PGP

Lets have a look at the zip technology first and then we will discuss radix 64. Lets say that we have this message “the brown fox jumped over the brown foxie jumping frog” with the compression what we are doing is we are checking the repetition of the word and replace something with something else and that is the whole idea, so im going to look at the repetition and those repetition will be removed from those code and for decompression im going to remove those code and replace original names and sentences. So that is the idea to do that I have two technique I have 8 bit pointer and 4 bit length , information or 12 bit pointer and 6bit length information why we need it I will explain , if we look at these the brown fox jump over or the brown foxie jumping frog ,do you see any repetition somewhere the brown fox is repetition what is the length of this ?..including spaces is 13 characters and im going to replace this one with a code and that code will have the pointer and the length information so the on the decompression side I will be able to replace it ,what im going to do is im going to live the character brown fox and im going to count the characters to this beginning how many characters ?..26, so this brown fox is going to be replaced 26 13 code and it says im using 8 bit then this will be 0 and if I m using 12 bit, its goingto be 1 ,so instead of this brown fox im going to write 0 26 13 after over so whne I receive this compressed message , I have to recover the brown fox, what im going to od is im going to go to pointer and it will point the beginning and it will tell take 13 characters from there and 13 characters is the brown fox jump, then this code will be replaced with the brown fox ,s o this is the algorithm as you see it is a simple algorithm , do you see any other thing over here?..this one right, right here this jump including space so how many characters?.length is 5 and im going to replace with space and jump so you are going to count all the characters upto this point that’s going to be the pointer and they counter already it is 27 so it is 0 27 5 is the code that is replaces the space jump , so on the decompression side they are going to check the pointer and find the beginning and take 5 characters from that point onwards and im going to take 5 pointer from that message and im going to recover the original message ,this is called LZ77 which ZIP is using it is using in more liberate way ,because it is not easy to explain repetition they have very good algorithm to detect those repetitions but some times if there are not much repetitions inside the text when you compress the text , compressed size of text might be bigger than original text for example for 5 characters I already replaced 0 5 , 6 ,5 character based on number of the character that im using in other words if you are using small sentences or words then you might be increasing the size of the message it really saves lot of space and the name of the algorithm is LZ77 and the second important thing is the email compatibility , radix 64 conversion,so what we are doing is , we are doing is we are doing after the encryption because encrypted file is binary file and im going to select 24 bits, so this 24 bits are going to be blocked into 6 bits ,and then im going to have a table it is called radix 64 encoding for each 6 nbit they assign a character for example 0 is for A and 24 is for Y and 47 for v, they are going to change it to the character based on the radix 64 table , these characters are represented with 8 bit ASCII code , so then what im going to have is im going to have 4 character ,and 4 character with the ASCII is goingto be transferred to the destination so lets do one example, in the example I have this 24 bit, 00110011, 01011100,10010001, three 8 bit block total 24 bits what im going to do is im going to split into 6 bits , this is 1 block and 2nd block and the 3rd and the 4th , what is the decimal equivalent of this number ,t hat is 8 what about other this 8,53 , 50 and this is 17 are we clear so what we will do is we are going to go radix 64 look up table, and what Is the 8

represent I, 53 is 1, 50 is y, and then 17 is R, so you see I changed the binary data into characters and then these I, 1 , y,R is going to be represented using 8 bit ASCII, is going to be used, so this way now I have the text and I can transfer the text to destination so when I received this message on the destination side what im going to do?...i have to go back here as this is encrypted file and if I decrypt this , we are not going to get anything so in other words we have to go back from ascii to this and from this to this ,and then reconstruct the 24 block, when you d o that you recover the encrypt the email and then you are going to decrypt the email and this is called basic radix 64 email compatibility , that PGP is using, clear so far?.. so lets have what pgp does for authentication here is the message this is email hash is going to be email and hash of the message is going to be encrypted with the private key of source A that is this is DS source A is signing the mail and this signing the mail and the result is zipped , compressed file and transferred it to the destination on the side first of all we have to unzip and recover the original message and from there calculate the hash and then from then the signature part and decrypt the signature part by using public key of source A and recover the hash and recalculate and recover message is compared and if this Is same then it is successful and I know that only source A can send this email because only source A can sign this message so this is the authentication part. Let have a look at confidentiality message is going to be zipped and then the message is going to be encrypted with the session key if the message is for idea algorithm then this key is for those guys but I have distribute this key im going to be encrypt this key with the public key of destination B and appended to the message ,message is encrypted so the format of the message and message is encrypted with the session key and the session key is encrypted with the public key of the destination, but before encryption I Zipped the file so this encryption is on the compressed file ,when destination receives this message ,encrypted session key and encrypted message with the session key first of al it is going to recover session key and it is going to decipher this part by using the corresponding private key so session key is recovered now it can decipher this whole message so the decipherment result is compressed message ,when you unzip it you have the original message back , so this way we are providing confidentiality and also key distribution with the first one we have the authentication so what is missing over here ?.. some thing is missing , in this, you cannot transfer message like this it is a binary file , before sending it you have to convert this one into radix 64 conversion and it is going to come here and on the destination side inverse radix 64 will play an important , this just shows confidentiality n but it has to come u p with it but I want confidentiality and authentication what im going to do this is the overall structure of PGP it does this message hash will be calculated and encrypted with private key of source A this is the signature part and then appended to the message so the message has message itself and the signature ,so the signature is calculated on original message and then signature and message compressed and signature message is going to be encrypted with the session key and the session key is encrypted with the public key and send to the message , so what im saying is this, I have that message and then I m calculating the signature ,what is the signature ,that is encrypted with the private key and this whole thing is going to go to zip and it will be encrypted with session key and it will be encrypted with the public key this is going to be the over all message for the PGP so you received this email, so this email you received what ware you going to do? First of all you are going to use private key to decipher to get the session key and when we encrypt it the result is zipped version of it and so unzip it , then decipher the whole message and am I right?..no we did the encryption after zip we going to do the decryption before zip so this is the decryption part then unzip it ,because result of the decryption is the

compression version of the file , unzip it now I have the signature and the there we are , you have confidentiality , you have the authentication you have integrity and key distribution , 4 major services that you need key distribution ,but this is not going to end here there are lot of problems as we shouldn't using radix 64 her , as they are radi64 conversion this is just a beginning of the confidentiality and authentication in pgp ,because the important thing is going to be for us is the key distribution but this time public key how you are going to get the public key of destination, pgp will come up with a new concept and that is called web of trust it is not going to use x509 , pgp is not using x509 either, pgp instead using web of trust , to accept someone s public key it is going to be required , for example imagine I do trust these guy 100% if I receive email from this guy this is my email that this is my public key and imagine I don't trust this lady 100% but I need her public key , but I know this guy is a friend of her and If he signs her public key and sends to me then I will accept his public key ,if I don't trust 100% and jus t I trust 50% a and 50% to get the public key from the lady I need to have two signature from the lady so it works something like this ,so you are going to assign a trust value for everybody you know , but if nobody knows this guy and I need his public key im struck because I cannot find any trustworthy entity to sign his key , it is called idle mode and it is upto you to decide whether to accept his public key or into it is going to work like this using web of trust concept but before going into that my point over here is key distribution is this , each user may have more than one public key , so how this user destination knows that when its going to use this public key corresponding private key but I have lot public private key pair ,which one of the key I have to use here and this is going to complicate a station a little bit , so how can you solve that problem ?..you have multiple pairs of public key s , if you receive multiple signs of message or PGP ,how do you know the sender use the particular key pair?..when you receive this one you will not receive public key information as this guy used some of your public key s and it is not sent and some of relation hasn't sent the public key and it is upto you to decide and you are assuming that for example I have one public and private key pair ,so you will say I have received first so im going to use it to recover the session key ,but the problem arises when you have multiple private and public key pairs as I said sender source doesn't send any information which key of yours is used for the encryption and also it is true for digital signature and how do you know if the sender has multiple keys which public key you are going to use to verify the signature to the sender, just set the public key as it is public , but although you are solving problem you are creating problem , as public keys are not similar entities remember 2048 bits along with the modulus you are going to increase the size to 4k immediately , and if you want to send to multiple users then the size of message will be huge and hash of the message what they are doing is they are least 64 bit of the message so that other have the probability that others have that least bit as same a 2 to power 64 and that 64 bit is the identifier of the key and we are going to include id of keys so that this thing will be little bit better