Kryptosäkerhet

2007-Mars-27, KY-Enköping.

(Uppdaterad 2007-Mars-31)

Glenn Larsson

Programmerare

Västerås

Vem är jag:

- Glenn Larsson, bor i Västerås
- INFOSEC-D, KY Enköping 2004-2006
- Programmering, Nätverkstekniker, Systemansvarig, PC-Tekniker, Utbildare.

Vad ska jag prata om?

- Kryptologi i största allmänhet.

Varför?

- Magnus bad mig komma hit och prata lite kort om kryptering.

AGENDA

- Introduktion
- Algoritmer
- Kryptonycklar
- Algoritmen
- Säkerhet
- Message Digest
- Certifikat
- Verkligheten
- Avslutning

INTRODUKTION

Lite historia om kryptologi:

"Caesar Cipher"

A (1) + 3 = D (4)
B (2) + 3 = E (5)
C (3) + 3 = F (6)

...osv...

Z (26) + 3 = Ö (29)
Å (27) + 3 = A (1)
Ä (28) + 3 = B (2)
Ö (29) + 3= C (3)

Säkerhet: Tar nån milisekund för en gammal PC från 80 talet att knäcka detta krypto.

Polyalfabetiskt krypto.

En bokstav kan vara flera bokstäver i en bestämd sekvens:

```
A B C D E F G H I J K ...   (huvudalfabet)
J K L M R S T U E F G ...   (alt 1)
G H I N V W X Y Z A B ...   (alt 2)
B C D O X R S T U E F ...   (alt 3)
```

Första gången blir A till J, nästa gång G, för att sen bli till B och när det sista alfabetet har passerats så blir A till J igen.

Säkerhet: Stoppar kanske lillasyster, men inte storebror.

Enigma och liknande krypto.

En nyckel används för att styra vilka bokstäver som ska användas. Något komplexare men inget revolutionerande.

Säkerhet: Det dög inte 1945 och inte heller idag.

DES – Data Encryption Standard

Den nya skolans kryptologi. Arbetar inte med tecken utan med bitar och logiska operatorer. Ett tag så var det populärt att utveckla nya varianter istället för att komma på något nytt.

Säkerhet: Hyffsad, värt att notera är att DES är av Amerikanska försvaret *endast godkänd för icke klassificerad information.*

Vem gör kryptering? Vem gör designen?

70-talet och tidigare: “The dark ages…”

Underättelsetjänster (NSA, KGB)
I viss mån företag (IBM projekt “Lucifer”)
Public Key systemet RSA skapades

80-talet: Det börjar bli mindre ”hush-hush”

Företag fick mer och mer digitala hemligheter att skydda
PC boomen
Skolor, Universitet

90-talet: Öppenheten tar fart

Privata individer (AES ”tävlingen”)

All modern kryptologi bygger på *Kerchoffs Theorem*:

-"Säkerheten ligger i nyckeln och inte att algoritmen är hemlig."

Varför ska allt vara öppet då?

Den algorithm som helt öppet kan stå pall emot tusentals som försöker knäcka algoritmen har större trovärdighet än en algoritm som är gjorda av få personer och aldrig öppet testad - dvs Flera kockar = Bättre soppa.

Det är ganska arrogant att hävda:

- "Vi har världens bästa kryptologer och vi kan inte knäcka våran egen algoritm, därför är den säker"

KRYPTOALGORITMER

Asymmetrisk vs Symmetrisk kryptering:

Asymmetrisk kryptering:

TVÅ nycklas används:

* En **publik** som sändaren har (icke hemlig)
* En **privat** som mottagaren har (hemlig)

Inte snabb, men 2 parter som inte känner varandra kan skicka meddelanden mellan varandra. Kan även användas för att skicka krypto nycklar mellan 2 parter för Symmetriska kryptosystem t.ex. Nätverksprotokoll.

Symmetrisk kryptering:

EN nyckel används får kryptering och dekryptering

Snabb, men nyckeln måste levereras personligen.

Säkerhet:

Asymmetrisk kryptering:

Dessa algoritmer baserar sin säkerhet på ett antagande om att matematik är svårt och att ingen kommer att lösa problemet (faktorering av primtal m.m.)

Hittils har de haft rätt...

Symmetrisk kryptering:

Symmetriska algoritmer baserar sin säkerhet på att deras logiska komplexitet är svår att knäcka.

Sårbarheter hittas hela tiden i dessa algoritmer, men sällan så måste algoritmer bytas ut.

Asymmetriska (och liknande) algoritmer:

RSA

Rivest, **S**hamir, **A**delman. Namnet på 3st forskare som kom på att primtal hade vissa egenskaper som t.ex. att en nyckel bara kan kryptera medans en annan kan dekryptera information.

Diffie Hellman

Ett primtalsbaserat system liknande RSA men bygger på modulär exponentiering och ofta används i nätverksprotokoll för att låta datorer komma överens om en sessionsnyckel.x

ECC

Ett system baserat på Eliptiska kurvor skapat av Certicom. Använder inte lika stora nycklar som RSA då certicom anser ECC vara säkrare på grund av avsaknade av lyckade forceringar.

Symmetriska algoritmer:

RC4

En gammal algoritm. Skapad av Ronald Rivest (RSA). Ifrån början en hemlig algoritm, men den blev reverseenginerad och publicerad. Numera äger RSA bara namnet, kallas därför ibland för ”ARCFOUR”. Används fortfarande i t.ex. WEP.

DES

Gjord av amerikanska **National Institute of Standards and Technology** (NIST). Den är “Gammal som gatan” men används fortfarande i t.ex. Hårdvara och äldre IT-system.

AES

Relativt ny, gjord av två forskare. Vann en tävling som NIST höll i på slutet av 90 talet där de sökte en ersättare åt *DES*.

Två kategorier av symmetriska algoritmer:

Streamcipher

Snabba, oftast endast en runda per tecken. Används t.ex i filsystem/nätverk för att flytta data fort. Tillåter Random Access.

Exempel på streamciphers: RC4

Blockcipher

Mindre snabba, flera rundor per block. Används ofta för långsiktig datalagring. Tillåter inte Ej Random Access.

Exempel på blockciphers: DES, AES

Två enkla begrepp: Substitution/Permutation:

Substitution: (byta ut tecken)

"hej hur mår du?"
”Xup XYT abT pY?"

Permutation: (aka “Transposition”)

"hej hur mår du?"
"jeh ruh råm ud?”

Substitution används nästan exklusivt idag och i många algoritmer i en funktion som kallas *S-Box*.

S-Box:

Vad gör en S-Box?

- Ökar på säkerheten i algoritmen, fast man mäts aldrig då den är *statisk* i algoritmen, även när nyckeln påverkar S-Boxen (*t.ex Khufu & Teledyne/Ritter*)
- RC4 är i *princip* en stor S-Box då den bara kör XOR på data som ska krypteras.

KRYPTONYCKLAR

Nyckellängd är det säkerhet?

- Inte nödvändigtvis.

Asymmetriska algoritmer (RSA, ECC): +1024 bitar

Symmetriska algoritmer (DES, AES): 56-256 bitar

One Time Pads: Flera miljoner bitar.

Orsaker:

Kvaliteten på algoritmens design.
Identifierade sårbarheter.

Fakta: datorer blir snabbare och snabbare.

Vad påverkar kryptonycklars användning?

Livslängd

Hur länge ska den användas, t.ex. ska den skydda en backup i 50 år?

Politik

Paranoida regeringar som vill förbjuda stark kryptografi eller vill "äga" en del av alla kryptonycklar (Key-ESCROW).

Sårbarheter

När som hellst kan någon identifiera en sårbarhet i en algoritm. Då kanske vissa nycklar blir ”svaga” eller icke önskvärda.

Lite Terminologi om kryptonycklar:

Keyspace =	Den totalt mängden *kombinationer* en nyckel kan göra under vissa förutsättningar (t.ex sårbarheter). t.ex 2^128
Keysize =	Nyckelns *storlek* t.ex 128 bitar (16 bytes)
Keystream =	Den *datamängd* som nyckeln producerar för att *kryptera data* med.
Related Keys =	Nycklar med *"egenskaper" av andra nycklar*.
Weak keys =	Nycklar som producerar *svaga keystreams*. (RC4/DES)

Vad är "rundor" eller ronder ?

"Subkeys" applicerade på data flera gånger om.

Kryptonyckeln skapar X antal (t.ex.) 32 st subnycklar och i varje runda krypterar data med dessa nycklar.

Kallas även för "*Key Expansion*"

- Det ökar inte säkerheten, men väl nyckels logiska storlek och tiden det tar att kryptera data.

Exempel: Varje byte i en 128 bit nyckel kan generera ett 32 bitars värde som krypterar 32 bitar i taget eller en halva av 64 bitar.

Redundans i språket:

-“Eniglt en udnreösnknig på ett Egnelkst univreisett så kan man srkvia bkoestvär i vliekn ornding som hlest och det går ädnå att lsäa då hjrnäan upfpatatr odret så lnäge fösrta och sitsa bkostaevn är på rtät palts. Dttea breor på att vi itne lsäer vrjae bkotsav för sig uatn odret som hleteht...”

Vad ovanstående text visar är att själva orden i språk är redundanta, det gör det lätt för en kryptoanalytiker att gissa nästa tecken i ett ord. Redundansen eller “Entropin” i texten är beroende på längden av textmassan. Vanlig Engelska mäts enda ner till 1.5-1.3 bitars entropi per byte. (“Prediction and Entropy in Printed English”, 1951, Shannon)

Entropi:

Den verkliga kryptografiska styrkan i lösenord m.m.

Vanlig Engelska har (ner till) 1.3 bitars entropi (Tecken följd m.m.)

Vi skapar ett subset av tecken
[A-Ö] = 29 tecken
[0-9] = 10 tecken

Totala antalet kombinationer (per byte)

[**A-Z**] + [**0-9**] = **39** = **100111** (binärt)
bit **6 + 3 + 2 + 1** aktiverad.

Vi tar då bitarna (3,2,1) och ser hur många delar det är av bit 6; ***0.125+0.0625+0.03125***= **0.21875**.

Resultat = *6.21875 bitars entropi per byte*.

Entropi i praktiken:

Lösenord

[A-Ö] + [a-ö] + [0-9] = 29 + 29 + 10 = 68 = 1000100 (binärt)
= 7.0625 bitars entropi per tecken

"***pA5sW0rD***"
= 7.0625 (bitar) * 8 (tecken) = **56(+) bitars entropi**

Passerfras

[a-ö] = 29 = 11101 (binärt)
= 5.90625 bitars entropi per tecken

"***One ring to rule them all..***"
= 5.90625 (bitar) * 20 (tecken) = **118(+) bitars entropi**

Resultat: Mera entropi i passerfraser & lättare att komma ihåg!

ALGORITMEN

Funktioner (byggstenar) i algoritmer:

Rotera till vänster:

ROTL (**11110000** , 2) = **11000011**

Rotera till höger:

ROTR (**11110000** , 2) = **00111100**

XOR:

11110000 *Xor* **11111111** = **00001111**

NOT:

Not **10101010** = **01010101**

Modulär multiplikation:

257 * **135** modulo **531** = **180**

Modulär addition:

182 + **491** modulo **121** = **68**

Modulär exponentiering:

3 ^ 5 modulo 13 = 9

Komponenter i en algoritm:

Key schedule (Nyckel “schema”):

* Bygger t.ex. upp nyckeldata för rundor

S-Box:

* Gör substitution på data

Kryptering:

* Krypterar informationen med algoritmen.

En titt på en kryptoalgoritm: RC4 (aka ”arcfour”)

Den är simpel, går att komma ihåg i huvudet, därför är den också pedagogisk. Den har sina svagheter, men alla dessa går att fixa.

Fyll en lista: ***S(0-255)*** *med 0,1,2,3...N*
Fyll en lista: ***K(0-255)*** *med nyckelns bytes*

```
For i = 0 to 255
    j = (j + S(i) + K(i)) Modulo 256
    Byt S(i,j)
Next i

i = 0: j = 0

For x = 1 to Storleken(Data)
    i = (i + 1) Modulo 256
    j = (j + S(i)) Modulo 256
    Byt S(i,j)
    Vektor = (S(i) + (S(j))) Modulo 256
    Data(x) = Data(x) Xor S(Vektor)
Next n
```

SÄKERHET

Attacker = Reducera komplexiteten!

- Nyckeln (Weak keys och allt det där)
- Designfel dyker upp hela tiden! (IV'n i WEP t.ex.)

(med mera…!)

Omöjligt att skydda information i ett system:

- Keyloggers/Bakdörrar
- Timing attack (Kocher, *coolhetsfaktor*, emot *RSA Secure ID*)
- Virtuellt minne/Temp filer
- Osäkra processer (krashar, minnesaccess, sällan feltolerant)
- Slumptalsgeneratorn vid nyckelgeneration (se **www.protego.se**)
- Man in the middle-attacker
- Sidechannel attacker (Påverka systemet med andra faktorer)
- Orealistiska kryptoprotokoll
- "Rubber hose" cryptology

(med mera…!)

Finns det några 100% säkra system?

Ja: "One time pads":

Det är det enda bevisbart säkra kryptosystemet som existerar.

- Tyvärr också helt **opraktiskt** eftersom nyckel längden >= Meddelandets längd.
- Distributionen av kryptonycklar blir hemsk.. (kurirer)
- Fungerar på strategiska ubåtar och andra ställen som behöver extrem säkerhet. Vanliga företag har inte den organisation som krävs för att underhålla ett sådant kryptosystem.

Hur “One time pads” fungerar:

Nyckeln = Lika lång som meddelandet.

Exempel:
"hej hur mår du?"

```
abcdefg H ij K lmnopqr S tuvwxyzåäö
        1 1 1111111 1 222222222
1234567 8 90 1 2345678 9 012345678 9
```

h = 8

Talet Modulo 29
Dvs: om talet > 29, reducera med 29 ner till 1 igen

k = 11

Nytt tecken blir då 8+11 dvs 19 (s)

Dekryptering sker *baklänges*

*Men, inte ens **KGB** kunde använda OTP's korrekt (!)*
[http://www.nsa.gov/publications/publi00039.cfm]

MESSAGE DIGEST

Message Digest (MD) algoritmer:
(aka "One way hashing algoritm")

Vad är det? En *signatur* på data som blir lika stor, oavsett hur mycket data som du matar in.

Exempel:

Vi förändrar *en* bit i meddelandet "HUR MÅR DU" och jämför:

H = Bitar 64+8
I = Bitar 64+8+1

"HUR MÅR DU" **= 1A56EA75C625F9DE8A8244F98D4038EB**
"IUR MÅR DU" **= EAA5BBEE9ACCFE2DB9A6743FDA3F2406**

Vi får ett helt nytt resultat när bara EN bit förändras.

Standarder:

MD5	(Message Digest 5)
SHA/SHA-1	(Secure Hash Algoritm)
SHA-192/256/512	(Secure Hash Algoritm)

Exempel på användningsområden för MD:

Med **f ()** *avses en message digest funktion som MD5 eller SHA algoritmerna*

Bevisa att du har ett dokument utan att visa dokumentet:

X = f (DOKUMENT)

Bevisa att du har ett dokument ett visst datum:

X = f (DOKUMENT & TID)

Generera kryptonycklar ifrån lösenord/passerfraser:

X = f (PASS)

Validera att en fil inte har blivit korrupt:

X = f (FILINNEHÅLL)

Exempel på användningsområden för MD:

Dela upp en kryptonyckel mellan 3 olika parter:

A = f (PASS)
B = f (PASS)
C = f (PASS)
NYCKEL = **A** XOR **B** XOR **C**

Skapa identitetsbaserade variabler (Websidor m.m.)

X = f (IP_ADRESS & RÄKNARE & NYCKEL)

Skapa signaturer (Message Authentication Code)

X = Crypt (f (DOKUMENT) , NYCKEL)

Exempel på användningsområden för MD:

Följande exempel kräver asymmetrisk kryptering:

Skapa digitala signaturer

X = Crypt (f (DOKUMENT) , **PRIVAT**_NYCKEL)

Validera digitala signaturer

X = Crypt (f (DOKUMENT) , **PUBLIK**_NYCKEL)

Dokumentet skickas ut och den publika nyckeln användas om dokumentet behöver valideras.

Ingen med publika nyckeln kan återskapa dokumentets signaturen och endast den privata nyckeln kan skapa signaturen på dokumentet.

CERTIFIKAT

Certifikat:

Lösenord och kryptonycklar kan användas av exakt vem som helst så på något sätt måste användare kunna lita på att de loggar in på rätt server, och servern måste kunna lita på att användaren är den som han/hon utger sig för att vara.

Voila' – Certifikat

Certifikat är till för att underlätta identifiering av användare och data i nätverksprotokoll. Certifikat använder sig av **Asymmetrisk kryptering** samt **Message Digest** funktioner för att skapa digitala signaturer.

Certifikaten är skapade i en hierarki med ett root organisation längst upp som intygar autenticiteten av de underliggande certifikaten som i sin tur intygar autenticiteten på de dokument/certifikat som de signerar.

CA – CERTIFICATE AUTHORITY

En uppgift av en CA är att:

- Bestämma livslängd på certifikat (Användning)
- Skapa certifikat
- Revokera certifikat (CRL – Certificate Revocation List)

En CA är mer än bara en mjukvara, det är en organisation med fastigheter, skalskydd och allt det vanliga. En CA står som garrant och signerar data/nycklar och har ofta mycket hög säkerhet. Men en organisation behöver inte bygga upp en egen CA.

Det finns färdiga certifikat att köpa ifrån CA's som t.ex. **Verisign**, **Thawte** och liknande företag.

Certifikat säljs i perioder, ofta gällande i X antal år.

VERKLIGHETEN

Krypto protokoll:

Specifierar HUR någonting ska ske:

1. Alice krypterar meddelande M med nyckel K1
2. Alice signerar meddelande M med nyckel K2
3. Alice skickar meddelande M till Bob
4. Bob emottar meddelande M ifrån Alice
5. Bob validerar Signaturen på M med nyckel K2
6. Bob dekrypterar meddelande M med nyckel K1

Ett krypto system – mer än bara en algoritm:

Policys för t.ex. säkerhet, lösenord och krypteringssystemet

Mjukvara för kryptering

Hårdvara att köra systemet på (Underliggande säkerhet)

Krypteringsprotokoll (Applicering: hur ska saker göras?)

Lagring och Åtkomstkontroll (Åtkomst till krypterat data)

Slumptal (Generering av t.ex. sessionsnycklar)

Nyckeldeponering (Anställda som dör eller uppsägningar)

Spårbarhet (**Vem** har gjort **vad** och **när**?)

Certifikat (Skapande, revokering, livslängd, även policy)

Vad behöver vi kryptera, måste vi kryptera allt?

Nej. Det hela beror på själva informationen och dess värde.

Dagens lunch i organisationens cafeteria är troligtvis ingen viktig information men dokumentation om ännu icke patenterade uppfinningar ska krypteras.

Mellan dessa två exempel så får någon form av balans fastställas som bestäms av organisationens verksamhet.

Säkerhetsklassningen av dokument är en bra guide...

Säkerhetsklassning vs Kryptering:

Säkerhetsklass:	**Exempel på åtgärd:**
Offentlig	Ingen åtgärd
Begränsad spridning	Ingen åtgärd
Konfidentiell	Krypteras vid lagring
Hemlig	Krypterad vid användning och lagring
Kvalificerat hemlig	Krypterad vid användning och lagring samt åtkomstkontroll och loggning.

Notera: "Kvalificerat hemlig" för ett företag (t.ex. Källkod) är inte lika "Kvalificerat hemlig" som Försvarsmakten (t.ex. Sveriges försvarsplaner i krisgtid). Dessutom ställer data ibland krav på att data ska kunna användas dagligen, t.ex. Källkod till program som är under utveckling.

Så kan de faktiska åtgärdena variera beroende på verksamheten.

WIRELESS

Lösningar för WiFi nätverk:

Hårvarukryptering

Inbyggd. Snabb. Blir sårbar med tiden.

WEP	**RC4 (!Undvik!)**
WPA	**RC4 (med mjukvaru AES)**
WPA-2	**RC4/AES**

Mjukvarukryptering

Ligger på datorn, inte lika snabb som hårdvara. Går lätt att byta ut.

VPN Klient (Många tillverkare)

Applikationskryptering

SSL/TLS i applikationer (T.ex. webbrowser)

AVSLUTNING

Läsvärt:

* *Applied Cryptografy*

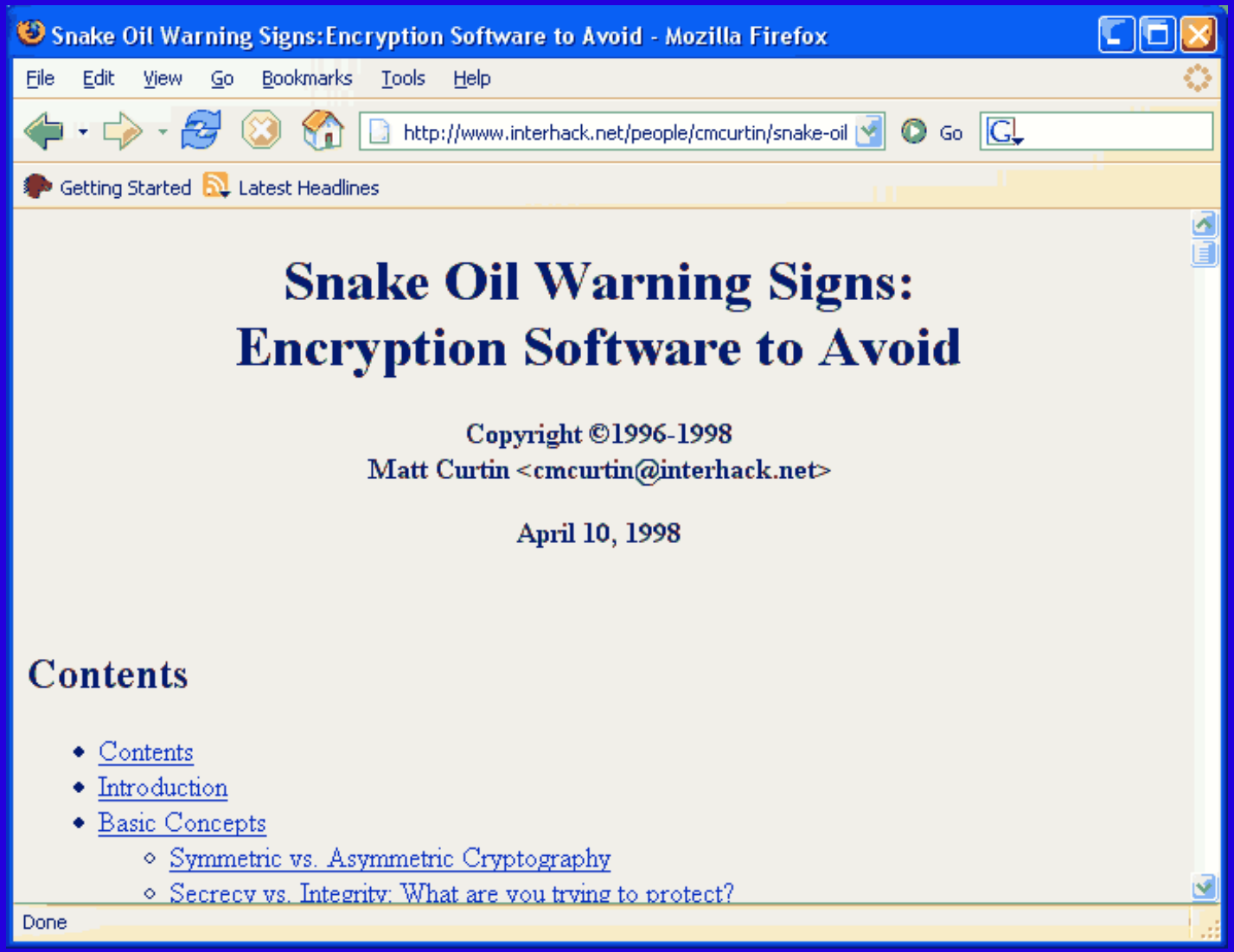
  (Bruce Schneier, Wiley förlag)
  ISBN: 0471128457

* *Cryptografy & Data Security*

  (Dorothy Denning, klassiker – permanent out of print! Kolla biblioteket.)

"Snakeoil FAQ"

http://www.interhack.net/people/cmcurtin/snake-oil-faq.html

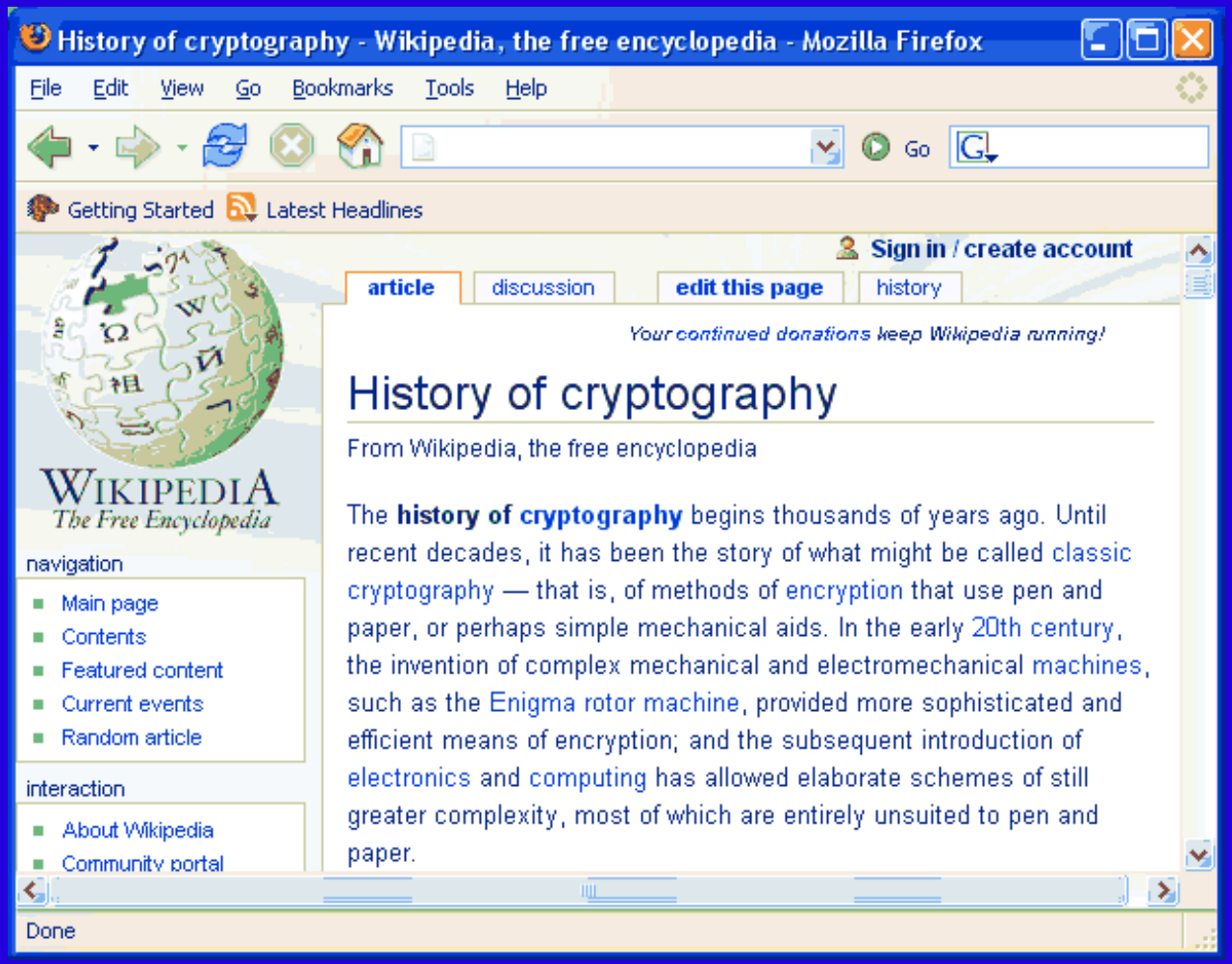


Wikipedia: WEP (med länkar till WPA och WPA2)

http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

Wikipedia: History of Cryptography.

http://en.wikipedia.org/wiki/History_of_cryptography



Frågor?

Glenn Larsson
[gurgel@postmaster.co.uk]