Getting Started With PGP

Digital Signature and Encryption
for E-mail and Files

Problem: E-mail is insecure

- More like a postcard than a letter
- Can be read by any number of people in transit
- If recipient's account is compromised, may be read by unauthorized person(s)
- Network sniffing may capture and reveal content to unauthorized person(s)

Problem: E-mail is easily forged

- **Current protocols date back to early 1970s**
- **Headers, especially “From:” are ridiculously easy to forge**
- **If sender account is compromised, may not actually have been sent by that person**
- **How can you verify the sender’s identity?**
- **A small demonstration (spoofmail.php)**

Solution: PGP ("Pretty Good Privacy")

- Developed in 1991 by Phil Zimmerman
- Based on public/private key pair cryptography principle developed by Whitfield Diffie
- Provides highly secure, portable encryption and signature for any digital data - including e-mail

PGP: Drawbacks

- Not all that “user friendly”
- Public/private key concepts take some getting used to
- Doesn’t scale well to large user groups
- If your private key is lost, corrupted or compromised, you have to generate a new one and start all over again

PGP: How Does It Work?

- **Let’s start with some basics of cryptography, and the high-level mathematics underneath it**
- **Don’t worry, this is just to give you an idea of what’s going on.**

Some Quick Definitions

- **Cryptography**: encrypting and decrypting messages
- **Algorithm**: a mathematical function used for encryption, decryption and verification
- **Key**: a string used encryption and decryption

(Digital) Keys

- **In computer cryptography, a key is simply a binary number of varying length, depending on the type of cryptography being used**
  - "Symmetric" cryptography generally relies on keys of 128 bits or more (e.g. most web browser SSL sessions)
  - "Asymmetric" cryptography requires keys of minimum 1024-bit length (2048 is typical these days)
- **A properly generated key can only be cracked by "brute force" - and only then after thousands of years (even at today's processing speeds)**

Keys: Encrypting/Decrypting (A bit oversimplified)

One typical way keys are used to encrypt/decrypt is by use of “exclusive-OR” (XOR) logic:

- If the bit to be encrypted is the same as the corresponding key bit, the encrypted bit is 0
- If they are different, the resulting encrypted bit is 1
- Without the key, you can’t be sure what the original bit was
- Simple keys like this are vulnerable to “known text” attacks

TEXT	KEY	RESULT
0	0	0
0	1	1
1	0	1
1	1	0

Penn
UNIVERSITY *of* PENNSYLVANIA

"Symmetric" Cryptography

- **Relies on both sender and recipient having identical copies of the encryption key - or access to a "third party" that can verify that both have the correct key (e.g. online "certificate authority")**
- **Can use relatively low key lengths**
  - 128 bits is more or less standard
  - 56 bits has been cracked
- **Problem: secure key distribution**

“Asymmetric” Cryptography (aka “Public/Private Key”)

- **Sender generates a “public/private” key pair**
- **The “public” key can be freely given out**
  - Published on website, in e-mail sigs, etc.
  - Some people put them on the back of their business cards
- **The “private” key is guarded at all costs**
  - If lost, corrupted or compromised, a new pair must be generated
  - All your contacts must be notified and supplied with the new public key
- **Data encrypted to the public key can only be decrypted by the corresponding private key - and vice versa**
- **PGP is based on “public/private” key cryptography**

How are public/private key pairs generated?

- **Oversimplifying a bit again, but they are essentially based on mathematical operations involving two really, really large prime numbers**
- **Prime numbers have been proven to have no known predictive pattern**
- **Finding prime numbers is easy for small numbers**
  - 14, 24, 49, etc
    - Even?…divisible by 2
    - Last digit is 5 or 0?…divisible by 5
    - And so on...
  - A LOT harder as the numbers get bigger - in fact, impossible

What is this number?

$$(2^{57885161})-1$$

(To write it out in decimal form would require 17.425.170 digits - at 1 digit per second, it would take more than 200 days, non-stop)

The (Currently) Largest Known Prime Number

$$(2^{57885161})-1$$

Source: primes.utm.edu/largest.html

(A really good site for "Stupid Number Tricks")

PGP Public/Private Key Pair Strength

- **The public key is mathematically derived from the private key**
- **The private key relies on the "computational difficulty" of deriving the large primes used to generate it**
- **Private key lengths of 1024 bits are probably OK, but 2048 is the accepted standard for most users**
- **Key lengths over 2048 bits are more secure, but tend to eat up processing power**

Recommended key size

Table 4: Recommended algorithms and minimum key sizes

Algorithm security lifetimes	**Symmetric key algorithms (Encryption & MAC)**	**FFC (e.g., DSA, D-H)**	**IFC (e.g., RSA)**	**ECC (e.g., ECDSA)**
Through 2010 (min. of 80 bits of strength)	2TDEA[23] 3TDEA AES-128 AES-192 AES-256	Min.: $L = 1024$; $N = 160$	Min.: $k=1024$	Min.: $f=160$
Through 2030 (min. of 112 bits of strength)	3TDEA AES-128 AES-192 AES-256	Min.: $L = 2048$ $N = 224$	Min.: $k=2048$	Min.: $f=224$
Beyond 2030 (min. of 128 bits of strength)	AES-128 AES-192 AES-256	Min.: $L = 3072$ $N = 256$	Min.: $k=3072$	Min.: $f=256$

From an archived publication:
Page 66: http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf

Algorithms

- **Used to induce "randomness" in encoded message**
- **Helps reduce "known text" vulnerability**
- **"Encrypting" algorithms are designed to be reversible (they have to be, don't they?)**
  - Some **well-known** ones: DSA, RSA
- **"Hashing" algorithms are "one-way", i.e., irreversible**
  - Used to provide "signature", or "shared secret" verification
  - Used for password storage on most operating systems
  - Well-known ones include SHA-1 or SHA-2 (used by PGP)

Example: SHA1

Input	Output
1	356a192b7913b04c54574d18c28d46e6395428ab
2	da4b9237bacccdf19c0760cab7aec4a8359010b0
3	77de68daecd823babbb58edb1c8e14d7106e83bb
4	1b6453892473a467d07372d45eb05abc2031647a
5	ac3478d69a3c81fa62e60f5c3696165a4e5e6ac4

Well-known Algorithms

- **The best-known algorithms are used to remove patterns and predictability from encoded messages**
- **They are not secret - in fact, they are made public and subjected to rigorous testing**
- **Encryption algorithms are reversible - they have to be**

Where’s the Security?

In a public/private key cryptography environment, the strength lies in the length and protection of the private key - and the algorithm

Digital Signatures

- **Usually applied to e-mail messages to verify the sender's identity**
- **Sender "signs" the message with his private key**
  - Hashing algorithm is applied as well
  - Hashing also provides check on message integrity
- **Anyone who has sender's public key can apply it to verify sender's identity**
  - ONLY someone who has the corresponding private key could have sent it
  - Hashing also verifies that the message was not altered in transit

What does a “Public Key” look like?

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: OpenPGP.js v0.9.0
Comment: http://openpgpjs.org

xo0EVOMXoAEEAM6uJZGrkiXd+bcjsrCs5JGUZo+SrqrnkevXL4a5oIUDWqG4
l/yf2s1VqfzTtci8uuKz0dVWWjZDLPLCCVWU6RXnr2+5F/VliaRkVRJ+FUzG
DWL3VKsX42L2+gsceOL3t+4rVIUGCA81AB/GRQ7Erm84UOGTrPCsLdrrUeSp
Z7pjABEBAAHNI0JhcnJ5IGRlIEdyYWFmZiA8YmdyYWFmZkBoaXZvcy5vcmc+
wrIEEAEIACYFAlTjF6AGCwkIBwMCCRBZYcp/r9V5bgQVCAIKAxYCAQIbAwIe
AQAA7bwEALX6EA6P8NnNlTRCBv1CZvmW43GpU7eA2wxg2Ym0NTaMOrXadcQe
anRIGahuaNRrhXWa9Lq5tDQAhy1OfZRnEPI8POOwmmX0EHO4jNveTUw38F17
kTkMPmGXp9W8piqAHvr+LXOs3ChIMpDeyBSvL9JCw3HK0BQcfIWsgqJfJ+7Y
zo0EVOMXoAEEALdeZkc1c24fhLgI7GCvMoKjm2oSnC3jHoJa/vgrT7Npy9jx
BfARB266dOGma9d2L9LVAnbPp/skInTUSlP+b08l2OlgxgZ41bXri29XLLQB
/CMbQU0Z2ARg6EWbmi5ugm0ZmGotkzbBhwbD0mRaIpdr2WPVAMUSTZetonyT
TGUZABEBAAHCnwQYAQgAEwUCVOMXoAkQWWHKf6/VeW4CGwwAAGtuBACqlIAt
ydWg7ctUvLkHxv8t8MX+uovAwBgwHa6XmXzYO4D2[illegible]XfXxRYPTaMpCGjW1qFN
601w01WHcT43zCSzbwZZy6zJ21Oq//vMS8BlXQ8f5[illegible]Xh+dWmgH2GA/bLGywK
tqe0GJex5MH3z7WLtM7twXKB67Ya+iI+Zt+JoQizQl[illegible]RA==
=2Ojo
-----END PGP PUBLIC KEY BLOCK-----
```

Hashed “fingerprint”:

1bf440b445d85c2ba39fb1365961ca7fafd5796e

(**Hexadecimal)**

How do I know that's really your public key?

- **Generating public/private key pairs does NOT require any authentication - you can associate any name and e-mail address with a key pair you choose**
- **To be completely certain that a given public key actually belongs to the person it claims, you need to physically <u>verify the person's identity and key *in person*</u>**
- **Check the key fingerprint**
- ***Do not trust third parties (security by the weakest link)***

Public Key Distribution

- ~~Published on a (personal) website~~
- Via a CD/DVD or USB drive
- ~~Send as e-mail attachment~~
- Send as e-mail attachment via internal mailserver ehhhmm.
- ~~Upload it to a “key server~~”
- Use a QR code on the back of your business card with your key fingerprint
- Print it out on a sheet of paper, along with the fingerprint not very practical...

Demonstration

Zimbra OpenPGP Zimlet

Training

Work with the
Zimbra OpenPGP Zimlet

Go to:
http://www.barrydegraaff.tk/training

Questions?