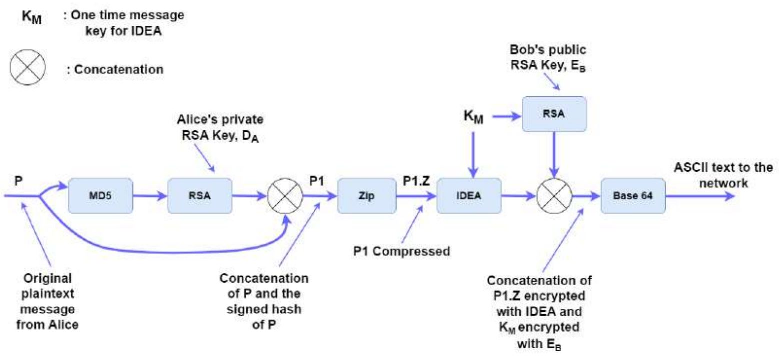
Network Security
Professor Gaurav S. Kasbekar
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Week - 05
Lecture - 29
Secure Email: Part 2

Hello, recall that we started our discussion of PGP in the previous lecture. We will now continue our discussion of PGP. Subsequently, we will discuss another system for secure email, namely S/MIME. So, suppose Alice wants to send an email to Bob. We introduced this notation last time.

D_A and D_B denote Alice and Bob's private keys, and E_A and E_B denote their public keys. Let P denote the plaintext message. We summarize the architecture of this system that is used to transform the original email message. So, we summarized this in the previous lecture. We now discuss this architecture in detail.

- Let D_A and D_B denote Alice and Bob's private keys, and E_A and E_B denote their public keys
- Let P denote plaintext message


K_M : One time message key for IDEA
: Concatenation
P
MD5
Alice's private RSA Key, D_A
RSA
P1
Zip
P1.Z
IDEA
K_M
Bob's public RSA Key, E_B
RSA
Base 64
ASCII text to the network
Original plaintext message from Alice
Concatenation of P and the signed hash of P
P1 Compressed
Concatenation of P1.Z encrypted with IDEA and K_M encrypted with E_B

So, P is the original email message. The hash of P is found using the cryptographic hash function MD5, and it is signed using the private key of Alice, that is D_A. So, this shows the computation of the cryptographic hash function MD5. So, the hash of P is found, and then it is signed using D_A. So, this is the digital signature.

The digital signature is obtained by finding the hash of P and then signing it using Alice's private key, D_A. So, this is the digital signature. And then the original message P is concatenated with the digital signature to get the digitally signed message. Let's call it P1. So, this is the digitally signed message P1 at this point.

Next, the message may be large, so we need to compress it. P1 is compressed using the ZIP program, which uses Lempel-Ziv compression. So, this shows the compression algorithm. P1 is compressed to get P1.Z, which is the compressed version of the message. So, this is P1.Z is the compressed version of the message with its digital signature.

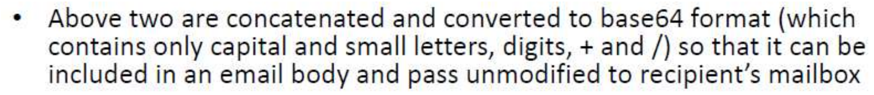
Next, we want to perform symmetric key encryption of the compressed message P1.Z. But for that, we need a secure symmetric key, which is shared with the receiver, Bob. So, for this purpose, the PGP prompts the sender Alice for some random input. So, Alice types some random characters on the keyboard. So, the content that Alice types as well as the typing speed, so these are random, so these are used to generate a 128-bit key, let's call it K_M.

So, this K_M is the random key that is used as the symmetric key for encryption using IDEA. So, this K_M is generated in this fashion. Alice just types a random input, and the content and typing speed are used to generate this 128-bit key, which is K_M. Next, the compressed message P1.Z is encrypted using IDEA with K_M as the key used for symmetric key encryption. This shows the encryption.

So, at this point, we have the encrypted message. The message P1.Z is encrypted using the IDEA cipher and the secret key K_M. Next, K_M is required at the receiver side, Bob, in order to decrypt the message. So, to transfer the key K_M securely to the receiver, K_M is encrypted using RSA, which is a public key cryptography system, and Bob's public key E_B. So, that encryption is shown here.

K_M is encrypted using RSA and Bob's public key E_B. So, this is the encrypted version of K_M. So, these two are concatenated, that is the encrypted message and the encrypted secret key, K_M. These are concatenated. So, at this point, we have the concatenation of P1.Z encrypted with IDEA and K_M encrypted with E_B.

So, here we have the encrypted compressed message and the encrypted secret key. Now, after concatenation, these are converted to base64 format, which contains only capital and small letters, digits, +, and /. So, there are 26 letters; hence, including capital and small letters, there are 52, then 10 digits, +, and /. So, there are 64 distinct characters in this format. And the reason for converting the message to base64 format is that it can then be included in an email body and passed unmodified to the recipient's mailbox.



- Above two are concatenated and converted to base64 format (which contains only capital and small letters, digits, + and /) so that it can be included in an email body and pass unmodified to recipient's mailbox

This has a historical reason for conversion to base64 format. Some old transmission protocols transformed some byte values. So, some byte values that appear in the ASCII code had special meanings to these protocols. So, they transformed some byte values, and that resulted in the modification of the email message. But these 64 characters were not changed by any transmission protocol.

So, for this reason, we convert to base64 format. So, it is a simple conversion of ASCII to base64 format, and then the message in base64 format is sent to the recipient's mailbox so that the transmission protocol is used for transferring the message; they don't modify the message. So, in this block diagram, this shows the base64 conversion, and after conversion into base64 format, it is sent over the network. So, how do we convert from ASCII format to base64 format? Bytes are grouped into groups of three bytes each.

So, three bytes have 24 bits. So, bytes are grouped into groups of three bytes each, and then four characters are used to encode each group. So, hence, every character will have 24 divided by 4, which is 6 bits. Every character will correspond to six bits, so 6 bits can encode 64 distinct characters, hence, base64 format requires characters of 6 bits each. So, this is how the conversion from ASCII to base64 is done.

We group bytes into sets of three bytes each, and then four characters are used to encode each group. So, that converts the message from ASCII to base64, and then the message is sent over the network. Now, let's discuss what happens at the receiver side. So, the receiver is Bob in this case. When Bob gets the message, he reverses the base64 encoding and converts the message back to ASCII.

Because the message has already been transferred over the network, due to base64 encoding, the transmission protocols do not make any changes to the message. But after reception, in order for Bob to be able to read the message, he reverses the base64 encoding and converts it back to ASCII. Then Bob decrypts the IDEA key K_M using D_B, which is Bob's private key. So, at the sender side, recall that the IDEA key K_M was encrypted using Bob's public key, so Bob reverses that process and decrypts the IDEA key K_M using D_B.

So, Bob knows the secret key K_M at this point, and he can use that to decrypt the message using the symmetric key decryption process of IDEA.

Now, Bob decrypts the message using K_M to get the compressed version of P1.Z and then decompresses it by reversing the Lempel-Ziv compression. So, Bob has obtained the decompressed version of the message. Now, Bob needs to check the integrity of the message. So, Bob has obtained the message P1 after decompressing the message. So, he separates P1 into P and the signature of P and verifies the message integrity.

That is that the message has not been tampered with and that Alice is indeed the sender using the public key of Alice, that is E_A. Recall that the digital signature created by Alice is $K^{-}(H(P))$, and then Bob applies Alice's public key, that is, K^{+}, to it to get back H(P) and checks whether that equals the hash of the message P. If yes, then the message integrity verification is successful. So, Bob verifies the digital signature in this step, and through this process, Bob is able to check the integrity of the message. One observation from this PGP is that RSA, which is a public key crypto system and it is slow, RSA is only used for signing the 128-bit MD5 hash of P and to encrypt the 128-bit idea key. So, RSA is used once at this point to sign the 128-bit MD5 hash of P, and it is used at this point to encrypt the 128-bit IDEA key.

So, the inputs of RSA in each case are just 128-bit numbers. So, hence, RSA can process them reasonably fast. So, RSA is not used for encrypting the entire message P, which can be very long. So, RSA is only used to process these values, the 128-bit MD5 hash of P and to encrypt the 128-bit IDEA key. So, that doesn't take much time since these are just 128-bit inputs.

The encryption of P, which may be long, is done using IDEA, which is a symmetric key cipher, and it is much faster than RSA. So, there is a combination of symmetric key cryptography and public key cryptography, which is used for transferring the message securely to the receiver side. So, in summary, PGP provides confidentiality, message integrity, and compression. So, the confidentiality is achieved through encryption using the IDEA cipher. Then, message integrity is provided by the digital signature, which is created at this point.

And compression is provided by the Lempel-Ziv algorithm, which is implemented in the ZIP program. So, that is at this point. So, PGP provides all these desirable properties of confidentiality, message integrity, and compression. So, now we discuss the mechanism used in PGP to distribute public keys. The mechanism is known as the Web of Trust.

It is an alternative to certification authorities. Recall that certification authorities, such as Symantec, Comodo, GoDaddy, and GlobalSign, can provide certificates that bind the public key to an identity, such as Robert John Smith, freesoft.org, and so on. So, certificates bind public keys to an identity. But what is the identity in the case of PGP? PGP secures email.

So, email addresses are the identities to which public keys are bound. So, in this case, in the context of PGP, we require certificates that bind public keys to email addresses. One of the objectives when PGP was designed was to protect email against government intrusion. So, hence, it was not always safe to use certification authorities because governments may interfere with certification authorities, and their security may be compromised. So, an alternative scheme called Web of Trust was used.

This Web of Trust scheme is an alternative to certification by CAs. So, this scheme was designed and was used in PGP. In this Web of Trust scheme, public keys are certified by users, and the certifier is an email address. So, notice the contrast with the CA architecture, where public keys are certified by certification authorities. In contrast, in Web of Trust, public keys are certified by users who can be ordinary users.

Each user decides whom they trust and how much they trust them. So, there is this additional feature in Web of Trust: confidence levels. That is, the certificate includes a confidence level which indicates how much the user trusts the party for whom they provide the certificate. The certificate includes a confidence level that indicates how confident the signer is of the binding of the public key to the email address claimed in the certificate. So, there is this additional notion, confidence level, which is not there in the certificates with CAs that we discussed earlier.

And a given user may not be satisfied with only one certificate. A user may ask for several certificates attesting to the same key binding before he or she is willing to trust it. Here’s an example. Suppose you have a certificate for Bob provided by Alice, who is moderately trustworthy. Then, using just this one certificate, you may trust this certificate moderately.

For example, you may encrypt moderately sensitive documents but not highly sensitive documents using it. So, in this case, when you have only one certificate for Bob, which is provided by Alice, who is moderately trustworthy, you sign only moderately sensitive documents, but you refuse to sign highly sensitive documents using this because you don’t have much confidence in the certificate. But later, suppose you receive certificates for Bob provided by some users C and D, each of whom is also moderately trustworthy. Then, this may significantly increase your confidence because now you have certificates for Bob provided by three different users, Alice, C, and D. So, this increases your confidence to the point that you are willing to encrypt highly sensitive documents using this public key. So, instead of getting only one certificate signed by a highly trustworthy entity, you may be willing to accept multiple certificates signed by different entities who are all moderately trustworthy.

So, that is how the notion of confidence level is used. So, the certificate includes a confidence level which indicates how confident the signer is of the binding of the public key to the email address claimed in the certificate. Another useful feature in Web of Trust is that multiple private and public key pairs per user are supported. They are useful when one is compromised. So, for example, a user Bob may have multiple private keys and corresponding public keys.

So, in case one of these private keys is compromised, then Bob can use the other private key-public key pairs. So, these other private key-public key pairs are useful as backups when the primary private key-public key pair is compromised. So, this is how the Web of Trust operates. We discussed the notion of certificate revocation during our discussion of public infrastructure. Certificate revocation is also used in PGP.

We may need to revoke or undo a certificate. For example, suppose Bob suspects that one of his private keys is stolen. In that case, he wants to undo the corresponding self-signed certificate. So, in this case, we require a mechanism for revoking certificates. We discussed the mechanism for certificate revocation, namely CRLs, Certificate Revocation Lists.

So, in this scheme, each user publishes, for example, the user may post on a website and periodically updates a digitally signed list of certificates that he or she issued, which have now been revoked. This is the Certificate Revocation List. The Certificate Revocation List can be used to revoke certificates that were earlier issued. But now the corresponding private keys may have been leaked. So, those certificates are no longer valid.

So, CRLs are a mechanism used for revoking certificates. When Alice receives a certificate for Bob signed by C, she checks the latest CRL issued by C. So, if that CRL includes the certificate, then Alice will not trust that certificate. If the CRL does not contain the certificate, then Alice will trust the certificate and use it to send messages to Bob. So, this notion of certificate revocation is used in PGP.

So, this concludes our discussion of PGP for securing email. We now discuss another alternative system for securing email, which is S/MIME. This stands for Secure Multipurpose Internet Mail Extension. This is another alternative system for securing email. So, first, before discussing S/MIME, we discuss MIME briefly.

MIME stands for Multipurpose Internet Mail Extension. It specifies a format for encoding arbitrary data in email text or including it as attachments. For example, pictures, rich text, audio and video files, binary files, and so on. These can be encoded as part of MIME. S/MIME is a security enhancement to the MIME email format standard.

So, it provides security mechanisms similar to those that PGP provides, namely message integrity, confidentiality, compression, and key management. The functionalities of S/MIME and PGP are similar. So, we'll discuss the block diagrams of S/MIME later, and we'll see that there are a lot of similarities between S/MIME and PGP. But the main differences between S/MIME and PGP are as follows: One is that S/MIME uses X.509 certificates that are issued by certification authorities or CAs.

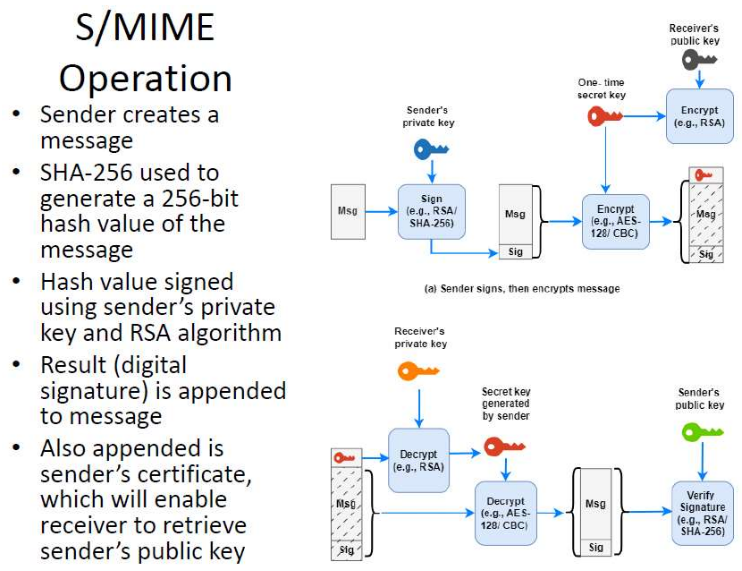
1) S/MIME uses X.509 certificates that are issued by Certification Authorities (CAs)

Recall that, in contrast, PGP uses Web of Trust. So, X.509 certificates issued by CAs are a standard mechanism for distributing public keys. So, S/MIME uses that, but one of the objectives of PGP was to guard against interference by governments. So, for that reason, it tried to avoid CAs and used the mechanism Web of Trust. So, that's one difference between S/MIME and PGP.

Another difference is that in S/MIME, the sender's certificate is included in the email message, from which the receiver can extract the sender's public key. And this public key is then used for verifying the digital signature. So, in contrast, recall that in PGP, recipients need to separately obtain the sender's public key. So, that's another difference. In S/MIME, the certificate is included in the email message, from which the public key of the sender can be obtained by the receiver.

Whereas in PGP, the recipients need to separately obtain the sender's public key. We now discuss the operation of S/MIME. The upper figure shows the operations at the sender, and the lower figure shows the operations at the receiver. The sender creates a message. That message is shown here.

And then a digital signature of the message is created for message integrity. Again, that digital signature is created by hashing the message and then applying the private key of the sender to it. So, as the cryptographic hash function, we use SHA-256 to generate a 256-bit hash value of the message. The message is hashed to find its hash value using the cryptographic hash function SHA-256. And then the hash value is signed using the sender's private key and the RSA algorithm.



S/MIME Operation

- Sender creates a message
- SHA-256 used to generate a 256-bit hash value of the message
- Hash value signed using sender's private key and RSA algorithm
- Result (digital signature) is appended to message
- Also appended is sender's certificate, which will enable receiver to retrieve sender's public key

Receiver's public key
One- time secret key
Encrypt (e.g., RSA)
Sender's private key
Msg
Sign (e.g., RSA/ SHA-256)
Msg
Sig
Encrypt (e.g., AES-128/ CBC)
Msg
Sig
(a) Sender signs, then encrypts message
Receiver's private key
Secret key generated by sender
Sender's public key
Decrypt (e.g., RSA)
Msg
Sig
Decrypt (e.g., AES-128/ CBC)
Msg
Sig
Verify Signature (e.g., RSA/ SHA-256)

So, this is the sender's private key, and it is used to sign the message. So, in case Alice is the sender, then $K^{-}(H(M))$ is the digital signature, as we discussed earlier. So, the private key is K^{-}, and the hashed message is H(M). So, this is the digital signature of the message. This Sig is the digital signature of the message. So, the digital signature is appended to the message, and that is shown here.

This is the original message, along with its signature, which is appended to it. Also appended is the sender's certificate, which will enable the receiver to retrieve the sender's public key. So, along with the message and its digital signature, the sender also includes the sender's certificate. That is not shown in this picture, but it is included along with the message and its digital signature. Then, the signed message is so far in the clear, so it needs to be encrypted.

The signed message is encrypted, typically using AES with a one-time secret key. For example, the key may be 128 bits long, and cipher block chaining is typically used. So, recall that to encrypt long messages with a block cipher, CBC is one of the techniques that can be used for encrypting long messages using a block cipher. So, AES is a block cipher, and for encrypting an email message using AES, we use it in combination with CBC, that is, cipher block chaining. So, this shows the encryption process.

A one-time secret key is used, and the message along with its signature is encrypted using AES with a 128-bit key and CBC, that is, cipher block chaining. A new random number is generated for each message and used as the one-time secret key. So, this is similar to PGP. So, a one-time secret key is used, and it is different for different messages. The one-time secret key is encrypted using RSA and the receiver's public key.

So, this is again similar to PGP. This shows the encryption of the one-time secret key using the receiver's public key. Again, a combination of symmetric key cryptography and public key cryptography is used. Public key cryptography is used to share the key used for symmetric key encryption, and symmetric key encryption is used to encrypt the actual message and its digital signature. Note that to reduce the encryption time, we use a combination of symmetric and public key encryption.

This is used in preference to using public key encryption to encrypt the whole message. So, that would be very time-consuming. So, to reduce the overhead, we use a combination of public key and symmetric key encryption. Compression can be used to reduce the email size. So, we call that the zip program is used in PGP.

Similarly, in S/MIME, we can use compression to reduce the email size. Now, this concludes our discussion of the sender's actions. Now, we discuss the actions at the receiver. So, the actions at the receiver are essentially the reverse of the actions performed at the sender. First, the receiver decrypts the one-time secret key using his or her own private key.

So, this shows the decryption of the one-time secret key using the private key of the receiver. So, recall that the one-time secret key was encrypted using the public key of the receiver. Now, to reverse that process, the receiver uses their private key to decrypt the secret key. And this is the secret key that is generated by the sender after decryption using the receiver's private key. And now, once the secret key is obtained by the receiver, the receiver decrypts the email message itself using the one-time secret key.

So, using this one-time secret key, the receiver takes a message which is in encrypted form and decrypts it. So, this process AES CBC is reversed to obtain the original message and its digital signature. And then the receiver verifies the signature using the sender's public key, which is extracted from the server certificate. Recall that the sender's certificate is included along with the message and its digital signature. The receiver uses that certificate to obtain the sender's public key.

Now, the sender had signed the message by taking its cryptographic hash function and applying their private key to it. The receiver applies the corresponding public key of the sender to the digital signature and verifies the digital signature. So, if M is the message, then $K^{-}\left(H(M)\right)$ is the digital signature, and by applying the public key to it, the receiver gets H(M) and checks whether that is equal to the hash of the original message M. So, in this way, the receiver verifies the integrity of the message. If the integrity check is successful, in that case, the receiver knows that the email message has not been tampered with and it has been sent by the legitimate sender. So, the receiver can go ahead and use the email.

The receiver can read the email and process it. So, in this way, S/MIME adds security mechanisms to email messages. So, in summary, we discussed different mechanisms for providing security to email. We discussed PGP first, and then we discussed S/MIME. So, as we have seen, the operations of PGP and S/MIME are very similar, but there are some differences which we discussed.

So, either of these systems may be used for achieving security of email. So, email security is one example of providing security at the application layer. Thank you.