Linköping University.
2004 – 05
Supervisor: Tina Lindkvist

Authors: Francois Arpin **fraar334@student.liu.se**
Peter Örnvall **petor960@student.liu.se**

How secure and user friendly is PGP

Information security report

How secure and user friendly is PGP

Francois Arpin, Peter Örnvall
fraar334@student.liu.se, petor960@student.liu.se

Abstract
In this document, we present the results of a study of the security and usability of PGP, which is a cryptographic tool to send emails on the Internet securely. We start by giving a brief introduction to the theoretical aspect of PGP schemes. Then, we investigate the user interface and the usability issues of the program. This investigation is based on our own experience as well as on the results of the article [4]. Finally we discuss the security problems that could affect PGP, looking at the possible types of attacks against PGP, practical as well as cryptanalytic.

Introduction

Utility of PGP

The Internet grew too fast to be secure. Especially it is difficult to talk about privacy when we exchange mails because it is quite easy to eavesdrop Internet communications of someone when he/she does not communicate on a protected local network. Even internally, certain malicious users may be interested by the content of the mails of another. It is also very easy for that user to study the traffic generated by the people on the network. Moreover, with the growth of wireless structure, it is even easier for anybody who is geographically close enough to simply listen to all the network traffic with as simple tools as a snack box and eavesdropping software.

Another problem may be rising soon. Certain countries start to change their law so mail exchange will not be longer considered as private there. Soon it may be possible being without any juridical protection against mail watching.

The problem of being sure of the identity of the sender of a mail is also crucial for justice or business matters. The email accounts are also very easy to hijack. So can we be sure that this person really sent the mail I am reading? Certainly no. So we have to find a tool to address those problems. PGP appears to be one solution.

PGP is an encryption tool that provides the possibility for anyone to use strong encryption and digital signatures. With digital signatures we can verify the identity of our correspondents. This implies that we can trust the message not to be forge. PGP offers a choice of encryption methods for the advanced user. The software uses asymmetric encryption with a private and a public key. The keys can be sent directly to the user one want to communicate with or be uploaded to a dedicated key server where the users can search for persons they want to communicate with privately.

History of PGP

One person with insight into cryptology followed what was happening in the cryptology community. From the middle of the 80's to the year of 1991 he had developed PGP (Pretty Good Privacy), an encryption program which was dependent on RSA, a strong encryption algorithm. The plan of releasing PGP as a shareware was not possible due to a patent on the RSA technology. PGP's inventor Phil Zimmerman claimed that he had been offered a free license from the owners of the RSA patent. But they denied that the case was so. In the same time it suddenly seemed that all encryption scheme, licensed or not, should be criminalized by the American government. Phil had put in years of work with PGP and had great debt to pay and felt desperate. He decided that PGP had to be spread to as many people and as fast as possible. He rewrote the conditions of the program to General Public License (GPL). Thereafter he gave the first version of source code to a friend who put the code out on Usenet. This lead to three year long investigation of Phil Zimmerman for crimes against the weapon export laws of the USA. The American government dropped the charges against Phil and by this time PGP had grown to be a de facto standard in the area of encryption tools for computers. After the charges had been dropped Phil founded the company PGP Inc (www.pgp.com). For more information, see [1].

How PGP works?

We will roughly describe in this part the way PGP works. As PGP provides encryption scheme, it uses encryption techniques. Those techniques can be classified into two categories: symmetric and asymmetric. We describe the principles that characterize those kinds of schemes, explain how they can be used and in which conditions. That will allow us to explain why and how they are used in PGP. We have the same procedure for the digital signing feature of PGP. This uses hash functions. So we explain the general principle of digital signing and we then explain how it is used in PGP. For those parts we provide examples as well. For more information about the basics of cryptology theory, refer to [2]. Refer to [3] for the principles of PGP scheme.

Symmetric encryption

Encryption is the transformation of data to a form in which it is impossible to read without the appropriate knowledge of the key. In symmetric encryption key only one key is used for both encryption and decryption.

The goal of using encryption is to keep a conversation private. If you talk in terms of computer security, the goal is to achieve confidentiality and integrity. Confidentiality would mean to keep the conversation private to others who may be listening and integrity would be the assurance that the unauthorized users do not alter the message.

Figure 1

This method requires the key to be sent over a secure medium/channel to be effective. It would be a good idea not to send the key over the Internet. The benefit with symmetric encryption is that it is much faster than the asymmetric encryption. The drawback is that it is required to send the key in a secure way (over a secure channel). [Figure 1] gives an illustration of symmetric encryption.

Some examples of cryptographic techniques would be IDEA, 3DES, CAST.

IDEA International Data Encryption Algorithm

IDEA Operates on 64 bit plaintext blocks and uses a 128-bit key. A 64 bit input block is divided into 4 blocks which becomes the input blocks of the first round of the algorithm. The total number of rounds is 8. In each round, the four sub-blocks are XOR:ed, added, and multiplied with one another and with 16-bit sub-blocks of key material. Between each round the second and the third sub blocks are swapped. Its speed is comparable with DES.

DES Data Encryption Standard

DES became outdated since the computer power is regularly increasing. DES was defined in 1976. It was built upon an algorithm from IBM, which was called Lucifer. DES is a block cipher, which means that the it operates on a fixed-length block of plaintext and converts it into a block of cipher text of the same size by using the secret key. In DES, the block size for plaintext is 64 bits. The length of the key is also 64 bits but 8 bits are used as parity bits. The key length used is therefore only 56 bits. Decryption is done by applying the reverse transformation to the block of cipher text using the same key.

The encryption process of DES is as follows:
The block of 64 bits plaintext is split in 2 halves (L_0, R_0), each of the halves is 32 bits long. DES also uses the 56-bit key to generate 16 keys of 48 bits each (K_i). These sub keys are used in 16 rounds. In each round the function F is applied to one half using a sub key (K_i). Then the result is XOR:ed with the other half. The two halves are then swapped and the process is repeated. All the rounds follow the same pattern except the last one, where there is no swap. The final result is the cipher text (L_1, R_1). So the plaintext (L_0, R_0) us transformed to the encrypted (L_1, R_1).

Decryption process of DES:
Decryption uses the same principles as encryption. The input is instead (L_1, R_1) which will generate the plaintext (L_0, R_0). The input keys are applied in reverse order. Other than that it is the same process as encryption.

3DES (or triple DES) is a further development of DES. In 3DES 3 stages of DES is used with a separate key for each stage. So the key length in 3DES is 168 bits.

Asymmetric Encryption

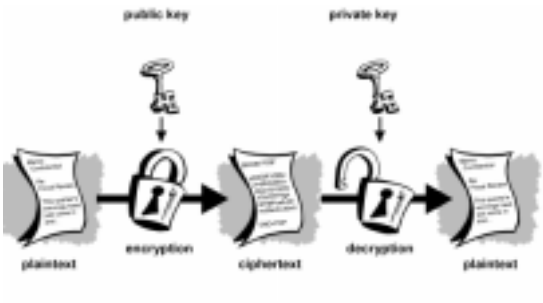

Figure 2

Public-key encryption [Figure 2] (also called asymmetric encryption) involves a pair of keys -one public key and one private key- associated with an entity that needs authentication and encryption. Each public key is published, and the corresponding private key is kept secret. Data encrypted with your public key can be decrypted only with your private key. You can freely distribute a public key, and only you will be able to read data encrypted using this key. In general, to send encrypted data to someone, you encrypt the data with that person's public key, and the person receiving the encrypted data decrypts it with the corresponding private key. It is computationally infeasible to deduce the private key from the corresponding public one.

RSA

RSA works in a fairly simple way. First you have to generate your private key and your public one. This takes five steps:

1. Generate two large prime numbers, `p and q`
2. Let `n = p*q`
3. Let `m = (p-1)(q-1)`
4. Choose a small number `e`, co prime to `m`
5. Find `d`, such that `de mod m = 1`

Then you just have to publish e and n as the public key and keep `d` and `n` as the secret key.

The encryption formula is `C = P^e mod n` where `C` is the cipher text and `P` is the plaintext. To decrypt, you perform: `P = C^d mod n`

Diffie-Hellman

Alice and Bob select two numbers, `q` and `n`. Then Alice selects the secret number `xa`. Bob selects the secret number `xb`. From the two public numbers, `q` and `n`, and her secret number `xa`, Alice calculates `ya` and sends the number to Bob. Using the same two public numbers, `q` and `n`, and his secret number `xb`, Bob calculates `yb` and sends the number to Alice. Alice and Bob have completed step one of the Diffie-Hellman process.

Alice calculates ya using the formula:

```
ya = (n ^ xa) mod q
```

This says, multiply `n` by itself `xa` times, then divide the product by `q` and save only the remainder.

Alice sends the number `ya` to Bob. In the meantime, Bob applies the same formula to the numbers `n`, `xb` and `q` to calculate the number `yb`:

```
yb = (n ^ xb) mod q
```

Bob sends the number `yb` to Alice. They are ready for step two.
Using the number `yb` received from Bob, Alice calculates

```
ka = (yb ^ xa) mod q
```

Again, this means multiply `yb` by itself `xa` times, and then save the remainder after dividing the product by `q`. When Bob receives Alice's number `ya` he calculates:

```
kb = (ya ^ xb) mod q
```

Alice and Bob have completed the Diffie-Hellman encryption process. Alice applied her secret number `xa` to Bob's value `yb` and calculated `ka`. Bob applied his secret number `xb` to Alice's value `ya` and calculated `kb`. It turns out that `ka = kb`, a number now known to Alice and Bob, but to no one else. Even though Eve, the eavesdropper, may have been monitoring their communications studiously, Eve cannot discover the number ka easily.

Benefits and drawbacks

The main benefit of using an asymmetric scheme is that you do not need a secure channel to exchange your private keys with other people. Moreover, the number of keys the system needs to make people communicate with each other is considerably reduced. Instead of having a private key for every possible couple-communicating people, you just need a couple of keys for every participant. It is also a great help for newcomers because

everybody does not need to get a private key to talk with them; they just have to get the public key from a server. This will be freely available.

PKI's privacy and authentication measures work well for any two-way communication. Authentication also works well for one-to-many communication, such as signing a document or an email that many people will read. However, privacy is another matter. Remember that privacy works by having the sender encrypting the information with the recipient's public key. What if there were multiple recipients on an email message that should be kept private? There is no simple answer for this.

This drawback of the public key infrastructure will have consequences in PGP use. But the drawback which has the most influence on PGP design is that public key encryption produces a lot of overhead (at least the output size of this kind of scheme is the twice bigger than the size of the input) and the processing time is unacceptable when data size shows up to be too big.

Encryption and Decryption in PGP

Encryption

PGP uses both conventional (symmetric) and public key encryption. When a plaintext is encrypted PGP will first compress the plaintext (unless it is too few bytes). Data compression saves transmission time, disk space and it also strengthens the cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext and thereby enhances the resistance against cryptographic attacks.

PGP then creates a session key, which is secret and only used one time. This key is a random number generated from the keystrokes and random mouse movement collected from the user. The session key uses a conventional encryption algorithm to transform the plaintext into cipher text. When the data has been encrypted the session key is encrypted with the recipients public key. The encrypted session key and the cipher text are then sent over to the recipient [Figure 3].

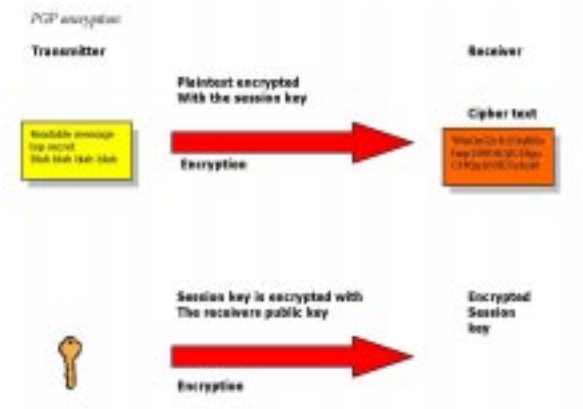

Figure 3

Decryption

Decryption works in the reverse to encryption [Figure 4]. The receiver uses his private PGP generated key to recover the temporary session key. This key is then used to decrypt the conventionally encrypted cipher text.

Figure 4

The combination of conventional and public key encryption combines the convenience of public key encryption with the speed of conventional encryption. Public key encryption provides a solution for key distribution. Conventional encryption is about 1000 times faster than public key encryption. Used together the performance will increase and key distribution issue is solved without a decrease of the security

Digital signing

Digital signatures enable the recipient of information to verify the authenticity of the information's origin, and also to check the integrity of the information. From a

technical point of view, the digital signing of a message is performed in two steps [Figure 5]:

In the first step of the process, a hash-value of the message (often called the message digest) is calculated by applying some cryptographic hashing algorithm (for example, MD2, MD4, MD5, SHA1, or other). The calculated hash-value of a message is a sequence of bits, usually with a fixed length, extracted in some manner from the message. All reliable algorithms for message digest calculation apply such mathematical transformations that when just a single bit from the input message is changed, a completely different digest is obtained.

In the second step of digitally signing a message, the information obtained in the first step is encrypted with the private key of the person who signs the message and thus an encrypted hash-value, also called digital signature, is obtained. For this purpose, some mathematical cryptographic encrypting algorithm for calculating digital signatures from given message digest is used. This can be summarized with the following schema:

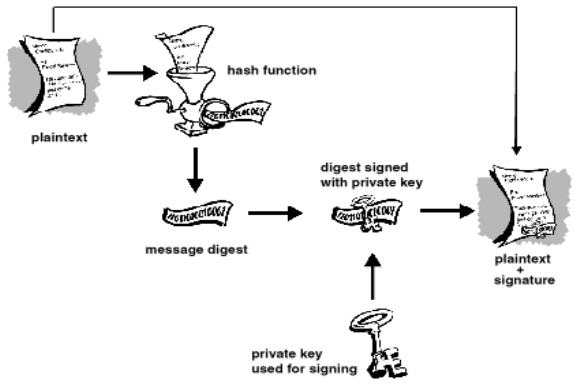

Figure 5

How user friendly is PGP?

PGP is a tool for using cryptography. Since it is an advanced security program this will have influence on the usability issues. In this section we will first give a definition of usability of security software. Then we provide the results of our own experience of the use of PGP as well as the result of a test with inexperienced computer users from the article [4]. We divide this into two parts before and during the use of PGP.

Usability in security: definition

Usability issues are specific when a software involving security has to be designed. [4] establishes a definition for usability in security software:

"Security software is usable if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don t make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it."

In order to start using PGP you probably have to know a lot on how cryptology is working. An essential part for the user to know about would be public key cryptography. It is shown [ref. Johnny] that inexperienced users with both computers and cryptology tried to encrypt messages with their own private key. Also PGP asks the user during the installation to generate keys. Maybe it is not clear to an inexperienced user that you have to generate keys in order to start using PGP. Knowledge about different types of keys would be good to have, since PGP have the option of letting the user choose. However using PGP would not be impossible for inexperienced users but their task could be made easier by a more usable interface.

Before the actual use

Getting the correct version.

Before you start using PGP you have to download a copy of the freeware version (or a commercial one if you need the extra features). However there are here a couple of things to be aware of. First you have to find a copy on a website that is located on an official domain like pgp.com for instance. Since the name PGP is protected, it will be safe to download a copy from a source like this. If you download from some other link belonging to some private person or other Internet sources there is a chance that you could suffer from a Trojan horse as mentioned in the practical attacks section of this document.

We first tried to use the freeware version of PGP 8.0 (the latest version of PGP) but when we tried to use it as a plug in of a web client, we were asked of paying for basic functions like encrypting or signing. After installing an earlier version (6.5.8) we could actually start using it. We got it from the website www.pgpi.com.

Key settings

Before actually generating the keys, you get some information about the public key structure. For people who got knowledge about this before, we think it is enough, but [4] shows that for average users, it can take more than that to be aware enough for using and managing keys correctly. Here we see that the key generation is not a problem by itself because the default settings are well adapted to average users but the problem is that they cannot understand from this phase what they are actually doing and why they are doing it.

In the version that we used, the configuration for key generation was done in four steps:

1. the choice of the key types
2. the choice of the key size
3. the choice of the key expiration
4. the choice of a pass phrase

In step 1, you have the choice between using DSS/DH (alias DSA/Elgamal) and RSA. Most of the users have DSS/DH. There is a clear explanation during this step justifying why DSS/DH is recommended as default. Another positive point for usability is the possibility to know about this specific problem by pushing the help button. By doing that you can get a more complete explanation. As a consequence, the amount of information does not frustrate you and, at the same time, you feel confident in your choice. It is also likely that you actually make the most suitable choice.

In step 2, you can select the size of the key pair that you will use. By default, it is set to a reasonable size to both keeping good security for most users (who are not supposed to be the target of heavy cryptanalytic structures) and staying small enough not to slowing encryption too much. The message from Philip Zimmerman might not be very useful as most of the people do not know whom he is but the content is clear enough to choose the key size without problem.

We have the same comments for step 3. We may add that the default setting of an infinite expiring time can be dangerous for beginners who are likely to perform tests on the first keys they generate. As a matter of fact, for reasons that we will see below, they might not be able anymore to revoke the public key they are going to generate. So these testing keys lasting forever might be a source of "pollution" for the key server. It will be even worst if people send information with no longer used public keys. It is very likely that those messages will remain secret for good.

Step 4 is more likely to be the cause for compromised security because it is a classic choice of password. Most of the users will choose bad quality pass phrases. However, a good point for PGP here is that it indicates graphically the quality of a pass phrase but it does not give any tip to the user for getting a good password. We will see that this choice for the password is crucial because it should be brute force resistant and easy to remember. For reasons that we see later, we think that the interface should warn the users to a greater extent on the importance of the choice of the password.

Creation of key backups

Due to the lack of warning about the consequences of accidentally revoking a public key or deleting a private key, the user may not understand why it is crucial to create key backups. It is also a bad point that it does not recommend to make a backup elsewhere than the used hard drive because a system crash is always possible. Here the interface just proposes to make a back up of the keys, without even recommending the creation of those backups.

During the use

As we saw in the definition of usability in security software, for a relevant use of PGP, users have to be aware of the tasks they have to perform for ensuring security. In the case of PGP, we think that users should know and realize the difference in using private and public keys. For example, to encrypt a message, users should use the public key of their recipient. Therefore, if they do not have already this key in their key ring, they should perform some actions to import it somehow. As we will see later, this is not always easy to do so. So, the users should be motivated for this task. Here, we study if the way this version of PGP is designed allows the users to understand what they are actually doing and why they are doing it.

The graphical description of the actions in PGP

[4] shows also that the icons used to represent the possible actions in the PGPTools are quite straightforward for novice users but they should be modified to give a better understanding to the user. [Figure 6] shows the principal icons that a user has to manipulate during the use of PGP.

Figure 6

First, the icon for encryption button should be more helpful to distinguish between public keys and private keys (used actually to decrypt). The problem with the metaphor of the lock is that it could mislead the user to think that the same key is used both for locking and unlocking. So the user may think that it is the same key for both encrypting and decrypting. Actually, when we push this encrypt button we do not know with which key we are doing it. Is it crucial? Would it be worthy to use more sophisticated metaphors to give this level of understanding to the user, risking confusing the user looking for a button to simply encrypt? The answer to those questions relies on the possibility of giving this understanding somewhere else in the software.

The same problem occurs with the signing button. It is even more problematic because the metaphor of the pencil does not link the signature with the use of any key. Users might not even realize that they use a key to sign (and as a consequence they are not aware of which key is used). The problem is also that this metaphor is one of the best for just giving the information about the nature of the action: signing.

Figure 7

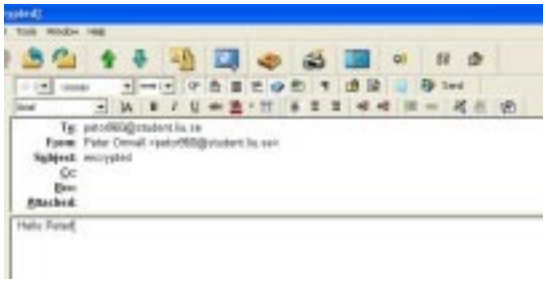

Figure 8

Figures 7 and 8 show the interface of plug in for both Eudora and Outlook Express. Concerning the decrypting/verifying button, things are getting worse because signature verification is not even represented graphically. This button with the open lock only refers to decryption. We encounter of course also the problem of knowing which key is used.

In the Windows Outlook mail client plug in [Figure 7], there is the same button without any icon. Because there might be already a lot of buttons used in the window for composing a new mail, it might occur the user cannot see all the buttons at first. In this case, you should push a button to scroll down the hidden buttons.

It would have been a good idea to group the PGP plug in buttons in Eudora [Figure 8] in a better way. Instead of spreading the PGP related icons out over the mail client as well as having different sizes of the buttons. Grouping the icons and using the same size would have been preferable. Another thing that would be better is to distinguish between verifying and decryption of messages with two buttons instead of one.

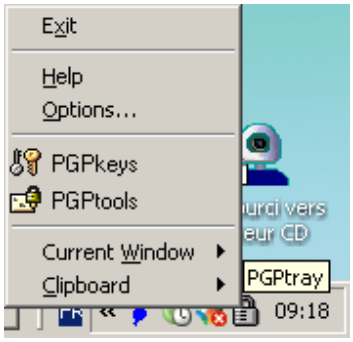

Figure 9

Finally the first contact you have with PGP when you are not using an email client is a quick launch shortcut represented with a lock [Figure 9]. Clicking on it, you have the choice between two main modules: PGPTools and PGPKeys. They respectively correspond to the access to the usual possible actions offered by PGP like signing or encrypting and to the management of the keys. This idea of separating the two tasks is good but the icons used are not sufficient to infer this separation. The user may be disturbed by this choice because he/she might not understand at first the management of the keys is actually as important as the normal actions.

Encryption/decryption phase

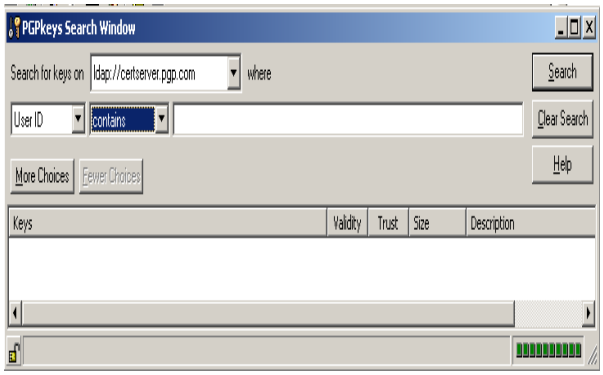

Figure 10

Importing the keys

To perform the action of encrypting you have to first get the key of the recipient. This is the point that caused us the most troubles. Even if you know the theory about the public key structure you might encounter difficulties in getting the right key to encrypt with. First if you are using the versions 2.x and less you are simply not able to encrypt a message to people using DSS/DH keys because this algorithm is not supported at all by those versions. Then, if you want to get the key of a user through a server you have to know on which server to search. There is no global public key searching tool that would consult all the servers. Once you finally ended up on the right server [Figure 10], other problems related to search by words in general might occur. We had the problem of having a too common name as user id. By just giving our first name, we could not easily retrieve our public keys on the server by searching by user id. Searching by email was not more successful for it was not giving any match. Several attempts proved it was not due to mistyping. The other search options like by key length and key value were not very useful since they appear like advanced user options.

Then we tried to exchange our public keys by mail. With the Outlook software, this was quite difficult as even if we could copy and paste the key block from the key file, you have to keep the overhead messages -----PGP BEGIN---- and -----PGP END------ to be able to generate an asc file from it. You have also to give some obscure suffix (asc), which is far from average computer user skills. The simple copy and paste in the key management window from the email window, as indicated in the help documentation, was simply not working and generated an error message. Using Eudora we encountered less problem since we just needed to click on an asc file to import it to the key ring.

[4] also stresses the fact that it is very difficult to understand the meaning of the difference in representation of RSA keys and DSS/DH keys. So if someone is using the two types of keys, one is likely not to know which key to use.

Once we finally got the key needed to encrypt to a recipient, we used our mail client to send each other an encrypted message. The problem was that we did not get much feedback about what we were actually doing. [4] shows also that a user might send a message plaintext, assuming that encryption was automatically done. So just remember the fact the buttons may be hidden, we understand that the accidental sending of plaintext message is possible with PGP for inexperienced users. What we did to be sure it worked as we thought was to send a message to ourselves and then looked at it on our web-mail site.

The encryption and signing buttons are not grouped together with the other PGP icons. They are also smaller. [Figure 11]

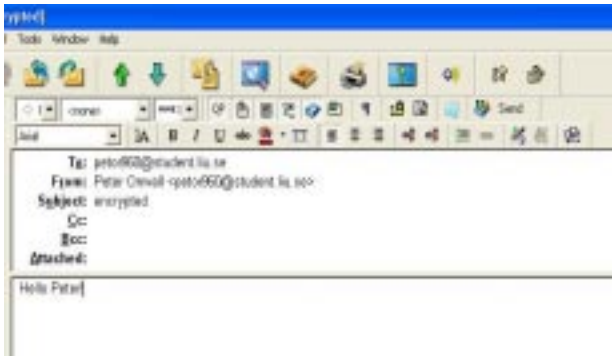

Figure 11

RSA Diffie-Hellman cohabitation is a source of problems

Originally, PGP used the RSA algorithm for encryption and signing. The versions after 5.0 use Diffie-Hellman. Most of the users nowadays use Diffie-Hellman. Those two algorithms use different types of keys. For compatibility reasons, the latest versions still support RSA, in the sense that one user is still able to use RSA keys to communicate with people using Diffie-Hellman. They just have to get his/her public RSA key. The RSA algorithm is also working in the last versions. But [4] shows this possible use of several types of keys leads to problems.

First there is an obvious compatibility problem since the users of the first versions supporting only RSA might not be able to check a message signed with Diffie-Hellman. The logarithm is not implemented on their software. So one Diffie-Hellman user might sign with his private DH key. The recipient can only check RSA signed message. It is even trickier when you are sending signed messages for several mixed recipients.

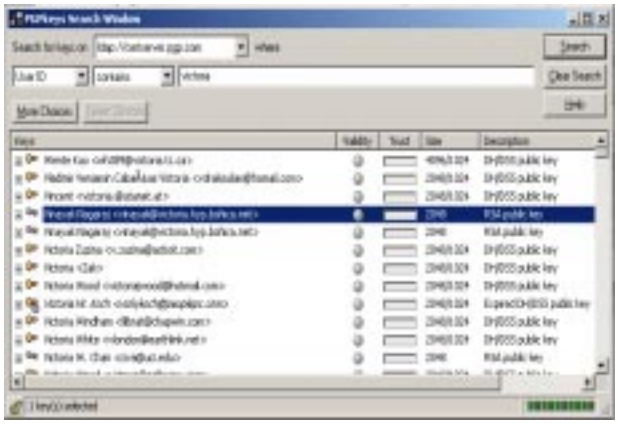

Figure 12

Secondly, the user should be conscious of the existence of the two types of key and should realize there is a compatibility problem. We saw before that it was already not easy to understand the difference of use between private and public keys. If there exists some extra distinction to make between types of public keys, we can be sure some users will be lost for good. When you have a look to the figure 10, you can see there is a difference of representation for RSA and Diffie-Hellman keys. Now imagine you are a novice user, who took the default key configuration without paying too much attention to the choice of the type of key. When you have a look to this difference of representation [Figure 12], you might take enough time to look into the documentation for forgetting what you were looking for.

Third, some users may use different types of keys for being able to communicate with old versions users. As we just saw, it is difficult to know what is the type of a key and if you do not know there is a difference between keys it is even more annoying because you have to make a choice and you are not motivated for it. So in the best case, you choose the key that the recipient is used to receive message encrypted with. But you might choose to encrypt with the RSA key although you are DH compatible. The recipient would not understand why you would have chosen RSA encrypting and try to decrypt first with his DH private key. This can disturb even experimented users.

Pass phrase related problems

To forget the pass phrase in PGP can be even more serious than in other secure systems because it encrypts messages with very strong algorithms. To forget the pass phrase can prevent you for good to access anymore a file that you would have encrypted. You can create a backup copy of the secret message/content but it is not the default settings.

Key Management

Too much information at the beginning of key management

As default in key management we automatically get a group of users from the PGP inc like Phil Zimmerman. We did not choose to import them in our key ring. In addition to this, we think that there is too much information in the key rings display by default. The users who not have particular needs or who are not experts don't need that. This is somewhat useless and may confuse a less experienced user. In the view menu this can be changed to only view the users, which we are communicating with. Therefore it is not the most serious problem.

Checking the validity of a key

A big issue in key management is to have information about the safety using the public key of someone we do not know. We have two ratings for helping us in decision-making. The first rating is "validity". It is a rating that users directly have an influence on. They set it according to their own experience of using the key they consider. They set it to "valid", for example, if they have tried the key and they have the proof it is a reliable one. This parameter says how sure the user is that using this key is safe. The second rating is "trust" and indicates how much faith the user has in the key from the information of how many users signed it and who signed it. The number of other users certifying that the key is valid influences this. But it is the user who actually sets this rating. That is why it is rather confusing.

[4] shows that this system of "web of trust" may appear quite vague too inexperienced users. The words "validity" and "trust" can be interpreted in rather similar ways so it might be confusing to make the difference between the concepts behind and to know where the trust we can have for a key comes from. It can be from our own experience (validity) or from the other users 's experience. We do not know at first why we should trust a key. There is no

information about those ratings except if we look deep in the documentation. We never get any message explaining the “web of trust” ’s scheme.

Deleting the private key

You have the option of deleting your own private key. This option is rather dangerous because if you delete your private key, you will not be able anymore to decrypt the messages encrypted with your public keys. A message asking the users if he/she is really sure of deleting the key appears but the user is not informed about the consequences of what he/she may do.

Accidental publicizing of the key

In PGP information is easier to add to a key server, than to remove. A user who is trying the program and is experimenting may end up generating a number of key pairs that are added to the key server without knowing that these keys cannot be removed if he/she forgets the corresponding pass phrase. As a matter of fact the keys can be revoked later giving the pass phrase. But without pass phrase it is impossible. The user can wait for them to expire but the key will remain on the server anyway. It is an idea to warn the user before uploading a key to a server. It is also better to have a default life length of a year during the installation of PGP.

The consequence of this kind of error is that the user has to notify all his correspondents. However there is one solution to this problem. PGP suggests that the user make a backup revocation certificate, so that if the pass phrase is lost, the user is able to use that certificate to revoke the public key. This may be a useful thing, but probably only to the expert users of PGP.

Accidental revoking of the key

If a user decide to revoke a key he should realize that the only way to undo the revocation is to restore the key is from a backup copy. Once the key is revoked, other will be unable to encrypt messages to this key. Here a message that explains to the user what is about to happen and that other will be unable to send encrypted messages to the user would be preferable.

Influence of the user-friendliness on security

In the previous part about the user-friendliness of PGP, we saw that this software was not easy to use and that mistakes could easily be done. The problem is that those difficulties the user can encounter can affect directly the security of his/her data.

Sending encrypted sensitive data

First, the user can seriously misunderstand the way the software works and the action he/she has to perform to encrypt data. For example, like we saw above, because of hidden button the user can simply forget to push encrypt button and send plaintext.

The web trust model

The level of difficulty to understand the web of trust model is a factor of insecurity in the sense that a user who would be confused by it would not use it as a tool. As a consequence, those users who are generally beginners who would be more vulnerable to corrupted public key attacks. On the other hand they are more likely not to be the target of such planned attacks.

Forgetting pass phrase

Forgetting the pass phrase could be dangerous for the access to archived files. As a matter of fact, to decrypt a file, you need both your private key and your pass phrase. We can imagine that after years, it would be easy to forget the pass phrase corresponding to the public key a file was encrypted with. Then the access to the file, because of the robustness of the encrypting algorithms, is seriously compromised.

Bad choice for pass phrase

Like for all other security schemes, it is proven that most of the users will choose an easy to guess pass phrase so they can remember it more easily. We saw that the system helps a little bit the user to evaluate the quality of a password. But it is likely that the user is still not motivated enough to be careful enough for this choice. The software does not stress enough on this aspect during the key generation

Practical attacks

In this part, we study the practical attacks that are possible to make against PGP. These are quite general in the sense they are not specific to only PGP. They mostly attack the environment of the user. For more information about those practical attacks, you can refer to [5], [6] and [7].

Trojan horse attacks

A trojan horse is a version of PGP, which has features unknown and unwanted by the user. The code of the software can be deliberately and maliciously modified to weaken the security for the user.

The Trojan horse attacks can hold in different ways:

- The Trojan horse could be hidden in the code of the binary patches (like in the installation files or in some upgrade sources). One can distribute a weakened version over the Internet with the same name as the legitimate version.
- It is possible to physically replace the legitimate copy with a weakened copy when the user is not present.
- The "man in the middle attack" is possible: a malicious user can operate between an official website and a user who wants to download PGP. The malicious user would substitute a weakened version to the normal one.

The possible malicious modifications would be:

- The random number generation routine is modified so that it will produce predictable results.
- The Session key routine could be modified so the same key is always used. IDEA, RSA or MD5 routines could be weakened.
- Wipe routine could be modified to leave data remnants.
- Messages could always be encrypted with an additional covert key.

The countermeasures are to download your copy from trusted reputable source, compile your own source code and validate all signatures.

Operating system attacks

Deleted files

If a file is deleted it is only the file allocation information that is changed, the contents of the file is still there on the hard drive of the computer and the contents still reside on the disk until it is overwritten by another file. There are many software packages available to retrieve files or parts of them when they are deleted. It is not only these files that are vulnerability. Some programs use temporary copies of a document as you work on it. The file is deleted when the session is finished. Users may not be aware that temporary files exist but if someone has access to the hard drive the contents of that file could be viewed.

Wipe utilities is a feature of PGP that overwrites data several times before the file is deleted. This prevents the snoopers from using a file recovery tool to retrieve sensitive deleted file or scan a hard disk for information.

Swap files

In Windows for example where swap files are used to make an effective use of the memory, data and code are written from memory to a cache file and then swapped in and out as necessary. All it takes is a copy of the swap file and a file editor then scanning through the bytes

A solution would be to wipe the swap file before turning the computer off, install lots of memory and turn the swapping off.

Slack space

DOS/Windows OS allocates disk space in clusters. Slack space is the space between the end of the physical file and the end of the allocated cluster. While a wipe utility overwrites the physical file, bytes in the slack space typically remain.

When bits and plaintext is left on the hard drive this is of course a risk. What the eavesdropper does is simply to search the hard drive for a text string related to some secret/private information. Defragmenting the hard drive will not remove data stored in the slack space. The only option is to use a utility designed to wipe the unallocated bytes at the end of the clusters.

The only countermeasure for this is to use a slack space eraser tool on a regular basis.

Keyboard monitoring

Routines of an operating system can be patched to capture keystrokes. The recorded keystrokes are then transmitted over the network or stored to disk. There are numerous keyboard-monitoring tools available on the Internet that could subvert a pass phrase.

The solution here would be to use tools designed to protect against keyboard monitoring.

Network attacks

As the Internet continues to grow the threats to users connected to the WWW increases. The days when PC's only where connected to a small network are in most cases over. Remote attacks launched over the Internet are now a realistic threat. This type of attack is very much likely to increase in the future.

Virus attacks

If a virus infects a computer, and the virus sends information over the network then the damage is done. The virus could have a keyboard monitor feature or it could open connections to the infected computer that could allow other computers to connect to the infected computer.

This problem can be addressed by using a virus checking utility and keep it up to date.

Browser and hostile applet attacks

As the browsers is one of the most common tools used for today's computer users, the risk of someone exploiting security flaws increases. The flaws could allow remote, unauthorized access of a users hard drive. These flaws could be used against PGP to.

1) Steal a key ring
2) Modify a key ring
3) Install a Trojan horse
4) Access plaintext files.

While the vendor's releases patches that address the flaws, not all users may be aware of the security problems with older versions of the browsers.

As the companies are competing and rushing to release their new versions of browsers, many more bugs are seen than in a normal development life cycle. The more bugs that the software contains the greater the chance is that these bugs will lead to a security hole.

The architecture of the mobile code programming languages provides a possibility of remote attacks through web page access.

Even though the java security model offers a theoretical model named the sandbox to keep away the applets from access to the users file system security bugs have occurred. ActiveX is relying on a signature based trust model where the users common sense is the countermeasure against malicious code. Also one trusted program might be designed to let all the other ActiveX programs have access to the users file system and thereby get access to some sensitive files. ActiveX there provides no real security apart from the common sense of the users.

Here we advice several solutions for defeating malicious mobile code:

- Download the latest browser patches for security.
- Do not trust web pages that contain mobile code.
- Use two computers: One for secure information and is isolated from the network and the other the browse the WWW containing unsecured information.

Tempest attack

The word tempest includes the study of electromagnetic emanations and the countermeasures required for preventing leakage of information. For example, CRT monitors give off electromagnetic emanations. Under the right conditions with the right equipment, it would be possible to recreate the contents of the screen from a remote location. With this type of attacks you would be able to see the pass phrase when it is visible on the screen. As a matter of fact, because it may be difficult to type correctly a long pass phrase, PGP offers the possibility to the user to enter the pass phrase in an unmasked way. So the pass phrase can actually be displayed on the screen. That is why this type of attack could be dangerous for PGP users. However, for the majority of the users, tempest attacks are not a realistic threat. It is also very difficult to recreate a captured computer screen. Only if the target computer has high value information and the eavesdroppers have access to very high amount of resources there may be a concern about tempest attacks. But if the user wants to protect against that type of attack, he/she should use common radio frequency shielding techniques and components to reduce electromagnetic leakage from the PC, peripherals and cablings and purchase tempest-shielded hardware.

Non technical attacks

Technology is not the only solution to security problems, consider social engineering and the attacks that are using people, which are usually more effective and cost efficient. So users should not underestimate the human element. Think of threats and bribery, breaking and entering theft, and misplaced thrust. All these are very realistic risk and could compromise the security of a computer system including PGP.

Counter measures consist in not writing the pass phrase down neither entering the pass phrase in front of anyone and keeping key rings on a floppy disk in a secure location.

Cryptanalytic attacks

We will briefly study here the robustness of the algorithms used in PGP to see if any cryptanalytic attack is possible.

Brute force on symmetric ciphers

The size of IDEA keys is 128 bits. One has to search for half of the key space: 127bits. this is big enough to prevent anyone from even hoping to success.

Brute force on asymmetric ciphers

Here we present the different forms of attacks on asymmetric ciphers.

Brute force factoring against RSA

Let's recall the math's of RSA, we have:

1. p and q are two large prime numbers
2. n = pq
3. Let m = (p-1)(q-1)
4. e, co prime to m
5. d, such that d*e = 1 mod m

An attacker has access to the public key. So he knows what is e and n. He wants d. The difficulty is to factor n in p and q. Trying to deduce (p-1)(q-1) from e*d = 1 mod m is even more difficult. For factoring n, different algorithms exist.
The Trial Division consists in trying all the primes numbers less than sqrt(n). This is the least efficient solution.
"Quadratic Sieve (QS)", "Multiple Polynomial Quadratic Sieve (MPQS)", "Double Large Prime Variation of the MPQS", "Number Field Sieve (NFS)" are faster algorithms to solve the problem. NFS is the fastest one.

We can present the times taken for breaking some keys:

Key Size	**MIPS-years required to factor**
512	30,000
768	200,000,000
1024	300,000,000,000
2048	300,000,000,000,000, 000,000

A MIPS-year is a 1,000,00 instructions per second computer running for one year. So we see that RSA still can be considered as secure enough for most users even if technologies are more powerful years after years.

Discrete log attack against Diffie-Hellman

The complexity of this type attack is comparable to the brute force factoring of RSA

Chosen cipher text attack on RSA

We can imagine the following scenario: an attacker Eve has access to a covert channel between Alice and Bob. Eve can listen to all the encrypted messages that Bob sends to Alice. When Bob sends the cipher text c, we have, if p is the plaintext: c=p^e mod n.
So Eve, who knows n, decides to choose a number r<n. Then she computes x=r^e mod n. She multiplies x modulo n with the cipher c she got from the covert channel, and she gets y. Finally she computes t=r^(-1) mod n, the multiplicative inverse of r mod n, which will be useful after a while.
She just has to ask to Bob to sign the message. By doing this, he performs the decryption for Eve. As a matter of fact, he does: y^d = (r^e)^d *(c)^d mod n= r*p mod n.
Then Eve just has to perform: t * (r*p) mod n =p mod n and she has access to p. A more precise description of chosen- cipher text attacks is done in [8]

Timing attacks

[7] shows that cryptographic operations take discrete amounts of time. An attacker can exploit timing differences in RSA and Diffie-Hellman operations.

Attacks on MD5

Brute force against MD5

We can draw the same conclusions as for the brute force against IDEA. The key size is the same: 128 bits.

Birthday attacks on MD5

It is proven there is an acceptable probability to find two messages which give the same output through any given hash function. This can be exploited by the attacker to break the signing scheme. Given n inputs and k possible outputs, (MD5 being the function to take n -> k) there are n(n-1)/2 pairs of inputs. For each pair, there is a probability of 1/k of both inputs producing the same output. So, if you take k/2 pairs, the probability will be 50% that a matching pair will be found. If n > sqrt(k), there is a good chance of finding a collision. In MD5's case, 2^64 messages need to be tried. This is not a feasible attack given today's technology. If you could try

1,000,000 messages per second, it would take 584,942 years to find a collision. (A machine that could try 1,000,000,000 messages per second would take 585 years, on average.)

Conclusions

PGP does not fulfil the usability requirement for security software. As we experienced by ourselves and also by reading Whitten's investigations, we realized that PGP is not user-friendly software.

First, key management is really the key problem in PGP. Just to get the public key of a correspondent into our key ring is a very difficult task, even for expert users. The web of trust model is also hard to understand. A theoretical background is required to manage this software in a proper way. So if you are a normal user, you have to get this needed knowledge from the software. This is another big issue in PGP. The interface in mail plugins could be revised in order to give the user a better understanding of what is going on. But this problem remains of teaching users about encryption/decryption concepts remains open.

Even if PGP is not that user-friendly, it is the most relevant attempt to provide cryptography to a broader audience. We believe that it would be possible for novice computer users to grasp the main functionality and principles if they read some documents about cryptography concepts like public keys. They can try to collaborate in learning the software to start sending signed/encrypted messages to each other.

PGP encounters security problems that all software are likely to have. PGP is just a strong link in the security chain. Most of the security issues it encounters are related to the use environment Cryptanalytic attacks are quite hard to perform compared to practical ones. The users mistakes are likely to be the most dangerous source of problems for security.

Appendix 1

Interview with advanced computer users who works with cryptology

Q: What kind of user are you?
A: Advanced computer users, good knowledge in cryptology.

Q: Which OS are you using?
A: Unix Solaris

Q: Why did you start to use PGP?
A: To show an example to make people start using PGP
Continuing for sending certain mail, and signed emails.

Q: Which version are you using?
A: GnuPG 1.2.3, Enigmail (GUI), and Mozilla mail client.

Q: Installation
A: Pre installed by admin

Q: Did the Configuration go well?
A: Some problems with signing keys, an issue about a user using
PGP and we used GPG.

Q: How long time did it take to get GPG working, and that you could do what you want with the software?
A: Maybe one week, but effective time a couple of hours.

Comments
New buttons in the graphical interface lets the user easy encrypt/decrypt mails.

Q: Which features of PGP are you using?
A: Sign email primary use, encryption some times with sensitive files.

Comments from the interview:
The users do not use any key server. They send their public key by mail to each other. GPG checks automatically the integrity of the mails. Sometime they don't fully understand how PGP works even if they are experts in cryptology.

Q: Do you change your keys often?
A: No, except when we forget the password. Yes once a year.

Q: What kind of encryption do you use?
A: Default. Diffie-Hellman, key length 1024.

Comments
They know only one user using RSA
They don't check public keys of other users.

References

[1] B. Leuf
Sekretess, kryptering och PGP
Datormagazin
2001 – 08

[2] W. Trappe, L. Washington
Introduction to Cryptography with Coding Theory
2001 – 09
Prentice-Hall; ISBN: 0130618144

[3] Network Associates, Inc. and its Affiliated *Companies How PGP works*,
Chapter 1 of the document Introduction to Cryptography in the PGP 6.5.1 documentation
URL: http://www.pgpi.org/doc/pgpintro/
1999

[4] A. Whitten, J. D. Tygar,
Why Johnny can't Encrypt: A Usability Evaluation of PGP 5.0
Proceedings of the 9th USENIX Security Symposium, 1999

[5] Joel McNamara,
Practical Attacks on PGP
URL: http://www.privacy.com.au/pgpatk.html,
1997 - 08 – 07

[6] Philip Zimmermann,
URL:http://www.stud.uni-hannover.de/stud/serv/pgpdoc/pgpdoc2/pgpdoc2.html
1994 -11 – 04

[7] W. Unruh,
PGP Attacks
URL: http://axion.physics.ubc.ca/pgp-attack.html
1996 – 02

[8] K. Jallad, J. Katz, B. Schneier,
Implementation of Chosen-Ciphertext Attacks against PGP and GnuPG,
Information Security Conference 2002 Proceedings, Springer-Verlag.