GnuPG: Open Encryption, Signing and Authentication



What is GnuPG?

GnuPG is the GNU project's complete and free implementation of the OpenPGP standard as defined by RFC4880. GnuPG allows to encrypt and sign your data and communication, features a versatile key management system as well as access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. A wealth of frontend applications and libraries are available. Version 2 of GnuPG also provides support for S/MIME.



OK. What is GnuPG?

- Implementation of public-key cryptography
- Conforms to an open standard (OpenPGP)
- Allows for:
 - Encryption of Data & Communication
 - Signing of Data & Communication
 - Authentication



Outline

- Background
 - Terminology
 - Motivations
 - General Theory
- Getting Started
 - Key Generation
 - Choices
 - Key Signing

- Best Practices
 - Threat Modeling
 - Key Separation
- Integration & UIs
 - Uls
 - E-mail
- Advanced Topics
 - Smart Cards
 - Authentication

Terminology

- PGP Pretty Good Privacy
 - Original implementation, 1991, by Phil Zimmerman
 - Source Available until 2000
- OpenPGP Standard for implementations
 - RFC 4880 (Replaced RFC 2440) (Message format)
 - RFC 3156 (e-mail format, PGP/MIME)
- GnuPG GNU-Project, GPL Implementation
 - Mostly PGP Compatible
 - Implements all of RFC 4880



5

Motivations: Encryption

- Protect messages against being read except by intended recipient(s).
- Intended recipient could be yourself.
- Can exchange secret communications without needing any pre-shared secrets.



Motivations: Signing

- Digital signatures prove that you wrote/signed a given chunk of data. (Non-repudiation)
- Used heavily for code signing, signed packages, etc.
- Message integrity (unmodified)



Shortcomings

- Encryption
 - Anyone with the private key can decrypt message
 - Have to know what key to encrypt to (anyone can generate a key with any UID)

- Signing
 - Anyone with the private key can sign a message
 - No proof of WHEN it was signed
 - No way to prove that you did NOT write a message



How it Works (Simplified)

- Public Key Encryption
 - Pair of Keys (Public, Private)
 - A message encrypted to one key can only be decrypted by the other key
 - Computationally infeasible to reverse calculation

Encryption

- Sender uses public key to encrypt
- Recipient uses private key to decrypt
- Signing
 - Signer uses private key to sign (encrypt)
 - Recipient uses public key to verify (decrypt)

9

Some Technical Details

- Messages are not
 really encrypted with public key
 cryptography
 - Encrypted with symmetric
 cryptography
 - Key then encrypted with public-key cryptography

- Likewise, messages not signed across the entire message
 - Hash is calculated
 - Signed with public-key cryptography
- Signing + encryption
 - Signed first
 - Only recipient verifies



OpenPGP Algorithms

- Public-key (Asymmetrical)
 - RSA(*)
 - DSA
 - ElGamal
 - (Future) ECC

- Symmetrical
 - IDEA
 - 3DES
 - CAST5
 - AES (*)
 - Blowfish
 - Twofish

ink)r



OpenPGP Algorithms

Most often used

- Compression
 - ZIP
 - ZLIB (*)
 - BZIP2

- Hashing MD-5
 - SHA-1 (*)
 - RIPE-MD/160
 - SHA-2 (Family)
 - SHA-256
 - **SHA-384**
 - SHA-512
 - **SHA-224**



ink)r



Getting Started: Key Generation

\$ gpg --gen-key

Please select what kind of key you want:
 (1) RSA and RSA (default)
 (2) DSA and Elgamal
 (3) DSA (sign only)
 (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits



Algorithm Choice

RSA

- "Safe bet" very commonly used in a variety of applications
- Based on Integer Factorization Problem
- DSA/ElGamal
 - A few cryptographers suggest it is SLIGHTLY stronger
 - Less researched
 - Based on Discrete Logarithm Problem
- Both are believed to be secure



Key Length

- Do not generate new 1024-bit keys!
- NIST suggests 2048 is secure until 2030.
 - 3072 secure until ~2040.
 - 4096 secure until ~2050.
- Quantum computing could change everything.
 - Topic for another day, and probably another group.
- Estimates against enterprise/government level attackers.
- Keylength.com



Getting Started: Key Generation

Please specify how long the key should be valid.

0 = key does not expire <n> = key expires in n days <n>w = key expires in n weeks <n>m = key expires in n months <n>y = key expires in n years Key is valid for? (0) 1d Key expires at Thu 17 Mar 2011 11:06:24 PM EDT Is this correct? (y/N) y



Key Expiration

Expires

- Key will fall out of use if you lose private key
- Update key periodically
- Regenerate key and get new signatures

- Never expires
 - No need to update date or regenerate
 - May never fall out of use if you lose your key or compromised



Getting Started: Key Generation

gpg --gen-key

You need a user ID to identify your key; the software constructs

the user ID from the Real Name, Comment and Email Address in this form:

Real name: Santiago Figueroa Email address: santi<u>@example.com</u> Comment: Demo Key Only You selected this USER-ID: "Santiago Figueroa (Demo Key Only) <santi<u>@example.com</u>>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o



Your Key



Getting Started: Finding Keys

- gpg --recv-keys <keyid>
 - gpg --recv-keys 5DEA789B
- gpg --search-keys <UID substring>
 - gpg --search-keys <u>santi@example.com</u>
- Keyserver
 - pool.sks-keyservers.net
 - pgp.mit.edu
- gpg --refresh-keys



Getting Started: Sending K

- gpg --send-key
- Make sure you really want the key out there
 - Don't publish test keys
- Use again after signing keys
 - Only if the original key was on the keyserver
 - Considered rude to publish someone's key



Keysigning

- Why sign keys?
- Alice wants to e-mail Carol, but doesn't have her key
- Alice downloads Carol's key from a keyserver
- But wait! Anyone could generate a key for <u>carol@example.com</u>
 - Never forget who might have access to e-mail



Keysigning

- Alice knows Bob who knows Carol
- Alice has met Bob, verified Bob's key, signed Bob's key
- Bob has met Carol, verified Carol's key, signed Carol's key
- If Alice trusts Bob, Alice can believe this key really belongs to Carol



Web of Trust

- Connections of signatures between users/keys
- gpg –list-sigs
- OpenPGP model instead of PKI (Certificate Authorities)
 - Some CAs may not be trustworthy, so some consider Web of Trust superior
 - Certainly, individuals I trust more than many CAs



24

Keysigning Parties/Events

- Help expand your Web of Trust
 - Helps verify not only those at party, but also those just past that point
- Most effective in cases where you want to communicate within that "social circle"



Signing Philosophies

ID-Based

- Present ID (often 2)
- Match Names to UIDs
- Sign Key

- E-mail based
 - Signer sends encrypted email to signee
 - Signee responds with signed email
 - Proves control of email address



Best Practices: Key Security

- Keep a copy of your key in a secure location
- Use a strong passphrase
 - If the file that contains your key is compromised, it is encrypted with this passphrase
- Keep a pre-generated revocation certificate offline "just in case"
 - This should be secured too



Best Practices: Threat Modeling

- U.S. Government
 - U.S. v. Boucher
 - Probably nothing will protect you
- Foreign Government
 - Might have law compelling you to disclose passphrase
 - Only if you are there or commit crime there

Corporation

- Unlikely to have resources
- Termination for improper computer use
- Malicious Attacker
 - Theft of Key
 - Keylogger



Best Practices: Key Separation

- Key Capabilities
 - Sign
 - Certify
 - Encrypt
 - Authenticate

- Use --expert option to gpg
- Separate keys: if weakness found in one key, other keys may be fine





Best Practices: Key Separation

4096R/5DEA789B created: 2010-12-19 expires: never pub usage: C validity: unknown trust: unknown expires: 2012-12-18 3072R/3F0A7DEA created: 2010-12-19 sub usage: S 3072R/63469263 created: 2010-12-19 expires: 2012-12-18 usage: E sub expires: 2013-02-22 2048R/8D1C060E created: 2011-02-23 sub usage: A [unknown] (1). Santiago Figueroa <santi@eample.com>



Best Practices: When to Sign E-Mail

Always

- Some suggest it builds history
- Still doesn't prove an unsigned message didn't come from you
- Be careful what you sign only the body is signed
- Important e-mail
 - Signifies email as significant
 - My personal practice



Best Practices: Signing Files

- Be careful signing files you didn't create
 - Binary files (including doc, docx, odt, etc.) may have multiple data streams, hidden text, etc.
- Sign "significant" files
 - Off-site backups (really!)
 - Code, packages, etc.
- Not currently in use for legal contracts
 - May change soon, but need "legal" keyholder verification





Best Practices: E-mail encryption

- Encrypt everything (to recipients with OpenPGP)
 - Some overhead
 - Many mobile devices don't support GPG or users don't use GPG on there
- Encrypt only the important
 - Tells an attacker which messages are important
 - Allows casual messages to be read everywhere



Integration: UIs

- GPA
 - Standard, Cross-Platform
 - GTK-based
- Seahorse
 - In most Gnome Installations
 - Highly Integrated
 - GPG/SSH/etc.

KGPG

- KDE based
- GPG only
- (Non-Linux) GPGTools
 - OS X Suite
- (Non-Linux) Cryptophane
 - Windows

ink)r



Integration: GPA

× GNU	Privacy As	ssistant - Keyring Editor						
<u>Fi</u> le	<u>E</u> dit <u>K</u> eys	s <u>W</u> indows <u>H</u> elp						
Edit	Belete	e Sign Import Export Brief Detailed Files						
R	Keyrin	ng Editor						
	Key ID	User Name	4					
-3	946E6762	Matthias Welwarsky <mwelwarsky@web.de></mwelwarsky@web.de>						
-3	D3P9318D	Mauricio Saint-Supery <tas@subdimension.com></tas@subdimension.com>						
-3	DF1DD471	Michael H. Warfield <mhw@wittsend.com></mhw@wittsend.com>						
	9A4C704C	Michael Nahrath <michael@nahrath.de></michael@nahrath.de>						
	E99DD5FD	Michael Richardson (General Purpose Key 2002) <mcr@sandelman.ca></mcr@sandelman.ca>						
-9	3D97C149 Michael Stone <mstone@debian.org></mstone@debian.org>							
-3	3103F52B	Microsoft Security Response Center <secure@microsoft.com></secure@microsoft.com>						
23	27FC3CA8	Miguel Coca <mcoca@gnu.org></mcoca@gnu.org>						
-9	DC8514F1	Miguel Mendez <flynn@energyhq.homeip.net></flynn@energyhq.homeip.net>						
-9	EBBD5271	Mike Caudill (Cisco PSIRT) <mcaudill@cisco.com></mcaudill@cisco.com>						
-9	13D9873D	Mirko Dziadzka (Key Certification only, please use my other keys for communication	1) <m< td=""></m<>					
-9	92E3B914	MoebiuZ <moebiuz@terra.es></moebiuz@terra.es>						
-	C425F18E	NG SEC Research Team <labs@ngsec.com></labs@ngsec.com>						
	52D1CAB1	Nathalie Weiler <weiler@tik.ee.ethz.ch></weiler@tik.ee.ethz.ch>	•					
•			Þ					
Deto	ils Signatur	185						
	The	key has both a private and a public part						
Lkor	Name: Mia	uel Cosa < massa @anu aras						
User	Mane. Mig	uel Coca <e970095@zipi fi="" upm.es=""></e970095@zipi>						
	Key ID: 27FC	C3CA8						
Fing	erprint: E60/	A CBF4 5C6F 914E B6C1 C402 8C4D C7B6 27FC 3CA8						
Exp	pires at: nev	rer expires						
Own	er Trust: Ultin	nate						
Key V	alidity: Fully	/ Valid						
Ke	y Type: DSA	1024 bits						
Crea	ted at: 03/2	2//00						
Selecte	ad Default K	(ev: Miauel Coca <mooca@anu ora=""> /</mooca@anu>	27EC3CA8					
00010	A Doradi N	ingeriood (noodegratog	2/1000/10					



INK)r

Integration: Seahorse

😽 Enci	ryption Key Manager 📃 🗖 🗙							
<u>K</u> ey <u>E</u> dit <u>R</u> emote	e <u>V</u> iew <u>H</u> elp							
Properties Exp	Filter:							
My <u>P</u> ersonal Keys	Keys I <u>T</u> rust Keys I've <u>C</u> ollected							
Name 🗸 Key ID Validity								
First time options:								
To get started with encryption you'll need keys. 🔯 Help								
Import existing keys from a file:								
Generate a new k	ey of your own:							
	Error opening directory '/home/gnome/.ss							



ink)r

Integration: KGPG

lame	 Email Tru 	st Expiration	Size	Creation	ID				
Lydia Pintscher	lydiapint	ter brocketter	1024 / 2048	2004-04-01	72C4ECD6				
S David Faure	dfaure@	_	1024 / 1024	2001-09-12	09B06E10				
- David Faure	dfaure@			2009-11-02	09B06E10				
+ E David Faure	<u> </u>	Key Server	– KGpg		SSSS				
Bavid Faure	Transie Insurances								
+ E David Faure	import Export								
 Second Subkey 	Key server:								
A L Spehr	hkp://keys.gnupg.net				~				
 Tario Andres Rodrigue 	Text to search or ID of the key to import:								
	House HILLS hundred								





ink)r

Integration: E-Mail

- Thunderbird
 - Enigmail
- KMail
 - Integrated
- Evolution
 - Integrated
- Mutt
 - Integrated

- Also, transparent outgoing
 - GNU Anubis
 - Freenigma
- See Also
 - Vim integration
 - Emacs integration



Advanced Topic: Smartcards

- Physical device that generates and stores keys and performs signing and encryption operations
- OpenPGP Smartcard v2 allows for up to 3 RSA keys, each up to 3072 bits in size
 - Sign/Certify
 - Encryption
 - Authentication
- Sold by Kernel Concepts out of Germany



Smartcard-Specifc Terms

PINs

- Admin PIN
- PIN
- Similar to passphrase; cards limit length; use only digits if you intend to use a reader that has a PIN pad
- 3 strikes rule



Card Readers

- Any CCID or PC/SC-compliant smart card reader should work
 - Very common (Amazon, eBay, etc.) with use of CAC cards for U.S. Military
 - Also available from Kernel Concepts
- Requires GPGSM on Debian-derivatives (S/MIME support for GPG)
- pcscd and pcsc-lite tools (required for PC/SC)
 - Provides more details if you run into issues



Usage

- gpg –card-status
 - Use to get card "recognized"
- gpg --card-edit
 - admin
 - passwd
 - url
 - fetch
 - Generate
- gpg --edit-key
 - keytocard



Authentication

PAM

- Poldi
- SSH
 - gpg-agent is a drop-in replacement for ssh-agent
 - enable-ssh-support
 - Must disable standard SSH agent, Seahorse, etc.
 - gpg --card-status
 - ssh-add -I, ssh-add -L (public key)

