

Public Key Crypto & Gnu Privacy Guard & CaCert

Svenne Krap
svenne@krap.dk

Indhold

- PGP
 - Motivation & teori
 - Praktisk
 - Løst og fast
- CaCert
- (Keysigning+Assurance)

PGP

Pretty good privacy

Motivation

Motivationen er at kunne kommunikere sikkert og/eller fortroligt med en anden part.

- Med **sikkert** menes at man har bevis for modpartens identitet
- Med **fortroligt** menes at ingen andre end kommunikationens parter kan følge med

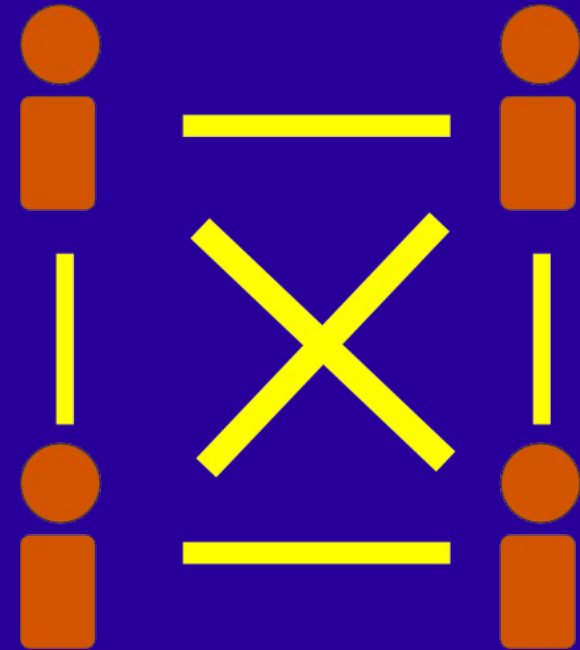
Kryptografi er som altid en del af løsningen....

Hvis ting er krypterede er et kun parter med adgang til nøglen, der kan læse dataene.



Men i traditionel crypto vokser
antallet af nødvendige nøgler meget
stærkt.

Antallet af nøgler er
Generelt $n*(n-1)/2$



Løsningen er asymmetrisk crypto (også kaldet public key crypto)

- Hver deltager har to nøgler (kaldet et nøglepar):
 - En **privat** nøgle, der skal holdes hemmelig
 - En **offentlig** nøgle, der skal udveksles med interesserede parter
 - Når den offentlige nøgle er signeret af andre, kaldes den et **certifikat**.

De specielle egenskaber ved asymmetrisk crypto:

- Hvis noget kodes med den **private** nøgle kan det kun åbnes af den **offentlige**
- Hvis noget kodes med den **offentlige** nøgle kan det kun åbnes af den **private**

Et kort sidespring

En **hash** funktion er en funktion, der omdanner et variabelt længde input til et fast længde output

En hash er en envejs funktion, hvor det er let at finde output fra input, men svært* at finde input pga. output

$$\text{Fx. } 8616460799 = 89681 * 96079$$

Kendte eksempler er MD5**, SHA1 og SHA2

* = ikke væsentligt nemmere end at afsøge systematisk

** = MD5 er dog ikke kryptografisk sikker mere

Nu har vi følgende situation:

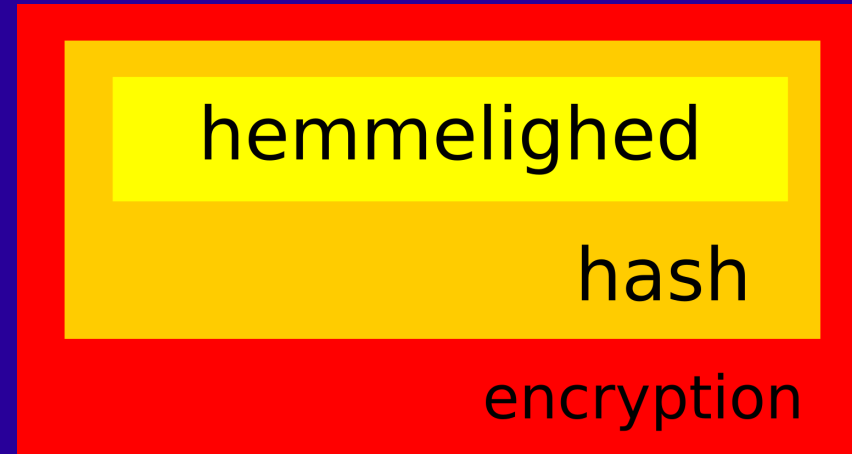
Primitiver: encrypt, decrypt, hash

Nøgler: PrivA, PubA, PrivB, PubB

Hvis vi har en **hemmelighed** kan vi nu:

- 1) Hash **hemmelighed** => sechash
- 2) Encrypt sechash med **PrivA** => signature
- 3) Vedhæft signature til hemmelighed => signeret hemmelighed
- 4) Encrypt signeret hemmelighed med **pubB** => cryptogram

- 1) Hash **hemmelighed** => sechash
- 2) Encrypt sechash med **PrivA** => signature
- 3) Vedhæft signature til hemmelighed => signeret hemmelighed
- 4) Encrypt signeret hemmelighed med **pubB** => cryptogram



Nu er der så mulighed for:

- A) Decrypt cryptogram med **privB** (kun B kan decrypte, A ved B er eneste modtager)
- B) Decrypt signature med **pubA** (alle kan decrypte, kun A kan have encryptet)
- C) Hash hemmelighed og sammenlign (hvis match er hemmeligheden uændret)

En praktisk overvejelse

Public key crypto er LANGSOMT da nøgler er store og operationerne er matematisk tunge

Derfor bruges der i praksis et mix af symmetrisk og asymmetrisk crypto

- 1) Generer en random nøgle af passende længde ift. symmetrisk cipher
- 2) Encrypt data med symmetrisk cipher med random nøgle
- 3) Encrypt random nøgle med asymmetrisk encryption og vedhæft data

Public-key crypto historie

- 1874, W. Jevons bemærker at visse matematiske funktioner er lettere den ene vej end den anden
- 1974 Merkle's puzzles
- 1976 Diffie & Hellman laver Diffie-Hellman key exchange
- 1977 Rivest, Shamir, Adleman (MIT) opfinder RSA
- 1991 Phil Zimmermann: Petty Good Privacy
- 1993: The People vs. Zimmermann
- 1998: OpenPGP standard (rfc 2440)
- 1999: Gnu Privacy Guard 1.0
- 2001: PGP/Mime (rfc 3156)
- 2006: Gnu Privacy Guard 2.0
- 2007: OpenPGP ny standard (rfc 4880)

Hvor bruges public key crypto?

- Sikring af mail
- Sikring af websites (HTTPS)
- Sikring af kode (signerede binaries..Xbox,Iphone ...)
- Digital signatur
- Netbank login
- OpenOffice.org
- PDF
- XMPP/Jabber (Google Talk)

Key management

Problem 1: Hvordan ved man at nøglen er stærk nok ?

Nøglen skal være stærk nok til ikke at kunne kompromitteres indenfor den tid hemmeligheden er interessant...

Overvejelser kan ses på <http://www.keylength.com>

Min personlige holdning: overkill is the right way to kill.....

Key management

Problem 2: Hvordan ved man at man har modpartens rette nøgle?

BEDST: Nøglen er personligt overrakt af modparten, hvilken du kender personligt

GOD: Nøglen findes et offentligt sted. Du kan få et fingerprint af modparten gennem en måde hvor du er sikker på hans identitet.

GOD: Du kan finde en nøgle online. En du stoler på, stoler på den anden nøgle er korrekt (Web of Trust)

MELLEM: En "trusted" 3.part (CA) har signeret den

SKIDT: Du hiver nøglen ned fra nettet uden at verificere den.

De to filosofier

OpenPGP: Folk stoler på folk. Trust kan beregnes i et "spindelvæv". Denne metode kaldes Web of Trust.

På **Keysigning Events** signerer man hinandens nøgler uden at kende hinanden (men ser billed-id). Trust er lavere, men hvis en person har mange (10? 100? 1000?) "fremmede" til at signere hans nøgle er den nok ok?

X509: Man har en eller flere pålidelige "myndigheder" der garanterer ægtheden.

Nu kaldes public delen af nøglen tit et certifikat!

MEN: Hvordan ved man om man kan stole på "myndigheden"?

PRAKTISK

Install

- aptitude install gnupg
- yum install gnupg
- emerge gnupg
-

Verify

```
$ gpg --version
```

```
gpg (GnuPG) 2.0.11
```

```
libgcrypt 1.4.4
```

```
Copyright (C) 2009 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
```

```
This is free software: you are free to change and redistribute it.
```

```
There is NO WARRANTY, to the extent permitted by law.
```

```
Home: ~/.gnupg
```

```
Supported algorithms:
```

```
Pubkey: RSA, ELG, DSA
```

```
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH
```

```
Hash: MD5, SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
```

```
Compression: Uncompressed, ZIP, ZLIB, BZIP2
```

gpg.conf

```
no-greeting
default-recipient-self
charset utf-8
keyserver hkp://subkeys.pgp.net
default-key 59B4E19C9078B10E75C52CA7844B0957AF6CA4CD
utf8-strings
encrypt-to AF6CA4CD
```

Key generation

Den hurtige:

```
gpg --gen-key ... vælg 1 DSA og Elgamal
```

Den stærke:

```
gpg --gen-key ... vælg 5 RSA (Sign Only)
```

```
gpg --edit-key 0x... "addkey" ... vælg 4 Elgamal (encrypt only)
```

Revocation certificate

```
gpg --output 0x...-revoke.asc --gen-revoke 0x....
```

Husk at opbevare den her sikkert!

Eksporter din public-key

Manuel (i filer):

```
gpg --armor --export 0x.... > 0x....asc
```

Med keyserver:

```
gpg --send-key 0x.... --keyserver
```

Importer din vens

Manuel (i filer):

```
gpg --import <fil>
```

Med keyserver:

```
gpg --search-keys "Svenne"  
gpg --recv-key 0x....
```

Trust og sign (for begge):

```
gpg --edit-key 0x...  
trust  
sign
```

eller `caff*`

Dagligt brug

Gode options:

- a (armor) laver ren tekst udgave
- r (recipient) vælger modtager

Kørsler:

- e (encrypt)
- d (decrypt)
- s -b (detach sign)

Frontends

- KGPG
- Enigmail

Løst og fast

Signing-party

Debian-produceret pakke til keysigning events, findes i dag til de fleste distros.

Vigtigst er `caff` og `gpg-key2ps`

`~/.caffrc`

```
$CONFIG{'owner'} = 'DEMO KEY';  
$CONFIG{'email'} = 'presentation@krap.dk';  
$CONFIG{'keyid'} = [ qw{4F186BEAC15A26A6} ];
```

Andet brug

Signering af dokumenter ... fx referater ... kan gøres nemt via **-abs**

DIY password manager... husk at pipe data

Kan bruges sammen med LUKS

Mere info

- GNU Privacy Guard handbook
- Google

CaCert

Formål

- At udstede X509 certifikater (som kan bruges til HTTPS)
 - via tillid (Web of Trust)
 - Gratis

Status

- Er ikke universelt i browsere
 - Bliver formodentlig aldrig universel (IE)
 - Nogle linux distro har dem andre har ikke
 - Ellers må man selv importere root certificate (som findes på cacert.org) for at kunne bruge dem.

Hvad kan man så bruge den til?

- Steder hvor man kan sikre at "klienten" har root-cert installeret
 - Ens egne devices (printere, routere osv)
 - Intranet mm. med en begrænset brugerskare
 - Kan signe PGP nøgler
- Fordele vs. self-signed
 - Certifikater har faktisk trust og kan ikke bare erstattes med et falsk

Lidt om point

- Cacert bygger på et point system
 - Man får tildelt point ved assurance (≤ 35 pr gang)
 - Ved 50 point får man navn med i certifikater og de holder 24 mdr (mod tidligere 6)
 - Ved 100 point: code signing, bliv assurer (efter test)
 - Over 100 point: jo flere point du har (op til 135) jo flere point kan du assure med

Procedure

- Opret dig på cacert.org
- Udprint "CAP Forms – A4 - WoT"
- Medbring WoT og 2 styk officiel billed-id (pas + kørekort) – evt. 1 billed-id + sygesikring – til assurance event
- Vent på indtastning
- Bemærk: assurer beholder din WoT-form og har pligt til at gemme den i 7 år.

Læs mere

<http://cacert.org>