<u>TITTLE:</u>

Activity 1:

Install and send a message using PGP. http://openpgp.org/ There are many options for using PGP, try different ones and see how they work together.

Activity 2:

Send email using Python. Look at your email server settings. Update the info in the Python file. Python file: (download link) (https://gist.github.com/anir0y/be6d613dd3d793e54f3e6d2be1dd0f65)

- **Can you email multiple people?**
- **Could you pull the list of people to email from an external file?**
- **How can you personalize the email for the recipient?**

Discussion:

(1)What could you do to ensure privacy when sending email?

(2)What expectation of privacy do you have when sending e-mail?

(3)If you had a secret message to send, how would you do it?

(4) How could you automate e-mailing many people?

Assessment Questions:

(5)Why do email services "read" your email? What is their goal?

(6)How does PGP secure email differently than GMAIL?

(7)Why don't people use services like PGP more often?

(8)What is phishing?

(9)What is spear-phishing?

COURSE NAME: CYBER SECURITY (MAJOR PROJECT)

DUE DATE: 2/06/2020

CLASSROOM NAME: CS04B4

EXECUTIVE SUMMARY

The main objective of this major project was to able to understand and explore the security of e-mail, disposable email accounts and use a secure email

system. As we all know email is one of the most-used forms of communication, be it an IT employee or a college student, most of the important messages or announcements are communicated by the company or universities through these emails. Since these emails play a crucial role in communication, major question arise in everyone’s mind as to whether these emails or email service providers safe enough? Does anybody see my emails? How secure are they, if not what are the consequences? All these major questions are being discussed in the assignment. Are there any alternative solutions for these problems? Yes there are solutions for that and we have discussed one such solution i.e., sending email using PGP which is more secure form of email communication and performed few activities related to that.

We have also discussed sending automated mails using python and also performed a few activities under this section too.

CONTENT:

AN INTRODUCTION

PGP (**P**retty **G**ood **P**rivacy) encryption has become a mainstay of internet privacy and security for one main reason: it allows you to send a coded message to someone without having to share the code beforehand. There's a lot more to it, but this is the fundamental aspect that has made it so useful. It's a conundrum that PGP has managed to solve with something called public-key encryption (don't worry, we'll cover what this is later), which allows its users to send secure and encrypted messages to people even if they've never met before, let alone had a chance to prearrange a code.

PGP's core function is to enable its users to send secure messages without needing a prior introduction, but that's not all it does. It also allows recipients to verify whether a message is authentic or if it has been tampered with. It does this by using something called digital signatures, which we will cover later in the article.On top of this, PGP can be used to encrypt other things besides email. You can use it to encrypt your hard drive, instant messages, files and more. While these are all important features, this article will mainly focus on using PGP encryption for email, PGP's most widespread use. Despite the name–which was inspired by a radio program's fictional shop called Ralph's Pretty Good Grocery–PGP is a form of encryption that does not have any publicly known ways of being broken. This means that as long as it is used properly, you can be confident in the security, privacy and integrity of your messages and files.

Once data has been encrypted with the recipient's public key, it can only be decrypted by their private key. This is why public keys are freely handed out, but private keys need to be guarded carefully. If your private key is compromised by an attacker, it enables them to access all of your PGP encrypted emails.In PGP, public-key encryption isn't used to encrypt the message, just the one-off session key that was generated to encrypt it. Why? Because public-key encryption is simply too inefficient. It would take too long and use a larger amount of computational resources. Since the body of the message usually contains the bulk of the data, PGP uses the more economical symmetric-key encryption for this. It reserves the lumbering public-key encryption for the session key, making the whole process more efficient. The process makes digital signatures essentially impossible to forge unless the private key has been compromised. The message digest is then encrypted with the sender's private key.

Python comes with the built-in smtplib module for sending emails using the Simple Mail Transfer Protocol (SMTP). smtplib uses the RFC 821 protocol for SMTP. The examples in this tutorial will use the Gmail SMTP server to send emails, but the same principles apply to other email services.

There are two ways to start a secure connection with your email server:

Start an SMTP connection that is secured from the beginning using SMTP_SSL().

Start an unsecured SMTP connection that can then be encrypted using .starttls().

In both instances, Gmail will encrypt emails using TLS, as this is the more secure successor of SSL. As per Python's Security considerations, it is highly recommended that you use create_default_context() from the ssl module. This will load the system's trusted CA certificates, enable host name checking and certificate validation, and try to choose reasonably secure protocol and cipher settings.

If you want to check the encryption for an email in your Gmail inbox, go to More → Show original to see the encryption type listed under the Received header.

smtplib is Python's built-in module for sending emails to any Internet machine with an SMTP or ESMTP listener daemon.

Using .starttls()

Instead of using .SMTP_SSL() to create a connection that is secure from the outset, we can create an unsecured SMTP connection and encrypt it using .starttls().

To do this, create an instance of smtplib.SMTP, which encapsulates an SMTP connection and allows you access to its methods. I recommend defining your SMTP server and port at the beginning of your script to configure them easily.

The code snippet below uses the construction server = SMTP(), rather than the format with SMTP() as server: which we used in the previous example. To make sure that your code doesn't crash when something goes wrong, put your main code in a try block, and let an except block print any error messages to stdout.

TERMS OF REFERNCE

On 15/05/2020, we got this assignment from our mentor. We all are very excited about learning more facts about cybersecurity in the future from him. It was a great opportunity that we got to learn so many new things from him.

This project is basically a method to explore the security of e-mail, disposable email accounts and use a secure email system. Email is one of the most-used forms of communication. How secure are the systems that we use daily and what are the implications of insecure systems? We will also look at alternatives that would ensure security. To determine whether the message is authentic and hasn't been tampered with, all the recipient's program has to do is compare the message digest from the email they received against the message digest that they derived from the digital signature. Sometimes it can be difficult for new users to find someone to sign their certificates and verify the relationship between themselves and their public key. It can be particularly challenging if they don't know other PGP users in real life.

The due date of our report is 2nd June,2020. We got enough time to complete and it was a great opportunity to work under such a great mentor. This project let us know about so many new things. We feel glad to be a part of this. We put all our efforts to make this project wonderful. We all decided to be a part of this because we are having a great interest in cybersecurity and cybercrimes and also it was offering a certificate from verzeo and Microsoft so it was great for us to do this.

ACTIVITY 1

Install and send a message using PGP. http://openpgp.org/ There are many options for using PGP, try different ones and see how they work together.

PROCEDURE:

You'll have to reboot (restart) your system after the PGP (Pretty Good Privacy) software has been downloaded and installed, so save any work on your computer and quit any open programs other than your web browser before you proceed. This tutorial has been designed for users of Windows PCs. A version of the tutorial for the Mac platform is in the works and will be ready soon. The tutorial describes the basics of the PGP software in order to help beginners get up and running using encryption.

Back to the Table of Contents

Step 1: Downloading PGP

1. In your browser, go to the download center at the PGP International Homepage
2. Click on the PGP Freeware link where it says: "Here you may download the latest freeware PGP version for your platform."
3. On the next web page, you'll be asked to answer a few simple questions. You have to answer Yes to all the questions as a way for you to declare that you won't misuse the software once you've downloaded it. Read each of the questions carefully before selecting Yes as your answer. Then click on the Submit button.
4. On the next web page, click on the first download link under PGP Freeware v6.5.8 (Windows 95/98/NT/2000). You'll be prompted now to decide if you want to open the software right away or download it (Save it) to your computer hard drive. You want to Save this file to disk, so make sure this option is selected in the File Download dialog box, then click on OK.
5. Now you have to tell your browser where you want to save the PGP program. Be sure to select a location on your hard drive where later you'll be able to easily find the zip file of the PGP software, then click on the Save button. The download will take a while, depending on the speed of your connection to the web.

Back to the Table of Contents

While you're waiting.... Where did PGP come from and how does it work?

Where did PGP come from?

Rarely does anything of significance arise out of the blue. PGP (Pretty Good Privacy) is the culmination of a long history of cryptographic discoveries. Cryptography is the science of writing messages in secret codes. It is nothing new. Since the human race became a species of its own, we have pondered the challenge of concealing our communications from others. Secrecy--stealth--is not a preserve of the human species. It is a matter of survival for all our brothers, sisters and cousins in the animal world from which we have evolved. Whether in times of peace or in times of war, we all harbor secret thoughts, feelings, desires, objectives, and so forth that we want to share only with those we absolutely trust, and that we want to carefully conceal from those who would take advantage of us if they knew what we had in mind.
Encryption makes this possible, and one of the strongest encryption tools available to us today is PGP.

Phil Zimmermann invented PGP because he recognized that cryptography "is about the right to privacy, freedom of speech, freedom of political association, freedom of the press, freedom from unreasonable search and seizure, freedom to be left alone." You can read Phil Zimmermann's fuller explanation as to why you need PGP. In the development of PGP, Zimmermann was greatly assisted by his knowledge of the long history of cryptography. Like Sir Isaac Newton, Zimmermann was able to achieve what he achieved because he "stood on the shoulders of giants" who went before him.

How does PGP work?

OK, here goes; put your thinking cap on... If this gets overly technical for you, and your eyes start to glaze over, don't worry about it. It's nice if you can understand what's going on with Public and Private Key encryption, but it's not necessary right away. You'll understand it better as you start to use it and as you interact with others who use it and can explain what's going on. For now, it's sufficient to just follow the sets of numbered steps carefully in order to learn the skills required to use PGP. But read over what follows and understand it as best you can.

When you have successfully completed Step 3 of this tutorial, you'll have created two keys to lock and unlock the secrets of your encoded information. A key is a block or string of alphanumeric text (letters and numbers and other characters such as !, ?, or %) that is generated by PGP at your request using special encryption algorithms.

The first of the two keys you'll create is your Public Key, which you'll share with anyone you wish (the tutorial also will show you how you can put your Public Key on an international server so that even strangers could send you encrypted data if they wanted). Your Public Key is used to encrypt--put into secret code--a message so that its meaning is concealed to everyone except you

Then there is your Private Key, which you'll jealously guard by not sharing with anyone. The Private Key is used to decrypt--decode--the data (messages and so forth) that have been encrypted using your Public Key. This means that the message encrypted (encoded) using your Public Key can only be decrypted (decoded) by you, the owner of the corresponding Private Key.
The designation of one of the two keys (Key1, say) as Public and the other (Key2) as Private is purely arbitrary since there is no functional difference between the two. PGP chooses one to act as the Public Key and designates the other as the Private Key. If it chooses to designate them in the other order (Public=Key2 and Private=Key1), it would make no difference. This is because when either key is used to encrypt something, the other will act as the corresponding decrypting key to convert the encrypted data back into its original form. This capability is at the heart of the "Signing" process mentioned in Steps 8 through 10 below.
Public and Private Key encryption solves one of two major problems with older methods of encryption, namely that you had to somehow share the key with anyone you wanted to be able to read (decrypt) your secret message. The very act of sharing the key meant that some untrustworthy so-and-so could intercept it--and frequently did. Which meant your code was practically useless.

The second major problem with older methods of encryption was the relative ease with which the code could be broken. Codes have to be incredibly complex if they're to foil the attempts of astute humans to crack them. This is all the more the case today when we have increasingly powerful computers to do the dirty, "brute force," work of trying every conceivable combination of key possibilities for us. PGP, and other similar encryption systems, use a key that is really--well--astronomically large, meaning that the number of binary bits (1s and 0s) used to create it has an astronomically large number of possible combinations and the actual decimal (base 10) value they represent is--well--huge. Unlike earlier encryption methods, the security of PGP encryption lies entirely with the key. Earlier encryption methods relied on "security through obscurity" (ie: keeping secret the method used to do the encryption). The methods used to do PGP encryption are known and documented. It is PGP's selection of the complex keys used to do an encryption that makes it next to impossible to crack.

The size of the key can be increased whenever necessary to stay one step ahead of advances in technology. Time alone will tell if PGP can stand the test of time, but for now it's one of the best encryption technologies you'll find.
If you would like to read the history of encryption and understand the origins of Zimmermann's PGP program, an excellent account is given in Simon Singh's CODE BOOK (Doubleday, New York, NY, 1999). Find out more about PGP at the International PGP home page. The CryptoRights Foundation is another good website for information regarding privacy issues. You might also like to join the PGP-BASICS User group where you can find speedy and informed answers to questions that might arise as you get started using PGP. Once you're more

experienced with the program, you can join the PGP Users Mailing List so you can keep in touch with issues related to privacy.

Back to the Table of Contents

Step 2: Unzipping and installing the PGP software

Once the download is complete, you'll have the zipped version of the PGP program on your hard drive. Now you have to unzip it. For this, your best bet is to use the shareware WinZip which you'll need to have installed on your computer. You may already have this program from when you had to install other software. You can check if you have the WinZip program by simply double clicking on the file you just downloaded (PGPFW658Win32.zip). If you don't have WinZip installed on your computer, or if you're in doubt, you can go get it (download it) from the web. The best place to do this is at http://www.winzip.com/ddchomea.htm. Follow the directions to install the WinZip software on your computer. With a program such as WinZip installed on your computer, you are now ready to unzip and install PGP. Here are the steps to follow:

1. Now locate the PGP zip file PGPFW658Win32.zip wherever you downloaded it on your computer and double click on it to unzip it.
2. Next you'll be prompted to run the PGP Setup program, which will begin immediately. You'll be presented right away with the PGP Freeware 6.5.8 Setup Wizard.
3. Follow the Wizard's step-by-step directions, clicking on the Next button as you go along. The first three screens contain info about PGP (licensing, etc.). Read them before clicking on the Next button.
4. The 4th screen asks you to enter your name and the name of the company you work for. Next you're asked to confirm the folder where you want the PGP Wizard to install the program. Unless you have other ideas, accept the default for this item.
5. Keep your wits about you on the next screen, which asks you which PGP components you want to install. You do NOT need to install PGPnet Virtual Private Networking, since for most people it's unnecessary and, as Steve Kinney points out, can create nasty network configuration problems. So Deselect this item by removing the check mark next to it before clicking on the Next button.
6. On each of the ensuing screens, read what the Wizard has to say. When asked, accept the defaults and let the Wizard do all the setup for you.
7. Once the PGP software is installed, you will have to reboot your system. PGP will prompt you to go ahead and Restart.
8. After your system has been restarted, you are ready to create your Public and Private Keys.

Back to the Table of Contents

Step 3: Setting up (Creating) your Public and Private PGP keys

Now that you have the PGP software installed on your computer, you need to create a Public and Private Key pair. This you can do at any time. Remember as you complete the steps that follow that your Public Key is so-called because you will willingly share it with others so that they can use it to send you secret information. Your Private Key is so-called because it alone will decode any information encoded with your Public Key. As long as you alone have knowledge of your Private Key, your privacy is assured. Here are the steps to follow:

1. Open PGPkeys by selecting Start/Programs/PGP/PGPkeys or by clicking on the PGPtray icon in the lower right corner of your screen and selecting PGPkeys in the pop up menu.
2. The PGPkeys window opens up, listing various people's Public Keys, among which in a short while will be yours and any others (your correspondents) that you choose to add to the list.
3. In the PGPkeys menu bar, click on the Generate New Keypair icon to bring up the PGP Key Generation Wizard. Read the introductory dialog, then click on Next.
4. The PGP Key Generation Wizard now asks you to enter your name and e-mail address. Do this now. You can use any name you like and it's a good idea to use a genuine e-mail address so you can take advantage of the PGP feature which will look up the correct key for you when you are writing to a particular correspondent. Click Next when you're done entering your name and e-mail address.
5. Now the PGP Key Generation Wizard asks you to select a key type. Accept the default (Diffie-Hellman/DSS) and click Next.
6. The PGP Key Generation Wizard next asks you to specify a size for your new keys. Again accept the default (2048 bits, which will give you a key so large that it would be well nigh impossible to figure out even by the most powerful computer in the world) and click Next.
7. Now the PGP Key Generation Wizard asks you when you want your key pair to expire. Accept the default (Key pair never expires) and click Next again.
8. The PGP Key Generation Wizard now asks you to enter a passphrase. Think about this before you proceed. Choose a passphrase that has at least eight (8) characters (that's a minimum of 8 characters as a requirement), with a mix of upper and lowercase letters or other characters. The greater the mix of characters and the longer the passphrase, the better. As Herb Kanner explains, "The size of the passphrase, and the inclusion of mixed case and non-alphabetics is to increase the difficulty of a brute force attack on your passphrase." So, if you use a longer passphrase (Herb's is 15 characters long, and Bernie's is 33!!) even if someone used a supercomputer, it would take an intolerably long time for it to try all combinations till it hit on your passphrasse. If you'd like to read more about this important subject of Passphrases, take a look at The Passphrase FAQ. Arnold G. Reinhold's DiceWare Passphrase HomePage is another excellent resource.
9. Enter your passphrase once you've decided what it will be, hit Tab, and re-enter it for confirmation. Then click Next again.
10. If you have entered an inadequate passphrase, the PGP Wizard will warn you and ask you to go back and re-enter another passphrase. But if all is

well, the PGP Key Generation Wizard will now go ahead and generate your key pair. You may be prompted to move your mouse around or hit random keys on the keyboard to help the Wizard create a more secure key. Click Next when the Wizard has finished generating your key.

11. You'll now be asked if you want to send your new Public Key to a server where others around the globe can find it and use it when they want to encrypt data they wish to send you. This is optional, so click in the box only if you wish to do this, then click on Next once more.

That's it! You're done creating your PGP Public and Private Keys. Now all you have to do is share your Public Key with anyone with whom you wish to exchange secure information. The next sections tell you how to do this, and how to use your key and those of your correspondents to encrypt and decrypt the data that you exchange.

Back to the Table of Contents

Step 4: Changing your Passphrase

After a while, as you become more accustomed to using PGP, you may well want to change your passphrase, especially if the one you first chose is not complex enough for your liking, or if it has become compromised by someone else discovering what it is. Changing your Passphrase is a simple process.

To change your Passphrase, here's all you do:

1. Open PGPkeys by selecting Start/Programs/PGP/PGPkeys or by clicking again on the PGPtray icon in the lower right corner of your screen and selecting PGPkeys in the pop up menu.
2. Highlight the key you want to change the passphrase for, then from the Keys menu select Properties.
3. In the dialog box that pops up on the screen, you'll see the button to Change Passphrase. Click on Change Passphrase, and in the next dialog box, as you might expect, you're asked to enter your current Passphrase. Go ahead and do this, then click on OK.
4. Now all you have to do is decide on a new Passphrase, write it down if necessary so you don't forget it, then in large letters write on the note the word "DESTROY" or "BURN" to remind yourself to do this once you've used the new Passphrase often enough to know it by heart.
5. When you're ready, enter it in the New Passphrase dialog box, and Confirm the New Passphrase by entering it again, then click on OK.

Back to the Table of Contents

Step 5: Distributing your Public Key

When you want to exchange Public Keys with a particular individual or group of individuals with whom you intend to exchange encrypted information, the best way to do this is to send it as an e-mail to whoever you want to have it. Read what follows carefully, however, so you understand how PGP works.

The recipient of your Public Key will have to have PGP installed on their own computer if they want to be able to add your Public Key to their keyring and use it to encrypt the data they want to send you. Likewise, you must have anyone else's Public Key on your keyring in PGPKeys if you want to send them encrypted

data. This is a bit tricky to understand at first, but think about it. Anyone who uses PGP has two keys, a Public Key and a Private Key. Your Public Key is used by other people to encrypt information they want to send you so no one else but you can know what the information contains. When you receive an encrypted message from someone (could be any kind of data, not just text), you use your Private Key to decrypt it. The neat thing is that you're the only person who can decrypt the secret message because you're the only person who has the Private Key, with the passphrase that unlocks it (unless you share your Passphrase and Private Key with someone else, which would defeat the purpose of PGP!).

If you want to, you can put your Public Key on one or more servers that form an international server chain. Effectively, this makes your Public Key available to anyone anywhere who would like to exchange secure communications with you. Step 6 below explains how to do this.

To include your Public Key in an e-mail message, here's all you do:

1. Open PGPkeys by selecting Start/Programs/PGP/PGPkeys or by clicking again on the PGPtray icon in the lower right corner of your screen and selecting PGPkeys in the pop up menu.
2. Locate your keypair among the list of keys in the dialog box and select it (by clicking once on it). Then copy it (Edit/Copy or control-C)
3. Start a new message in your e-mail editor, in the To: box enter the e-mail address of the recipient, and type a subject header such as "My Public Key"
4. Now click to put the cursor in the body of the e-mail, Paste your Public Key (Edit/Paste or control-V) into the body of the e-mail, and send it.

Back to the Table of Contents

Step 6: Making your Public Key available through a certificate server
It's a good idea eventually to place your Public Key(s) on what's called a public certificate server. This is a server where anyone can access your Public Key and use it to send you encrypted messages. You'll still be the only one who can decrypt the message because you alone have the Private Key, so you never need worry that your privacy will be compromised just because you made your Public Key public. After all, that's why it's called a Public Key. However, as a beginner to PGP, you may not want to do this right away, since you may well decide to change your Public Key at a later date for one reason or another. The thing is that, once you put your Public Key on a certificate server, you can't remove it--ever, and there's no point littering the server with keys that are never going to be used. So keep this section of the tutorial in mind for later, after you've got used to using the program and have settled into using a particular Public Key.

Here, then, are the simple steps to make your Public Key available through the certificate server at MIT. It doesn't matter which server you post your Public Key to, by the way, since they are all interlinked. Wherever you post your Public Key, it will be available worldwide.

1. Start by connecting to the internet, so that PGP can access the web site (in our case a server at MIT) where your Public Key can be sent and included in the database of Public Keys.
2. Open PGPkeys by selecting Start/Programs/PGP/PGPkeys or by clicking on the PGPtray icon in the lower right corner of your screen and selecting PGPkeys in the pop up menu.
3. In the PGPKeys window, among the list of keys you see there, click on the icon representing your Public Key. This is the key you want to post to the certificate server at MIT.
4. Now pull down the Server menu, select Send to and then select the link to the MIT server at http://pgpkeys.mit.edu:11371.
5. PGP will now access the server for you and post your Public Key there. When it's done, it'll inform you that the key was posted successfully.

Back to the Table of Contents

Step 7: Obtaining and Adding someone else's Public Key to your keyring
Once again this is simple enough. There are two ways to do this. You can either have someone send you their key in an e-mail and then paste it into your keyring from their e-mail or, if they have their key already posted to a certificate server, you can go get it yourself. Here is all you do if you get someone's public key in an e-mail:

1. First you tell your friend or friends to follow Step 5 above to send you their Public Key in an e-mail message.
2. Open the e-mail message containing the Public Key you wish to add to your keyring.
3. Drag to select from -----BEGIN PGP PUBLIC KEY BLOCK----- all the way down to -----END PUBLIC KEY BLOCK-----.
4. Then copy it (Edit/Copy or control-C)
5. Open PGPkeys by selecting Start/Programs/PGP/PGPkeys or by clicking again on the PGPtray icon in the lower right corner of your screen and selecting PGPkeys in the pop up menu.
6. In the PGPkeys window, paste the Public Key you wish to add to your keyring (Edit/Paste or control-V).

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP Personal Privacy 6.5.8
mQGiBDYehAsRBAD9Mmgv7gV3JsMQ0sMLWjk9zX4O07h2XHUow0P5hTIL+lEtcjBn
W6152pufGD0cZjEdVHctH9x/nTOoUdcepTLyIHoLVD+MboepvlizEvqfCQNWEadH
BRE3fzSMqxWTl9nAQyXjqDBi+M92mswMQXUNqhg7cxk50Aymb2rQFoc1PwCg/0mu
fQgQ0UmNTKWfovuGhxKV0/8D/1luI30t5sb4DHFB4mgY9an0r0PtSYdHU3M5jrIx
kbHnWPYtydKNWkhBOncZJOnARAQ6q9SzmdoS9bzVIf0FXAVB3TG7IqgkaxXzkCAw
xkyrcxLjuT0GAbFg3t0kAqzsVnmIgfTCCycg/Xfnn+0Nak0Q06yHtOsPz2g8xhYa
K0MDBACIPH5tpJekxd+fZtF4dHqEotrXPcslPECi3BZELAEsntoAHRS/hYtQUFFZ
7bls3/wdMYX9etlxUbfUXhdxtuxJnpT2S0VoVI4h53cnAAhe8jzCOK5qVBUXSsjX
gYgD03BcXfwM4pMIpxbk+5i+oE5E0w2hIH9sfKFbgLHBaj0ZxLQyUm9iZXJ0IEEu
IFJvc2VuYmVyZyA8Ym9iLnJvc2VuYmVyZ0BkaWdpdHNjb3JwLmNvbT6JAEsEEBEC
AAsFAjYehAsECwMCAQAKCRCnsU6BJsi/YQuOAKC8K2jIl841wmyzLE2EXxE9u1Hw
XACg3X9G1Ad9jNGOkGVOLugXnxE36FS5Aw0ENh6ECxAMAMwdd1ckOErixPDojhNn
l06SE2H22+slDhf99pj3yHx5sHIdOHX79sFzxIMRJitDYMPj6NYK/aEoJguuqa6z
ZQ+iAFMBoHzWq6MSHvoPKs4fdIRPyvMX86RA6dfSd7ZCLQI2wSbLaF6dfJgJCo1+
Le3kXXn11JJPmxiO/CqnS3wy9kJXtwh/CBdyorrWqULzBej5UxE5T7bxbrlLOCDa
AadWoxTpj0BV89AHxstDqZSt90xkhkn4DIO9ZekX1KHTUPj1WV/cdlJPPT2N286Z
4VeSWc39uK50T8X8dryDxUcwYc58yWb/Ffm7/ZFexwGq01uejaClcjrUGvC/RgBY
K+X0iP1YTknbzSC0neSRBzZrM2w4DUUdD3yIsxx8Wy2O9vPJI8BD8KVbGI2Ou1WM
uF040zT9fBdXQ6MdGGzeMyEstSr/POGxKUAYEY18hKcKctaGxAMZyAcpesqVDNmW
n6vQClCbAkbTCD1mpF1Bn5x8vYlLIhkmuquiXsNV6UwybwACAgwAyBlC3K4DLlAq
KOf/gzd0YoazkUyx6qte4IF/wTw/wg9wK7mDqab75zAN1DxcsmpJLaPmWAFu2rWd
U1UqeB5+hnpnrYhFkxzL+TnOa9ckI9S33iLjFPCU185FZJNlVlcgclLeog5DSzCV
TjgUeOEMsSUn5d4a7DSkHPfT8TBMlNPQBTuJGGC15H8cCC+2QmWUuLNkq90z6MR+
E5JOajj6z/7qOvfL4SOVPtQxvF5iz2zduapxGgTz4UqeVwA1X7HkXx7Cumdipg0S
Wn63j4IiHHoU4hDyanEkLX32l5PfhaQ8zfpztcy1TxaKXPPtgOGFNDCMo+dG6HfQ
AO2qsL5rDdvgu9hLeCkHaxVDgCKZ4XZ5T9SgL8v3UKep8JomEFt1kdq4KCBJ/gS/
EvXEbnj6Vs63pYtrKgoWxGCCqva8/fqUAfvsX+llGDLBCWLreMTDPisj4fXmJS7h
Cbdg4bSMbVzRUWLNQ/wJbRhR7eeMcP7vrT/q6rx3eL3QCD5cIOCiiQBGBBgRAgAG
BQI2HoQLAAoJEKexToEmyL9hzEIAoL1BDnbaEbsiFPdGsIOz302dNRNCAKCpB9tr
K5MB6twgr+Ww52xyQf6xww==
=BBVS
-----END PGP PUBLIC KEY BLOCK-----
```

Step 8: Using the PGP encryption software to send (encrypt and sign) and receive (decrypt) secure e-mails

You are ready now to start using the PGP program to generate secure, encrypted digital information. In this section you'll learn how to encrypt messages or other data before you send them, and how to decrypt messages or other data that you have received.

First, the encryption process.

1. Compose the e-mail you want to send in whatever natural language you want to use (French, English, Spanish, German, etc.).
2. When you have finished composing the e-mail, make sure the cursor is still somewhere in the body of your message, and click on the PGPtray icon in the lower right corner of your screen.
3. In the PGPtray pop-up menu, select Current Window, then in the Current Window sub menu, select Encrypt & Sign. This will bring up the PGPtray Key Selection dialog box where you should see the list of Public Keys including that of the person or persons to whom you wish to send your message. Note: The Private Key is kept in a file called the Private Keyring. It is encrypted with your selected passphrase so even if, somehow, someone gets access to your Private Keyring, it will be unusable without access to the

Passphrase to decrypt the Key for use. Every time PGP needs access to the Private Key (to Decrypt an Encrypted Message or to Sign an Outgoing Message or someone's Public Key) the Passphrase will need to be re-entered. By default, PGP will remember [cache] your Passphrase for two Minutes so that you do not have to re-enter it if needed more than once within this time frame. However, two minutes isn't much time and the odds are you'll need to re-enter your passphrase every time unless you change this default. Step 14 explains how to do this, along with warnings about how to use the cache wisely and without risk.

4. Double click on the Public Key of the person to whom you wish to send your message (this selects the key and moves it to the recipients box just below). When you have made your selection, click on OK.
5. You will be prompted to enter your passphrase. Type it in carefully, then hit OK. If you did everything correctly, the message will be converted to unintelligible gobbledygook (aka "ciphertext", as it's called in the world of cryptography). The ciphertext will look like the following:

```
-----BEGIN PGP MESSAGE-----
Version: PGP Personal Security 7.0.3
qANQR1DBwk4DepqGz+tv7awQC/sGOyvgkqLDEz3QOc4AkDuoTVl9O2y7X260NR47
w77OngPn3z/01yEpVDmkfrpdXKYmVhylICPg1yvNYTyx6EW5LIOYt1yuxLc+bjKS
piwrBdCxz5+VT8z9IQz7BNu75GBP5YMJyhZUgwFRDahPITz0ziqL9nBZeUX27PGL
ZIc32bm/18zLwbLUZi4CSPlnc9PzXTeubwnsaC0ZU1PT+WokkhPRxPrgBHLU/rMj
zqOoh2/dXGMUFY7F0zitGw1jcj+jIf49hpzPZ5oWChZQjnQdREZgaRenx3jRomol
BnT0KgGk+cBp8BIM65DyoYdMKE878n+ngTgIYUYkBLnYXfQv9pgagPlQUgmMWSK/
zRkLS3PpKJFTv629iBXKKDeCteqD4668TRty3N1sEXaFbpMZtaNWJvqlXpbbrAkO
rvKAxMq9gpA+asf6415NSX29FT4uv4D7FWF3fp2e9it8c30//9yKXQ8pJb0vfz8B
vZCwIO1me371DScIwI2D8/8EHzQMALxye7O/tpDW3BEU+NEqsHM2nXdebKl7mPk8
5voUyYZb3vz3PQHNJ+Jg14KybK8Jn7KGji19nHFgFtHN0Qoz4e5aTlZtMksWDaX+
dT6xfrKBo5wOaQHGAX3NHBAMTCqUoZajsGxsc+dQ/WB7Qw4qdZjmLtzj35HcF7s0
5RwOWZ2F9cqSj0b994llaT9zo2jXs5ZM/fAZUBPsCp55EFpe52NFKJgyJY92mYi0
1SK26VMNMdHdp4zHWZdNkhPPG0EgDsz1g+EtY6YXWQYwIKPnQUivf5mhDdhPmWK
6
sAR4D7s2Vgqs2gQnvuFxpkDMc5l2rMTAE5+x228SpMPau27BDxBDKLw1i6ak23C+
l2qmiqQg0qeSFy2o7+HmyKWCENl2V84N8eLhoE+iyXj5fL2UvMlqVJePTT76Rz6p
+tD/15JYZo/8uAxIBivaB7P7k2Bqu0bmrCD4wdSKOLzhScxAj15Dtu0kWgEKGs80
VgTMu2iQLtphN7oObhWzUIf9O3MlqMnBCiOp4VFGebnJcDvullUB4OYZD6ZLIecN
8BsqsVlqawJbtWpmRf8973Yg2bicP0ISCwFaoDvR8C+wb3h9nJ9EZeO/mZGjJweR
A6yXK7wyp6JHnvACwFhUkTno7nrdq8cDaG4ssolsUSKnON87ycLFWq/mNs9fhqzF
Y3y7Q4f7hA4EL83+bxc4YGqzirWHeVXetZdft018+0Oz2Au8gRG5AVd+DX+xlr56
mJlkrlzYWG7HuEl8CRS7rAZHgRAIV3I7WDeNEYyBQNt/MfzUQY9+BmbtCsTlOnda
j8IkiL0QIW/9ZyvifxpvzKGKhxdXoqJWVSXLKHGk1qvY9epgw7QWk15crlti0Q4+
aDXvNieN9imk3UNQe2rncqzIKlbxasjparCKXiErQGFjldtTLrZcf7KjNOJuVG9J
HoOZC39ur8rkVrgWuSzrvzhpeQl0VlmdviZpocErZYPtnDQGgA3TbXX4lXoMiM1a
bOxTskUcgIBzN2L9nNfIhVaxJxMd3260SpJxElJ27V6Be97Q+YX4TF9xlH4zWFM3
NpGg1iXWNRb4VSwPE2+ZEiKirrlMsgXxfZNvAy3bAuSm0b1u7Isa/Jjab96DHff6
5g5K
=WRFH
```

```
-----END PGP MESSAGE-----
```

Now send the message just as you would normally do.
Next, the decryption process.

1. Open the e-mail containing the encrypted message. All you'll see is unintelligible ciphertext (as shown in the examples above).
2. Drag to select the block of ciphertext.
3. Click once more on the PGPtray icon in the lower right corner of your screen.
4. In the PGPtray pop-up menu, select Current Window, then in the Current Window sub menu, select Decrypt & Verify. This will bring up the dialog box asking you to enter your passphrase.
5. Type your passphrase into the PGP Enter Passphrase dialog box that pops up on the screen, and hit OK. The decrypted message will come up in a new window for you to read. If you wish to keep the decrypted version, you can copy it and paste it into a word processor of your choice before saving it to disk. The decrypted message will look like the following (Note that the message is now readable and the signature has been verified):

```
*** PGP Signature Status: good
*** Signer: Robert A. Rosenberg <bob.rosenberg@digitscorp.com>
*** Signed: 06/30/2001 at 00:51
*** Verified: 06/30/2001 at 00:52
*** BEGIN PGP DECRYPTED/VERIFIED MESSAGE ***
This is a sample of what the above Encrypted&Signed message looks like
after it has been decrypted and the signature has been successfully
verified. Since the Public Key that was used to encrypt this text belongs
to Robert Rosenberg, only he can decrypt the message to extract this
message. An Encrypted&Signed message is a Clear Signed Message (such as the
sample in Step 10 below) prior to the Encrypt Stage and after the Decrypt
Stage. While it is possible to just Encrypt a message, it is usual to also
sign it to prove its origin.

*** END PGP DECRYPTED/VERIFIED MESSAGE ***
```

That's all there is to it. To find out about the many other features of the PGP program, check out the Manual that was originally downloaded with the software. It's a .pdf file which will print out beautifully on your printer so you can read it at your leisure over a nice cup of tea :) Well, maybe you'll need something a bit stiffer to help you figure it all out...

On a technical note: The actual encryption/decryption is NOT being done with the Public/Private keys of your recipient(s) but with a special one-time key that is generated for use in this specific encrypt&sign operation. Every time you do an encrypt&sign, a new one-time key is generated. Unlike the Public/Private key pairs where anything encrypted with one key needs the other key to do the decrypt, these one-time keys have the ability to decrypt anything that they encrypt (hence its being known as a Symmetric Key). When you encrypt any data, this one-time key is used to do the actual encryption. The Public key of each recipient is then used to encrypt the one-time key and added to the encrypted text created with the one-time key. Thus what results is a list of recipients with the one-time key supplied encrypted with each person's Public Key along with the common copy of the one-time key encrypted ciphertext. This format allows a message to be sent to multiple people at the same time yet allow each to use his or her own Private Key to read it. The decrypting process involves the recipient's PGP Program scanning the list of encrypted one-time keys looking for the copy that was encrypted with their Public Key. This copy is then decrypted with the Private key to recover the one-time key which then can be used to do the actual decrypting. The Signing/Verification actions that occur during an encrypt&sign and decrypt&verify are covered in Step 10 below and occur prior to the encryption itself and after the corresponding decrypting of the data.

Step 9: Using your Default Public Key to save a backup, encrypted, decipherable copy of all your e-mail messages

There's something you need to know right away about PGP encryption: once you encrypt a message using the Public Key of the person to whom you're sending it, you won't be able subsequently to decrypt it and read it yourself since you don't have your correspondent's Private Key. Most of the time this doesn't matter because you may not need to keep a copy of every message you sent. But sometimes (maybe often, if you consider it necessary) you want to keep your own encrypted copy of a message for the record and you need to be able to decrypt it, if and when you want to read it at a later date. The best thing to do is tell PGP to encrypt all your messages using your correspondent's Public Key as well as one of your own Public Keys (called the Default Public Key).

```
----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is a sample of a clear signed message. Note that it can be read even
if you do not have PGP or verify the signature. Doing the Verify is only
needed if you want to verify that it was written by who it claims to be
from and/or that it has not been altered after being signed.

-----BEGIN PGP SIGNATURE-----
Version: PGP Personal Privacy 6.5.8
iQA/AwUBOzJwnqexToEmyL9hEQLCPACePdnPEau8VKKcxsD78ysTlWbgHFUAmwZe
mx/Q+qWDRsftiGGjeImc4tjL
=cFLq
-----END PGP SIGNATURE-----
```

And this is what the clear signed message looks like after being verified:

```
*** PGP Signature Status: good
*** Signer: Robert A. Rosenberg <bob.rosenberg@digitscorp.com>
*** Signed: 6/21/01 at 18:09
*** Verified: 6/21/01 at 18:11
*** BEGIN PGP VERIFIED MESSAGE ***

This is a sample of a clear signed message. Note that it can be read even
if you do not have PGP or verify the signature. Doing the Verify is only
needed if you want to verify that it was written by who it claims to be
from and/or that it has not been altered after being signed.
```

```
*** END PGP VERIFIED MESSAGE ***
```

Step 11: Weaving the Web of Trust--Signing someone else's Public Key

Here is a comment from a respected member of the Public Key Encryption community (Nick Andriash) in response to a request he received to sign a cyberfriend's Public Key. "With respect to signing each other's Public Keys," Nick replied, "I have already done so with a non-exportable signature, because we have been in constant communication, and I obtained your Public Key from your web site; I am confident enough in knowing the messages are coming from the same person at the same address... I just don't know who that person is, and that is why I cannot sign your Public Key with an exportable signature, where it will always travel with the Public Key. For that, I insist on face to face meetings, along with an exchange of photo ID, etc., as this is the only way to maintain the integrity of one's own Web of Trust. All of the people who have signed my Key, I have met personally, and that is as it should always be, unless we are introduced to each other by a Trusted Introducer whose signature appears on both our Public Keys."

Step 12: Using the PGP encryption software to protect (encrypt) your personal documents

On your computer in the office or at home, you may well have private documents that you do not want others to be able to read. You can use your own Public Key to encrypt these documents. You can easily and quickly encrypt a single file or a set of files. To decrypt the files, you simply reverse the process that follows by selecting the option to Decrypt instead of Encrypt from the PGP menu. Here are the steps to follow to encrypt a single file or document:

1. Right click on the Start menu in the lower left corner of your Windows screen, select the Explore option in the pop-up menu, then in the left hand column of the Explore window select the C drive, for example, and you'll see the contents of your C drive listed in the right hand side of the Exploring window.
2. Right click on any document you have listed there (in the right hand side of the Exploring window) and you'll see a new item (PGP) in the pop-up menu.
3. Select PGP in the pop-up menu and then you'll see the sub-menu option to Encrypt the document you've highlighted. Click on Encrypt.
4. Now you're presented with the Key Selection dialog box. Double click on your own Public Key (or drag it down to the Recipients box below) and click on OK. PGP has now created a second, encrypted, version of the document with a .pgp extension.
5. All you need do now is delete the original, non-encrypted document, so that all you have left on your disk is the encrypted file which only you can read. Do this right away by right clicking on the original and selecting Delete from the pop-up menu.

Step 13: Using PGP to Wipe files from your disks
When you delete a file, is the data it contains removed from your disk? Answer: No! You may not be able to see the name of the file anymore if you list the contents of your disk, but someone who knows what they're doing can easily resurrect it and, if it's not encrypted, read it. When you delete a file, all you're doing is removing the link to it from the disk's index of files. It's like a card catalog in a library. Every book in the library has a card in the catalog which helps you find it on the shelves. If you remove the card from the catalog, you'll have a problem finding the book--but it's still out there on the shelves. When you delete a file on your disk, it's like removing the card from the catalog. The file's still there on the disk, even though you can't easily get to it. To remove it completely, you must Wipe that part of the disk clean, and this is what the PGP Wipe function does for you. Let's try it for practice.

- Use your word processor to create a dummy file and save it with the name Dummy. Put any old garbage in it, since you're going to Wipe it off your disk in a minute.
- Now locate the Dummy document using the Explore option in the Start menu (as you did just now in Step 12).
- Right click on the Dummy document and select the option in the pop-up menu to Wipe the file. Simple as that. PGP writes a bunch of random data to the place on your disk where the Dummy file was saved, effectively removing all trace of the original data. Neat, huh?

Step 14: Useful PGP Options you should know about
We'll be adding explanations for more PGP Options over the next few weeks. For now, here is an explanation of how you can tweak the time frame of the cache that PGP uses to remember your Passphrase. You'll also find out here how to Purge your Passphrase cache, a simple task which is very important to remember to do when you leave your computer unattended. Finally, for your convenience, we've added a table listing the hotkeys available in PGP.

FINDINGS:

There's been a lot of discussion of the problems with PGP, how it uses ancient crypto, etc. Unfortunately, I don't think a lot of the discussed replacements actually meet the same use cases. I've read the PGP Problem but am unsatisfied with the suggestions. Maybe I'm just being cranky, but I'd love some feedback on the problems I see with the suggested alternatives.

I currently use PGP for 4 use cases:

1. Occasional encrypted email, usually for vulnerability reports or discussing undisclosed bugs.
2. Encrypting files to others. Usually associated with 1) above.
3. Encrypting files to myself (in the future -- ooh, time travel). More seriously, backups using duplicity.
4. Signing git tags and the encrypted backups in 3. Oh and some email, because I can.

Are there modern replacements for all of these use cases?
Signal is often touted as the replacement for (1), but that requires giving my phone number to anyone I want to communicate with (associating my communications with my real-world identity) and also precludes having multiple identities. Signal also doesn't have a way to easily archive my communications (in fact, it seems bound and determined to avoid that) as well as an inability to run on multiple mobile devices. It makes it very hard for multiple individuals to receive the same messages (e.g., for receiving bug bounty reports, as suggested in the Latacora blog post). Signal also seems vulnerable to SIM porting attacks if users ignore the "key has changed" message. (Also, Signal is not decentralized, but I guess that is a preference more than a technical objection.)
For (2), magic wormhole is mentioned, but this seems to be encryption in transit and not encryption at rest? I guess that meets some of the needs of encrypting to others, but it seems I need to keep my machine available to them, so it makes it hard for transferring files from, say, my laptop, if the other user is not currently available. What are good options for encrypting a file that I can just drop into Dropbox, Google Drive, or even (shudder) email?
For (3), tarsnap is suggested, but that ties you to a particular service provider. Is there a modern alternative where I can store the backups on external hard drives or machines of my choice? I don't want to depend on just the tarsnap service in the case that it goes under or suffers a technical failure of its own.
For (4), signify/minisign is mentioned, but it's not clear to me how one gets the original key, other than mentioning posting it in a bunch of places. Seems like it basically depends on https at best.

Original need: Linux distributions and many other software update mechanisms use PGP signatures to prevent malicious mirrors or network attackers from altering the contents of their packages.

Modern alternative: a lightweight signing tool like OpenBSD's signify or minisign, either of which you could quickly build in Go using x/crypto/ed25519. I wrote one as a party trick last month – it's less than 200 lines of code and that includes some silly key parsing tricks.

These tools are extremely simple and robust because they only support one algorithm, lack state or any notion of a keyring, and are usually decoupled from complex messaging formats (which should be a separate concern from your signature cryptography). As a bonus they're mostly compatible with each other because the keys are all EdDSA keys. Need I even mention that EdDSA is much, much faster than RSA?
If you're a maintainer of self-serve packaging infrastructure or otherwise have more complex needs, you should take a look at TUF. It's a modern toolkit meant to resist all kinds of update-related attacks that PGP signatures simply don't address. For an idea of how it works in practice, check out the designs for Python's package infrastructure (PEP 458, PEP 480). The Notary project is a productionized and audited implementation.

CONCLUSION:

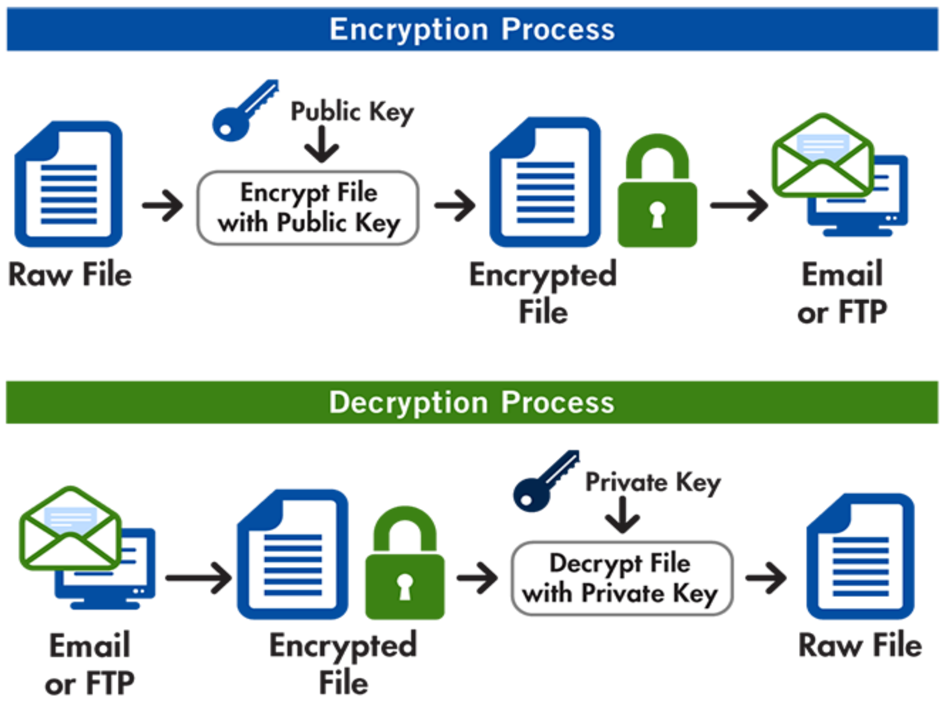
PGP is very easy to understand, on the surface. Imagine you want to send your credit card information to a friend and you write it on a piece of paper. You then put the paper in a box and send it by mail.

A thief can easily steal the box and look at the paper that contains your credit card information. What could you do instead?You decide to put a key lock on the box, but you realize that you have to send the key along with the box. That's no good.

What if you meet your friend in person to share the key beforehand? That could work, right? It could, but then both of you have a key that allows to unlock the box. You, as the sender, will never need to open the box again after closing it. By keeping a copy of a key that can unlock the box, you are creating a vulnerability.

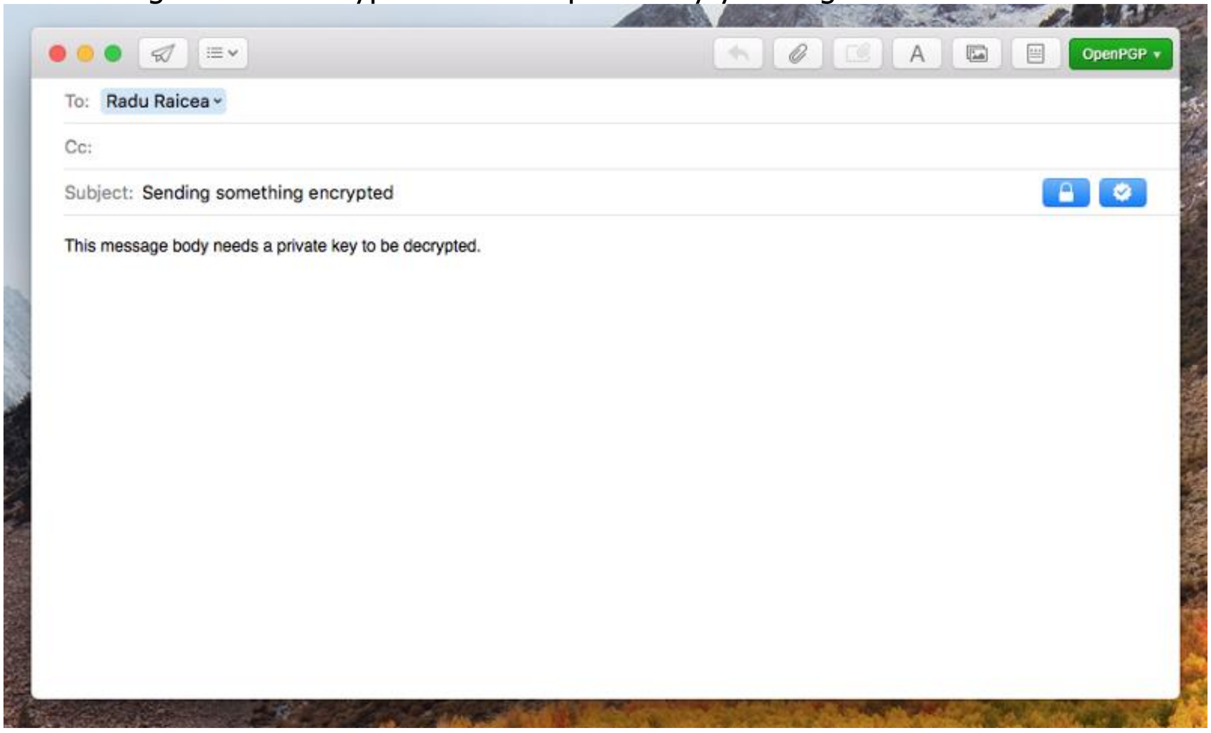
Finally, you found just the right solution: you'll have two keys. The first key will only be able to lock the box. The second key will only be able to open the box. That way, only the person who needs to get the content of the box has the key that allows them to unlock it.

This is how PGP works. You have a **public key** (to lock/encrypt the message) and a **private key** (to unlock/decrypt the message). You would send the public key to all your friends so that they can encrypt sensitive messages that they want to send to you. Once you receive an encrypted message, you use your private key to decrypt it.



A Brief Example

There are plenty of software tools that implement the OpenPGP standard. They all have different ways of setting up PGP encryption. One particular tool that works very well is Apple Mail.
If you are using a Mac computer, you can download the GPGTools. This application will generate and manage your public and private keys. It also integrates automatically with Apple Mail.Once the keys are generated, you will see a lock icon in the subject line, when composing a new message in Apple Mail. This means that the message will be encrypted with the public key you've generated.



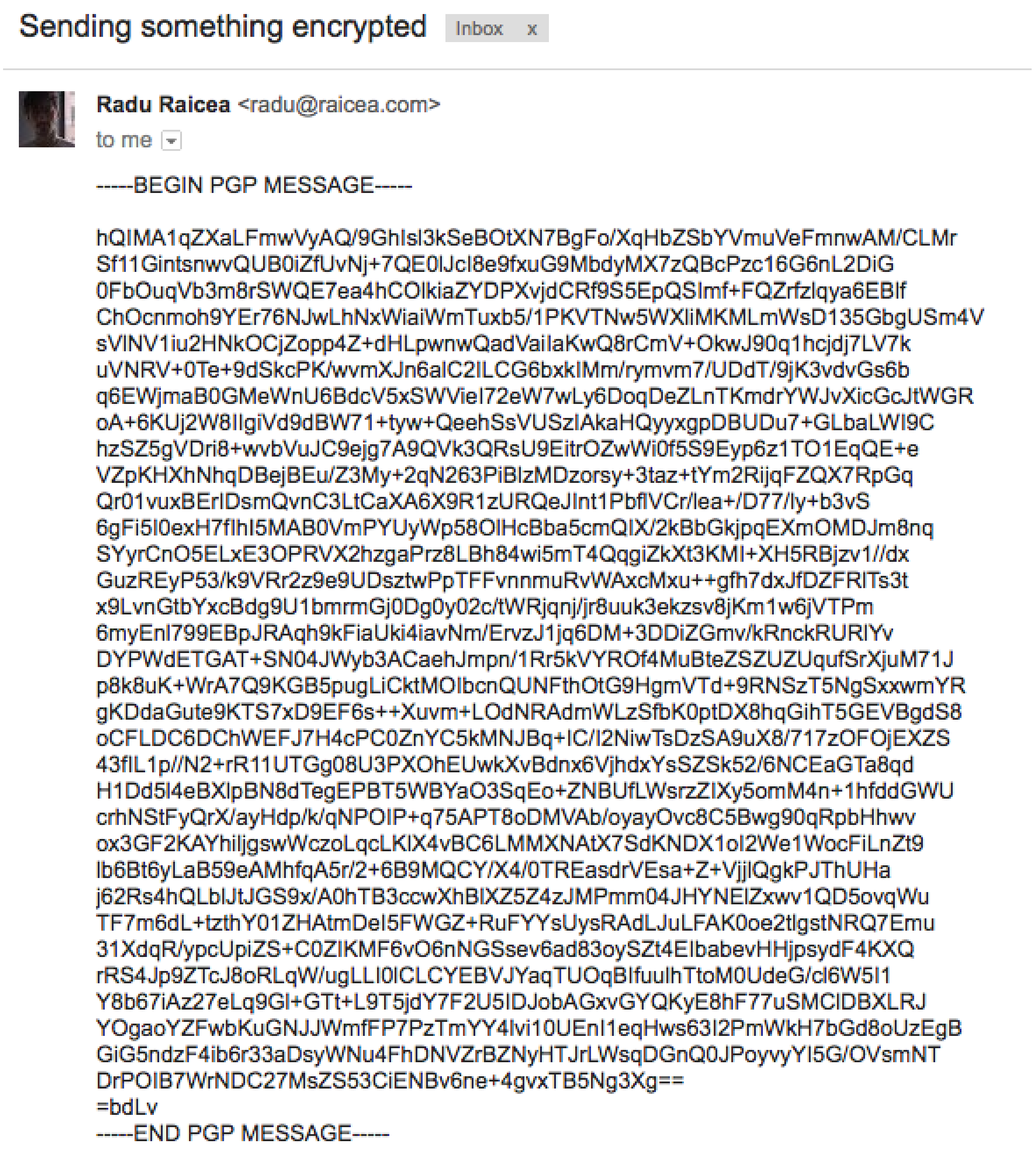
To: Radu Raicea
Cc:
Subject: Sending something encrypted

This message body needs a private key to be decrypted.

Composing a PGP encrypted e-mail using Apple Mail

After sending the e-mail to someone, it will look like this. They will not be able to see the content of the e-mail until they decrypt it using the private key.

Note that PGP encryption does not encrypt the subject line of an e-mail. Never put any sensitive information in the subject line.



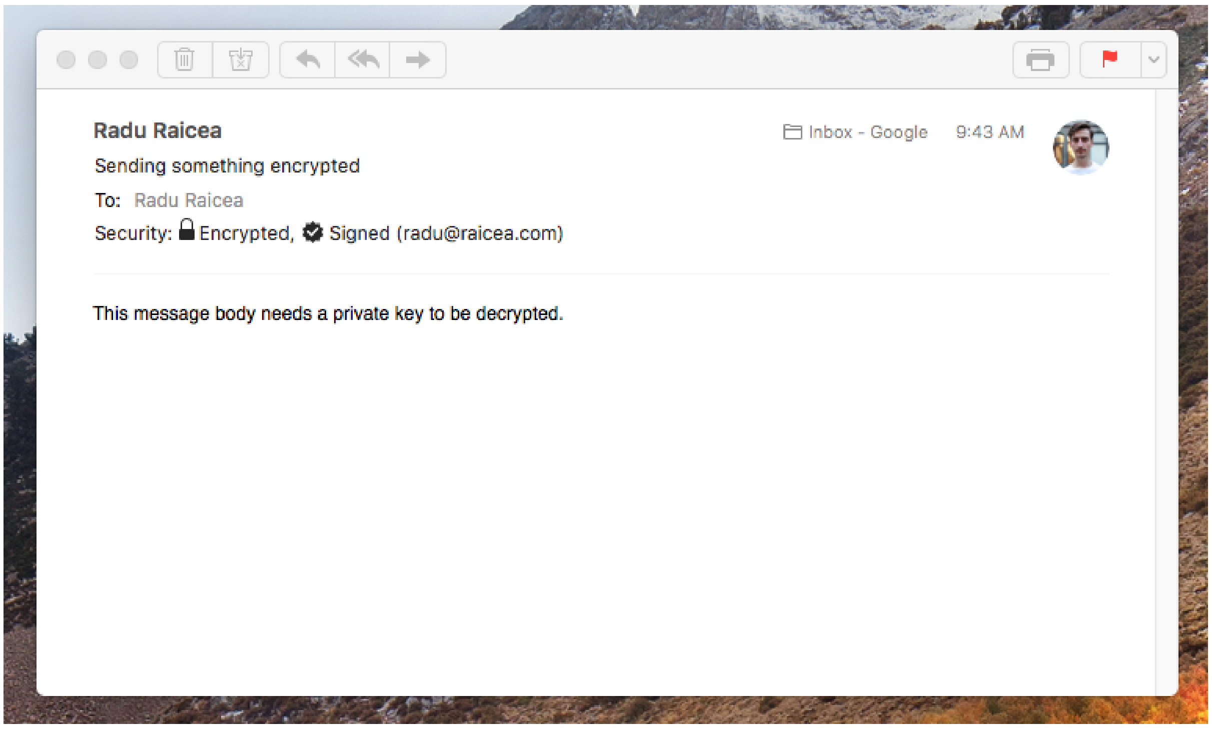
Sending something encrypted Inbox x

Radu Raicea <radu@raicea.com>
to me

```
-----BEGIN PGP MESSAGE-----

hQIMA1qZXaLFmwVyAQ/9GhIsl3kSeBOtXN7BgFo/XqHbZSbYVmuVeFmnwAM/CLMr
Sf11GintsnwvQUB0iZfUvNj+7QE0lJcl8e9fxuG9MbdyMX7zQBcPzc16G6nL2DiG
0FbOuqVb3m8rSWQE7ea4hCOlkiaZYDPXvjdCRf9S5EpQSImf+FQZrfzlqya6EBIf
ChOcnmoh9YEr76NJwLhNxWiaiWmTuxb5/1PKVTNw5WXliMKMLmWsD135GbgUSm4V
sVINV1iu2HNkOCjZopp4Z+dHLpwnwQadVailaKwQ8rCmV+OkwJ90q1hcjdj7LV7k
uVNRV+0Te+9dSkcPK/wvmXJn6alC2ILCG6bxkIMm/rymvm7/UDdT/9jK3vdvGs6b
q6EWjmaB0GMeWnU6BdcV5xSWViel72eW7wLy6DoqDeZLnTKmdrYWJvXicGcJtWGR
oA+6KUj2W8IlgiVd9dBW71+tyw+QeehSsVUSzIAkaHQyyxgpDBUDu7+GLbaLWI9C
hzSZ5gVDri8+wvbVuJC9ejg7A9QVk3QRsU9EitrOZwWi0f5S9Eyp6z1TO1EqQE+e
VZpKHXhNhqDBejBEu/Z3My+2qN263PiBlzMDzorsy+3taz+tYm2RijqFZQX7RpGq
Qr01vuxBErlDsmQvnC3LtCaXA6X9R1zURQeJInt1PbflVCr/lea+/D77/ly+b3vS
6gFi5l0exH7flhI5MAB0VmPYUyWp58OlHcBba5cmQIX/2kBbGkjpqEXmOMDJm8nq
SYyrCnO5ELxE3OPRVX2hzgaPrz8LBh84wi5mT4QqgiZkXt3KMI+XH5RBjzv1//dx
GuzREyP53/k9VRr2z9e9UDsztwPpTFFvnnmuRvWAxcMxu++gfh7dxJfDZFRITs3t
x9LvnGtbYxcBdg9U1bmrmGj0Dg0y02c/tWRjqnj/jr8uuk3ekzsv8jKm1w6jVTPm
6myEnl799EBpJRAqh9kFiaUki4iavNm/ErvzJ1jq6DM+3DDiZGmv/kRnckRURIYv
DYPWdETGAT+SN04JWyb3ACaehJmpn/1Rr5kVYROf4MuBteZSZUZUqufSrXjuM71J
p8k8uK+WrA7Q9KGB5pugLiCktMOIbcnQUNFthOtG9HgmVTd+9RNSzT5NgSxxwmYR
gKDdaGute9KTS7xD9EF6s++Xuvm+LOdNRAdmWLzSfbK0ptDX8hqGihT5GEVBgdS8
oCFLDC6DChWEFJ7H4cPC0ZnYC5kMNJBq+IC/I2NiwTsDzSA9uX8/717zOFOjEXZS
43flL1p//N2+rR11UTGg08U3PXOhEUwkXvBdnx6VjhdxYsSZSk52/6NCEaGTa8qd
H1Dd5l4eBXlpBN8dTegEPBT5WBYaO3SqEo+ZNBUfLWsrzZIXy5omM4n+1hfddGWU
crhNStFyQrX/ayHdp/k/qNPOIP+q75APT8oDMVAb/oyayOvc8C5Bwg90qRpbHhwv
ox3GF2KAYhiljgswWczoLqcLKIX4vBC6LMMXNAtX7SdKNDX1ol2We1WocFiLnZt9
Ib6Bt6yLaB59eAMhfqA5r/2+6B9MQCY/X4/0TREasdrVEsa+Z+VjjlQgkPJThUHa
j62Rs4hQLbIJtJGS9x/A0hTB3ccwXhBIXZ5Z4zJMPmm04JHYNElZxwv1QD5ovqWu
TF7m6dL+tzthY01ZHAtmDeI5FWGZ+RuFYYsUysRAdLJuLFAK0oe2tlgstNRQ7Emu
31XdqR/ypcUpiZS+C0ZIKMF6vO6nNGSsev6ad83oySZt4ElbabevHHjpsydF4KXQ
rRS4Jp9ZTcJ8oRLqW/ugLLI0ICLCYEBVJYaqTUOqBIfuulhTtoM0UdeG/cl6W5I1
Y8b67iAz27eLq9GI+GTt+L9T5jdY7F2U5IDJobAGxvGYQKyE8hF77uSMCIDBXLRJ
YOgaoYZFwbKuGNJJWmfFP7PzTmYY4lvi10UEnl1eqHws63l2PmWkH7bGd8oUzEgB
GiG5ndzF4ib6r33aDsyWNu4FhDNVZrBZNyHTJrLWsqDGnQ0JPoyvyYI5G/OVsmNT
DrPOIB7WrNDC27MsZS53CiENBv6ne+4gvxTB5Ng3Xg==
=bdLv
-----END PGP MESSAGE-----
```

Receiving a PGP encrypted e-mail

If you are using software that automatically decrypts the message using your private key, like Apple Mail, it will look something like this:



Radu Raicea
Sending something encrypted
To: Radu Raicea
Security: Encrypted, Signed (radu@raicea.com)

This message body needs a private key to be decrypted.

Decrypted PGP e-mail

```
--paw

------- Forwarded Messages

From: Ruth Milner <rmilner@aoc.nrao.edu>

Are they compatible? I.e. can they understand each other's
keys?
If not, then I'd stick with PGP for broader interchange with
other
sites.

We chose PGP (6.5.2) with RSA support, largely because we
weren't
sure about PGP/GPG compatibility when we gave encryption
software
some attention a little over a year ago. So far nobody here
has
encountered a need to revisit that choice. We don't have any
real
infrastructure in place yet, though; just made sure that all
sites
had the capability and that key security folks had keys.

Ruth.
```

REFERENCES :

(1) https://www.howtogeek.com/187961/why-no-one-uses-encrypted-email-messages/
(2) http://philzimmermann.com/EN/background/index.html
(3) https://www.goanywhere.com/managed-file-transfer/encryption/open-pgp
(4) http://www.pitt.edu/~poole/PGP.htm

ACTIVITY 2

PROCEDURE:

Some of the major security issues that the most commonly used email services include :

1.Sending Confidential Data-When essential data is not encrypted it acts like an open ground for hackers to feed upon.

2.Malware-The one place where malwares can be found the most are in emails where in they easily trace their way in and cause destruction.as long as an email looks legitimate no one will suspect it

3. Phishing -one of the biggest email security threats are phishing scams .Links in emails that look harmless could be scams only for users to loose data because of one click.

4.Stolen Devices-This does not point straight as an email threat but when a phone or any device is stolen everything that device hold is compromised

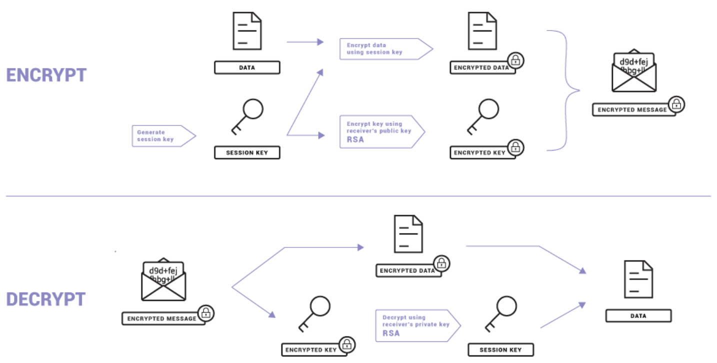
5. Weak Passwords-In order to avoid remembering huge complicated passwords people choose to keep simple passwords they avoid two factor authentication, which definitely adds to our bigger problem.

What is PGP?

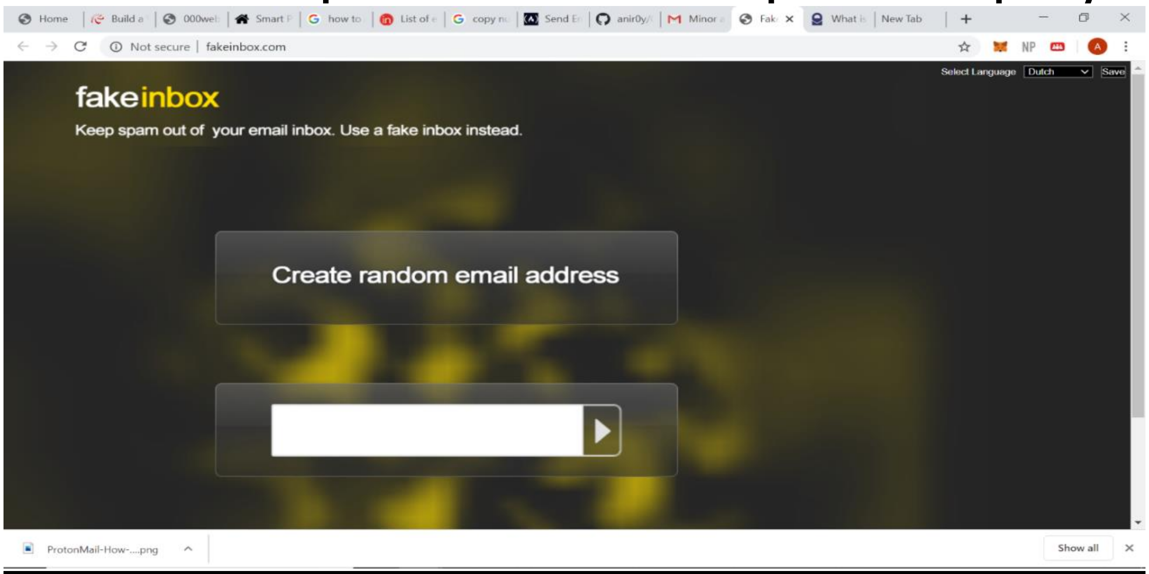
PGP is a cryptographic method that lets people communicate privately online. When a mail is sent using PGP, the message is converted into unreadable ciphertext on the device before it passes over the Internet. Here only the recipient has the key to convert the text back into the readable message on their device, it also authenticates the identity of the sender and verifies that the message was not tampered with in transit. Before PGP, a lot of people could read your emails. PGP was developed in the 1990s to allow email and other types of messages to be exchanged privately. Today, PGP has been standardized into OpenPGP, enabling anyone to write PGP software that is compatible and interoperable with other implementations.

Working of PGP?

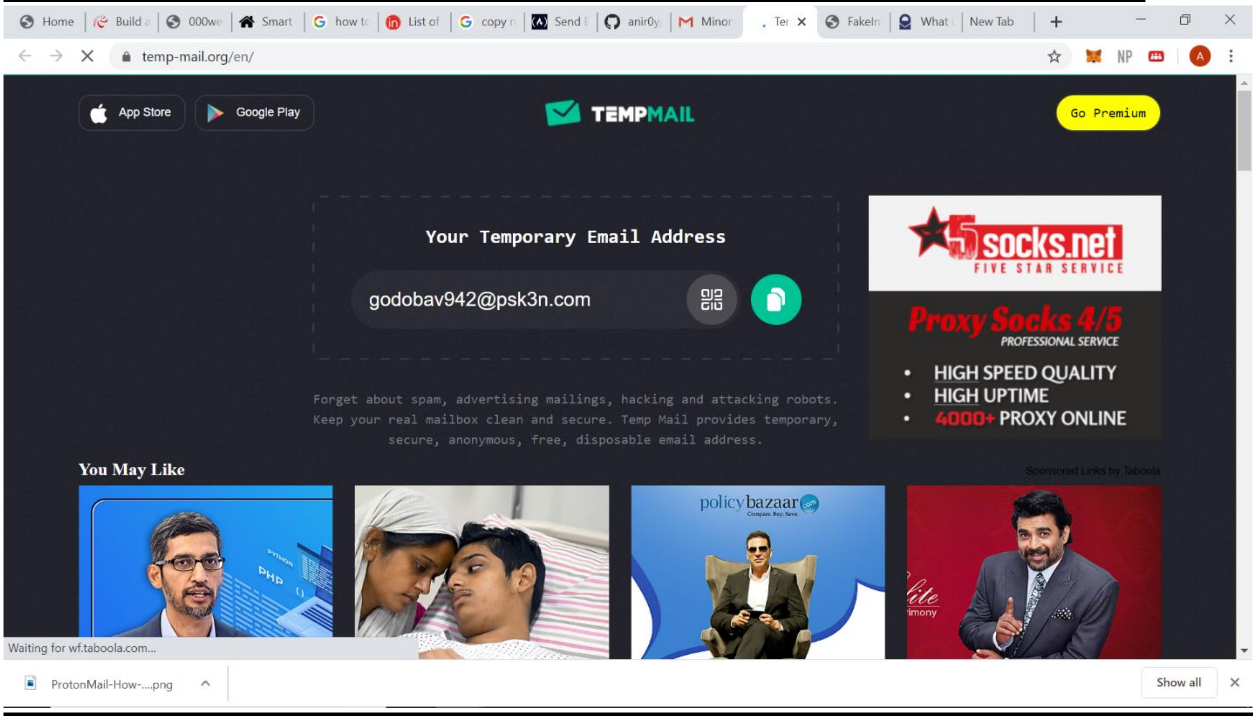
It's useful to see a diagram to understand how PGP encryption works. PGP uses a combination of symmetric key encryption and public key encryption.



Sites such as tempmail and fake inbox help create temporary mails



Keep spam out of your email inbox. Use a fake inbox instead.



Your Temporary Email Address

godobav942@psk3n.com

Forget about spam, advertising mailings, hacking and attacking robots. Keep your real mailbox clean and secure. Temp Mail provides temporary, secure, anonymous, free, disposable email address.

How to send email using python ?

Here are four basic steps for sending emails using Python:

1. Set up the SMTP server and log into your account.
2. Create the MIMEMultipart message object and load it with appropriate headers for From, To, and Subject fields.
3. Add your message body.
4. Send the message using the SMTP server object.

What is mail merge in Python?

Mail merge is a mechanism for integrating information from a database to an email template with placeholders, in order to add personalization to the multiple emails. To put it simply, if you need to target a list of users by email, and address them by name, you can do it with a simple table and script using mail merge.

How to use mail merge?

We can look at the following example we took from the mailmerge page on pypi.org.

I. Installation:

First, we need to install mail merge into our systems using the following Pip command:

```
$ pip install mailmerge
```

II. Create a sample template email, database, and config:

Use the following code:

```
$ mailmerge –sample
```

This will result in the following:

- Created sample template email "mailmerge_template.txt"
- Created sample database "mailmerge_database.csv"
- Created sample config file "mailmerge_server.conf"

Edit these files, then run mailmerge again.

III. Edit the SMTP server config:

The defaults are set up for gmail. Be sure to change your username. Use the following format:

```
[smtp_server]

host = smtp.gmail.com

port = 465

security = SSL/TLS

username = YOUR_USERNAME_HERE
```

IV. Edit the template email message:

The TO, SUBJECT, and FROM fields are required. The remainder is the body of the message. Use {{ }} to indicate customized parameters that will be read from the database. For example, {{email}} will be filled in from the email column of mailmerge_database.csv.

Use the following format:

```
TO: {{email}}

SUBJECT: Testing mailmerge

FROM: My Self <myself@mydomain.com>

Hi, {{name}},

Your number is {{number}}.
```

V. Edit the database mailmerge_database.csv:

Notice that the first line is a header that matches the parameters in the template example, for example, {{email}}.

Use the following format:

```
email,name,number

myself@mydomain.com,"Myself",17

bob@bobdomain.com,"Bob",42
```

VI. Run the python code and send all the emails.

Can you email multiple people?

Yes, we can email multiple people using python. We can use mail merge to achieve this. When we want to send the same invitations to many people, the body of the mail does not change. Only the name (and maybe address) needs to be changed.

Mail merge is a process of doing this. Instead of writing each mail separately, we have a template for body of the mail and a list of names that we merge together to form all the mails.

Could you pull the list of people to email from an external file?

Yes, we used an external file named emailids.txt which stored the addresses of the recipients. We extracted the different email ids using the following code:

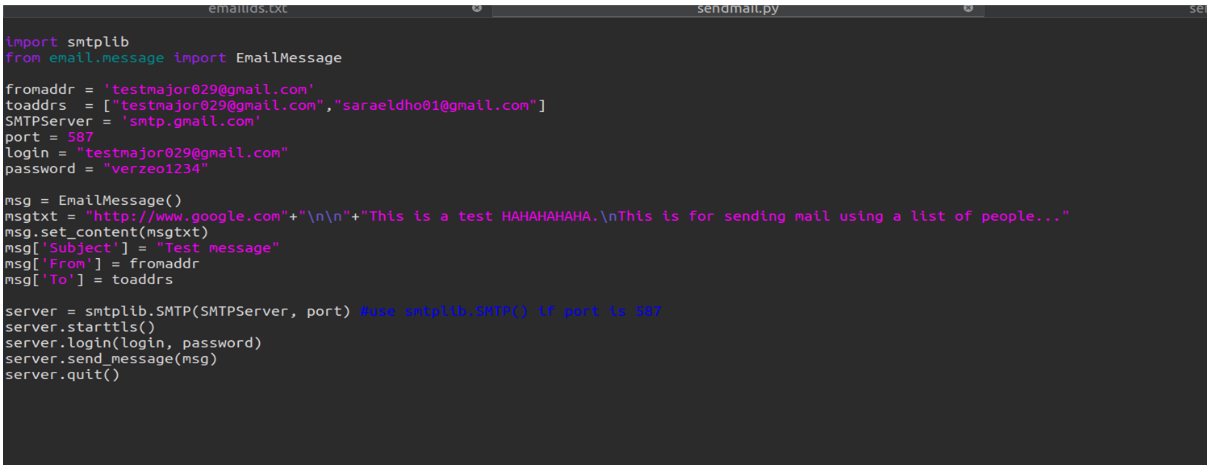
```
my_file = open("emailids.txt", "r")

content = my_file.read()

print(content)

content_list = content.split()

print(content_list)
```

This way it's easier to store the data in an organized way and helps in better functioning of the program.

How can you personalize the email for the recipient?

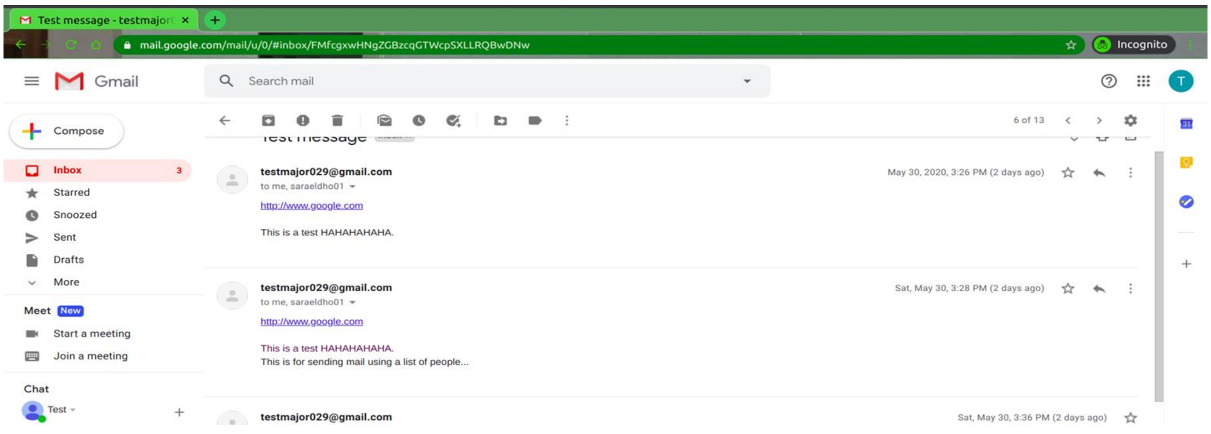
If you would like to send a personalized email to each person, and do not want to type each one by hand, python can automate this for you. All you need is the mailing list in some kind of structured format, and then you can go through it line by line to create and send emails.

You can use a csv file to store your data in a structured format and then use it to send personalized emails to your recipients. You can access the data from the python code and use placeholders in your emails to substitute the personalized texts in the emails.



```
import smtplib
from email.message import EmailMessage

fromaddr = 'testmajor029@gmail.com'
toaddrs  = ["testmajor029@gmail.com","saraeldho01@gmail.com"]
SMTPServer = 'smtp.gmail.com'
port = 587
login = "testmajor029@gmail.com"
password = "verzeo1234"

msg = EmailMessage()
msgtxt = "http://www.google.com"+"\n\n"+"This is a test HAHAHAHAHA.\nThis is for sending mail using a list of people..."
msg.set_content(msgtxt)
msg['Subject'] = "Test message"
msg['From'] = fromaddr
msg['To'] = toaddrs

server = smtplib.SMTP(SMTPServer, port) #use smtplib.SMTP() if port is 587
server.starttls()
server.login(login, password)
server.send_message(msg)
server.quit()
```

FINDINGS:



testmajor029@gmail.com
to me, saraeldho01
May 30, 2020, 3:26 PM (2 days ago)

http://www.google.com

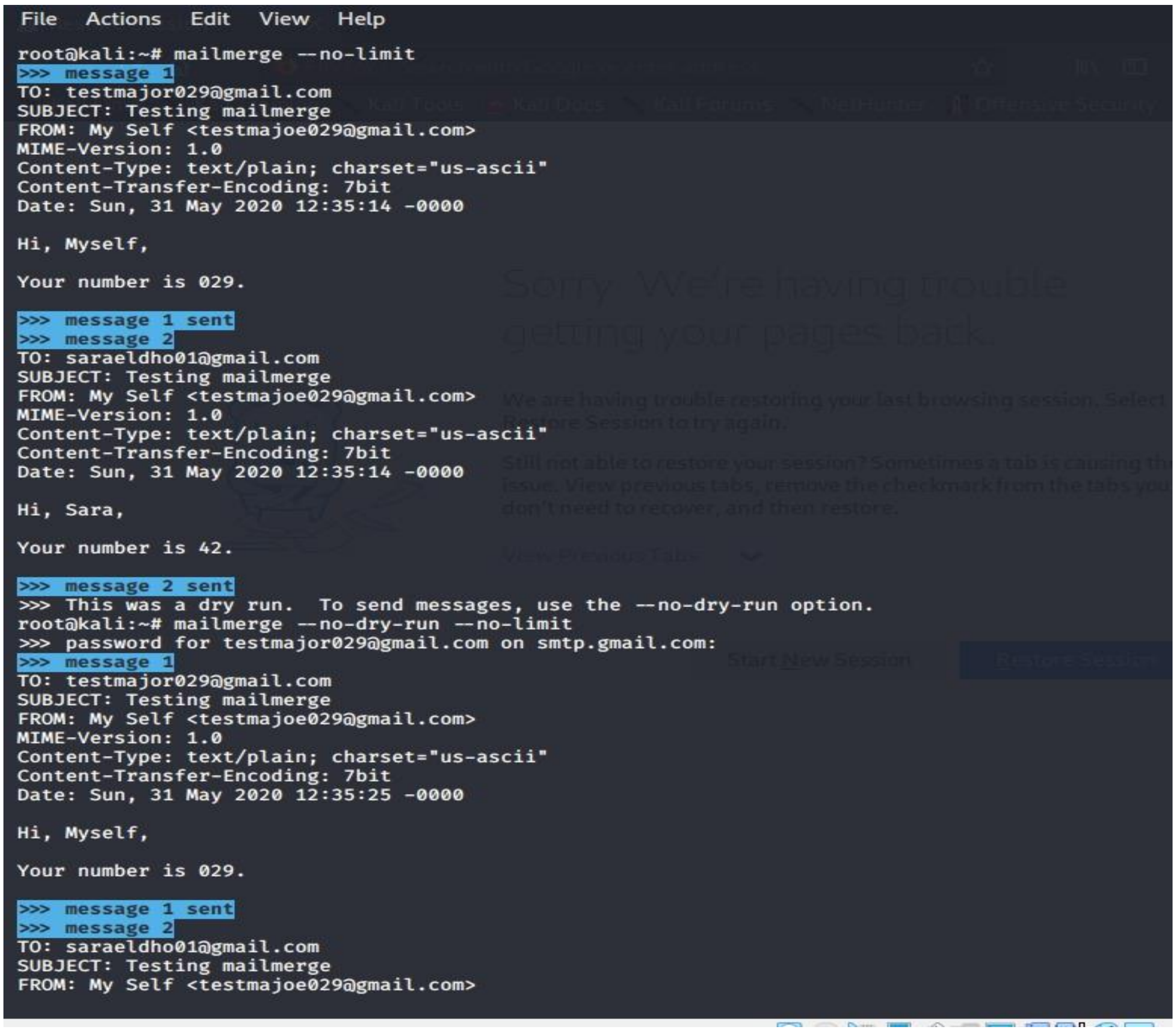
This is a test HAHAHAHAHA.

testmajor029@gmail.com
to me, saraeldho01
Sat, May 30, 3:28 PM (2 days ago)

http://www.google.com

This is a test HAHAHAHAHA.
This is for sending mail using a list of people...

testmajor029@gmail.com
Sat, May 30, 3:36 PM (2 days ago)



```
root@kali:~# mailmerge --no-limit
>>> message 1
TO: testmajor029@gmail.com
SUBJECT: Testing mailmerge
FROM: My Self <testmajoe029@gmail.com>
MIME-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
Content-Transfer-Encoding: 7bit
Date: Sun, 31 May 2020 12:35:14 -0000

Hi, Myself,

Your number is 029.

>>> message 1 sent
>>> message 2
TO: saraeldho01@gmail.com
SUBJECT: Testing mailmerge
FROM: My Self <testmajoe029@gmail.com>
MIME-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
Content-Transfer-Encoding: 7bit
Date: Sun, 31 May 2020 12:35:14 -0000

Hi, Sara,

Your number is 42.

>>> message 2 sent
>>> This was a dry run.  To send messages, use the --no-dry-run option.
root@kali:~# mailmerge --no-dry-run --no-limit
>>> password for testmajor029@gmail.com on smtp.gmail.com:
>>> message 1
TO: testmajor029@gmail.com
SUBJECT: Testing mailmerge
FROM: My Self <testmajoe029@gmail.com>
MIME-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
Content-Transfer-Encoding: 7bit
Date: Sun, 31 May 2020 12:35:25 -0000

Hi, Myself,

Your number is 029.

>>> message 1 sent
>>> message 2
TO: saraeldho01@gmail.com
SUBJECT: Testing mailmerge
FROM: My Self <testmajoe029@gmail.com>
```

CONCLUSION:

The SMTPlib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. For details of SMTP and ESMTP operation, consult RFC 821 (Simple Mail Transfer Protocol) and RFC 1869 (SMTP Service Extensions).

Class smtplib.**SMTP**(*host='', port=0, local_hostname=None*,
[*timeout*,]*source_address=None*)

An SMTP instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations. If the optional host and port parameters are given, the SMTP connect() method is called with those parameters during initialization. If specified, *local_hostname* is used as the FQDN of the local host in the HELO/EHLO command. Otherwise, the local hostname is found using socket.fqdn(). If the connect() call returns anything other than a success code, an SMTPConnectError is raised. The optional *timeout* parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). If the timeout expires, socket.timeout is raised. The optional source_address parameter allows binding to some specific source address in a machine with multiple network interfaces, and/or to some specific source TCP port. It takes a 2-tuple (host, port), for the socket to bind to as its source address before connecting. If omitted (or if host or port are '' and/or 0 respectively) the OS default behavior will be used.

For normal use, you should only require the initialization/connect, sendmail(), and SMTP.quit() methods. An example is included below.

The SMTP class supports the with statement. When used like this, the SMTPQUIT command is issued automatically when the with statement exits. E.g.:

```
>>> from smtplib import SMTP
>>> with SMTP("domain.org") as smtp:
...     smtp.noop()
...
(250, b'Ok')
>>>
```

REFERENCES:

(1) https://gist.github.com/anir0y/be6d613dd3d793e54f3e6d2be1dd0f65

(2) https://realpython.com/python-send-email/

DISCUSSION QUESTIONS

(1) What could you do to ensure privacy when sending email?

Ans: Of course, that doesn’t mean that secure, private email isn’t possible. It’s just up to you to take a few precautions to keep your email safe:

1. Use two-factor authentication

The basic principle of two-factor authentication is simple: combine something you know with something you have. One example is a debit card, which requires you to have both your physical card and your PIN to verify your identity. By enabling two-factor authentication (or two-step verification), you aren’t putting all of your faith in a password. That’s a good thing, considering how weak many of our passwords are. For Gmail, setting up two-step verification is as simple as clicking a button and entering in your mobile number. For Windows Mail, or Outlook, it’s a similar process. Just log in, go to your “Password and security” tab and click “Set up two-step verification.” Now that you’ve enabled two-factor authentication, a hacker with your password is out of luck — unless they’ve also managed to steal your cell phone.

2. Limit forwarding

When we’re sent a message we want to share, we often click “Forward” without thinking about the consequences. Where is the message going? Who will see it? Where will it be stored? If your email is hosted on a corporate server, it is likely there are certain security measures in place to protect any sensitive information contained in your private email. When someone forwards an internal email to a recipient outside of your company, however, you are exposing that data (as well as any other emails in the forwarded chain) to potentially unsecured, unencrypted servers.Similarly, if you’re a covered entity sending email containing protected health information (PHI) to a business associate, all it takes is one employee to forward that email to an unauthorized recipient to violate HIPAA.

3. Set expiration dates on your messages

While some of us can't stand a messy inbox, the average user doesn't bother cleaning up their private email, often seeing deleting email as a waste of time. Considering more than 50 percent of us receive at least 11 emails a day, can you blame them?That means that any sensitive information you send to a client could very well be sitting there months later. At that point, you no longer control the fate of your data.Luckily, Virtru lets you set an expiration date on your email, so that after a certain date, it will no longer be readable by the recipient (or anyone else, for that matter).

4. Understand your service provider's TOS

Your email provider's terms of service can tell you a lot more than their media interviews and advertisements can. For starters, it'll let you know what kind of security they are offering you. Are they encrypting messages on their server? Do they have protections against brute-force attacks? Is there any guarantee that your data is being protected? While you might think your email provider has your best interests in mind, there's a good chance that they don't have the same expectations you do. Take Google for example, which openly passes private email through automated scanning. After reading your email provider's TOS, you'll likely realize that keeping your private email secure isn't their first priority — that's entirely up to you.

5. Encrypt your email

The best way to keep your private email away from prying eyes and hackers is to use encryption. Encryption protects your private email by jumbling up your messages, making them impossible to decipher unless you explicitly authorize someone to read them.If you are using a client-side encryption service like Virtru, even if your inbox is compromised, the contents of your message will be unreadable. Likewise, you don't have to worry about your messages being intercepted after you send them, either by hackers or nosey service providers. As an added bonus, if your email ends up getting stored on a server outside of your control, you still have power over who gets to see it — and you can revoke that permission at any time.

(2) What expectation of privacy do you have when sending e-mail?

Ans: People continue to rely on their belief that the contents of e-mails, like phone calls, are sacrosanct and what is "said" in e-mail communication remains "confidential" to everyone other than the parties to them. However, that expectation of privacy is breaking down by the day. E-mails should more properly be viewed as a "postcard" or a conversation over a speakerphone, both open and available to a passerby to hear or see, than like a private "confidential," "sealed" letter.1

Seeking to ensure that e-mails remain confidential, however, does not take much effort on the part of the sender or recipient. That, no doubt, is why courts are increasingly rejecting arguments seeking to prevent the disclosure and use of putatively confidential e-mails in court proceedings where the e-mail user has done little to maintain its alleged confidentiality.

Reasonable Expectation of Privacy in Text Messages

In reaching the determination that text messages can attract a reasonable expectation of privacy, the SCC considered whether there was an expectation of privacy in the text messages and if so, whether this subjective expectation was objectively reasonable.

In evaluating the reasonableness, the SCC looked at three factors further discussed below: (i) the place of the search, (ii) the private nature of the subject matter, and (iii) control over the subject matter.

i. Place of the Search What is the place of an electronic text message conversation? The SCC determined that, unlike traditional methods of communication involving physical documents, electronic conversations such as text messages do not occupy a physical place; rather, they create "private chat rooms" between individuals. These rooms, although electronic, are the place of the search and this suggests there would be a reasonable expectation of privacy in text messages.

ii. Private Nature of Text Messages

The SCC concluded that given the intrinsic private nature of texting wherein individuals specifically choose the recipient of their texts and are more inclined to discuss personal matters, it is reasonable to expect these private interactions to remain private.

iii. Control over Text Messages

Finally, and most importantly, the SCC analyzed the notion of control over text messages once they are sent. Both the prosecution and the dissenting judges believed that once control over a message is lost – when it is received by another party capable of disclosing it — the reasonable expectation of privacy of the message is also lost. The majority of the SCC disagreed and stated that the risk that the contents of a text message can be disclosed does not negate a reasonable expectation of privacy in such an electronic conversation. In essence, a person does not lose control over the information simply because another person possesses it or can access it.

(3) If you had a secret message to send, how would you do it?

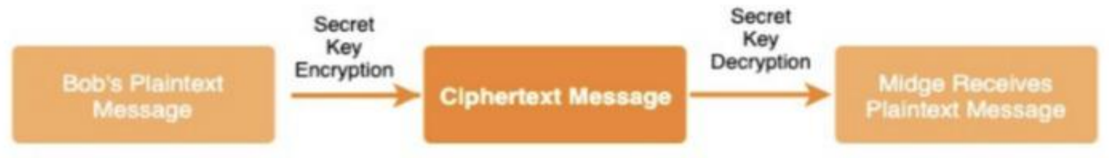
Ans: Cryptography is the science of using codes and ciphers to protect messages, at its most basic level. Encryption is encoding messages with the intent of only allowing the intended recipient to understand the meaning of the message. It is a two way function (you need to be able to undo whatever scrambling you've done to the message). This is designed to protect data in transit.One of the earliest ciphers involved a simple shift. For example, if you just shift all the letters in the alphabet by a few, the alphabet might look like the following:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

NOPQRSTUVWXYZABCDEFGHIJKLM

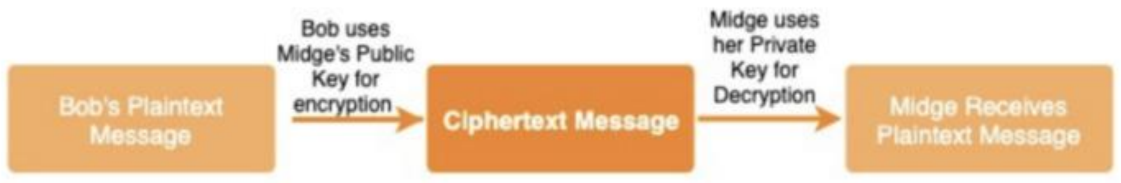
Then, each letter of the alphabet corresponds to a different letter, but it is difficult to figure out which one, if you don't already know. Using this cipher, the message, 'Hello' translates to 'Uryyb'.Unfortunately, advances in analysis, particularly pattern analysis driven by very powerful computers, made these types of cyphers very easy to break.In response to that, we've developed very strong, complex algorithms. These can be broken down into two basic types of encryptions—symmetric algorithms and asymmetric algorithms.

Symmetric algorithms are also known as 'secret key' algorithms, and asymmetric algorithms are known as 'public key' algorithms. The key difference between the two is that symmetric algorithms use the same key to encode and decode (see the first figure below), while asymmetric algorithms use different keys for encryption and decryption (see the second figure below).



As you can see in the above figure, with symmetric encryption, if Bob and Midge want to communicate, Bob first encrypts his message with the secret key (the encrypted message is called ciphertext). Then he sends it to Midge. Midge then decrypts the message with the same secret key and is able to read the message. To send a message back, the process is reversed.This process is fast, scalable, and very secure. The problem with it is that it requires both parties to already have the same secret key. If they don't, they need to pass it along insecure channels, which essentially removes the security of the encryption. With Asymmetric encryption, as in the above figure, if Bob and Midge want to communicate, Bob encrypts his message with Midge's public key and sends it to her. She then decrypts the message with her private key to read it. To send a message back, the process is reversed.

In this way, anyone can send Midge a message, as she can make her public key available to anyone, but only she can decrypt a message (as she keeps her private key secret). It also solves the need to pass a secret key along insecure channels, because there is no need to pass a secret at all. The disadvantage is that it requires everyone who wants to communicate to have two different keys (not scalable), and it is relatively slow.


Bob's Plaintext Message
Bob uses Midge's Public Key for encryption
Ciphertext Message
Midge uses her Private Key for Decryption
Midge Receives Plaintext Message

(4)How could you automate e-mailing many people?

Ans: Start from scanning your inbox and exploring which repetitive actions you recently made.

Combining Gmail + Tasks

Take advantage of other Google services like Google Tasks. This simple-to-use tool comes integrated with Gmail so you can easily add emails to your to-do lists if you want to read them later. You can set up a due date and link it to your Calendar, add some notes, and define subtasks.

What is more, you can use built-in filters to automate tasks right within Gmail, mark and sort emails, and improve your overall productivity.

Ways to Automate Email Tasks in Gmail

You may not know it yet, but Gmail offers a built-in functionality to manage your inbox. Let’s look into some of the tasks you can automate.

Use Labels to Automatically Sort Incoming Messages

You are receiving tons of emails and they all have different level of importance. To manage them properly, you can create several categories and then sort messages manually, but it can take a lot of time. Luckily, there is a way automatically sort emails into folders in Gmail.Right in the search bar, there is a dropdown menu that allows you to sort messages by subject, sender, or date. What is more, you can also set up certain terms as labels for more specific filtering. Gmail can automatically label incoming messages so you can just click on this mark and read all related emails.

Moreover, you’re free to set up as many labels as you need to organize your inbox as precisely as possible.

Step-by-step guide

Click on the dropdown arrow next to the search bar;

Type the label word into the subject line and use "Create filter" command;

Check the "Apply the label:" box, open the dropdown menu and click "New label…";

Type in your label, click "Create" and "Create filter";

Now you can see the label next to all incoming messages with subjects that contain the respective search criteria.

Use Filters to Automatically Forward and Mark Emails

Another trick to keep your inbox organized is to deal with irrelevant messages or requests that can be completed by your assistants. For example, if someone sends you an email and mentions a meeting in a subject line, you might want to forward it to your secretary and don't worry about scheduling.

You can sort Gmail emails by sender or subject and automatically forward them to your assistant or relevant department. You can also mark emails as read or important, move them to another folder, etc.

Step-by-step guide

At the top of the screen find the search bar and click the dropdown button;

Type in the keyword in the "Subject" field and use "Create filter" command;

Find the "Forward it" box and fill in the address of your choice;

Also, you can click "Mark as read" so the messages won't be displayed as unread;

ASSESSMENT QUESTIONS:

(1) Why do email services "read" your email? What is their goal?

Ans: -> To answer this question first we need to go deep in the history and the answer lies under it. One of the early email services which where quite safe was Gmail which was started in 2004 which was free to use for everyone.

Free email services are usually paid for by showing you advertisements. Some email services scan your emails in order to show you personalised or targeted ads. You could argue that that's a benefit, because you'll see ads in which you might have some interest. You could also argue that your emails are private, so it's an invasion of privacy. Either way, it's different from scanning your emails to stop viruses and phishing attempts, which nobody wants to stop.

There are several options, but the problem is bigger than you may think, and likely to get worse. When Google launched Gmail in 2004, it did indeed scan your emails for advertising purposes. You could only avoid it by using a paid-for version of the service. That changed last summer, when Google announced that it would no longer scan emails to tailor adverts.That didn't mean you would get adverts picked at random. As Alex Hern explained here, "the adverts will now be targeted in the same way as other Google services, based on information gleaned from other activity on users' profiles, such as their searches, browsing activity, and even physical locations".

Google has the world's largest known tracking system, thanks partly to Google Analytics – which is used on far more websites than any Facebook tracker – and its DoubleClick advertising business. Add Google Search and it already knows so much about you that scanning your emails may well be superfluous. The good news is that you can delete a lot of it, with the help of a Guardian article by Dylan Curran: Here is all the data Facebook and Google have on you.

The goal behind reading your emails-

The problem is that Google has lots of other reasons to read your email, and you may find some of them useful. For example, Google might tell you about traffic jams or flight delays, put appointments in your calendar, or offer to write quick email replies for you. These features involve AI software reading your emails, tracking your website visits, and your location. In the future, it may include information from home speakers (Google Home etc), household gadgets (Nest etc), fitness bands, smartwatches, cars (Waymo) and other connected devices.

(2) How does PGP secure email differently than GMail?

Ans: PGP ("Pretty Good Privacy") is a powerful, free cryptography package that lets people exchange files in a private, encrypted format, and also provides message authentication. PGP is called a public key system. Each person using PGP has both a public and a private key. Each key is actually a digital signature (a small file with a stream of uniquely generated characters). The public key is widely distributed to any correspondents, while the private key is guarded with secrecy.An encrypted message in PGP is scrambled in a complex way to make it unreadable to anyone other than the intended recipient. You can use your private key and the recipient's public key to generate a message that can be unscrambled only by the recipient who has your public key as well as a personal private key, allowing you to easily exchange encrypted messages with anyone with whom you have exchanged public keys.

Conventionally, a single-key system is used to encrypt files; the same key used to encrypt a file is also used to decrypt it. The key thus must be carefully transferred from person to person, since if it were found, anyone could decode messages. This type of keying is known as symmetric.

PGP is also used for authentication. A private key generates a unique digital signature attached to a message. Anyone holding the creator's public key can verify that the message was generated by the person holding a genuine private key, and whether the message was altered in transit or not.

PGP secure differently than Gmail because in Gmail your mail are been read by the Gmail service provider for their benefits for an extent. Whereas PGP only the person whom the mail is been send can access it and even the Gmail service provider’s can’t even see.

(3)why dont people use services like PGP more often?

PGP is pretty good privacy. It is an encrypted email solution that masks your messages before you send them. PGP works using public-key cryptography, meaning that every user has their own key pair — one private key, and one public key. The private key is used to decrypt messages, while the public key is used to encrypt them.
Your public key can be openly shared with just about anyone without incurring any risk. Even if a potential bad guy were to obtain your public key, they wouldn’t be able to decrypt your messages. For that, they would need to access the private key, which only you have. While PGP is very secure, it’s also a huge hassle. If you’re going to send someone an encrypted message, not only do they have to be using PGP, but you also have to exchange keys first. Likewise, once you have a collection of keys, it’s up to you to hold on to them. If they get corrupted, or if your computer goes up in smoke, you have to repeat the process. Because in the modern world,

PGP is only necessary if you don't trust your email provider, and basically everyone does. Assuming you and the person you're emailing both use a reasonably mainstream provider, your connection to the email website or email server is encrypted by SSL, and the connection between the two email servers is encrypted. Your email provider can access your email, but most people simply aren't worried about that. This also means that such a service would only really be useful to businesses, and only inter-business communication, at that. The services that do offer encryption that works past their servers often simply use various forms of PGP, meaning that the same hurdles present with that system are present here.

(4)What is Phishing?

Phishing is a cyber-attack that uses disguised email as a weapon. The goal is to trick the email recipient into believing that the message is something they want or need — a request from their bank, for instance, or a note from someone in their company — and to click a link or download an attachment. If users take the bait and click the link, they're sent to an imitation of a legitimate website. From here, they're asked to log in with their username and password credentials. If they are gullible enough to comply, the sign-on information goes to the attacker, who uses it to steal identities, pilfer bank accounts, and sell personal information on the black market. Phishing is the simplest kind of cyberattack and at the same time , the most dangerous and effective.

Types of phishing:-

(1)SPEAR PHISHING

(2)CLONE PHISHING

(3)PHONE PHISHING

FEW SIGNS TO IDENTIFY PHISHING

The email makes an offer that sounds too good to be true. It might say you've won the lottery, an expensive prize, or some other over-the-top item.

- You recognize the sender, but it's someone you don't talk to. Even if the sender's name is known to you, be suspicious if it's someone you don't normally communicate with, especially if the email's content has nothing to do with your normal job responsibilities. Same goes if you're cc'd in an email to folks you don't even know, or perhaps a group of colleagues from unrelated business units.
- The message sounds scary. Beware if the email has charged or alarmist language to create a sense of urgency, exhorting you to click and “act now”

before your account is terminated. Remember, responsible organizations do not ask for personal details over the Internet.

- The message contains unexpected or unusual attachments. These attachments may contain malware, ransomware, or another online threat.

(5)What is spear phishing?

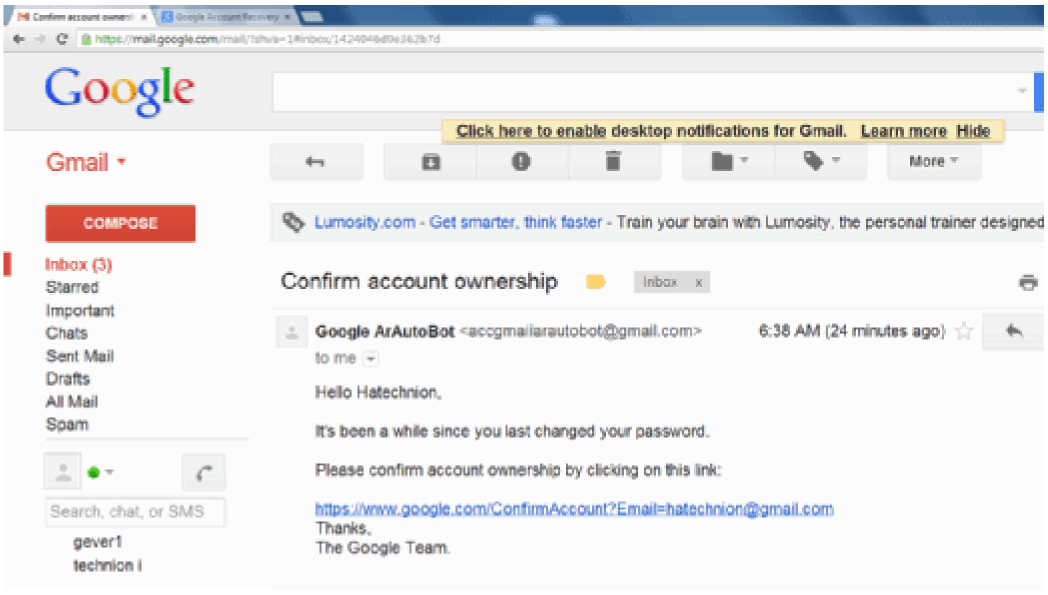
Spear phising is a type of Phishing. While most phishing campaigns send mass emails to as many people as possible, spear phishing is targeted. Spear phishing attacks a specific person or organization, often with content that is tailor made for the victim or victims. It requires pre-attack reconnaissance to uncover names, job titles, email addresses, and the like. The hackers scour the Internet to match up this information with other researched knowledge about the target's colleagues, along with the names and professional relationships of key employees in their organizations. With this, the phisher crafts a believable email.

For instance, a fraudster might spear phish an employee whose responsibilities include the ability to authorize payments. The email purports to be from an executive in the organization, commanding the employee to send a substantial payment either to the exec or to a company vendor (when in fact, the malicious payment link sends it to the attacker). The sender may request that the user reply directly to the email, or the message may include a malicious link or attachment that installs malware on the target's device, or directs the target to a malicious website that is set up to trick them into giving sensitive information like passwords, account information or credit card information.

Spear phishing can be more difficult to identify than phishing attacks due to the personal details that give the messages an air of validity. However, some of the characteristics that are common to phishing emails are also common to spear phishing emails:

- The sender’s email address is spoofed. The email address looks like it’s from a trusted individual and/or domain, but closer inspection reveals a typographical error or the exchange of one alphanumeric character for another that closely resembles it.
- A sense of urgency, particularly as it relates to performing a task that goes against company policy. Attackers evoke a sense of urgency to exploit the recipient’s desire to do good or to simply be helpful.

- Poor grammar, typographical errors or unlikely language within the body of the message. The body of the email does not sound like other messages from the supposed sender. Perhaps the tone is too informal or the jargon is incorrect for the recipient's geographic location or industry.


Confirm account ownersh
Google Account Recovery
https://mail.google.com/mail/?shva=1#inbox/142404d9e362b7d
Google
Click here to enable desktop notifications for Gmail. Learn more Hide
Gmail
More
COMPOSE
Inbox (3)
Starred
Important
Chats
Sent Mail
Drafts
All Mail
Spam
Search, chat, or SMS
gever1
technion i
Lumosity.com - Get smarter, think faster - Train your brain with Lumosity, the personal trainer designed
Inbox

Confirm account ownership

Google ArAutoBot <accgmailarautobot@gmail.com> 6:38 AM (24 minutes ago)
to me

Hello Hatechnion,

It's been a while since you last changed your password.

Please confirm account ownership by clicking on this link:

https://www.google.com/ConfirmAccount?Email=hatechnion@gmail.com
Thanks,
The Google Team.

APPENDICES:

Table of Contents

The OpenPGP keys of the FreeBSD.org officers are shown here. These keys can be used to verify a signature or send encrypted email to one of the officers. A full list of FreeBSD OpenPGP keys is available in the PGP Keys article. The complete keyring can be downloaded at https://www.FreeBSD.org/doc/pgpkeyring.txt.

D.1. Officers

D.1.1. Security Officer Team <security-officer@FreeBSD.org>

```
pub   rsa4096/D39792F49EA7E5C2 2017-08-16 [SC] [expires: 2023-01-02]
      Key fingerprint = FC0E 878A E5AF E788 028D  6355 D397 92F4 9EA7 E5C2
uid                           FreeBSD Security Officer <security-officer@FreeBSD.org>
sub   rsa4096/6DD0A349F26ADEFD 2017-08-16 [E] [expires: 2023-01-02]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBFmT2+ABEACrTVJ7Z/MuDeyKFqoTFnm5FrGG55k66RLeKivzQzq/tT/6RKO9
K8DaEvSIqD9b0/xgK02KgLSdp0Bucq8HLDFYUk3McFa6Z3YwjobNCWkxc72ipvVl
uAOGN4H6fuoYOpeg4cLK1H9pktUIrzONTCixaZzc/Bu6X+aX4ywGeCfsuu8g5v03
fLCPBLLgf3Bm5wsyZ6ZaGmsmILrWzd+d/rbr35Mcc5BekdgywUI4R191qo1bdrw9
mEJP1V7Ik3jpExOsNnuhMTvm5OQMeCTfUvVEOtBU15QtbT+1LXF5FIOgML0LwS5v
RHZN+5w/xvzSnEULpj24UuMKLDs/u9rj8U/zET8QaE+oG7m/mr4jJWZEmdX8HKdO
WrpnVj6UAppk72qdBIEfLsOW2xB/NOjJpppbCQH3+sw7DRYA2UnKE9Mptj/KKiE4
cs4c8Cupo2WSu93lEZDC5rCrULpT2lFeEXnRYlC/5oIgY5w9sFide9VI4CzHkkWX
Z2NPW/i1w3mFhoXjvnNLGOYMfAMKPxsRC2/Bn3bY0IhKvuIZ4rAeu7FTmKDDqFKQ
YEcrUOW74ZVng17AB29xzjWr4zNJVvp/CybFiUb8JoKkwtVWRqAVZIEgenAjU40d
G5+W4e+ccL0mfTQfEBbXRjnL2BL2tnaoBR42cTfbZGRucPHz7MrlKBEeZQARAQAB
tDdGcmVlQlNEIFNlY3VyaXR5IE9mZmljZXIgPHNlY3VyaXR5LW9mZmljZXJARnJl
ZUJTRC5vcmc+iQJUBBMBCgA+FiEE/A6HiuWv54gCjWNV05eS9J6n5cIFAlmT2+AC
```

GwMFCQoek4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ05eS9J6n5cKd9A/9
Fz3uGjNy28D0ALT1d/JJGzdQ2R3YwspHk9KHBr1LePkog9wf1WRalwCeNtPmA+g5
cn24psuzOeh1tRElImTZ2eE2ENPZ9XzK/J0ok0nK42MvmIwmMCyz+CaWv9GXW+FK
0oXnFmHi4YaQUVN3p+45TGkD9T+O5biVww7P47n/NnWsTfhLx0bzC7LyjPKXINai
/LgPgtlcOgY65/YhW/qhADCkoU7qMp9is41jMjTu1WB3OBPJkUkNpHfu6r15y8FN
Wqsk7K4W6Obr/WQ6VKGGXgh/a5mTcaEoFGMO16uHijAY4nXeb2HGZlBKxgmPH9Ur
aT4A9Pz/n+rIRMrK+rs+msFPemQHHNBYxy+x99uBpRBNyT2Su6GouZIxu5J16aIM
V0ZyOy/dy7m/uJ4sMhJPqKkd8a+MoQs/2L1M1y1EAzsO/QZqIrKrCluaftNN9k/B
qU0XClSDqB6sRMF7HFzYqb+f+M6cwSL/3Cp1Yx4rZ/onEE/MdWp64+3R87dETTXd
5tWXQw04qOhfPri5cBTI7r3t/qMO1iNXCGSG5RJbGkas6N6t6Mj83L4ItjI8doLf
aSIWZjj1XP3/me2hFJ6h2G5y5A+khO4ZwhC0ATFSq1fYbVGHw5AtfthIgNn8FoWu
+Sb8h7/RqTr7F6LgWagAoAh0GtVj02SVABZjcNZz/AKJAjcEEAEKACEWIQQc9/9v
rfXKn74bjLLtZ+zWXc9q5wUCWZPcTAMFAngACgkQ7Wfs1l3PauflkRAAgYcaBX0Y
ic4btxKoP/eOVpgUciOPPKEhDCiloQDyf4XQnZFDoMfjgcHpbLTBZ6kiAz2UzDGr
fJ4yUqrD+xfixUfCd5YpwzsaSpCGzDzSxOBcP/SpuAFhe40awSOIf5MruQar9Mlf
33JyslDLULXXeewAq2pcGk0/WrrOragI6Cs2vPGy9XP96VvLxyhjrWjlKmnO+//w
UF8oIO5hhKoqbtoxxlcqJgsWVyHch0mnPzvr6GWwoPhFXocnh1oPdbLjX1AwmGm9
ltEYMge4QxONIXlXJR0TvuDuJOaLNvTOC3OI8L97fdBcZS7eNJrG5FAYR5Ft3ISf
KJowIsSLGDt/cYApqpyP2pv7FpCvnwHgXHYar7/q4zhngCFRxQ2DPUx1cIJQ3Bgh
HZolKyK1X7XE5ZVDfZ3s3gcHSVKS89pipgHHZNr4sSmOanA8rXHcyHS4o2zSi1ie
r4iBwnOk6cCd6UNzEIiq0y/XhP/sc7xeL0mn3wDuV7jDBP9sp65sexL1qtIAfnzL
pLQevm0z41ifrUH5nNeL6RdbXpaoXc8M4PJJeQKJDu04KzLcQpZdUdCJsbS6QO9w
srWR8enQXPEhz2CO4L77bM9TgYO29222jTqEPcbXcmxF/klxO1rpssTTHUnHHi1Z
LUGYCbZPjt+laTJ2YPHTjUtN1Jw85vSKCEuJATMEEAEKAB0WIQS7KNQLNg7uk2rt
FW/l97zLo73d+AUCWjSYRwAKCRDl97zLo73d+JKyB/9N5Ytao12nD5QzMLvceGh5
otCLN99TUryYiDVDLoNkBivq3jHQA/hOX2rwEueFq0+LF8/2DnglJuUICNtCxIzL
WXXf/Hr5iWBUQ0JxYNPQzzjdMSXGE0WMwYVpAbCGxHpIsetKLdHUCwneYhaywe3I
KzmRJSDJGV1IJB0sAfoFtgybZXHgIR61jQjtnNmmyYXliYCd0wmIhXQDFN91tzzG
+EZdJ3Fao9JsMC+x55jO6EOLVySZgRF5E8vCeKUWemQciKFC7EhKcljILPYAA21u

```
NmHCAgRHKWU9JMdFK0w9lQuN2HQaNfkahjarTNM/Q6LwxY0dLG0vVYifE085WFAf
uQINBFmT2+ABEACxi39m5nQZexzY3c9sg/w5mUYCD89ZNSkj427gduQMYYGn7YW6
jSPfVJ/V3+PDK824c0a0XasyDapQFY1CPTZYrReRPoyjb8tJjsSVGXXCTFpJZlFU
br6kS9mgcx58Sypke2PMVk73+W1N1Yco+nahfTECRuM2/T2zHHr0AdKuBPF28U+H
TxyLatKoIgQwHDs4E/f4ZTbAoHvu3PixAl7XHVXCgz0cHaLhRljXizbZDXngOdGm
lqdFlAIpL6/l8E3m1Er0m3IfFo6qSzWRHg/KaBGIL4YKetJ6ACjlkCe5qbatDpmk
gWlg3Ux4RBVjyCK834Xh7eZpEcNf2iwpm28glWh7XMHGUplTHkU3PWQ4vGfNxXB8
HBOd9r02/cHL6MiHwhCAfIzZGVtqR0i9Ira57TMdXTpJWNXUcgsCMsi/Bg2a+hsn
aiYLrZc18uNL5nqOqsqKG3c1TcmeN7nbxVgnrNST4AjteulkhmB9p8tNOXA3u979
OO0T5LPwdqIpobdZ0lfw4URnAGw4Wd4Sm9PtRw0RvuAk2M2e5KXNyxPWAuMVkoRR
a7wG6h/R8pki54Gexyc+JkfB4ZcOrzHNLurw6DhxroyfRs8WEgX0wNIGmJvCXSBG
54jb5w9qudYwzIg4YPfvuX8sfeY8MTNhal3rF0tvVloGj3l709wlaWlBYwARAQAB
iQI8BBgBCgAmFiEE/A6HiuWv54gCjWNV05eS9J6n5cIFAlmT2+ACGwwFCQoek4AA
CgkQ05eS9J6n5cKhWw/+PT0R4r2gPAxI8ESEe380BYOmneNAH24MFOgWXqWCj4zX
Uz992BVnW2aL5nH4O5d822LGeCrYUC7SCpQvlifdHZHjobgtizLTwuu40bc3gSOz
cxWlx2jKfx3Ezn6QQz2mhhK6fZ1AO0ObiQxQq25ldURep95L78E/C8XkCe11YlUR
ng3wQKeHM7awZWRw/QBC92haHuVtU3cx7At+zQL7jTBKSZqd34zzs0uoXIhk2h94
O07MMDZ8z8MeU337vdL+RKYtD2bljLwpf7/kqg1D/q44RJ4ZpZcha9G0GvtLaQg2
+MAPlLg1vOWZ8wOTLaQHm+uzYRpkqxkIV8OuVd4UikCd8t3VNjNG5rG/YRNIAX0A
UEzs6oMF5YOFE8LmykesbUHAbC07Vcb0AsT5u3XKixDiIpPdnYSwGlkvoOVVLdeh
q/aXLK9V8BpViG5+a8xP2fdF1eMqdnrKAsiO4GEiq193PN/FA049VeIs3fd0izAa
x7+ag1MGtoF5Pij5iTVJm6phH5SUd1P3FY3OmclxWj/MbL4ba/G/6FWcy5NXxdw9
L1bRqaM2KEHJ67aF6NZz7UMldwExAWzFbUon1LUpKysAukxVf0EnntydBeVOQ+JO
HdqEpirrVLMpxPttUB2xxbo947nMj7/Bnme2gvb0vxaC9xSGVxrpW9cg5iCwSdc=
=8rds
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.2. Security Team Secretary <secteam-secretary@FreeBSD.org>

```
pub   4096R/3CB2EAFCC3D6C666 2013-09-24 [expires: 2018-01-01]
      Key fingerprint = FA97 AA04 4DF9 0969 D5EF  4ADA 3CB2 EAFC C3D6 C666
```

```
uid                      FreeBSD Security Team Secretary <secteam-
secretary@FreeBSD.org>
sub   4096R/509B26612335EB65 2013-09-24 [expires: 2018-01-01]
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
mQINBFJBjIIBEADadvvpXSkdnBOGV2xcsFwBBcSwAdryWuLk6v2VxjwsPcY6Lwqz
NAZr2Ox1BaSgX7106Psa6v9si8nxoOtMc5BCM/ps/fmedFU48YtqOTGF+utxvACg
Ou6SKintEMUa1eoPcww1jzDZ3mxx49bQaNAJLjVxeiAZoYHe9loTe1fxsprCONnx
Era1hrI+YA2KjMWDORcwa0sSXRCI3V+b4PUnbMUOQa3fFVUriM4QjjUBU6hW0Ub0
GDPcZq45nd7PoPPtb3/EauaYfk/zdx8Xt0OmuKTi9/vMkvB09AEUyShbyzoebaKH
dKtXlzyAPCZoH9dihFM67rhUg4umckFLc8vc5P2tNblwYrnhgL8ymUaOIjZB/fOi
Z2OZLVCiDeHNjjK3VZ6jLAiPyiYTG1Hrk9E8NaZDeUgIb9X/K06JXVBQIKNSGfX5
LLp/j2wr+Kbg3QtEBkcStlUGBOzfcbhKpE2nySnuIyspfDb/6JbhD/qYqMJerX0T
d5ekkJ1tXtM6aX2iTXgZ8cqv+5gyouEF5akrkLi1ySgZetQfjm+zhy/1x/NjGd0u
35QbUye7sTbfSimwzCXKIIpy06zIO4iNA0P/vgG4v7ydjMvXsW8FRULSecDT19Gq
xOZGfSPVrSRSAhgNxHzwUivxJbr05NNdwhJSbx9m57naXouLfvVPAMeJYwARAQAB
tD9GcmVlQlNEIFNlY3VyaXR5IFRlYW0gU2VjcmV0YXJ5IDxzZWN0ZWFtLXNlY3Jl
dGFyeUBGcmVlQlNELm9yZz6JAj0EEwEKACcFAlJBjIICGwMFCQgH7b8FCwkIBwMF
FQoJCAsFFgIDAQACHgECF4AACgkQPLLq/MPWxmYt8Q/+IfFhPIbqglh4rwFzgR58
8YonMZcq+5Op3qiUBh6tE6yRz6VEqBqTahyCQGIk4xGzrHSIOIj2e6gEk5a4zYtf
0jNJprk3pxu2Og05USJmd8lPSbyBF20FVm5W0dhWMKHagL5dGS8zInlwRYxr6mMi
UuJjj+2Hm3PoUNGAwL1SH2BVOeAeudtzu80vAlbRlujYVmjIDn/dWVjqnWgEBNHT
SD+WpA3yW4mBJyxWil0sAJQbTlt5EM/XPORVZ2tvETxJIrXea/Sda9mFwvJ02pJn
gHi6TGyOYydmbu0ob9Ma9AvUrRlxv8V9eN7eZUtvNa6n+IT8WEJj2+snJlO4SpHL
D3Z+l7zwfYeM8FOdzGZdVFgxeyBU7t3AnPjYfHmoneqgLcCO0nJDKq/98ohz5T9i
FbNR/vtLaEiYFBeX3C9Ee96pP6BU26BXhw+dRSnFeyIhD+4g+/AZ0XJ1CPF19D+5
z0ojanJkh7lZn4JL+V6+mF1eOExiGrydIiiSXDA/p5FhavMMu8Om4S0sn5iaQ2aX
wRUv2SUKhbHDqhIILLeQKlB3X26obx1Vg0nRhy47qNQn/xc9oSWLAQSVOgsShQeC
6DSzrKIBdKB3V8uWOmuM7lWAoCP53bDRW+XIOu9wfpSaXN2VTyqzU7zpTq5BHX1a
+XRw8KNHZGnCSAOCofZWnKyJAhwEEAEKAAYFAlJBjYgACgkQ7Wfs1l3PaudFcQ//
```

UiM7EXsIHLwHxez32TzA/0uNMPWFHQN4Ezzg4PKB6Cc4amva5qbgbhoeCPuP+XPI
2ELfRviAHbmyZ/zIgqplDC4nmyisMoKlpK0Yo1w4qbix9EVVZr2ztL8F43qN3Xe/
NUSMTBgt/Jio7l5lYyhuVS3JQCfDlYGbq6NPk0xfYoYOMOZASoPhEquCxM5D4D0Z
3J3CBeAjyVzdF37HUw9rVQe2IRlxGn1YAyMb5EpR2Ij612GFad8c/5ikzDh5q6JD
tB9ApdvLkr0czTBucDljChSpFJ7ENPjAgZuH9N5Dmx2rRUj2mdBmi7HKqxAN9Kdm
+pg/6vZ3vM18rBlXmw1poQdc3srAL+6MHmIfHHrq49oksLyHwyeL8T6BO4d4nTZU
xObP7PLAeWrdrd1Sb3EWlZJ9HB/m2UL9w9Om1c6cb6X2DoCzQAStVypAE6SQCMBK
pxkWRj90L41BS62snja+BlZTELuuLTHULRkWqS3fFkUxlDSMUn96QksWlwZLcxCv
hKxJXOX+pHAiUuMIImaPQ0TBDBWWf5d8zOQlNPsyhSGFR5Skwzlg+m9ErQ+jy7Uz
UmNCNztlYgRKeckXuvr73seoKoNXHrn7vWQ6qB1IRURj2bfphsqlmYuITmcBhfFS
Dw0fdYXSDXrmG9wad98g49g4HwCJhPAl0j55f93gHLGIRgQQEQoABgUCUkGO5gAK
CRAV1ogEymzfsol4AKCI7rOnptuoXgwYx2Z9HkUKuugSRwCgkyW9pxa5EovDijEF
j1jG/cdxTOaJAhwEEAEKAAYFAlJBkdUACgkQkshDRW2mpm6aLxAAzpWNHMZVFt7e
wQnCJnf/FMLTjduGTEhVFnVCkEtI+YKarveE6pclqKJfSRFDxruZ6PHGG2CDfMig
J6mdDdmXCkN//TbIlRGowVgsxpIRg4jQVh4S3D0Nz50h+Zb7CHbjp6WAPVoWZz7b
Myp+pN7qx/miJJwEiw22Eet4Hjj1QymKwjWyY146V928BV/wDBS/xiwfg3xIVPZr
RqtiOGN/AGpMGeGQKKplkeITY7AXiAd+mL4H/eNf8b+o0Ce2Z9oSxSsGPF3DzMTL
kIX7sWD3rjy3Xe2BM20stIDrJS2a1fbnIwFvqszS3Z3sF5bLc6W0iyPJdtbQ0pt6
nekRl9nboAdUs0R+n/6QNYBkj4AcSh3jpZKe82NwnD/6WyzHWtC0SDRTVkcQWXPW
EaWLmv8VqfzdBiw6aLcxlmXQSAr0cUA6zo6/bMQZosKwiCfGl3tR4Pbwgvbyjoii
pF+ZXfz7rWWUqZ2C79hy3YTytwIlVMOnp3MyOV+9ubOsFhLuRDxAksIMaRTsO7ii
5J4z1d+jzWMW4g1B50CoQ8W+FyAfVp/8qGwzvGN7wxN8P1iR+DZjtpCt7J+Xb9Pt
L+lRKSO/aOgOfDksyt2fEKY4yEWdzq9A3VkRo1HCdUQY6SJ/qt7IyQHumxvL90F6
vbB3edrR/fVGeJsz4vE10hzy7kI1QT65Ag0EUkGMggEQAMTsvyKEdUsgEehymKz9
MRn9wiwfHEX5CLmpJAvnX9MITgcsTX8MKiPyrTBnyY/QzA0rh+yyhzkY/y55yxMP
INdpL5xgJCS1SHyJK85HOdN77uKDCkwHfphlWYGlBPuaXyxkiWYXJTVUggSjuO4b
jeKwDqFl/4Xc0XeZNgWVjqHtKF91wwgdXXgAzUL1/nwN3IglxiIR31y10GQdOQEG
4T3ufx6gv73+qbFc0RzgZUQiJykQ3tZK1+Gw6aDirgjQYOc90o2Je0RJHjdObyZQ
aQc4PTZ2DC7CElFEt2EHJCXLyP/taeLq+IdpKe6sLPckwakqtbqwunWVoPTbgkxo

```
Q1eCMzgrkRu23B2TJaY9zbZAFP3cpL65vQAVJVQISqJvDL8K5hvAWJ3vi92qfBcz
jqydAcbhjkzJUI9t44v63cIXTI0+QyqTQhqkvEJhHZkbb8MYoimebDVxFVtQ3I1p
EynOYPfn4IMvaItLFbkgZpR/zjHYau5snErR9NC4AOIfNFpxM+fFFJQ7W88JP3cG
JLl9dcRGERq28PDU/CTDH9rlk1kZ0xzpRDkJijKDnFIxT2ajijVOZx7l2jPL1njx
s4xa1jK0/39kh6XnrCgK49WQsJM5IflVR2JAi8BLi2q/e0NQG2pgn0QL695Sqbbp
NbrrJGRcRJD9sUkQTpMsLlQTABEBAAGJAiUEGAEKAA8FAlJBjIICGwwFCQgH7b8A
CgkQPLLq/MPWxmZAew//et/LToMVR3q6/qP/pf9ob/QwQ3MgejkC0DY3Md7JBRl/
6GWfySYnO0Vm5IoJofcv1hbhc/y3OeZTvK4s+BOQsNokYe34mCxZG4dypNaepkQi
x0mLujeU/n4Y0p0LTLjhGLVdKina2dM9HmllgYr4KumT58g6eGjxs2oZD6z5ty0L
viU5tx3lz3o0c3I9soH2RN2zNHVjXNW0EvWJwFLxFeLJbk/Y3UY1/kXCtcyMzLua
S5L5012eUOEvaZr5iYDKjy+wOxY4SUCNYf0GPmSej8CBbwHOF2XCwXytSzm6hNb3
5TRgCGbOSFTIy9MxfV5lpddQcdzijmuFSl8LySkL2yuJxjlI7uKNDN+NlfODIPMg
rdH0hBSyKci6Uz7Nz/Up3qdE+aISq68k+Hk1fiKJG1UcBRJidheds29FCzj3hoyZ
VDmf6OL60hL0YI1/4GjIkJyetlPzjMp8J7K3GweOUkfHcFihYZlbiMe7z+oIWEc7
0fNScrAGF/+JN3L6mjXKB6Pv+ER5ztzpfuhBJ/j7AV5BaNMmDXAVO4aTphWl7Dje
iecENuGTpkK8Ugv5cMJc4QJaWDkj/9sACc0EFgigPo68KjegvKg5R8jUPwb8E7T6
lIjBtlclVhaUrE2uLx/yTz2Apbm+GAmD8M0dQ7IYsOFlZNBW9zjgLLCtWDW+p1A=
=5gJ7
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.3. Core Team Secretary <core-secretary@FreeBSD.org>

```
pub   rsa2048/0CB403E4E95B96EC 2018-06-30 [SC] [expires: 2020-06-29]
      Key fingerprint = 9F02 836F 50D3 AD5A B75A  C588 0CB4 03E4 E95B 96EC
uid                           FreeBSD Core Team Secretary <core-secretary@freebsd.org>
sub   rsa2048/133C3338A5B95A60 2018-06-30 [E] [expires: 2020-06-29]
      Key fingerprint = FA37 B8AA C667 C3AA D310  751D 133C 3338 A5B9 5A60


-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBFs3wcYBCAC7nlaUTMqyT7PBSFLtW/LleSz7BNUwqSTo8LfUVJOY5G/pzWt5
```

```
Mqjqh4oJcW/MvKFTDeRaJ2mHp+vELxIP7wO3gcP36dXgImw6sXwBTkPlKpmmFRm1
M+QqnCCrrLHtCznWaDg+1fTHmyQpFHpg37XzA1Z5ev6PryEUYJkcBP77oNCTY933
86sXOqRAJRywvN/LEkAoaawqBz0CpkNTOBACoJZRV8i9CIklEOy8J+hNzGtJpHkg
FxUOXWj7z+2y6UOR4GzSpYAWJGbtwEcpGPfhqJk5M5eZ6PJcwzZ6LeLKgGFzNi6r
tlShQh5LT7wAKkTrBsZ9vckyyuTEtqgdGCmhABEBAAG0OEZyZWVCU0QgQ29yZSBU
ZWFtIFNlY3JldGFyeSA8Y29yZS1zZWNyZXRhcnlAZnJlZWJzZC5vcmc+iQFUBBMB
CgA+FiEEnwKDb1DTrVq3WsWIDLQD5OlbluwFAls3wcYCGwMFCQPCZwAFCwkIBwMF
FQoJCAsFFgMCAQACHgECF4AACgkQDLQD5OlbluyRZAf/VG9VWpIsofcoHwDxhYAL
mm+xbuP/eq1/Q8HeO3XVhA/HZF5nvSKZbD8F+ujaHDH/waNStWb3wUK87l9AfB6G
QFMVYjVQWrPwgpwFtGjL9zLMCBS3T+ysuub+xSuPhr1KQHgKB4+t6NLoBlSwP+76
sLLx0SILGwTpsb0r84etaECgp5ymAXijbzIB0Pu44Y+DjZimBEVuw2YRZ4/Ug/3z
pcNQqpjbrHNYjU6AOZEHXftbXwuWfgdjINnrWpvTwkKVnU0FhGXV9UYWP2UAxE5u
OyAvIyYFbX10iSFQGUXle3eg6IuHncT5u6P1IxQM++d/TJIbKrQW+xdr+1I+vUrS
rokCMwQQAQoAHRYhBHLPrCF5vLAktbVFkANvbJ7n856/BQJbOJDdAAoJEANvbJ7n
856/1swQAN2QKGe1riRm9jKVxC8AMy57+Tzu1ITGDDUf6dH2+gxx0K5GoVmtdhLL
2qrmDJEqP7K232T25cU5zStQnaTHpEIUklY8Rn1Fati8+IZBdpemG4BXTzGnNDQ0
FS6PxuxOFvcELOFvuUil3PP7ArMKI9jfjxisEkOWFuwQVyIPeApcQuf8vyqrfTnV
/Qes/XhySrvsEL+ehq2OEorl6YjMB2/lVK2lVWYrWJ91Oq8Vwp0G09whZEMhMabQ
D1OxlmM6kofkTioM8DOmbGTbOXhiiiiCUI41pOAOzF9SrCqCpLV2OyrPFz7J+GU9
6u+DPPZyy7O8NmjdDsyrDg2hhbTwWC4dvW+QMJSWZ8Bo8eMx8b5ti9RX0XPEIwao
KrCKh3aemGgkP8zcVbFWOzOji8aXrpWrRr/oxQmJxE49d2j1oF4LydIfhDxOnfOF
428pVhDXDLjf0xdUIVQqCsOBQvzwVPWTQVOFSakVFNRYP6/SXyF5eUf5E6iSExKn
fn+G4FtrJd6QNwNUquI2LF8CEhJBpLNBqjJW3WEv1tDzU+rqS9QpHzSmLzLqtiE+
5HqynvOPXGRRsAcUOLmV4fMUGRH8tpNoH4iBEc7LmoFTQXIf6oJClaiwRkFKuT9c
2XlkJ4ca6fxU4KyoHtR6pmMNkLIcehfpoL11+TPyyBjNd2TwLpLbiQIzBBABCgAd
FiEEwHv14xCuZL9hILD2NqfAX+Hs+bsFAls4kV0ACgkQNqfAX+Hs+bvRrw//QVea
9diHHbzxq84yp4eOGQoj86usPSV+IOZN27+e6QDYR8ZsxqFE5wQycSAdyqo0n42Q
EDE6tnn+/HhyFogr7kF8CRJMtsSlwKgDrMMYjVPnP2fP5VFxAF36epSRgcGC0Lqh
Ris+xjfSzXM2oNiiebPu2MOe8qOe8LVGJMyuxJZbb/OuEfgLGLKtjcJ1SujKhzLl
```

```
TVS8JSSVRbxk62huh/Mo80eCKHMV+/NmbHP4QKZBOVSWn0U/lrm+SyDR78l3EhtN
x/KIfhiPZENYTjSBSxa8F/Vg19bcmUedLapcN9J8q2KVNx7VuiPz+X2ww/dOKFR0
FxwOvCweGFRNRyoytF4ziw0Gwt78RHw4OdhQg8YH38kbrRFvf2YqiddGUA2UWwKi
HRdj9ZGemzL++OE/MZvgODVhZA6V5QU/B9bR3xfnVcBsPyGTrlQ8XZ9aY1wBMTrS
TTbS3sD7HuyS4PO8rt3iZy50UDMc5v55Pr5SIPiaUdyV8Y401oOWnKvKgKtHzBtC
2ADT+iZk/I4a3iDj4hw07Y+O1Voqp72LaACGhqWqkN0zqoKq3TvD/ukEZwgsvDdP
ErzPUanN31gn055PlpWYQBVoLjupH8SXahrdTmo15Xjdr97VHCuABNT4Kh3QDELU
vQtF0IB+S+VQfTVR5wkC1OLj8J1edvoXlsVzREW5AQ0EWzfBxgEIAMZxwaI3hZ2G
je7L8N1TFfPJA62kMGzzFDvFqeH8mDPOXkd4JC4y2EIBySPS36y0c1MJM79oOkKI
6DQLyUb3p4hGZbEVKidAwXvp4t5x1QJ0bpodHc/7xh95EP11Lf8C/DFP5Js3YVPl
MsdeVhx7J8itQuivoLJrZVTgKSgFepatLuXXKUttYAJNcU11ziPwTlzjEuTx4X6V
RimPrp8+/dbkRmPhsDqMXrqJmjeNarYK9F0xKlaWnIhtyZnNXtHrdtQE/VOBjoXN
0NXiuJg02JZGqZuBM80Ig7yBdmUlZdPrxkYw92+kxHIdySM3+WYbGu/e6T/VY6wx
7KW2IV3u3b8AEQEAAYkBPAQYAQoAJhYhBJ8Cg29Q061at1rFiAy0A+TpW5bsBQJb
N8HGAhsMBQkDwmcAAAoJEAy0A+TpW5bsp0AH/Rht32xeJQk59UgDf7BPHiiphgg8
P1qmRVd6OZJ6GoVYWjJ87+gU9sChbZUTCFioiIYLWPbhm9AJKy1KDrcnP0zYjWL2
SKjezMbru9cgFYk6R3LO+mK5DwtGMgyzipKAN8Kh92pX2WERUeMFulkYa4+rdVkP
kBtB49hmDj25GPw/72Vuksg5m7sbpEZzt6JjXQN0ynDjBuizE/HYm2E8VW5tH1aH
wdzVGruNVIOMMF3gHKbJbrxKiq/SPJfph0YGeL6v5bF9mgizGamEUn9YHVkCqZ7z
wDuSIDVTSiQQOJesD58WOADCDINEP3uXFhlI1A0Au7X+XYyjIjHCdyTNhBI=
=5VKx
-----END PGP PUBLIC KEY BLOCK-----
```

D.1.4. Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>

```
pub   rsa2048/D8294EC3BBC4D7D5 2012-07-24 [SC]
      Key fingerprint = FB37 45C8 6F15 E8ED AC81  32FC D829 4EC3 BBC4 D7D5
uid                           FreeBSD Ports Management Team Secretary <portmgr-secretary@FreeBSD.org>
sub   rsa2048/5CC117965F65CFE7 2012-07-24 [E]
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----


mQENBFAOzqYBCACYd+KGv0/DduIRpSEKWZG2yfDILStzWfdaQMD+8zdWihB0x7dd

JDBUpV0o0Ixzt9mvu5CHybx+9lOHeFRhZshFXc+bIJOPyi+JrSs100o7Lo6jg6+c

Si2vME0ixG4x9YjCi8DisXIGJ1kZiDXhmVWwCvL+vLInpeXrtJnK8yFkmszCOr4Y

Q3GXuvdU0BF2tL/Wo/eCbSf+3U9syopVS2L2wKcP76bbYU0ioO35Y503rJEK6R5G

TchwYvYjSXuhv4ec7N1/j3thrMC9GNpoqjVninTynOk2kn+YZuMpO3c6b/pfoNcq

MxoizGlTu8VT4OO/SF1y52OkKjpAsENbFaNTABEBAAG0R0ZyZWVCU0QgUG9ydHMg

TWFuYWdlbWVudCBUZWFtIFNlY3JldGFyeSA8cG9ydG1nci1zZWNyZXRhcnlARnJl

ZUJTRC5vcmc+iQE4BBMBAgAiBQJQDs6mAhsDBgsJCAcDAgYVCAIJCgsEFgIDAQIe

AQIXgAAKCRDYKU7Du8TX1QW2B/0coHe8utbTfGKpeM4BY9IyC+PFgkE58Hq50o8d

shoB9gfommcUaK9PNwJPxTEJNlwiKPZy+VoKs/+dO8gahovchbRdSyP1ejn3CFy+

H8pol0hDDU4n7Ldc50q54GLuZijdcJZqlgOloZqWOYtXFklKPZjdUvYN8KHAntgf

u361rwM4DZ40HngYY9fdGc4SbXurGA5m+vLAURLzPv+QRQqHfaI1DZF6gzMgY49x

qS1JBF4kPoicpgvs3o6CuX8MD9ewGFSAMM3EdzV6ZdC8pnpXC8+8Q+p6FjNqmtjk

GpW39Zq/p8SJVg1RortCH6qWLe7dW7TaFYov7gF1V/DYwDN5iEYEEBECAAYFAlN2

WksACgkQtzkaJjSHbFtuMwCg0MXdQTcGMMOma7LC3L5b4MEoZ+wAn0WyUHpHwHn
n

pn2oYDlfAbwTloWIiQEcBBABAgAGBQJQDuVrAAoJENk3EJekc8mQ3KwIAImNDMXA

F8ajPwCZFpM6KDi3F/jpwyBPISGY1oWuYPEi1zN94k5jS90aZb3W8Y8x4JTh35Ew

b6XODi3uGLSLCmnlqu2a80yPfXf5IuWmIQdFNQxvosj9UHrg+icZGFmm+f0hPJxM

TsZREv3AvivQfnb/N3xIICxW4SjKSYXQcq4hr4ObhUx7GKnjayq+ofU2cRlujr87

uOH0fO3xhOJG4+cX5mI1HGK38k0Csc1zqYa/66Qe5dnIZz+sNXpEPMLAHIt1a45U

B967igJdZSDFN33bPl1QWmf3aUXU3d1VttiSyHkpm4kb9KgsDkUk1IJ5nUe9OXyd

WtoqNW5afDa5N0aIRgQQEQIABgUCUA7lwwAKCRB59uBxdBRinNh2AJ41+zfsaQSR

HWvSkqOXGcP/fgOduwCfUJDT+M1eXe2udmKof/9yzGYMirKJASIEEAECAAwFAlAa

IT8FAwASdQAACgkQlxC4m8pXrXwCHAf+J7l+L7AvRpqlQcezjnjFS/zG1098qkDf

lThHZlpVnrBMJZaXdvL6LzVgiIYVWZC5CSSazW9EWFjp9VjM7FBHdWFZNMV7GAuU

t0jzx6gGXOWwi+/v/hs1P11RyDZN5hICHdPNmyZVupciDxe+sIEP9aEbVxcaiccq

zM/pFzIVIMMP5tCiA42q6Mz3h0hy6hntUKptS8Uon6sje5cDVcVlKAUj1wO2cphC
```

qkYlwMQfZV5J9f/hcW5ODriD3cBwK8SocA2Cq5JYF8kYDL1+pXnUutGnvAHUYt87
RWvQdKmfXjzBcMFJ2LlPUB1+IFvwQ13V9R8j9B/EdLmSWQYT9qRA2okCHAQTAQoA
BgUCV1XMpwAKCRCtu/hhCjeJt2CyD/9JLe+Ck23CJkeRSF8oC+4SFOUdSAmejSzn
klPwmEClffABYd/kckO1T6um+2FUcXuJZQE1nKKUNvZ8pBWwsm1RDHsyroKi/XB1
0a1Tdx/rvlU88ytbeLfUCLzoCrf6pkMQWoU6/3qS6elV0WwOlDufk+XjD1sja2wu
sshG8y+1WCA5JjP3rZdD9NVdzo5DgkotTRUfuYN1LJIN4zlDgHj7FVP7wW7+R0cZ
FoOiNsLJCA0FN8SiyU98UysjawLiIY9dTJz6XVA0DgB0TZWO3mWiDjITeKrdGcqf
PNiJhmvUKBkn07YpTPNfkoTT/p/q5ChYmu0ubGeyS1ELKjmklJ+DzynfZLzvnXYX
Ngo5ckeuqEqUNxM0J63v8lmfhDRROFveqHWdp0XMxXVmR5bMunSldg5EZsoLyQbN
+ScIPnDTAEPGrCtf0t84RQxNQeET6/WBbZfzeSeAFmpBFCdicsZ6Mjwtwjr4+o15
n1QMTZco1NaTqf8vXwzl9wM4aYtg1OkF4z8HdHuy50CHCet4mT5eJgwZUfFvXdbM
pHXprEI0Y9OOL4aMinC1egF3dXt/0n57i6CE+E2k3UJPNvMrtp0HaDEnKZ8cfkBU
EBzkUYi5wwqntHV2JRisqoRnHdvJT7ImlHMe7WaJsifBK874PnToaKg8P6K1Tph+
FyLxULaYjYkCHAQSAQgABgUCVBg2zwAKCRDqsDxYv9xHj1klEADXYJdHC3zsdx7w
DsJsttWdykcZoOd/VUKUdN0BAU72nLV0tLn4uFjETA6MhHZVxzwIDTeLB8kqyEpc
fZnoVbqJIUJz1sJXMdOty7CwZzlZlAwmUaIfFiazJY1p398JbyYfSrVKNOpw9wCm
Db7WP9dBritwvjaLzu8HQsiztO0S/5ha/EDfTU3qocBUTjbCtGR9LqAmPE4X8+li
F2EfZMEoJd3rJWsYv2y/k6pSgC/MpQewnyr6f+JQ/781UoZB6PpxCxfu4D6xlOyd
ERBUg+FfDAWYR+KX+DGOalRlUyaSz8Nvxl8/b0Im/AQhx9afqyEZxIDpg52zt8jJ
t3wx23YP8EQGUgwF8pIrj3wFSBSG3a/cskiBNUIhChIR9hQrVPUahN/jx7DGAGxk
/Ka9qsRGYTHfSr9jjTUQ+htfeFBRDR0nkZKMo5+Wk/cAcBKVbPlBpwvnzT3fh+wL
cF3ErBbx5jp+BoFee8D6ATeUvQxMcgVbDPUkgMsy3EtKMVO10jhIoXoVV+Sg9GZ8
zMEy1tORKn0zsd2ZgXC2sRJOm5ttCSdYQ4ddbM1A9jg6tiRx4hES16GDywvkL8P2
M9+qyIfjQxjGU33f/r8zp9DyNT1VlrtwhFxtOoMdmrsbYOCTja4Xg14hK1hRac0k
GB7bj6w97p8uMrQT3PlSMtoyrRyo7bkBDQRQDs6mAQgAzNxJYpf5PrqV8pdRXkn3
6Fe45q671YtbZ2WrT7D0CVZ8Z+AZsxnP/tiY1SrM2MepCeA2xBAhKGsWBWo1aRk5
mfZOksKsiXsi2XeBVhdZlCkrOMKBTVian7I1lH59ZnNIMX0Nl0tlj3L1IjeWWNvf
ej43URV81S9EmSwpjaWboatr2A+1oJku5m7nPD9JIOckE1TzBsyhx7zIUN9w6MKr
7gFw8DCzypwUKyYgKYToVm8QlkT/L3B0fuQHWhT6ROGk4o8SC71ia5tc1TzUzGEZ

1AQO8bbnbmJLBDKveWHCoaeAkRzINzoD9wAn9z4pnilze59QtKC1cOqUksTvBSDh
6wARAQABiQEfBBgBAgAJBQJQDs6mAhsMAAoJENgpTsO7xNfVOHoH/i5VyggVdwpq
PX8YBmN5mXQziYZNQoiON8IhOsxpX4W2nXCj5m6MACV6nJDVV6wyUH8/VvDQC9nH
arCe1oaNsHXJz0HamYt5gHJ0G1bYuBcuJp/FEjLa48XFI7nXQjJHn8rlwZMjK/PW
j1lw2WZiekviuzTEDH8c3YStGJSa+gYe8Eyq3XJVAe2VQOhImoWgGDR3tWfgrya/
IdEFb/jmjHSG5XUfbI0vNwqlf832BqSQKPG/Zix4MmBJgvAz4R71PH8WBmbmNFjD
elxVyfz80+iMgEb9aL91MfeBNC2KB1pFmg91mQTsiq7ajwVLVJK8NplHAkdLmkBC
O8MgMjzGhlE=
=iw7d
-----END PGP PUBLIC KEY BLOCK-----

D.1.5. <doceng-secretary@FreeBSD.org>

```
pub   rsa2048/E1C03580AEB45E58 2019-10-31 [SC] [expires: 2022-10-30]
      Key fingerprint = F24D 7B32 B864 625E 5541  A0E4 E1C0 3580 AEB4 5E58
uid                           FreeBSD Doceng Team Secretary <doceng-
secretary@freebsd.org>
sub   rsa2048/9EA8D713509472FC 2019-10-31 [E] [expires: 2022-10-30]
```

-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBF27FFcBCADeoSsIgyQUY8vREwkTikwFFlNg31MVy5s/Nq1cNK1PRfRMnprS
yfB62KqbYuz16bmQKaA9zHN4FGfiTvR6tl66LVHm1s/5HPiLv8sP14GsruLro9zN
v72dO7a9i68bMw+jarPOnu9dGiDFEI0dACOkdCGEYKEUapQeNpmWRrQ46BeXyFwF
JcNx76bJJUkwk6fWC0W63D762e6lCEX6ndoaPjjLBnFvtx13heNGUc8RukBwe2mA
U5pSGHj47J05bdWiRSwZaXa8PcW+20zTWaP755w7zWe4h60GANY7OsT9nuOqsioJ
QonxTrJuZweKRV8fNQ1EfDws3HZr7/7iXvO3ABEBAAG0PEZyZWVCU0QgRG9jZW5n
IFRlYW0gU2VjcmV0YXJ5IDxkb2Nlbmctc2VjcmV0YXJ5QGZyZWVic2Qub3JnPokB
VAQTAQoAPhYhBPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsDBQkFo5qABQsJ
CAcDBRUKCQgLBRYDAgEAAh4BAheAAAoJEOHANYCutF5YB2IIALw+EPYmOz9qlqIn
oTFmk/5MrcdzC5iLEfxubbF6TopDWsWPiOh5mAuvfEmROSGf6ctvdYe9UtQV3VNY

```
KeeyskeFrIBOFo2KG/dFqKPAWef6IfhbW3HWDWo5uOBg01jHzQ/pB1n6SMKiXfsM
idL9wN+UQKxF3Y7S/bVrZTV0isRUolO9+8kQeSYT/NMojVM0H2fWrTP/TaNEW4fY
JBDAl5hsktzdl8sdbNqdC0GiX3xb4GvgVzGGQELagsxjfuXk6PfOyn6Wx2d+yRcI
FrKojmhihBp5VGFQkntBIXQkaW0xhW+WBGxwXdaAl0drQlZ3W+edgdOl705x73kf
Uw3Fh2a5AQ0EXbsUVwEIANEPAsltM4vFj2pi5xEuHEcZIrIX/ZJhoaBtZkqvkB+H
4pu3/eQHK5hg0Dw12ugffPMz8mi57iGNI9TXd8ZYMJxAdvEZSDHCKZTX9G+FcxWa
/AzKNiG25uSISzz7rMB/lV1gofCdGtpHFRFTiNxFcoacugTdlYDiscgJZMJSg/hC
GXBdEKXR5WRAgAGandcL8llCToOt1lZEOkd5vJM861w6evgDhAZ2HGhRuG8/NDxG
r4UtlnYGUCFof/Q4oPNbDJzmZXF+8OQyTNcEpVD3leEOWG1Uv5XWS2XKVHcHZZ++
ISo/B5Q6Oi3SJFCVV9f+g09YF+PgfP/mVMBgif2fT20AEQEAAYkBPAQYAQoAJhYh
BPJNezK4ZGJeVUGg5OHANYCutF5YBQJduxRXAhsMBQkFo5qAAAoJEOHANYCutF5Y
kecIAMTh2VHQqjXHTszQMsy3NjiTVVITI3z+pzY0u2EYmLytXQ2pZMzLHMcklmub
5po0X4EvL6bZiJcLMI2mSrOs0Gp8P3hyMI40IkqoLMp7VA2LFlPgIJ7K5W4oVwf8
khY6lw7qg2l69APm/MM3xAyiL4p6MU8tpvWg5AncZ6lxyy27rxVflzEtCrKQuG/a
oVaOlMjH3uxvOK6IIxlhvWD0nKs/e2h2HIAZ+ILE6ytS5ZEg2GXuigoQZdEnv71L
xyvE9JANwGZLkDxnS5pgN2ikfkQYlFpJEkrNTQleCOHIIIp8vgJngEaP51xOIbQM
CiG/y3cmKQ/ZfH7BBvlZVtZKQsI=
=MQKT
-----END PGP PUBLIC KEY BLOCK-----
```