

PGP: What, Why, When, Which, How, and More...

UUASC
02/02/06

(formerly USC LUG 11/17/05)

Phil Dibowitz

What We Will Cover

- (Briefly) What is PGP?
- Why use PGP?
- When to use PGP?
- Which *kind* of PGP to use and when (Traditional vs PGP/Mime)
- How to use PGP
- PGP Web of Trust
- Comparisons to X509/SMIME

What We Won't Cover

- How encryption works
- How digital signatures work
- How PGP differs, cryptographically, from other methods

What is PGP?

- Pretty Good Privacy
- I'm not a history professor
- Non-centralized

WHY?

- Why should I use PGP?

Why: What Can I Do With It?

- Encrypt data to yourself or others
- Digitally verify both unencrypted and encrypted data: authenticity of content and author
- Protect your data
- Protect yourself

Let's see how...

Why: Protecting Your Data

- Keep your passwords in one secure place

```
gpg --encrypt passwords.txt
```

Creates passwords.txt.gpg

```
gpg --decrypt passwords.txt.gpg
```

With a passphrase and the key (2-piece auth!), this dumps the contents to stdout

- Assumes you have keys setup

Why: Protecting Your Data

- Encrypted email – without the key and the passphrase – no one can read that email
- Lots of support:
 - Thunderbird+Enigmail
 - Outlook+GPGol
 - Apple Mail+GPGMail
 - Mutt
 - Pine+Pine Privacy Guard
 - Evolution

Why: Protecting Yourself

- Signing email – with your public key, anyone can check you sent it, and it is unmodified.
- Managers: Directives can't be modified or forged
- Employees: Have proof directives were sent
- “Shutdown server X”
- “Sure, give away my ticket, I can't make it”
- “You have to remove file X to break the deadlock”
- “Peter is the new guy, he should be granted access to the data center.”

WHEN?

- When should I use PGP?

When: Encryption

- When you need it:
- Distribute new passwords
- Encrypt your personal password store
- Directives with sensitive data
- Credit card numbers
- Personal messages having to go through untrusted 3rd parties
- Telling your girlfriend in the next office something, erm, “personal” via work email

When: Signing: Sometimes

- The *sometimes* argument
- Signed messages came from you
- Signed messages weren't modified in transit
- Unsigned messages might be from you, might not
- Unsigned messages might be modified in transit, might not
- Still vectors for attack

When: Signing: Always

- The *always* argument
- All messages are from you
- All message guaranteed unmodified
- Unsigned messages can just be /dev/null'd, or treated with high-suspicion
- No forged email!
- No mis-quoting
- Once it's setup, no work to sign – why not?
- But... what about ugliness? Lack of support? What kind? We'll cover it.

WHICH?

- Which type of PGP email should I use?

Which: Traditional vs PGP/Mime

- Traditional, aka, “clear-signed” - just a PGP header and footer, and a signature below that – all in the body of the email or document.
- PGP/Mime splits the body and the signature up so the user always sees the body as intended. If they support PGP, their client will use the signature for verification. If not, they'll just see a small attachment.

Which: Traditional

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA1

This is an email with a PGP signature.

This will go into my presentation on PGP.

PGP is a great and wonderful thing.

Oh, and of course. Hello World!

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v1.4.2 (GNU/Linux)

Comment: Using GnuPG with Thunderbird - <http://enigmail.mozdev.org>

iD8DBQFD3xldN5XoxaHnMrsRAhGEAJ4kQqrQeULjd+mHBupiBDq14077TACfYKmm
l7B4drp8B7m3T9HtAvdEfJ8=

=AygA

-----END PGP SIGNATURE-----

Which: PGP/Mime

This is an email with a PGP signature.

This will go into my presentation on PGP.

PGP is a great and wonderful thing.

Oh, and of course. Hello World!

- assuming MIME support (everything but Outlook Express)

Which: Traditional

- PROS
- Simple: Just text in the body
- Anything can read the message
- Don't need MIME or modern mail readers
- Can be easily verified by news readers, web-based readers, etc.

CONS

- Messy – hard to read the message without PGP support
- Obsolete

Which: PGP/Mime

- How does it work? Split the message into two parts using MIME:
- Body:
Content-type: text/plain
(implies Content-disposition: inline)
- Signature:
Content-type: application/pgp-signature
Content-disposition: attachment
- RFC's 3156, 2015, 2046

Which: PGP/Mime

- PROS
- Anything with MIME support sees the message clearly, with the signature as an insignificant attachment (for PGP-inept)
- Can sign a forwarded message with a signature in it easily (using MIME-encapsulation)
- Looks cleaner
- CONS
- Non-MIME compliant readers have trouble

Which: Application/PGP

- Horrible wanna-be standard replaced by PGP/Mime
- Takes a **traditional-signed message**
- Body is Content-type: application/pgp and Content-disposition: inline

- Almost **nothing** displays properly. Only mutt 1.4x ever supported it. Mutt 1.4x used this when set to “traditional” ...

Which: The Decision

- The decision should ultimately be based on who your readers are:
- What mail client is most common? What other mail clients are used?
- Are they going to be doing verification? Can they install a plugin?
- Once you have that...

Which: Support

| CLIENT | TRADITIONAL | APPLICATION/PGP (mutt 4x only?) | PGP/MIME |
|---------------------------|---|---|--|
| Outlook (2k & 2k3) | Displays message (with extras) | Does not display message | Displays message |
| Outlook Express | Displays message (with extras) | Does not display message | Does not display message Can't even double-click to view - must download and open separately |
| Thunderbird | Displays message (with extras) | Displays message IF "display attachments inline" is set | Displays message |
| Thunderbird + Enigmail | Displays & verifies message | Displays & verifies message IF "display attachments inline" is set | Displays & verifies message |
| Pine | Displays message (with extras) | Does not display message | Displays message |
| Pine + Pine Privacy Guard | Displays & verifies message (with extras) | Does not display message | Displays message |
| Mutt | Displays & verifies message | Displays & verifies message | Displays & verifies message |
| Apple Mail | Displays message (with extras) | Does not display message | Displays message |
| Apple Mail + GPG Mail | Displays & verifies message (with extras) | Does not display message | Displays & verifies message |
| Eudora | <i>unknown</i> | <i>unknown</i> | <i>unknown</i> |

Which: No, Really Phil, Which one?

- Assuming the people who will be verifying can handle either, how do you piss off the PGP-inept the least?
If most recipients use:
- **Outlook Express:** (they need help) Traditional so they can see your email (but consider only signing as needed)
- **Outlook/Pine** (where most people aren't verifying): PGP/Mime – they'll see a clean message inline
- **Outlook/Pine** (with most people verifying): Traditional – only way plugins can verify/decrypt
- **Thunderbird/Netscape/Mail/Mutt:** PGP/Mime – clean and verifiable

HOW?

- I now know Why, When, Which – and I like it!
But how do I do it?

How: Get PGP

- Windows: GnuPG for windows: gnupg.org
- Linux/Solaris/BSD: GnuPG: gnupg.org
- Mac: MacGPG: macgpg.sf.net

How: Create Keys

- \$ gpg --gen-key
- Type:
 - (1) DSA and Elgamal (default)
 - (2) DSA (sign only)
 - (5) RSA (sign only)
- Size: 1024 - 4096
- Expiration
- Real Name, Comment, Email
Phil Dibowitz (Work Key) phil@example.com
- **PassPHRASE**
- More details at <http://www.phildev.net/pgp/>

How: Now What?

- Now you can:
Encrypt/decrypt/verify/sign files and keys
- OK, but what about email?!

How: MUA Level

- **Thunderbird:** Enigmail (enigmail.mozdev.org)
- **Apple Mail:** GPGMail (www.sente.ch)
- **Mutt:** Included
- **Outlook:** GPGol (www.g10code.com)
- **Pine:** Pine Privacy Guard (quantumlab.net)
- **Evolution:** Included

Install instructions:

<http://www.phildev.net/pgp/>

How: MUA Level

- Configure GnuPG: (in `~/gpg.conf`):
 `keyserver pgp.mit.edu #only one!`
 `keyserver-options auto-key-retrieve`
Most MUA support multiple servers, i.e. enigmail.
Add as many as you want. `pgp.com` is good.
- In your MUA Plugin:
- Traditional or PGP/Mime?
- Sign by default?
- Verify by default!

TRUST

- Who do I trust, how do I trust them, and why?

Trust: Intro

- Unlike in x509/SSL, trust isn't controlled by one authority
- Down with the man! No expensive certificates!
- Multiple signatures == better verification
- Verification by people YOU trust
- More trust requires more work
- Multiple “levels” of signing enable fine-grained trust
- Private and public trust

Trust: Implicit vs Explicit

- Signing someone's key indicates some level of trust and/or verification of identity
- You can also add local “trust”
- If a key you receive is signed by a key you have signed, there is implicit trust there
- If a key you receive is signed by a key you have locally trusted, there is implicit trust
- Signing a key is also implicit trust
- Explicit trust is rarely used, but neat.

Trust: Signing Keys

- **Why:** To vouch for the identity of one whose identity you have checked.
- **When:** Key signing parties, at the office, when you meet someone interested, etc.
- **How:** Check 2 forms of ID, check key fingerprint, and optionally check the email address, then sign the key and send it to them

Trust: Verifying Identity

- **ID:** At least one photo ID, usually two forms of some ID. You're saying to the world this key belongs to a person!
- **Fingerprint:** Verify the key you're signing has the fingerprint they have on their local copy.
- **Email:** (optional) encrypt an email to them with a secret word of your choosing and a number they chose. They must email you back the word and a number you chose.

Trust: Actually Signing

- `gpg --sign-key $KEYID --ask-cert-level`

Trust options:

- (0) I will not answer. (default)
 - (1) I have not checked at all.
 - (2) I have done casual checking.
 - (3) I have done very careful checking.
-
- Email or `gpg --send-key`
 - In `gpg.conf`: “ask-cert-level”

Trust: Local Trust

- To add local trust: `gpg --edit-key $KEYID`
- From the `gpg` shell, “trust”
- Trust options:
 - (1) I don't know
 - (2) I do NOT trust
 - (3) I trust marginally
 - (4) I trust fully
 - (5) I trust ultimately
- “save”

Trust: Local Trust

- In general, only trust your keys at 5
- If you know someone does a very thorough job verifying identify, trust them at 4, and you'll trust keys they sign even if you haven't signed their key, or your sig expires
- Local trust is in a trustdb, no one sees it
- Local signing is another way to do local trust, we won't cover it...

Trust: Calculated Trust

- So.. uh, this key is signed at 3 by a key that's signed at 2 by a key that's trusted by my key, which means I, uhm, what?
- It's complicated, and infinitely tunable, but by default...
- A key is trusted if it meets **both**:
 1. It is signed by enough valid keys, meaning one of the following:
 - * You have signed it personally,
 - * It has been signed by one fully trusted key
 - * It has been signed by three marginally trusted keys
 2. The path of signed keys leading from K back to your own key is five steps or shorter.

Trust: Example 1

- Phil signs Stewie (full)
- Stewie signs Peter (full)
- Peter signs Brian (full)
- Brian signs Meg (full)
- Meg signs Lois (full)

- Phil trusts all of them

Trust: Example 2

- Phil signs Stewie (marginal)
- Phil signs Meg (marginal)
- Phil signs Peter (marginal)
- Meg signs Chris (full)
- Stewie signs Chris (full)

Phil doesn't yet trust Chris...

- Peter signs Chris (full)
- Phil trusts Chris

Trust: Example 3

- Phil signs Stewie (full)
- Stewie signs Meg (full)
- Meg signs Peter (full)
- Peter signs Brian (full)
- Brian signs Lois (full)
- Lois signs Chris (full)

- Phil trusts everyone **except** Chris

Trust: Example 4

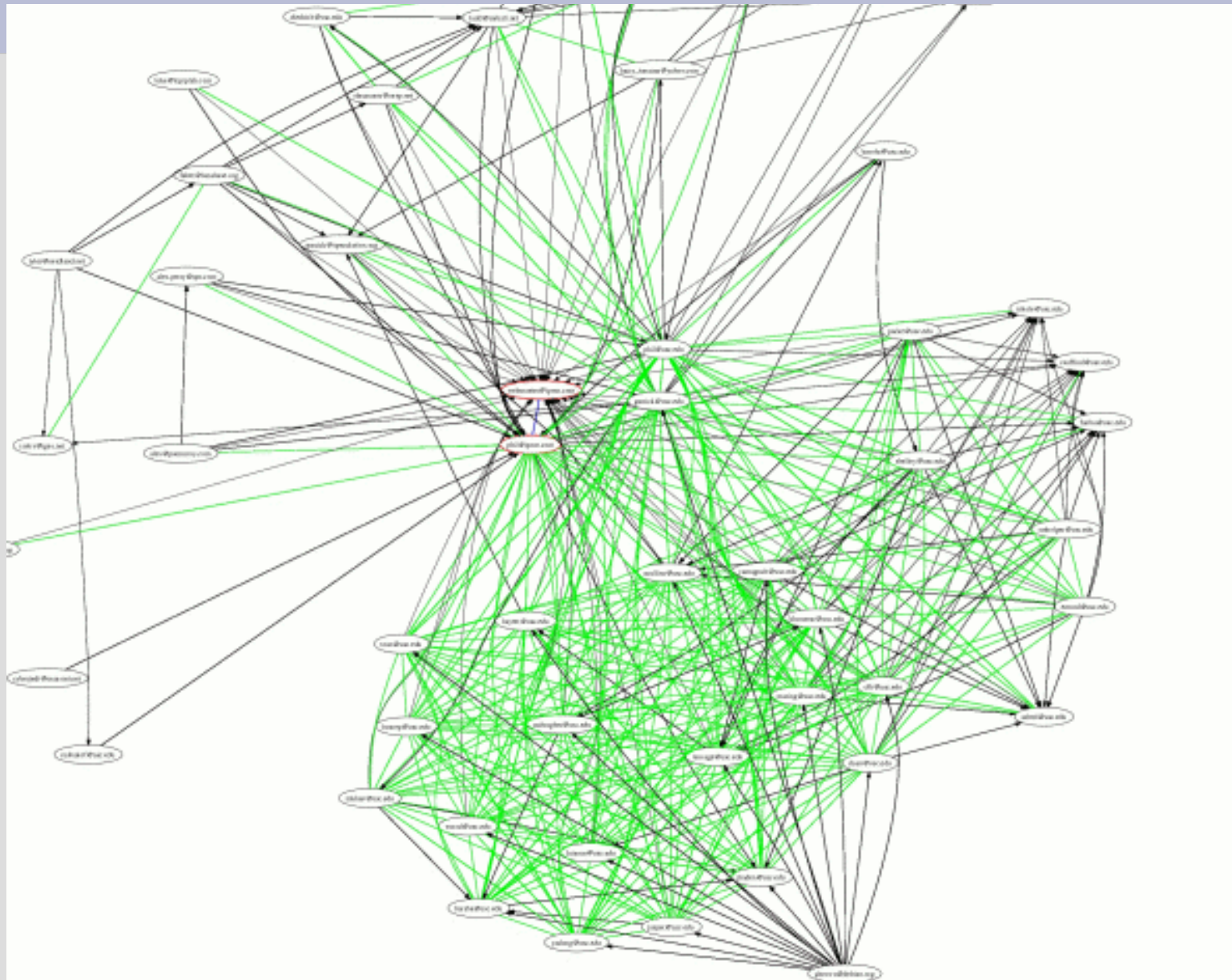
- Phil signs Brian (marginal)
- Phil signs Peter (marginal)
- Brian signs Peter (marginal)

- Phil trusts Peter since he directly signed it, even though it's “marginal”

Trust: Tunables

- --max-cert-depth n
How deep the chain go and still be trusted
- --completes-needed n
How many completely trusted keys are needed
- --marginals-needed n
How many marginally trusted signatures are needed

Trust: Web of Trust



Trust: Web of Trust



Trust: Signatures

```
$ gpg --list-sigs phil@ipom.com
pub 1024D/A1E732BB 2003-09-12
uid      Phil Dibowitz <phil@ipom.com>
sig 3    A1E732BB 2004-11-20 Phil Dibowitz <phil@ipom.com>
sig 3    X 8CAFF3DF 2003-09-12 Phil Dibowitz <phil@usc.edu>
sig 3    X 96E6F473 2003-09-12 Garrick Staples <garrick@usc.edu>
sig 2    X 819FD62E 2003-09-24 S. Tyler McHenry <tyler@nerdland.net>
sig 3    X E65FF97B 2003-11-03 Ted Faber <faber@lunabase.org>
sig 3    A1E732BB 2003-09-12 Phil Dibowitz <phil@ipom.com>
sig 3    X 808D0FD0 2003-11-24 Don Armstrong <don@donarmstrong.com>
sig 3    X FEA48B61 2003-11-24 Garrick Staples <garrick@speculation.org>
sig 3    X A4B1D0D4 2003-11-25 John Mullins <mullins@usc.edu>
sig 3    X 5811ED5F 2003-11-25 Carl Hayter <hayter@usc.edu>
sig 3    X EBA65398 2003-11-25 Asbed Bedrossian <asbed@usc.edu>
sig 3    X FC2FC9F0 2003-11-26 Chet Burgess <cfb@usc.edu>
sig 3    AE127015 2003-11-23 Todd A. Lyons (Cannonball) <todd@mrball.net>
sig 3    X 6AFD6695 2003-12-01 Linda Savage <lsavage@usc.edu>
sig 3    C730C0E4 2003-12-04 Harry Tanama (nick name pr0gm3r)
<harry_tanama@yahoo.com>
sig 3    X 398D7394 2003-12-05 Brian Emord (no one) <emord@usc.edu>
```

...

Trust: The Pretty Picture

- I know geeks like the nifty pictures...
- `$ sig3 -d 2 'phil@ipom.com webmaster@ipom.com' >keys.dot`
- `$ neato keys.dot -Tps -o keys.ps -v -Goverlap=scale -Gsplines=true`
- `$ convert keys.ps keys.jpg`
- `$ rm keys.ps keys.dot`
- Wtf are “sig3” and “neato”?

Trust: Helper Utilities

- sig2dot – Takes the output of `gpg –list-sigs` and creates a “dot” file.
<http://www.chaosreigns.com/code/sig2dot/>
- sig3 – same idea as sig2dot, but with better support for large keychains. Written by Carl Hayter from USC.
<http://www.phildev.net/pgp/>
- neato is part of the graphviz package
Takes data files and creates postscript plots.
- Convert is part of ImageMagick

COMPARISON: X509/SMIME

- How does it compare?

Comparison: X509/SMIME

- SMIME requires a certificate authority for your X509 certificates
- One entity must be trusted by everyone, or it doesn't work
- Usually costs money
- SMIME signatures are **vastly** longer...

Comparison: X509/SMIME

- MIAGCSqGSIB3DQEHAqCAMIACAQExCzAJBgUrDgMCGGUAMIAGCSqGSIB3DQEHAQAAoIIGFJCCAs8w
ggI4oAMCAQICAwW++jANBqkqhkiG9w0BAQQFAADBiMQswCQYDVQQGEwJaQTEIMCMGA1UEChMcVGhh
d3RIIEVnbN1bHRpbmcmcgKFB0eSkgtHRkLjEsMCoGA1UEAxMjVGHhd3RIIFBlcnNvbMFSIEZyZWVt
YWVlElzC3VpbmcmG0EwHhcNMDQwNTA0MDMyMDAyWWhcNMDUwNTA0MDMyMDAyWjBDMR8wHQYDVQQD
ExZUaGF3dGUgRnJlZW1haWwWgTWVtYmVyMSAwHgYJKoZlIhvcNAQkBFhF3YXNzYUBiZW1waGlzLmVk
dTCCASlWdQYJKoZlIhvcNAQEBBQADggEPADCCAQoCggEBANm8wiPSE3aeqFpA2o/adOYCZa2T7oHi
fjZE8IK2EXtgcBZmC2AmmPvPQTtU3sB4AV4bBaKT75qefvNQMSYS2mZjTuCiUSFvLgJ/pTwBf
GWquvXJH+Q57e1mqkSTKqSRWwA2GsZ1u7YytAaeiyB5p3FiWHC/zOT8V0CB9NYFqn4rNvL5NIEn
keK4y42/aMDX0XdywR+Bep1vUfRMBlludq/f49hLUGDbfuY5x44Z/RXbjBH6eClhCmyhgmRD/fcg
/cHwkwESA4Hisz3UErTsrZShEkuloG8W5SJSRFzXh3hqOibQxiU2bN0Gq3q9UD3kBFuXOUg2PGVL
oha/QfUCAwEAAaMuMCwwHAYDVR0RBBUwE4ERd2Fzc2FABWVtcGhpcy5lZHUwDAYDVR0TAQH/BAlw
ADANBqkqhkiG9w0BAQQFAAOBqQAhuI7KCwfeiJy/4ndpV2lHr4Tw2mt2QMkGrVFC9OgzD10BuNDX
UD4y7qnHvzbZbavlsomxZ4HtNX0NLYNQKTRL2WtSeT2NAlaGSFHXGJDQAHFYX6+ZLnP7MKRU4Ovv
OssPfrQH71qMtOhnDoGXypFrpjefQLPoGnm0+PvDz6SGDCCAz8wggKooAMCAQICAQ0wDQYJKoZl
IhvcNAQEFBQAwdExCzAJBgNVBAYTAipBMRUwEwYDVQQIEwXZlXN0ZJulENhcGUxEjAQBGNVBACT
CUNhcGUgVg93bjEaMBGGA1UEChMRVGHhd3RIIEVnbN1bHRpbmcmcgKDAmbGVBAsTH0NlcnRpZmlj
YXRpb24gU2VydmljZXMGRI2aXNpb24xJDAiBgNVBAMTGT1RoYXd0ZSBQZXJzb25hbCBGcmVlbWVp
bCBDQTErMCKGCSqGSIB3DQEJARYccGVyc29uYWwWtZnJlZW1haWwAdGhhd3RlLnNvbTAEF0wMzA3
MTcwMDAwMdBaFw0xMzA3MTYyMzU5NTlaMGlxZCzAJBgNVBAYTAipBMSUwIwYDVQQKExxUaGF3dGUg
Q29uc3VsdGluZyAoUHR5KSBMdGQuMSwwKgYDVQQDEyNUaGF3dGUgUGVyc29uYWwWgRnJlZW1haWwWg
SXNzdWluZyBDQTCBnzANBqkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAxKY8VXNV+065yplaHmjAdQRw
nd/p/6Me7L3N9VvyGna9fww6Yfk/Uc4B1OVQCjDXAmNaLlkVcl7dyfArhVqqP3FWy688Cwfn8R+R
NiQqE88r1fOCdz0Dviv+uxg+B79AgAJk16emu59I0cUqVIUPSAR/p7bRPGEEQB5kGXJgt/sCAwEA
AaOBIDCBkTASBgNVHRMBAf8ECDAGAQH/AgEAMEMGA1UdHwQ8MDowOKA2oDSGMmh0dHA6Ly9jcmw
dGhhd3RlLnNvbS9UaGF3dGVQZXJzb25hbEYyZWVtYWIwS0EuY3JSMAsGA1UdDwQEAwIBBjApBgNV
HREIjAgpB4wHDEaMBGGA1UEAxMRUHR5KSBMdGQuMSwwKgYDVQDEYyNUaGF3dGUgUGVyc29uYWwWgRnJlZW1h
aWwWgSXNzdWluZyBDQIDDD76MAkGBSsOAwIaBQCgggFTMBGCSqGSIB3DQEAzELBqkqhkiG9w0B
BwEwHAYJKoZlIhvcNAQkFMQ8XDTA1MDEyMDIyYzU5NTlaMGlxZCzAJBgNVBAYTAipBMSUwIwYDVQQKExxUaGF3dGUg
Q29uc3VsdGluZyAoUHR5KSBMdGQuMSwwKgYDVQQDEyNUaGF3dGUgUGVyc29uYWwWgRnJlZW1h
aWwWgSXNzdWluZyBDQIDDD76MA0GCSqGSIB3DQEBAQUABIIBAI5DZx5NbAg2
0cvrNT74279t6xvSpYudfuk81Ryb8ZAj3152ilKgSpTzRoWlVusZOLj+3YnRvy7Fb2ZtClkj+dS
vvy+Y1YegzVB33DW1huhjF6t9RQEYYy1xkWir0VjXSMaiyYlMrOIBo2pJXQU85RrzvF1NCamak8j
AIWDVzdyp4XBvQvs4LtfOgajf4QlkkOXtOVHxDgFDJqxdof03UJ/etZVwep6OBgOaz0cm+K4F6Dh
8hZq9r/SsOedZQ1C5s4vt1Ysz7dztR6N9bdCs1+UKjManw7kZhcAWIbRIAznmkmNNpRfJMD0xbq
ebGZMAAr9pTlgHXn1zhsdV/mXAAAAAAAAA=

Comparison: X509/SMIME

- Recall the PGP sig:

```
iD8DBQFD3xldN5XoxaHnMrsRAhGEAJ4kQqrQeULjd+mHBupiBDq14077TACfYKmm  
l7B4drp8B7m3T9HtAvdEfJ8=  
=AygA
```

- Which would you rather have attached to an email?

Comparison: X509/SMIME

- SMIME has been around for a long time – support in Netscape 4
- Never caught on
- I2 trying to revive it
- Potentially useful when one entity **is** trusted by everyone.
- Many other **great** uses for X509 and SSL...

More Info / Sources

- All of this information – with less brevity – and MUCH more is available at my PGP site:
<http://www.phildev.net/pgp/>
- RTFM:
<http://www.gnupg.org/gph/en/manual.html>
- RFC 3156: MIME Security with OpenPGP
- RFC 2015: MIME Security with Pretty Good Privacy (PGP)
- RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

Things To Take With You

- Always sign – it'll save your ass
- Encrypt as needed – it'll also save your ass
- For most people PGP/Mime is the best to use (unless everyone you know uses OE)
- *Good* MUAs will do all the work for you:
Thunderbird, Apple Mail, mutt

Fin

- I'm finally done!
- Questions?
- Comments?
- Heckles?
- Concerns?

- <http://www.phildev.net/pgp/>

Thanks for coming!