ABSTRACT

This document gives a basic primer on the wide subject of release signing. See under Furth for links to authoritative sources of deeper information.

Release managers are the target audience.

This document is informative and does not constitute policy.

IMPORTANT

All new **RSA** keys generated should be at least **4096** bits. **Do not** generate new **DSA** keys.

Recent research has revealed weaknesses in SHA-1, and thus in the DSA and 1024 bit RSA O keys which must use this algorithm. Though no realistic attacks have been made public, exp with similar weaknesses in MD5 suggests that further advances may well lead to practical a within the next few years. This accords with current NIST guidance on DSA.

The future impact of this weakness on Apache can be mitigated by action now. What needs is a little involved. So, complete instructions have been prepared. Please read and follow th

- Committers without a code signing key should read this document and follow these ir (openpgp.html#generate-key).
- Committers with a DSA key or an RSA key of length less than 2048 bits should generat for signing releases. The original key does not need to be revoked yet. Follow this guic (key-transition.html).
- Committers with RSA keys of length 2048 or more do not need to generate a new key y should reconfigure their client to avoid the weakness by following these instructions (openpgp.html#sha1) and wait for the next major OpenPGP revision.

How to find the length of your key is described here.

CONTENTS

- What Does 'Isolated Installation' Mean?
- What Key Length Is Recommended?
- Is MD5 Still Secure?
- Is SHA-1 Still Secure?
- What is SHA-3?
- Which Standard Crytographic Hash Algorithms Are Secure?
- How Do You Generate A Code Signing Key?
- What OpenPGP User-ID Should I Choose For My Code Signing Key?
- What OpenPGP Comment Should I Choose For My Code Signing Key?
- What Is A Public Key Server?
- How Do You Upload A Key To A Public Key Server?
- How Can I Ensure My Local Web Of Trust Is Up To Date?
- How Do You Export A Key?
- What Is A Key ID?
- What Is A Sub Key?
- How Do I A Use Sub Key To Sign Emails?
- How Can I Find Out More?
- Is There A Quick Way To Sign Several Distributions?
- How Can I Transfer A Secret Key?
- Why Do Some People Have Two Keys?
- What Is A Transition Period (For Keys)?
- How Should I Transition From A Short To A Longer Key?
- I Have A New Key. Which Apache Documents Need To Be Updated?
- What Is RSA?
- How Do I Find The Length Of A Key?

HELP WANTED!

Help to finish this document by contributing documentation patches (infra-site.html) ! If th information you seek isn't in this document, then please submit a patch once the infrastruc have answered your question.

FURTHER READING

- The Guide (openpgp.html#gnupg) to using GnuPG at Apache
- The Apache Key Transition Guide (key-transition.html)
- RFC 1321 (http://www.ietf.org/rfc/rfc1321.txt) MD5 Message-Digest Algorithm
- RFC 2440 (http://www.ietf.org/rfc/rfc2440.txt) OpenPGP Message Format
- RFC 3174 (http://www.ietf.org/rfc/rfc3174.txt) Secure Hash Algorithm 1 (SHA1)
- The GNU Privacy Guard project documentation (http://www.gnupg.org/documentatio
- An introduction to PGP public key cryptography (http://www.pgpi.org/doc/pgpintro/)
- Applied Cryptography (http://www.schneier.com/book-applied.html) by Bruce Schnei

- Windows centric PGP FAQ (http://www.mccune.cc/PGPpage2.htm) by Tom McCune
- Henk Penning's Apache home page (http://home.apache.org/~henkp/)

Note this is not normative

BASIC FACTS

Every artifact distributed by the Apache Software Foundation *must* be accompanied by one
containing an OpenPGP compatible ASCII armored detached signature and another file con
MD5 checksum. The names of these files *must* be formed by adding to the name of the artif
following suffixes:

- the signature by suffixing `.asc`
- the checksum by suffixing `.md5`

An SHA checksum *should* also be created and *must* be suffixed `.sha` .

Release managers *must not* store private keys used to sign Apache releases on ASF hardwa

THE KEYS FILE

The KEYS file is a plain text file containing the public key signatures of the release manager
optionally other committers) for the project. For example, Apache Ant KEYS (http://www.ap
/dist/ant/KEYS). It is traditional to include the following header (which explains its usage):

```
This file contains the PGP keys of various developers.


Users: pgp < KEYS
or
       gpg --import KEYS


Developers:
    pgp -kxa <your name> and append it to this file.
or
    (pgpk -ll <your name> && pgpk -xa <your name>) >> this file.
or
    (gpg --list-sigs <your name>
    && gpg --armor --export <your name>) >> this file.
```

The commands above will generate a descriptive comment describing the key, followed by
itself. [Key handling software ignores the comments when importing a key file]

The KEYS file is stored alongside the release archives to which it applies, e.g. at the top leve
mirror area for the project. This is to ensure that it is available for download by users, and
automatically archived with historic releases.

For example, the Ant KEYS file is in the directory http://www.apache.org/dist/ant

(http://www.apache.org/dist/ant). The corresponding SVN area is at https://dist.apache.org/r
/dist/release/ant (https://dist.apache.org/repos/dist/release/ant).

Since the KEYS may be needed to check signatures for archived releases, it is important tha
that have ever been used to sign releases are retained in the file. Entries should only be adc
described above), not removed.

Note: this system will be replaced by a better process in the near future. In preparation, ple
that public keys are connected as strongly as possible to the Apache web of trust and are av
from the major public key servers.

Applied cryptography is a subject that has considerable depth. Luckily, it's possible to get st
signing releases without being an expert. Just remember that (from time to time) you will e
situations that will require research and learning. Hopefully the FAQ will be a reasonable f
call.

You will need an application to manage keys and create signatures. GNU Privacy Guard
(http://www.gnupg.org/) is recommended and the Apache documentation generally assume
what you're using. (Documentation patches for other tools welcomed.) Read the Apache usa
(openpgp.html#gnupg) and keep the manual (http://www.gnupg.org/gph/en/manual.html) h
that GnuPG can handle MD5 and SHA checksums as well as PGP signatures. It is your one-st
cross-platform tool for release signing and verification.

It can be hard for newbies to be confident that they have executed operations correctly. Cor
some practice first.

WHY WE SIGN RELEASES

A signature allows anyone to verify that a file is identical to the one created by the Apache r
manager. Using a signature:

- users can make sure that what they received has not been modified in any way, either
  accidentally via a faulty transmission channel, or intentionally (with or without malic
- the Apache infrastructure team can verify the identity of a file

OpenPGP signatures confer the usual advantages of digital signatures: authentication, integ
non-repudiation (http://www.pgpi.org/doc/pgpintro/#p12). MD5 and SHA checksums only p
integrity part as they are not encrypted.

SECURITY BASICS

- Protect your private key
- Choose a good passphrase
- Opt for a reasonably long key length

KEY BASICS

To sign releases, you need to generate a new master key-pair for code signing. Follow these instructions (openpgp.html#generate-key).

SIGNING BASICS

Signatures should be ASCII armored and detached.

Your public key should be exported and the result appended to the appropriate `KEYS` file(s

That's all you need to know to sign a release.

HOW DO I SIGN A RELEASE?

Create a OpenPGP compatible ASCII armored detached signature for the released artifact. U signature with the released artifact.

See here for a basic overview.

WHAT IS AN OPENPGP COMPATIBL ASCII ARMORED DETACHED SIGNATURE?

It is an OpenPGP compatible ASCII armored detached signature.

To create one using GNU Privacy Guard (http://www.gnupg.org) for file `foo.tar.gz` type:

```
$ gpg --armor --output foo.tar.gz.asc --detach-sig foo.tar.gz
```

WHAT IS OPENPGP?

OpenPGP (http://www.openpgp.org/) is an RFC (http://www.ietf.org/rfc/rfc2440.txt) describi for interoperable public key cryptography. Various implementations exist. Apache recomm *Privacy Guard*.

GNU Privacy Guard (http://www.gnupg.org/) (GnuPG) is an open source OpenPGP compatib implementation. It comes with good documentation (http://www.gnupg.org/(en)/documenta /guides.html) describing not just GnuPG but also giving a good general introduction to publ cryptography. A guide to using GnuPG at Apache is available (openpgp.html#gnupg).

WHAT IS PUBLIC KEY

CRYPTOGRAPHY?

Public key cryptography is asymmetric. One key can be used to encrypt a message which or other key can decrypt. Knowledge of the first key can be made public without compromisin security of the second key. One key is therefore called the public key and one the private ke

When using public key cryptography, the public key can be freely distributed but the privat be secret. It is vital that private key files are protected. Private keys are typically stored in f protected by symmetric encryption. Strong passphrases must be chosen to protect them.

WHAT IS AN DETACHED SIGNATUR

A digital signature is created from an original document using a private key. Possession of t corresponding public key allows verification that a given file is identical to the original doc attached signature is attached to the document whereas a detached signature is contained i separate file.

WHAT IS ASCII ARMORING?

An encoding format that converts a binary file into a string of ASCII characters. This forma human readable and more portable.

WHAT IS AN MD5 CHECKSUM?

MD5 is a well known (http://www.faqs.org/rfcs/rfc1321.html) message digest algorithm.

Many tools are available to calculate these sums. For example, OpenSSL (http://www.opens be used:

```
$ openssl md5 < file
```

Platform specific applications are also common. For example `md5sum` on linux:

```
$ md5sum file
```

With GnuPG:

```
  $ gpg --print-md MD5 [fileName] > [fileName].md5
```

Please run the command in the same directory as the file so the output only contains the fil with no directory prefixes.

Please note that the security of MD5 is now questionable and is only useful as part of a defe depth.

WHAT IS A SHA CHECKSUM?

Like MD5 , SHA (http://www.ietf.org/rfc/rfc3174.txt) is another message digest algorithm.

Using GnuPG, you can create a SHA1 signature as follows:

```
$ gpg --print-md SHA1 [fileName] > [fileName].sha
```

Please note that further use of `SHA-1` should be avoided.

`SHA256` and `SHA512` use the same `SHA` algorithm family with longer hash lengths (256 and 5
respectively). These longer variations are less vulnerable to the weaknesses found in the al
family than `SHA1` . SHA512 is recommended.

To create a `SHA512` checksum use:

```
$ gpg --print-md SHA512 [fileName] > [fileName].sha
```

Please run the command in the same directory as the file so the output only contains the fil
with no directory prefixes.

Other members of the `SHA` family (with different hash lengths) are rarely used.

This family of algorithms will be retired once SHA3 is available.

WHAT IS A MESSAGE DIGEST ALGORITHM?

A message digest algorithm takes a document and produces a much smaller hash of that do
good algorithm will produce different digests for very similar documents. A good algorithm
infeasible to create a message matching a given hash.

A trusted digest for a document can be used to verify the contents of an untrusted file. The
of the digest allows it to be delivered over a secure but expensive channel whilst the untrus
delivered over an insecure but inexpensive one. This is useful when distributing releases.

WHAT IS A WEB OF TRUST?

It is difficult to personally verify the identity of all useful public keys. However, having veri
identity of only a small number of public keys it is possible to deduce the identity of public
by the owners of these keys. This process can be repeated. This extended graph of trusted i
termed a web of trust (http://en.wikipedia.org/wiki/Web_of_trust).

Webs of trust can be used to solve the problem of verifying the identity of public keys.

Note: in order to take full advantage of a web of trust, it is important to actively build your
trust into the major public webs of trust. Conferences are an ideal opportunity but you mus

prepared.

For more information read Henk Penning's Apache home page (http://home.apache.org/~he
the GNU Privacy Guard User Guide (http://www.gnupg.org/(en)/documentation/guides.html

HOW DO I LINK INTO A PUBLIC WE
TRUST?

By an existing member of that web of trust signing your public key to verify your identity. S
Penning's Key Signing HOWTO (http://home.apache.org/~henkp/sig/pgp-key-signing.txt)

A short guide is available (openpgp.html#wot).

WHAT IS A KEY SIGNING PARTY?

A key signing party is a meeting organised to allow the exchange of public keys and so exte
web of trust.

See the Keysigning Party HOWTO (http://www.cryptnet.net/fdp/crypto/keysigning_party
/en/keysigning_party.html) and Henk Penning's Key Signing HOWTO (http://home.apache.o
/sig/pgp-key-signing.txt)

HOW CAN I LINK MY KEY INTO THE
APACHE WEB OF TRUST?

By meeting other Apache committers face-to-face and exchanging public keys. There are se
to achieve this:

- Key signing parties are organised at each ApacheCon (http://www.apachecon.com).
- If you are not able to attend (or the conference is a long way off) then consider organ
face-to-face meeting of local committers (http://people.apache.org/map.html).
- Subscribe to the `party` list and when you visit a new city, see if committers want to m

See Henk Penning's Key Signing HOWTO (http://home.apache.org/~henkp/sig/pgp-key-signi

WHAT DOES VERIFYING A SIGNATU
MEAN?

Public key cryptography can be used to test whether a particular file is identical (in content
original by verifying a signature. The signature file is a digest of the original file signed by a
key which attests to the digest's authenticity.

For example, when using GNU Privacy Guard (http://www.gnupg.org/) you verify the signat `foo-1.0.tar.gz.asc` for release `foo-1.0.tar.gz` using the following command:

```
$ gpg --verify foo-1.0.tar.gz.asc foo-1.0.tar.gz
```

Trust is required in the identity of the public key that made the signature and that the signa the original in question (and not some other file). When verifying a release from an untrust (for example, over P2P file sharing or from a mirror) it is therefore important to download signature from a trusted source. Signatures for all Apache releases are available directly fr `www.apache.org` and should be downloaded from there.

HOW CAN I CHECK THE INTEGRITY A RELEASE?

MD5 and SHA checksums provide a simple, means of verifying the integrity of a download. simply create a checksum (in the same way as the release manager) after download, and co result to the checksum downloaded from the main Apache site. Obviously, this process does provide for authentication and non-repudiation (http://www.pgpi.org/doc/pgpintro/#p12) as can create the same checksum.

The integrity of a release can also be checked by verifying the signature. More knowledge is to correctly interpret the result but it does provide authentication and non-repudiation. If y connected to the Apache web of trust then this also offers superior security.

WHAT DOES 'PUBLIC KEY NOT FOU MEAN (WHEN VERIFYING A SIGNATURE)?

Before a signature can be verified, the public key is required.

For example, when using GNU Privacy Guard (http://www.gnupg.org/) if you have never im appropriate public key a message similar to the following will be displayed:

```
$ gpg --verify foo-1.0.tar.gz.asc foo-1.0.tar.gz
gpg: Signature made Mon Sep 26 22:26:18 2005 BST using RSA key ID 00000000
gpg: Can't check signature: public key not found
```

Unknown keys can often be downloaded from public key servers. However, these should o trusted through a web of trust.

Apache projects normally keep the developers' public keys in a file called `KEYS` . You may be find that file on the project's website, or in their code repository. Use

```
$ gpg --import KEYS
```

to import the public keys.

WHAT IS A TRUSTED KEY?

OpenPGP uses a web of trust. The owner of a public key who trusts the identity of a second
mark this key as trusted by signing it. This has several major effects:

- In future, no untrusted key warning will be issued when a valid signature for this key
- Keys trusted by the owner of the key may also become trusted. In other words, a key v
  identity has been confirmed by the owner of a key whose identity you trust may be au
  trusted. This behavior is typically configurable.
- The next time you export your key, those who trust your key may start to trust the ide
  trusted key.

The transitive nature of the web of trust places a responsibility on the owner to verify the i
the owner of those keys marked as trusted.

For more information read Henk Penning's Apache home page (http://home.apache.org/~he
the GNU Privacy Guard User Guide (http://www.gnupg.org/(en)/documentation/guides.html

WHAT IS THE DIFFERENCE BETWEE VALID SIGNATURE FROM AN UNTRUSTED KEY AND AN INVALID SIGNATURE FROM AN UNTRUSTED KEY?

Trustfulness and validity are different concepts. You may elect to trust the identity of a key
degrees (or not at all). For a particular key, a particular signature for a particular file may b
other words, created by the private key from an identical file) or invalid (either corrupt or
from a different file).

You should not trust a file with an invalid signature. You can trust a file with a valid signatu
as you trust the identity of key that was used to verify the signature.

For example, when using GNU Privacy Guard (http://www.gnupg.org/) a message similar to
following indicates that the signature is invalid:

```
$ gpg --verify foo-1.0.tar.gz.asc foo-1.0.tar.gz
gpg: Signature made Mon Sep 26 22:26:18 2005 BST using RSA key ID 00000000
gpg: BAD signature from "someone@example.org"
```

whereas a message similar to the following indicates that the signature is valid but for an u
key:

```
$ gpg --verify foo-1.0.tar.gz.asc foo-1.0.tar.gz
gpg: Signature made Mon Sep 26 22:05:28 2005 BST using RSA key ID 00000000
gpg: Good signature from "someone@example.org"
gpg:                  aka "someone@anotherdomain.org"
gpg: checking the trustdb
gpg: checking at depth 0 signed=1 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: checking at depth 1 signed=0 ot(-/q/n/m/f/u)=1/0/0/0/0/0
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 00
```

The fingerprint may be used to ascertain the appropriate level of trust to assign to the key.

WHAT IS A PUBLIC KEY FINGERPRINT?

Public keys are long and even when ASCII armored are not very easy for humans to unders
compare. A fingerprint is a shorter digest of the key formatted in a way that makes it easier
humans to read and compare.

WHY INFEASIBLE AND NOT IMPOSSIBLE?

Responsible cryptography talks about infeasible cracks (rather than impossible ones) since
accurate. All current practical methods can be subjected to brute force attacks and so can b
So, a good question is whether attacks are feasible given the current state of the art.

WHERE SHOULD I CREATE THE SIGNATURES?

Creating signatures requires the private key. Limited copies should be kept of the private ke
must be kept confidential. Though the file used to store the private key is typically protecte
encryption, it is vulnerable to dictionary attacks on the passphrase. This file must therefore

secret. So, signatures should be created on the machine used to store the private key.

It is important that the private key is stored on secure hardware with limited read permissi
protected by a good passphrase. Consider using removable media or an isolated installatior

A master private key used to sign Apache artifacts (or to secure communications with the A
particularly valuable. If you want or need to be able to create signatures for other purposes
example, signing email messages) in other (less secure) locations it is recommended that yo
multiple sub keys and use sub keys for signing in less secure locations.

The private key *must not* be store on any ASF machine. So, signatures *must not* be created c
machines.

WHAT IS 'INSECURE MEMORY' AND SHOULD I BE WORRIED?

When you use GNU Privacy Guard (http://www.gnupg.org) you may see a warning similar t

```
gpg: WARNING: using insecure memory!
gpg: please see http://www.gnupg.org/faq.html for more information
```

If you are using GnuPG on Apache hardware, please read this. Sensitive operations using a
must not be executed on ASF hardware.

If you encounter this issue elsewhere then it indicates that GnuPG cannot lock memory pag
means they may be swapped out to disc. It would then be feasible for an attacker who had g
access to the machine to read the private key from the swap file. For more details, read the
(http://www.gnupg.org/faq.html).

WHAT IS A PASSPHRASE?

The term *passphrase* is often used in cryptography for what might be better known as a pas
other contexts. For example, an OpenPGP private key is typically stored to disc in an file en
a symmetric cypher keyed by a passphrase. This passphrase is one of the weakest elements
system: should anyone else gain access to the file then a dictionary attack will be feasible or
passphrase. So, choosing a strong passphrase is very important.

Passphrases are (unlike passwords) typically unlimited in length. Long passphrases are rec
This allows sequences of (at least seven) unrelated words to be used as well as more conver
mixtures of symbols and alphanumerics.

Note that even a good passphrase will offer only limited protection. Given the encrypted fil
enough time, a determined cracker will be able to break any passphrase. A good passphrase
important time in the event of a compromise but is no substitute for keeping the private ke
secure in the first place.

WHAT IS A REVOCATION CERTIFICATE?

OpenPGP defines a special type of signed message called a *revocation certificate*. This mess indicates that the signer believes that the key is no longer trustworthy. Typically, the revoca certificate will be signed by the key to be revoked (though the key may specify that other ke be trusted for revocation). The type of revocation and the comment included may be used t how much trust to place in a good signature by a revoked key.

A revocation certificate should be generated for each public key used. These should be stor securely and separately from the public key.

Each revocation certificates has a type specifying a general (machine readable) reason for t revocation:

- No reason specified
- Key has been compromised
- Key is superseded
- Key is no longer used

It is recommended that certificates are created to cover the first two cases. Note that if a ke can no longer be accessed (due to media failure, say) it is best to assume that the key has be potentially compromised. It is recommended that revocation certificates are printed and st to guard against media failure.

An ASCII armored revocation certificate for key `bob` can be generated and saved to `revoke` GNU Privacy Guard (http://www.gnupg.org) as follows:

```
$ gpg --output revoke.asc --armor --gen-revoke bob
```

The certificate produced should be securely stored.

If you are preparing a revocation certificate for future use, it is recommended that you test practice.

HOW DO I REVOKE A KEY?

To revoke a key (given a certificate) using GNU Privacy Guard (http://www.gnupg.org) impo revocation certificate :

```
$ gpg --import revoke.asc
gpg: key 4A03679A: "Some User <someuser@example.org>" revocation
certificate imported
gpg: Total number processed: 1
gpg:    new key revocations: 1
```

WHERE SHOULD A REVOCATION CERTIFICATE BE STORED?

The revocation certificate should be stored securely and separately from the key it revokes. onto CDROM or printing out onto hard copy are good solutions.

HOW DO I DISTRIBUTE A REVOCAT CERTIFICATE?

In the event of a compromise, a revocation certificate needs to be distributed to those using This process needs to be a mirror of the process by which the original key was distributed.

- The Apache infrastructure team should be informed by a post containing the revocati certificate
- The KEYS files containing the original key should be updated with the revocation cert
- The revocation certificate should be uploaded to the major keyserver networks
- An announcement should be posted to the appropriate lists with the revocation certif attached

WHAT IS THE DIFFERENCE BETWEE DELETING AND REVOKING A KEY?

When a key is deleted from a keyring, it is simply removed. It can be added again later.

When a key is revoked, the key is marked in the key ring. Whenever a message signed by th verified in the future, the user will be warned that the key has been revoked.

For example, when verifying a revoked key, GNU Privacy Guard (http://www.gnupg.org) iss following comment:

```
$ gpg --verify message.asc.message
gpg: Signature made Sat Apr  8 09:28:31 2006 BST using DSA key ID 4A03679A
gpg: Good signature from "Some User <someuser@example.org>"
gpg: checking the trustdb
gpg: checking at depth 0 signed=0 ot(-/q/n/m/f/u)=0/0/0/0/0/1
gpg: WARNING: This key has been revoked by its owner!
gpg:          This could mean that the signature is forgery.
gpg: reason for revocation: Key has been compromised
gpg: revocation comment:
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: 82D1 169B E6F1 9D14 DA76  A5DD 968E 66E4 4A03 679A
```

CAN I MARK A KEY AS LOCALLY TRUSTED?

On occasion, a key may be trusted by the user (who understands the risks) but is not consid trustworthy enough to be exported to the web of trust.

OpenPGP allows keys to be signed as local only. These trust relationships will not be export public web of trust but will be treated as trusted when the key ring is used locally.

For example, with GNU Privacy Guard (http://www.gnupg.org) use:

```
$ gpg --lsign-key someuser
```

HOW CAN I SAFELY PRACTICE USIN OPENPGP?

Use separate environments each with a different practice keyring.

For example, using GNU Privacy Guard (http://www.gnupg.org) :

- *(First time only)* Create a directory to contain the keyring
- Open the shell to be configured to use this keyring
- Change to the directory
- *(First time only)* `$ mkdir -m 700.gnupg`
- Setup environment `$ export GNUPGHOME=.gnupg`

WHAT IS THE DIFFERENCE BETWEI

PUBLIC AND A PRIVATE KEY?

A public key is used for verifying signatures and encrypting messages, a private key for gen signatures and decrypting messages. Public keys can be freely distributed safely whereas p must be kept protected. More details here.

HOW SHOULD MY CODE SIGNING PRIVATE KEY BE PROTECTED?

Anyone who possesses a copy of a private key used to sign releases can create doctored rele valid signatures. If this person intends harm then the consequences could be serious indee

It is therefore very important that this private key is kept secret.

- Only sign releases on a secure machine.
- Keep your signing application fully patched.
- Keep the key file safe and secret.
- Choose a good passphrase.

HOW SECURE DOES THE MACHINE USED TO SIGN RELEASES NEED TO BE?

If the code signing machine is owned (http://www.catb.org/~esr/jargon/html/O/owned.html) only a matter of time before the key is compromised.

At a minimum, the machine should well maintained (kept up to date with security patches, appropriate anti-virus and firewall software). The ideal is an isolated, well maintained inst used only for creating releases. This can be achieved with a little effort by creating an isola installation on a separate hard disc (which is physically disconnected when not in use signi or a live CD.

WHICH APPLICATIONS CREATE OPENPGP COMPATIBLE SIGNATURES?

There are many applications available (some commercial, some freeware, some software li

Whichever one you choose, please subscribe to the appropriate security lists and keep the a
fully patched.

Here are some used by ASF release managers:

- **Recommended** GNU Privacy Guard (http://www.gnupg.org) *Software Libre*

HOW SAFE DOES THE PRIVATE KEY NEED TO BE?

It is vital that the private key is kept safe and secure. Though the file is encrypted using a pa
given enough time any determined cracker will be able to break that encryption. It is there
essential that the private key file is kept safe and secure.

Basic precautions should include ensuring that the directories are readable only by the use

However, it is recommended that for code signing keys additional measures are taken. The
opportunity can be reduced by either using a isolated installation or by storing the private k
removable media (which should be removed to secure storage when not being used to sign

WHAT DOES 'ISOLATED INSTALLATION' MEAN?

An installation which is inaccessible when not being used to sign releases. For example, cre
installation on a separate hard disc or use a live CD.

WHAT KEY LENGTH IS RECOMMENDED?

There is no good, simple answer to this question.

The number of operations required to break a key by brute force increases with key size. H
cost of using the key also rises. So, the planned usage of the key must be a consideration. Ke
code signing will only be used rarely and in situations where performance is not the main c
This is a reason to err in favor of long key lengths.

Over time, the practice cost of attacking a key (of a given length) by brute force falls as com
power increases. So, a key whose length seems adequate today may be seem too short in a f
time. This is a significant issue for long-lived keys such as those used to sign ASF releases. A
a reason to err in the favor of longer key lengths.

Now that there is doubt about the medium term security of SHA-1 , the DSA keys and 1024 b
(which depend on this algorithm) should be avoided for new keys. It is uncertain whether 2
keys will be strong enough to remain secure until SHA3 (and the next generation of standar

It is therefore recommended that new keys should be at least 4096 bit RSA (the longest wide supported key length).

IS MD5 STILL SECURE?

There is no good, simple answer to this question.

Though feasible collision attacks are known, they are still computationally expensive. MD5 useful as an additional layer in a defense in depth. Not recommended.

IS SHA-1 STILL SECURE?

There is no good, simple answer to this question.

Research has revealed weaknesses in this algorithm. Though there are no practical attacks the time of writing, experiences with similar weaknesses in MD5 suggest that efforts should now to move away from this algorithm.

Breaking the longest members of this family (`SHA512` and `SHA256`) is still considered infeas SHA-3 is available, new uses of SHA-1 should be avoided and SHA512 (or SHA256) used inst

Configuration instructions for GnuPG can be found here (openpgp.html#sha1).

WHAT IS SHA-3?

SHA-3 is the designation for a new cryptographic hash algorithm to replace the SHA family. competition (http://csrc.nist.gov/groups/ST/hash/sha-3/index.html) has selected Keccak. The standard has not been released yet. Once the full standard is released work will need to be permit it to be used with OpenPGP.

WHICH STANDARD CRYTOGRAPHIC HASH ALGORITHMS ARE SECURE?

There is no good, simple answer to this question.

Feasible - though expensive - attacks on MD5 have been made public. Similar weaknesses h found in the SHA family of hashes though practical attacks are not yet publically known. H longer hash sizes offer considerable protection. So, larger members of the SHA family still l to be secure enough for a number of years.

RIPE has not been as well studied as either SHA or MD5. It is likely to offer no more securit

SHA512 is the strongest well studied widely used cryptographic hash. It is therefore the bes recommendation until SHA3 is available.

HOW DO YOU GENERATE A CODE SIGNING KEY?

The exact mechanics are application dependent. For GnuPG (recommended), follow the str generation instructions (openpgp.html#generate-key). Please think about the right key leng configure the tool to avoid SHA-1 and choose a good passphrase. Use the recommended id a comment.

WHAT OPENPGP USER-ID SHOULD CHOOSE FOR MY CODE SIGNING KI

It is recommended that your Apache email address is used as the primary `User-ID` for the key. For example, `rdonkin@apache.org` .

WHAT OPENPGP COMMENT SHOUI CHOOSE FOR MY CODE SIGNING KI

It is recommended that the comment includes *CODE SIGNING KEY*. This makes clear the pri for this key. This can be helpful if you later generate keys for other uses.

It is recommended that keys generated for other purposes include in the comment *NOT FO SIGNING*.

WHAT IS A PUBLIC KEY SERVER?

A public key server manages public keys. Available functions may vary but typically includ search and download.

Public key servers exist to distribute public keys. They do not vouch for the actual identity owner of each key. This must be established either directly or through a web of trust. Do nc key just because it has been downloaded from a key server.

The major public key servers synchronize their records regularly so a key uploaded to one disseminated to the rest.

Some well known public key servers:

- MIT (http://pgp.mit.edu)
- SKS OpenPGP Public Key Server (http://pgp.surfnet.nl/)

Although the following is also well-known, at present it does not synchronise keys with oth servers, so should not be used as the only public server for your key:

- PGP Global Directory (http://keyserver.pgp.com/)

HOW DO YOU UPLOAD A KEY TO A PUBLIC KEY SERVER?

There are two common ways to upload a key to a public key server :

- Most key servers allow exports to be uploaded through the website
- By using automatic facilities built into most OpenPGP implementations

For example using GNU Privacy Guard (http://www.gnupg.org) , the key with ID B1313DE2 c
exported to the default public key server by:

```
$ gpg --send-key B13131DE2
```

Note that each changed key must be exported separately.

HOW CAN I ENSURE MY LOCAL WEI TRUST IS UP TO DATE?

The public web of trust grows constantly as people sign new keys and upload the new signa
the network of public key servers. Public keys should be periodically refreshed to ensure th
local web of trust is as full as possible. Many OpenPGP clients allow keys to be easily refresl
querying a public key server.

For example, to refresh all keys using GNU Privacy Guard (http://www.gnupg.org) use:

```
$ gpg --refresh-keys
```

HOW DO YOU EXPORT A KEY?

A public key can be exported using OpenPGP (http://www.gnupg.org) by using `--export` . Ty
export should be ASCII armored. For example, to export all public keys to the command lin

```
gpg --export --armor
```

In most cases, it is better to export all keys - this ensures that signatures made on other keys
exported. However, it is possible to export just one key by specifying it on the command lin

Secret keys can also be exported. However, exporting secret keys poses a security risk and t
better solutions for most common use cases. For example, copying the `GNUPGHOME` directory
`~/.gnupg`) is a better way to transfer an OpenPGP (http://www.gnupg.org) keyring from on
to another.

WHAT IS A KEY ID?

A key ID is similar to a fingerprint but is much smaller in length. There is no guarantee that unique. Consequently, it is strongly recommended that the fingerprint is checked before sig The key ID is typically used for locating keys and identifying keys already contained within keyring. For these use cases, key ID should be unique enough in practice.

A short guide to discovering the key ID for a key is available (openpgp.html#find-key-id).

WHAT IS A SUB KEY?

Each OpenPGP keyring has a single master key. This key is signing only. It may also optiona number of sub keys (for encryption and signing).

If you wish to sign emails using a key related to that used to sign code, it is recommended th signing sub key is used.

HOW DO I A USE SUB KEY TO SIGN EMAILS?

To keep a code signing key safe and secure it is recommended that the key is not kept on a h a regular development machine. This means that the master key should not be used directly emails. However, there are occasions when digitally signed emails are desirable.

The recommended approach is to create a sub key for email signing and export it to the reg machine. The master key can then be kept safely offline. For more details, read:

- Subkey cross certification (http://www.gnupg.org/(en)/faq/subkey-cross-certify.html)
- Signing Subkey HOWTO (http://fortytwo.ch/gpg/subkeys)

Note that some public key servers do not handle sub keys correctly. It may be necessary to u the SKS (http://www.nongnu.org/sks/) network.

HOW CAN I FIND OUT MORE?

See this.

IS THERE A QUICK WAY TO SIGN SEVERAL DISTRIBUTIONS?

The private `https://svn.apache.org/repos/private/committers` repository contains scripts th with batch signing several distributions.

HOW CAN I TRANSFER A SECRET KEY?

This is application dependent. Instructions for GnuPG are available (openpgp.html#secret-key-transfer).

WHY DO SOME PEOPLE HAVE TWO KEYS?

When switching from an uncompromised key to another (typically stronger) one, it is conv use a transition period. During a transition, both keys are trustworthy but only (the newer) actively used to sign documents and certify links in the web of trust.

WHAT IS A TRANSITION PERIOD (F KEYS)?

When replacing one uncompromised key with a newer (typically longer) one, a transition p where both keys are trustworthy and participate in the web of trust allows - by trust transit to the old key to be used to trust signatures and links created by the new key. During a tran keys are trustworthy but only (the newer) one is actively used to sign documents and certif the web of trust.

HOW SHOULD I TRANSITION FROM SHORT TO A LONGER KEY?

If you have a short but uncompromised key and would like to transition to a longer one, fol instructions (key-transition.html).

If your key has been compromised then you **MUST NOT** transition but revoke the old key a with a new one immediately. **DO NOT** use a transition period.

I HAVE A NEW KEY. WHICH APACHE DOCUMENTS NEED TO BE UPDATE

A number of Apache documents need to be updated. Follow these instructions (openpgp.ht

WHAT IS RSA?

RSA is a well known public key cryptography algorithm which supports signing and encryp

See further reading for more details.

HOW DO I FIND THE LENGTH OF A KEY?

The easiest way to discover the length of a key (with id `KEYID`) is to use `gpg --list-keys KE` will print basic information about the key. The first line will include the size in the second c before the id.

For example:

```
$ gpg --list-keys B1313DE2
pub   1024D/B1313DE2 2003-01-15
uid                  Robert Burrell Donkin (CODE SIGNING KEY) <rdonkin@apache.
uid                  Robert Burrell Donkin <robertburrelldonkin@gmail.com>
uid                  Robert Burrell Donkin <robertburrelldonkin@blueyonder.co.
sub   4096R/40A882CB 2009-06-18 [expires: 2010-06-18]


$ gpg --list-keys A6EE6908
pub   8192R/A6EE6908 2009-08-07
uid                  Robert Burrell Donkin (CODE SIGNING KEY) <rdonkin@apache.
sub   8192R/B800EFC1 2009-08-07
```

shows that key B1313DE2 has length 1024 and A6EE6908 length 8192.

The Apache Way (/foundation/how-it-works.html)
Travel Assistance (/travel/)
Get Involved (/foundation/getinvolved.html)
Community FAQ (http://community.apache.org/newbiefaq.html)
(/foundation/marks/)
Contacts (/foundation/contact.html)
Policy (/foundation/policies/privacy.html)
Export Information (/licenses/exports/)
License/Distribution FAQ (/foundation/license-faq.html)
Contacts (/foundation/contact.html)