SHA-1 and OpenPGP/GnuPG

Christian Aistleitner
christian@quelltextlich.at

quelltextlich e.U.
Linz, Austria

21st October 2012, Gentoo Miniconf

Outline

Outline

SHA-1

SHA-1

- hash function turning messages into 160 bit digest
  (e.g.: "`Gentoo Miniconf yay!`" → "`4626c7881221e52e521c11b840a04131681cc321`")
- *cryptographic* hash function
  (e.g.: "`Gentoo Miniconf nay!`" → "`85f6867078cbf1f5a8cabf0d66ad8d2280bb597e`")
- easy to implement
- widely used (e.g.: OpenPGP, GnuPG, git)
- published 1995 in FIPS PUB 180-1

SHA-1

SHA-1

- hash function turning messages into 160 bit digest
  (e.g.: "`Gentoo Miniconf yay!`" → "`4626c7881221e52e521c11b840a04131681cc321`")
- *cryptographic* hash function
  (e.g.: "`Gentoo Miniconf nay!`" → "`85f6867078cbf1f5a8cabf0d66ad8d2280bb597e`")
- easy to implement
- widely used (e.g.: OpenPGP, GnuPG, git)
- published 1995 in FIPS PUB 180-1

SHA-1

SHA-1

- hash function turning messages into 160 bit digest
  (e.g.: "`Gentoo Miniconf yay!`" → "`4626c7881221e52e521c11b840a04131681cc321`")
- *cryptographic* hash function
  (e.g.: "`Gentoo Miniconf nay!`" → "`85f6867078cbf1f5a8cabf0d66ad8d2280bb597e`")
- easy to implement
- widely used (e.g.: OpenPGP, GnuPG, git)
- published 1995 in FIPS PUB 180-1

SHA-1

SHA-1

- hash function turning messages into 160 bit digest
  (e.g.: "`Gentoo Miniconf yay!`" → "`4626c7881221e52e521c11b840a04131681cc321`")
- *cryptographic* hash function
  (e.g.: "`Gentoo Miniconf nay!`" → "`85f6867078cbf1f5a8cabf0d66ad8d2280bb597e`")
- easy to implement
- widely used (e.g.: OpenPGP, GnuPG, git)
- published 1995 in FIPS PUB 180-1

SHA-1

SHA-1

- hash function turning messages into 160 bit digest
  (e.g.: "`Gentoo Miniconf yay!`" → "`4626c7881221e52e521c11b840a04131681cc321`")
- *cryptographic* hash function
  (e.g.: "`Gentoo Miniconf nay!`" → "`85f6867078cbf1f5a8cabf0d66ad8d2280bb597e`")
- easy to implement
- widely used (e.g.: OpenPGP, GnuPG, git)
- published 1995 in FIPS PUB 180-1

SHA-1 is broken!

SHA-1 has been broken. Not a reduced-round version. Not a simplified version. The real thing.

Bruce Schneier (2005)

Use of SHA-1 should now be avoided.

Apache Software Foundation (At least since 2009)

Due to weaknesses found with the SHA1 hashing algorithm Debian prefers to use keys that are at least 2048 bits and preferring SHA2.

Debian (At least since 2009)

> Although SHA1 is considered to be broken by some, [...]
That is plain nonsense.

Werner Koch (2012)

SHA-1 is broken!

SHA-1 has been broken. Not a reduced-round version. Not a simplified version. The real thing.

Bruce Schneier (2005)

Use of SHA-1 should now be avoided.

Apache Software Foundation (At least since 2009)

Due to weaknesses found with the SHA1 hashing algorithm Debian prefers to use keys that are at least 2048 bits and preferring SHA2.

Debian (At least since 2009)

> Although SHA1 is considered to be broken by some, [...]
That is plain nonsense.

Werner Koch (2012)

SHA-1 is broken!

SHA-1 has been broken. Not a reduced-round version. Not a simplified version. The real thing.

Bruce Schneier (2005)

Use of SHA-1 should now be avoided.

Apache Software Foundation (At least since 2009)

Due to weaknesses found with the SHA1 hashing algorithm Debian prefers to use keys that are at least 2048 bits and preferring SHA2.

Debian (At least since 2009)

> Although SHA1 is considered to be broken by some, [...]
That is plain nonsense.

Werner Koch (2012)

SHA-1 is broken?

SHA-1 has been broken. Not a reduced-round version. Not a simplified version. The real thing.

Bruce Schneier (2005)

Use of SHA-1 should now be avoided.

Apache Software Foundation (At least since 2009)

Due to weaknesses found with the SHA1 hashing algorithm Debian prefers to use keys that are at least 2048 bits and preferring SHA2.

Debian (At least since 2009)

> Although SHA1 is considered to be broken by some, [...]
That is plain nonsense.

Werner Koch (2012)

SHA-1: Status quo (full 80 rounds)

Complexity (compressions)	Estimate (years)[1]	Type	Source	Year
2^{160}	$2.0 \cdot 10^{31}$	preimage	(bruteforce)	1995
2^{80}	17000000	collision	(bruteforce)	1995
(avg.) 2^{77}	2000000	chosen pfx	Stevens	2012
2^{69}	8100	identical pfx	Wang, Yin, Yu	2005
$2^{60.3}$ to $2^{65.3}$	20 to 630	identical pfx	Stevens	2012

[1] using 2010 measurement of 2300 million compressions/second on ATI HD 5970.

> It's time to walk, but not run, to the fire exits. You don't see smoke, but the fire alarms have gone off.
>
> Jon Callas ($\leq$2005), quoted by Bruce Schneier

SHA-1: Status quo (full 80 rounds)

Complexity (compressions)	Estimate (years)[1]	Type	Source	Year
2^{160}	$2.0 \cdot 10^{31}$	preimage	(bruteforce)	1995
2^{80}	17000000	collision	(bruteforce)	1995
(avg.) 2^{77}	2000000	chosen pfx	Stevens	2012
2^{69}	8100	identical pfx	Wang, Yin, Yu	2005
$2^{60.3}$ to $2^{65.3}$	20 to 630	identical pfx	Stevens	2012

[1] using 2010 measurement of 2300 million compressions/second on ATI HD 5970.

It's time to walk, but not run, to the fire exits. You don't see smoke, but the fire alarms have gone off.

Jon Callas ($\leq$2005), quoted by Bruce Schneier

Outline

OpenPGP

- IETF RFC 4880 “OpenPGP Message Format” (STD1) (November 2007)
- Implementations using OpenPGP: GnuPG, PGP, Hushmail
- Transferred data is “OpenPGP message”.
- “OpenPGP message” is sequence of “packet”s
- “packet” is for example
  - Single Public key
  - Single Secret key
  - Single Signature
  - Literal Data
  - Encrypted Data

OpenPGP and hardwired SHA-1

Section	Name	Problematic?
§5.5.3.	Secret-Key Packet Formats (usage 254)	encrypted
§5.14.	Modification Detection Code Packet	encrypted
§12.2.	Key IDs and Fingerprints	?

OpenPGP's packets with selectable hash functions

Section	Name
§3.7.1.1.	Simple S2K
§3.7.1.2.	Salted S2K
§3.7.1.3.	Iterated and Salted S2K
§5.2.2.	Version 3 Signature Packet Format
§5.2.3.	Version 4 Signature Packet Format
§5.2.3.8.	Preferred Hash Algorithms
§5.2.3.25.	Signature Target
§5.4.	One-Pass Signature Packets

$\rightarrow$ one octet algorithm id allows to select used algorithm

Outline

SHA2 for GnuPG primer

- Add the following to your gpg.conf

```
s2k-digest-algo SHA512
digest-algo SHA512
default-preference-list SHA512 SHA384 SHA256
   SHA224 SHA1 AES256 AES192 AES CAST5 ZLIB
   BZIP2 ZIP Uncompressed
cert-digest-algo SHA512
```

- New keys
  - Just create them as usual. They are defaulting to SHA512 now.
- Existing keys
  - passwd your keys
  - setpref your keys
  - re-sign your keys

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	?
§3.7.1.2.	Salted S2K	?
§3.7.1.3.	Iterated and Salted S2K	?
§5.2.2.	Version 3 Signature Packet Format	?
§5.2.3.	Version 4 Signature Packet Format	?
§5.2.3.8.	Preferred Hash Algorithms	?
§5.2.3.25.	Signature Target	?
§5.4.	One-Pass Signature Packets	?

- one octet algorithm id allows to select used algorithm
  - 2 $\rightarrow$ SHA-1
  - 10 $\rightarrow$ SHA-512

Generating key

```
$ gpg --gen-key
gpg (GnuPG) 2.0.19; Copyright (C) 2012 Free Software Foundation, Inc.
[...]
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2012-10-18
pub   2048R/DA6AD20A 2012-10-17 [expires: 2012-10-18]
      Key fingerprint = 2FB1 8AD2 9D58 307C 51E6  CDF3 9A82 03DD[...]
uid                  John Doe
sub   2048R/32EA7448 2012-10-17 [expires: 2012-10-18]
```

String-to-key (I)

```
$ gpg --export-secret-keys 0xDA6AD20A | gpg --list-packets
:secret key packet:
[...]
        iter+salt S2K, algo: 3, SHA1 protection, hash: 2, salt: [...]
[...]
:secret sub key packet:
[...]
        iter+salt S2K, algo: 3, SHA1 protection, hash: 2, salt: [...]
[...]
```

String-to-key (II)

```
$ echo "s2k-digest-algo SHA512" >>~/.gnupg/gpg.conf
$ gpg --edit-key 0xDA6AD20A
[...]
gpg> passwd
Key is protected.

You need a passphrase to unlock the secret key for
[...]
gpg> save
$ gpg --export-secret-keys 0xDA6AD20A | gpg --list-packets
:secret key packet:
[...]
        iter+salt S2K, algo: 3, SHA1 protection, hash: 10, salt:[...]
[...]
:secret sub key packet:
[...]
        iter+salt S2K, algo: 3, SHA1 protection, hash: 10, salt:[...]
[...]
```

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	s2k-digest-algo + passwd
§3.7.1.2.	Salted S2K	s2k-digest-algo + passwd
§3.7.1.3.	Iterated and Salted S2K	s2k-digest-algo + passwd
§5.2.2.	Version 3 Signature Packet Format	?
§5.2.3.	Version 4 Signature Packet Format	?
§5.2.3.8.	Preferred Hash Algorithms	?
§5.2.3.25.	Signature Target	?
§5.4.	One-Pass Signature Packets	?

Signing

```
$ echo "abc" | gpg --default-key 0xDA6AD20A --sign | gpg --list-packets

You need a passphrase to unlock the secret key for
user: "John Doe"
2048-bit RSA key, ID DA6AD20A, created 2012-10-17

[...]
:compressed packet: algo=1
:onepass_sig packet: keyid 9A8203DDDA6AD20A
        version 3, sigclass 0x00, digest 2, pubkey 1, last=1
[...]
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
[...]
        digest algo 2, begin of digest dc cb
[...]
```

Signing

```
$ echo "digest-algo SHA512" >>~/.gnupg/gpg.conf
$ echo "abc" | gpg --default-key 0xDA6AD20A --sign | gpg --list-packets

You need a passphrase to unlock the secret key for
user: "John Doe"
2048-bit RSA key, ID DA6AD20A, created 2012-10-17

[...]
:compressed packet: algo=1
:onepass_sig packet: keyid 9A8203DDDA6AD20A
        version 3, sigclass 0x00, digest 10, pubkey 1, last=1
[...]
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
        version 4, created 1350496299, md5len 0, sigclass 0x00
        digest algo 10, begin of digest b1 40
[...]
```

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	s2k-digest-algo + passwd
§3.7.1.2.	Salted S2K	s2k-digest-algo + passwd
§3.7.1.3.	Iterated and Salted S2K	s2k-digest-algo + passwd
§5.2.2.	Version 3 Signature Packet Format	?
§5.2.3.	Version 4 Signature Packet Format	digest-algo
§5.2.3.8.	Preferred Hash Algorithms	?
§5.2.3.25.	Signature Target	?
§5.4.	One-Pass Signature Packets	*automatic*

Signing

```
$ echo "abc" | gpg --default-key 0xDA6AD20A --sign --force-v3-sigs \
  | gpg --list-packets

You need a passphrase to unlock the secret key for
user: "John Doe"
2048-bit RSA key, ID DA6AD20A, created 2012-10-17

[...]
:compressed packet: algo=1
:onepass_sig packet: keyid 9A8203DDDA6AD20A
        version 3, sigclass 0x00, digest 10, pubkey 1, last=1
[...]
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
        version 3, created 1350496570, md5len 5, sigclass 0x00
        digest algo 10, begin of digest aa 92
[...]
```

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	s2k-digest-algo + passwd
§3.7.1.2.	Salted S2K	s2k-digest-algo + passwd
§3.7.1.3.	Iterated and Salted S2K	s2k-digest-algo + passwd
§5.2.2.	Version 3 Signature Packet Format	digest-algo
§5.2.3.	Version 4 Signature Packet Format	digest-algo
§5.2.3.8.	Preferred Hash Algorithms	?
§5.2.3.25.	Signature Target	?
§5.4.	One-Pass Signature Packets	*automatic*

Preferred hash algorithms

```
$ gpg --export 0xDA6AD20A | gpg --list-packets
:public key packet:
[...]
:user ID packet: "John Doe"
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
[...]
        hashed subpkt 21 len 5 (pref-hash-algos: 8 2 9 10 11)
[...]
```

Further Ids:

- 8 → SHA256
- 9 → SHA384
- 11 → SHA224

Preferred hash algorithms (II)

```
$ echo "default-preference-list SHA512 SHA384 SHA256 SHA224 SHA1 "\
"AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP Uncompressed" \
>>~/.gnupg/gpg.conf
$ gpg --edit-key 0xDA6AD20A
[...]
gpg> setpref
Set preference list to:
     Cipher: AES256, AES192, AES, CAST5, 3DES
     Digest: SHA512, SHA384, SHA256, SHA224, SHA1
     Compression: ZLIB, BZIP2, ZIP, Uncompressed
     Features: MDC, Keyserver no-modify
Really update the preferences? (y/N) y
[...]
gpg> save
```

Preferred hash algorithms (III)

```
$ gpg --export 0xDA6AD20A | gpg --list-packets
:public key packet:
[...]
:user ID packet: "John Doe"
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
[...]
        hashed subpkt 21 len 5 (pref-hash-algos: 10 9 8 11 2)
[...]
```

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	s2k-digest-algo + passwd
§3.7.1.2.	Salted S2K	s2k-digest-algo + passwd
§3.7.1.3.	Iterated and Salted S2K	s2k-digest-algo + passwd
§5.2.2.	Version 3 Signature Packet Format	digest-algo
§5.2.3.	Version 4 Signature Packet Format	digest-algo
§5.2.3.8.	Preferred Hash Algorithms	default-preference-list + setpref
§5.2.3.25.	Signature Target	?
§5.4.	One-Pass Signature Packets	*automatic*

Signature Target

- OpenPGP:

```
29 = Reason for Revocation
30 = Features
31 = Signature Target
32 = Embedded Signature
```

- GnuPG (common/openpgpdefs.h):

```
SIGSUBPKT_REVOC_REASON  = 29, /* Reason for revocation. */
SIGSUBPKT_FEATURES      = 30, /* Feature flags. */

SIGSUBPKT_SIGNATURE     = 32, /* Embedded signature. */
```

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	s2k-digest-algo + passwd
§3.7.1.2.	Salted S2K	s2k-digest-algo + passwd
§3.7.1.3.	Iterated and Salted S2K	s2k-digest-algo + passwd
§5.2.2.	Version 3 Signature Packet Format	digest-algo
§5.2.3.	Version 4 Signature Packet Format	digest-algo
§5.2.3.8.	Preferred Hash Algorithms	default-preference-list + setpref
§5.2.3.25.	Signature Target	*not implemented*
§5.4.	One-Pass Signature Packets	*automatic*

Cheater!

```
$ gpg --export 0xDA6AD20A | gpg --list-packets
:public key packet:
[...]
:user ID packet: "John Doe"
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
        version 4, created 1350497669, md5len 0, sigclass 0x13
        digest algo 2, begin of digest 6a c6
:public sub key packet:
[...]
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
        version 4, created 1350493810, md5len 0, sigclass 0x18
        digest algo 2, begin of digest 54 65
```

Cheater! (II)

```
$ echo "cert-digest-algo SHA512" >>~/.gnupg/gpg.conf
$ gpg --edit-key 0xDA6AD20A
[...]
gpg> setpref
[...]
gpg> key 1
[...]
gpg> expire
[set expiration time again.
 For this to update the signature's algorithm, use patches from
 gnupg-dev mailinglist]
gpg> save
```

Cheater! (III)

```
$ gpg --export 0xDA6AD20A | gpg --list-packets
:public key packet:
[...]
:user ID packet: "John Doe"
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
        version 4, created 1350500867, md5len 0, sigclass 0x13
        digest algo 10, begin of digest 9c 74
[...]
:public sub key packet:
[...]
:signature packet: algo 1, keyid 9A8203DDDA6AD20A
        version 4, created 1350501179, md5len 0, sigclass 0x18
        digest algo 10, begin of digest 09 b9
[...]
```

OpenPGP's packets with selectable hash functions

Section	Name	For GnuPG
§3.7.1.1.	Simple S2K	s2k-digest-algo + passwd
§3.7.1.2.	Salted S2K	s2k-digest-algo + passwd
§3.7.1.3.	Iterated and Salted S2K	s2k-digest-algo + passwd
§5.2.2.	Version 3 Signature Packet Format	{cert-,}digest-algo
§5.2.3.	Version 4 Signature Packet Format	{cert-,}digest-algo
§5.2.3.8.	Preferred Hash Algorithms	default-preference-list + setpref
§5.2.3.25.	Signature Target	*not implemented*
§5.4.	One-Pass Signature Packets	*automatic*

- Update existing signatures
  - key: setpref (requires to apply patch)
    or kill sigs and re-sign
  - sub key: expire (requires to apply the same patch)
  - other's keys/user-ids: re-sign

OpenPGP §9.4. Hash algorithms

ID	Algorithm	Text Name
1	MD5	MD5
2	SHA-1	SHA1
3	RIPE-MD/160	RIPEMD160
4	Reserved	
5	Reserved	
6	Reserved	
7	Reserved	
8	SHA256	SHA256
9	SHA384	SHA384
10	SHA512	SHA512
11	SHA224	SHA224
100 to 110	Private/Experimental algorithm	

Implementations MUST implement SHA-1. Implementations MAY implement other algorithms. MD5 is deprecated.

SHA2 keys' interoperability (last three years' releases)

	Version	works?	Available in
	master (v1)	yes	
	v1.4.12	yes	Gentoo, Debian wheezy
	v1.4.11	yes	Ubuntu natty–quantal
2009-09-02	v1.4.10	yes	Debian squeeze, Ubuntu lucid
	master (v2)	yes	
	v2.0.19	yes	Gentoo, Arch, openSUSE 12.2, Debian wheezy
	v2.0.18	yes	openSUSE 12.1
	v2.0.17	yes	Ubuntu oneiric–quantal
	v2.0.16	yes	openSUSE 11.4
	v2.0.15	yes	
	v2.0.14	yes	Ubuntu lucid–natty
2009-09-04	v2.0.13	yes	Debian squeeze

Summary

- RFC 4880 allows moving away from SHA-1
- Interoperability may suffer
  (no problem on GnuPGs of last three years)
- Switch to SHA512 takes $<$10 minutes
- Only remaining hurdle: updating algorithm used for subkey binding

- Beware:
  - This does not prevent you from receiving SHA-1 digests.
  - This does not prevent you from sending SHA-1 digests.

SHA2 for GnuPG primer

- Add the following to your gpg.conf

```
s2k-digest-algo SHA512
digest-algo SHA512
default-preference-list SHA512 SHA384 SHA256
   SHA224 SHA1 AES256 AES192 AES CAST5 ZLIB
   BZIP2 ZIP Uncompressed
cert-digest-algo SHA512
```

- New keys
  - Just create them as usual. They are defaulting to SHA512 now.
- Existing keys
  - passwd your keys
  - setpref your keys
  - re-sign your keys

SHA2 for GnuPG primer

- Add the following to your gpg.conf

```
s2k-digest-algo SHA512
digest-algo SHA512
default-preference-list SHA512 SHA384 SHA256
   SHA224 SHA1 AES256 AES192 AES CAST5 ZLIB
   BZIP2 ZIP Uncompressed
cert-digest-algo SHA512
```

- New keys
  - Just create them as usual. They are defaulting to SHA512 now.
- Existing keys
  - passwd your keys
  - setpref your keys
  - re-sign your keys

Outline

OpenPGP §13.3.2. Hash Algorithm Preferences

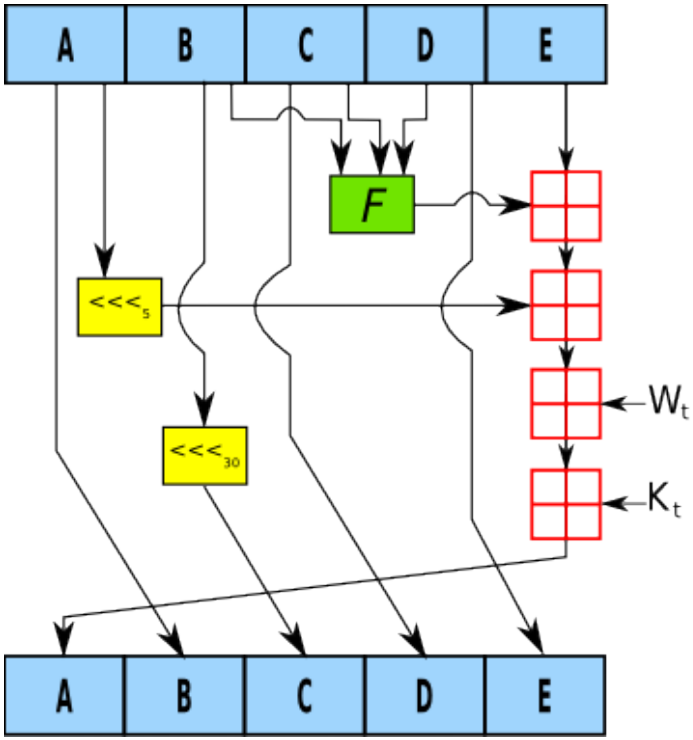
Since SHA1 is the MUST-implement hash algorithm, if it is not explicitly in the list, it is tacitly at the end. However, it is good form to place it there explicitly.

SHA-1 and black magic

- Construct message
- Pass message through black box
- Obtain message digest

SHA-1 and white magic

- Construct message
- Pad and append message size
- Chop into 512 bit blocks
- Initialize internal status
- Sequentially for each block
  - For each of 80 rounds
    - Simple bit shuffling (constants, $\wedge$, $\vee$, $<<$, ...)
- Obtain message digest



Schematics of single SHA-1 round (Wikipedia, Matt Crypto, Ppietryga, H2g2bob. CC-BY-SA 2.5)

Outline

Used abbreviations

FIPS	Federal Information Processing Standard
NIST	National Institute of Standards and Technology
SHA	Secure Hash Algorithm