the key generation algorithm and signs all those messages using the unknown signing key not known to the adversary.

But the corresponding verification key is given to the adversary and now in the output phase, the adversary submits a message and a forgery and the definition of the experiment is we say that the adversary has won the game denoted as saying that the output of the experiment is 1, if and only

if the forged message m* is different from the messages for which the adversary has got signature in the past.

And the signatures σ* is indeed a valid signature on the message m* under the verification key vk and we say that a signature scheme is unforgeable or simply secure, if for every poly time adversary participating in this experiment, the probability that it can come up with successful forgery is upper bounded by a negligible function in the security parameter.

(Refer Slide Time: 11:09)



So, now we compare the 2 primitives message authentication code and digital signature because on a very high level both the goal; the goals of both the primitive is the same : to prevent forgery or authenticity and integrity, so syntactically a signature scheme consist of a key generation algorithm, a signature algorithm and a verification algorithm and so is a message authentication code.

The difference is that for the signature, the key generation algorithm output 2 keys, whereas for the message authentication code, the key generation algorithm outputs a single key. For message authentication code the same key k is used for both generating the tag as well as for verifying the tag, whereas for the signature different keys are used for producing the signature and for doing the verification, so that is a difference on the syntactic level.

So, because of this, we have several pros and cons for both this with respect to both these primitives, so if we consider signature schemes key agreement is not at all a challenging task, in the sense signer can just run the key generation algorithm and it can make its verification key available in a public domain in an authentic fashion and any verifier could use that verification key to verify the signatures generated by the signer.

On the other hand, for message authentication code since it is a symmetric key primitive, key agreement is a very, very challenging task. If Sita wants to do authentic communication with 2 receivers say Ram and Hanuman, then she needs to have a pair of symmetric key or mac key for Ram and another independent mac key for Hanuman.

But when we come to the signature it is fine, if she just have one signing key and one verification key that will solve the purpose for every potential verifier. As a consequence, when it comes to signature schemes they are publicly verified, the signer signs the message, any verifier can just pick up the verification key of the signer and verify whether the signature is valid or not, whereas for the message authentication code, it is not publicly verified.

Only the receiver who has the same key k, with which the sender has generated the tag can verify the authenticity of the message, tag. As an implication of that, we get tag, since the digital signatures are publicly verified they are transferrable, that means if say Ram has obtained legitimately signed documents from Sita, then it can forward it and transfer it to Hanuman, claiming that indeed the document originated from Sita.

And Hanuman can also verify whether indeed that is the case or not but that is not possible in the case of message authentication code because its sender Sita has computed a message authentication tag for a message then that can be verified only by Ram and until and unless Ram transfers the key k_{SR} to Hanuman, the authenticity of the tag cannot be verified by Hanuman, in that sense message authentication codes are not transferable.

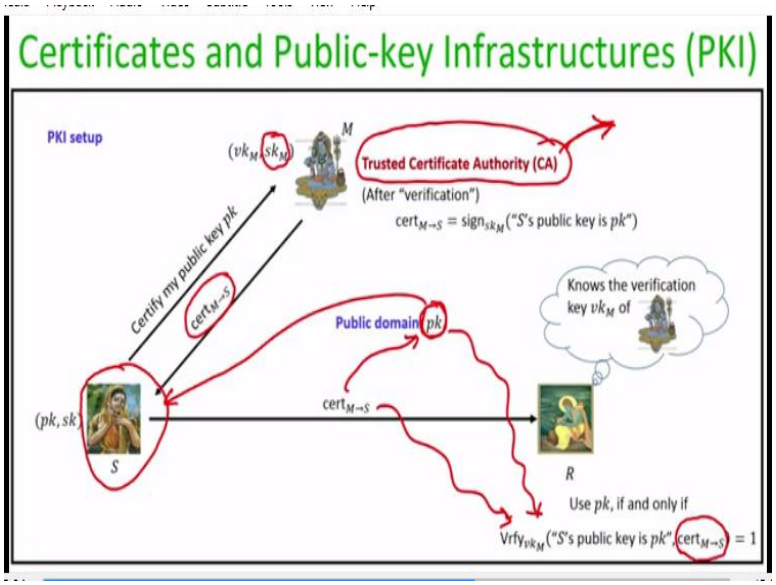
And because of those transferable property, digital signatures provides non-repudiation which is very useful in the legal applications namely, if Ram obtains a signed contract from Sita and later

if there are any violation by Sita with respect to the terms mentioned in the contract, then the Ram can go to any legal experts say, Hanuman and prove that indeed he has received a signed contract from Sita.

But now, Sita is not following the terms and conditions that are mentioned in the document and Hanuman can verify the claims of Ram but that need not be the case in the message authentication code, until and unless Ram can transfer the mac key to Hanuman or to any third party. So, it might look like that digital signatures have all the good features and message authentication codes have all the bad features.

But that is not the case because if you consider the computational aspect for message authentication codes and digital signatures it turns out that the amount of computations that are required for digital signatures, they are of order of several magnitude compared to the amount of computations required for message authentication codes; specifically, the signature schemes that we are going to see based on the RSA function and Diffie Hellman problem, where we will be performing modular exponentiation, modulo a very large modulus, whereas as we had already seen, message authentication codes can be designed very efficiently using block ciphers. That is a trade-off between message authentication code and digital signatures depending upon whether you prefer efficiency versus public verifiability, non-repudiation, transferability, you can use either message authentication code or digital signatures.

(Refer Slide Time: 16:18)



So, now let us discuss a very practical application of digital signature namely that of public key infrastructure and digital certificates. So, till now I was constantly saying that if there is an entity say, Sita, right, which has run the key generation algorithm of a public key encryption scheme and has obtained the public key and secret key, we were assuming that the public key of Sita is available in the public domain in an authentic fashion.

That means, anyone, any receiver or any entity, Ram who wants to do a secure communication with Sita can look for the public key of so-called Sita and we were assuming that there is a mechanism for Ram to verify that whether the so called public key indeed belongs to the intended Sita or not, we were assuming that an authentic copy of the public key pk of the intended Sita is available in the public domain.

But now, we would like to answer this question that how at the first place, Ram can verify whether indeed the public key pk, which is available in the public domain is the public key of the so-called Sita or not, right. So, we would like to now solve this problem and to solve this problem, we assume that we have or all the parties in the system have access to a trusted authority and we call certificate authority of CA, which I denote by M considering, it is the master.

And this master will have its own signing key and verification key for some secure signature scheme and we assume that somehow the verification key of this master is available to any entity Ram who wants to verify the authenticity of the public key of Sita, right, so that is a trust assumption we are making in this whole solution, right. Now, to convince Ram that indeed the public key pk belongs to Sita, what Sita can do is; it can, in the off-line it can go to the master and it can request the master or the certificate authority that please, certify my public key pk.

And to do the certification, what the authority can do is, it can perform the verification namely, it can verify the credentials of so-called Sita, it can verify whether indeed it is the right Sita or not, what exactly are its credentials and so on and if the credentials are verified successfully, the master authority or the certificate authority can issue a digital certificate which basically, is the signature of the master on a certificate stating that indeed the public key or the encryption key of the entity Sita is pk, right.

So, that is how digital certificate or basically a signature digitally signed document produced by the certificate authority and given to Sita, all this is happening in the off-line right, nothing as of now, no communication has happened between Sita and Ram, everything Sita is performing in the off-line. Now, once this certificate is obtained by Sita, what Sita can do is; if she now wants to convince Ram that indeed the public key pk belongs to her, what she can do is; she can transfer the public key pk to Ram and along with that she can send a certificate issued by the certificate authority and now, you see that Ram can verify whether indeed the so-called public key which is obtained from Sita indeed belongs to Sita or not because what she has to do is; she has to just verify whether the digitally signed certificate or the signature is indeed the right signature for the message that sender's public key or Sita’s public key is pk, under the verification key of the certificate authority which we are assuming is available to the entity Ram.

If the certificate verification is successful, Ram is convinced that indeed the public key pk belongs to the entity called Sita and this whole setup here that we had discussed here is what we call as public key infrastructure setup or PKI setup, where the minimal trust assumption that we are making here is that we have certificate authority available and its verification key for a signature scheme is available with every entity in the system.

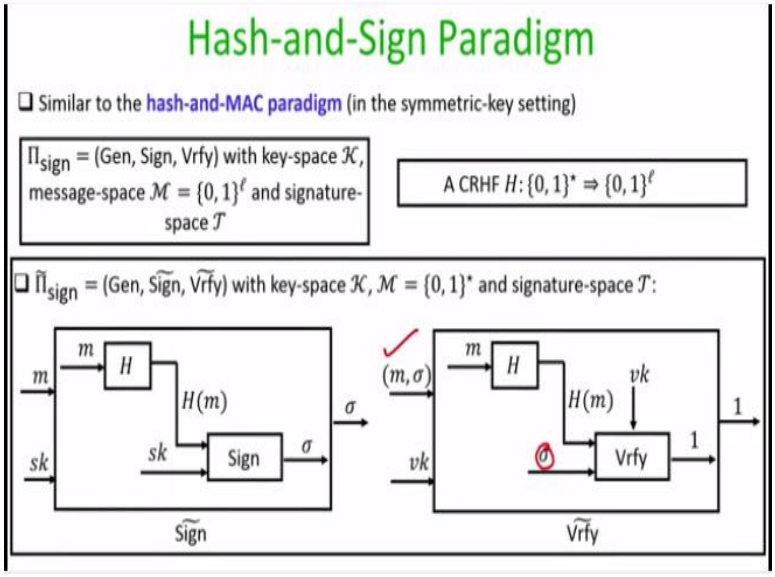
Any sender or any entity Sita, who wants to get its authentic copy of the public key can in the off-line go to the certificate authority and get a digital certificate by proving its credentials and getting a digital certificate stating that indeed its public key is pk and now if Sita wants to send or setup its public key, what she can do is; she can put a public key pk in the public domain along with this digital certificate.

Or if she wants, she can send a public key to any entity Ram who wants to do a secure communication with Sita along with the digital certificate and the authenticity of the public key can be verified by verifying the certificate and the so-called public key, so that solves our whole problem of setting up authentic copies of the public key for the entities, we just have to assume that we have a trusted authority called certificate authority available in our system whose verification key is available with everyone.

And this whole set up is called PKI setup, in this example I have just used one CA's, so you might be wondering that if there are millions of Sitas who want to get a public key verified by the master then that will put too much of load on a single certificate authority or if signing key of the certificate authority is compromised, then anyone can forge digitally signed documents, any corrupt Sita can forge this master authority's certificate on any invalid public key and that will create problem in the whole systems.

So, what we can do is instead of having a single certificate authority we can imagine that we have a hierarchy of certificate authority and so on, but the high level idea is that just by making a small trust assumption namely, the presence of a trusted certificate authority, this whole problem of setting up authentic copies of the public key can be solved.

(Refer Slide Time: 23:05)



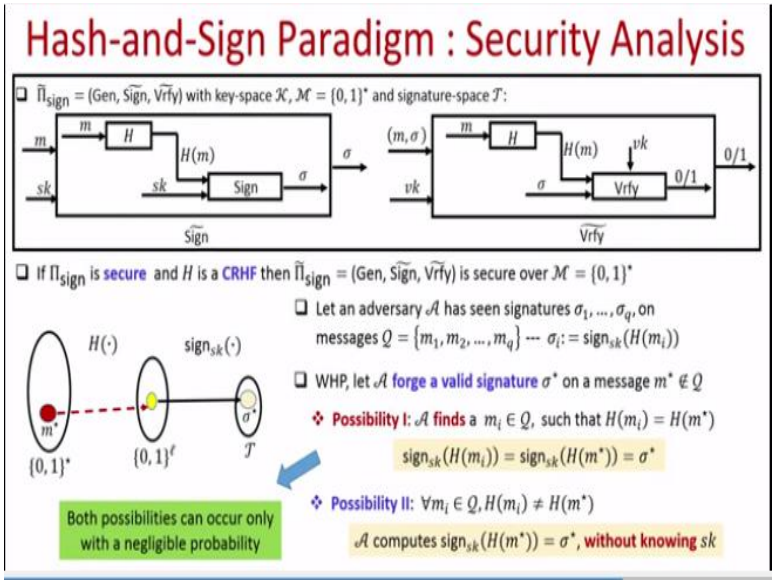
Next, we discuss an interesting paradigm, which we call as hash-and-sign paradigm, which can be used to sign arbitrary long messages and this is similar to your hash-and-Mac paradigms which we had used in the symmetric key setting. So, the scenario is the following; we have a signature scheme which is secure but which can support signatures on messages of length *l* bits and we are given a collision resistant hash function mapping arbitrary length bit strings to an output of *l* bit strings.

And our goal is to now come up with the secure signature scheme by combining these 2 primitives which can sign messages of any length, not just bit strings of length *l* bits. But the signature size is fixed size, namely the signature space of this new signature algorithm should be the same as it was for the secure signature scheme which we are given, that means, if my underlying signature scheme which I have given produces signature of say, length *l* bits, then it does not matter what is the size of the message; using the composed signature scheme, the new signature scheme, it does not matter what is the length of the message, my signature size will be of n bits, its size will be independent of the size of the message which I am going to sign and the way we are going to combine these 2 primitives is as follows : to sign a message, m, which could be of any length, what we do first is we hash the message and once we hash the message, we obtain a message digest of the message which is of length *l* bits and now, we compute the signature on the hash of the message and that is considered as the overall signature for my message *l*, to verify the signature we do the corresponding operations, if you are given a

(message, signature) pair, what we do is we first hash the message and then we verify whether the signature component that we have received is indeed a valid signature on the hash of the message that we have obtained.

If the verification algorithm is 0, then the overall output of the new verification algorithm is 0 that means, we reject the (message, signature) pair, on the other hand if the output of the verification algorithm of the underlying signature scheme is 1, then we accept the given message, signature pair in the higher level verification algorithm.

(Refer Slide Time: 25:40)



Now, we can prove that if the given signature scheme for fixed length messages is secure that is its unforgeable and if my hash function is collision resistant hash function, then the signature scheme for signing arbitrarily long messages that we have obtained is secure, and a security proof for this is again more or less same that we have used to prove the security of the hash and Mac paradigm, let me just give you an overview of the security proof here.

So, imagine that we consider an adversary participating in the signature forgery algorithm against this new signature scheme and say it has asked for the signatures of messages in the set Q consisting of q number of messages and as per the rule of the signature scheme that we have constructed each signature is nothing but a signature on the hash of the message that adversary has queried.

And now, imagine that with very high probability in this signature forge experiment, the adversary A after getting the signatures on the messages in the set Q could produce a forgery on message m* which does not belong to the set of messages for which it has seen the signatures and if that is the case, then there could be 2 possibilities. Possibility 1 is that there exist at least 1 message in the set of messages for which adversary has signature say m_i such that the hash of the message m_i and the hash of the forged message m* are same.

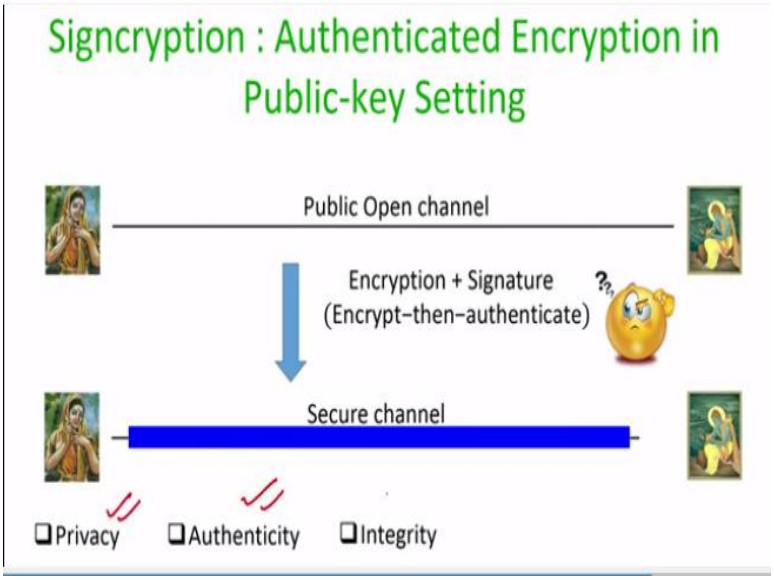
If that is the case then it turns out that both the message m_i as well as the message m* will produce the same signature σ* and since adversary has already seen the hash of the message m_i, it is σ*, it can simply submit the signature of the message, new message m* is also σ* and it is a valid forgery but if this is the case, then it means that adversary knows how to find out a pair of collision for the underlying hash function in polynomial amount of time.

But that goes against the assumption that my underlying hash function is a collision resistant hash function. On the other hand the second possibility could be that the forged message m* is such that its hash value is different from the hash value of all the messages for which the adversary has seen the signature, if that is the case then the only way the forged message m* and σ* is a valid forgery is that adversary has computed the signature on the fixed length message namely H(m*), from scratch without actually knowing the signing key sk.

But if that is the case, then it means that there exist a bit string of length *l* bits for which the adversary could compute the signature with high probability even without knowing the signing key sk but that goes against the assumption that my fixed length signature scheme is secure. So, these are the 2 possibilities with which any adversary could come up with a successful forgery with high probability against the new signature scheme that we have constructed.

And as we have argued that both these possibility can occur only with the negligible probability. We can formalise this intuition rigorously in the same way that we have used to prove the security of our hash and mac paradigm but I am not going into the formal details.

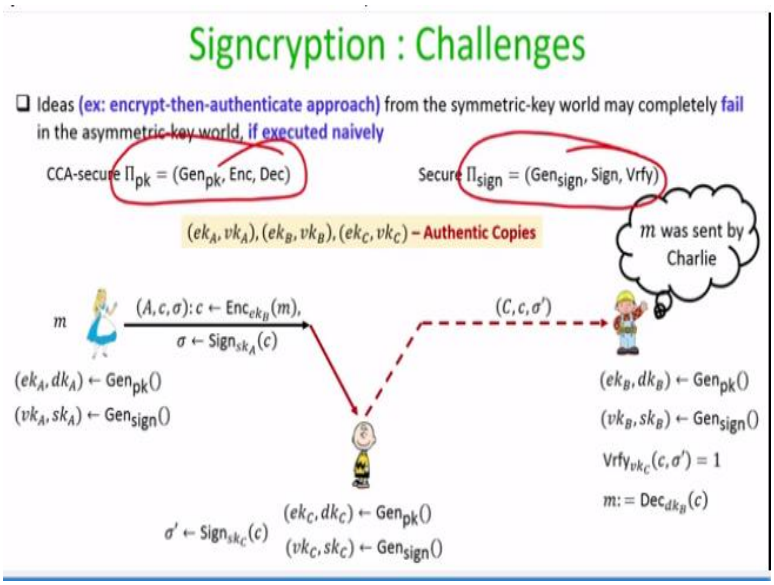
(Refer Slide Time: 29:21)



Now, let me finally end this lecture by asking the question that is it possible to combine signatures and encryption in the public key setting to obtain an equivalent notion of authenticated encryption. So, remember in the symmetric key world we have seen that if there is a sender and a receiver with no pre shared information and connected by a publicly open insecure channel.

Then by using an authenticated encryption scheme which we can obtain by generically combining a CPA secure symmetric encryption and a secure MAC, we get the affect of a virtual secure channel between the sender and receiver, which satisfies 3 properties namely, privacy, authenticity and integrity. A natural question will be that if we now go to the public key world, we know how to achieve privacy using encryption. And we have argued till now that how we can achieve integrity and authenticity in the public key domain using signature schemes, so is it possible now to combine these 2 primitives say using the encrypt, then authenticate approach and get an authenticated encryption scheme namely the affect of a virtual secure channel between the sender and a receiver satisfying privacy, authenticity and integrity.

(Refer Slide Time: 30:40)



It turns out that yes, we can get the effect of an authenticated encryption scheme even in the public key setting by a primitive which we call as signcryption, sign because we are going to use a signature scheme and cryption because we are going to use an encryption scheme but it turns out that the naive way of combining a signature scheme and encryption scheme is not going to give you an authenticated encryption scheme.

In fact, it turns out that ideas from the symmetric key world may completely fail in the asymmetric key world and what we are going to show here is if you see the encrypt then authenticate approach which we know always yields authenticated encryption scheme in the symmetric key world can completely fail, if executed naïvely in the public key world. So, to make my point clear, imagine we are given a CCA secure instantiation of public key encryption scheme.

And for the moment assume we are given a secure signature scheme. Till now, we have not yet seen a candidate signature scheme but for the moment assume, we are given a candidate signature scheme. Now, what I am going to demonstrate is that if I combine these 2 primitives using the encrypt then authenticate approach, then we will not get what exactly we are asking for.

So, to make my point clear imagine, we have 3 entities in the system; Alice, Bob and Charlie, you can imagine that the Bob is a maths teacher for the class and Alice and Charlie are 2 of the students and we imagine that each entity in the system has its own pair of encryption key and decryption key and its own pair of signing key and verification key, so for instance we assume that Alice has her pair of encryption key and decryption key. And it has her own pair of signing key and verification key, in the same way we have Charlie has its own pair of encryption, decryption key and its own pair of signing key and verification key and so is the case for Bob, and we assume that for the moment using say PKI setup and whatsoever you assume; we assume that here the public key and a verification key of all the entities, their authentic copies are available in the public domain.

Now, imagine our teacher Bob has given maths assignment and say Alice has prepared her assignment solution m and now she wants to send it to teacher Bob in an authentic fashion using the encrypt then authenticate approach, so what she does; she encrypts her assignment using the public key of Bob, obtain the cipher text c and then it authenticates the cipher text c by signing, using her signing key.

And she also puts her identity to show Bob that actually, this assignment is coming from an entity called Alice and she forwards her cipher text and signed cipher text to Bob. Now, imagine Charlie who has not done the assignment; he will eavesdrop this communication. What he can do is; it can retain the encryption of the assignment as it is namely, it does not change c but what it can do is instead of sending σ or signature to Bob, it can prepare its own signature on the assignment; encrypted assignment by producing his signature on this encrypted assignment, which it can do because it owns its own signing key and now what it does is; it changes A to C and sends the assignment to Bob saying that the assignment has been submitted to Bob. Now, what Bob is going to do is or the maths teacher is going to do is; it will verify whether the assignment is correct or not. So, to do the verification what it will do is; it will first verify whether indeed σ' is a valid signature of so called Charlie on the cipher text c or not.

And only if the signature is valid, Bob will proceed further and decrypt the cipher text to obtain back the assignment and in this case, everything will pass successfully and the maths teacher

Bob will be convinced as if the assignment was submitted by Charlie, which is actually not the case. So, you might be wondering that what went wrong here because we had proved in the symmetric world that in the symmetric key world, then encrypt then authenticate approach is secure.

But what we are now saying is that if we implement the encrypt then authenticate approach naïvely using a signcryption scheme, then we do not get what exactly we are asking for. The problem here is that since we are in the public key world, anyone can strip off the signature generated by a legitimate Alice and instead of that it can produce its own signature because we are in the public key world that is the main source of the problem.

And that is why we have to combine these 2 primitives namely, this public key encryption scheme and the signature scheme in a sophisticated fashion whose details are out of scope of this course but summary here is that we can achieve the notion of authenticated encryption by combining the signature; by combining a secure signature scheme and a secure encryption process.

So that brings me to the end of this lecture, just to summarise; in this lecture, we had introduced a very important public key cryptographic primitive namely digital signatures, we have seen its security and we have seen a generic construction of how to combine a fixed length signature scheme with the collision resistant hash function to sign arbitrarily long messages but with the fixed size signatures, thank you.