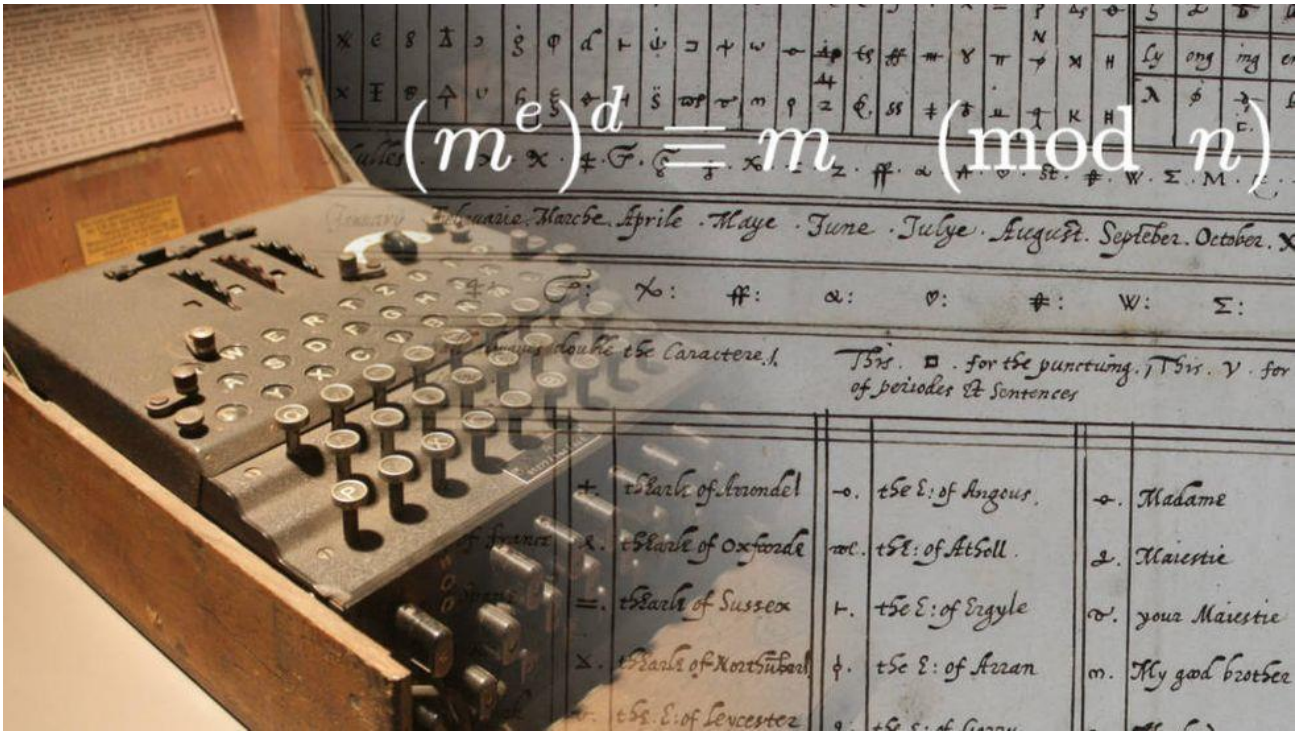
# KRYPTOGRAFINS HISTORIA

## med kuriosa





### Sammanfattning

En kortfattad beskrivning över kryptografins historia med de enklaste och mest använda metoderna



### Richard Frank

richard.frank@infotrust.se

Författare: **Richard Frank**

## Innehåll

**1 INLEDNING ..... 2**

**2 CEASAR OCH DE FÖRSTA CHIFFREN ..... 2**

2.1 Förskjutningschiffer ..... 2
2.2 Vigenère-chiffret ..... 2
2.3 Kodböcker ..... 3
2.4 Engångskryptot – det enda obrytbara kryptot ..... 4
2.5 Kryptomaskiner ..... 4
2.6 Matematisk hjälp ..... 4
2.7 Steganografi ..... 4
2.8 Kryptografi (är inte samma som Kryptologi) ..... 5
2.9 Substitution (Förskjutningschiffer) ..... 5
2.10 Scytale ..... 5
2.11 Krypto eller Chiffer ..... 5
2.12 De första kryptoanalytikerna ..... 6
2.13 Atbash ..... 7
2.14 Övergångstiden ..... 8
2.14.1 Att strö in "nollor". ..... 8
2.14.2 At såmm än krata staaua. ..... 8
2.14.3 Att använda en kod. ..... 8
2.15 Vigenérekryptot ..... 8
2.15.1 Användes Vigenérekryptot? ..... 9
2.15.2 Det stora chiffret ..... 9
2.15.3 Mannen med järnmasken ..... 9
2.15.4 Hur gick det för Maria Stuart, drottningen av Skottland? ..... 9
2.16 Mekaniserad säkerhet: ..... 10

**3 DJUPDYKNING I KRYPTOVÄRDEN ..... 11**

3.1 Vuxenutbildning (hands-on) ..... 11
3.2 DES (Data Encryption Standard) ..... 11
3.3 Lede Fi (Eve eller Eva) ..... 12
3.4 Gissningslekar ..... 13
3.4.1 Substitutionschiffer ..... 13
3.5 Obevakad minut ..... 14
3.6 Jag kan inte gå för det ..... 16
3.6.1 Vad som händer är detta: ..... 17
3.7 Din fantasi ..... 17
3.8 Letar du efter ett gott tecken ..... 19

## Historik

| Version | Författare | Datum | Beskrivning |
|---|---|---|---|
| 0.1 | Richard Frank | 17 Mars 2023 | Skapat första upplagan |
| 0.2 | Richard Frank | 18 mars 2023 | Revidering |

# 1 Inledning

Redan de gamla romarna skickade kodade meddelanden till varandra och genom historien fram till i dag har kryptering spelat en oerhört viktig roll i samhället. Tekniken har genom åren utvecklats historien och detta lilla kompendium visar de vanligaste metoderna på ett mer populärhistoriskt maner för att sedan göra en djupdykning i hur det fungerar Så det är dags att stänga dörrarna, dra för gardinerna för nu ska vi snacka om KRYPTERING

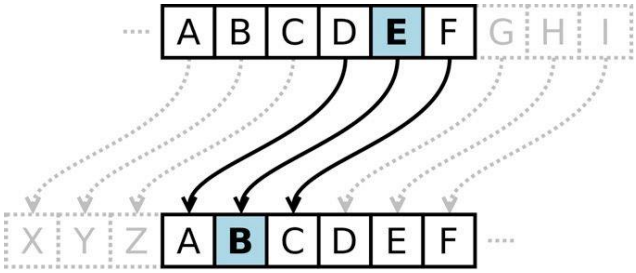
Före modern tid handlade kryptologi endast om sekretess av meddelanden (d.v.s. kryptering) - konvertering från ett begripligt meddelande till ett obegripligt och tvärt om, vilket gör meddelandet oläsligt för personer utan kunskap om det hemliga språket (det vill säga nyckeln som behövs för dekryptering av meddelandet). Kryptering har använts av bland annat spioner, militären och diplomater för att försöka säkerställa sekretess inom kommunikation. Under de senaste decennierna har området utvidgats till att innehålla bland annat tekniker för meddelandeintegritet, autentisering av sändare/mottagare, digitala signaturer och säkra betalningar.

# 2 Ceasar och de första chiffren

Romarna var inte först att skriva med chiffer. En mesopotamisk krukmakare på 1500-talet före vår tideräkning skyddade den värdefulla formeln för en lasyr, med ett krypto, och hebreiska lärda använde enkla chiffer tusen år senare. Julius Caesars utbredda användning av enkla förskjutningschiffer har gett namn åt dessa så kallade **Caesarchiffer**. Vägledningen riktar sig främst till den som praktiskt ska bedriva arbetet.

## 2.1 Förskjutningschiffer

I ett enkelt ***förskjutningschiffer*** byts bokstäverna i en text ut genom att flytta varje bokstav ett antal steg åt höger eller vänster i alfabetet. Att bryta sådana är en barnlek, så genom åren har kryptologer uppfunnit mer och mer intrikata sätt att byta ut bokstäverna.




## 2.2 Vigenère-chiffret

1553 uppfann Giovan Battista Bellaso Vigenère-chiffret som inte knäcktes fullständigt förrän 1863. På 1800-talet uppfanns de första ***polygrafiska*** chiffren där bokstäverna inte byts ut var för sig utan i grupper. Lite längre ner i texten förklaras Vigenère-chiffret

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| **B** | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A |
| **C** | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B |
| **D** | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |
| **E** | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D |
| **F** | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E |
| **G** | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F |
| **H** | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G |
| **I** | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H |
| **J** | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I |
| **K** | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J |
| **L** | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K |
| **M** | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L |
| **N** | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M |
| **O** | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| **P** | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| **Q** | Q | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
| **R** | R | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
| **S** | S | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| **T** | T | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S |
| **U** | U | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| **V** | V | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
| **W** | W | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
| **X** | X | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
| **Y** | Y | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X |
| **Z** | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |

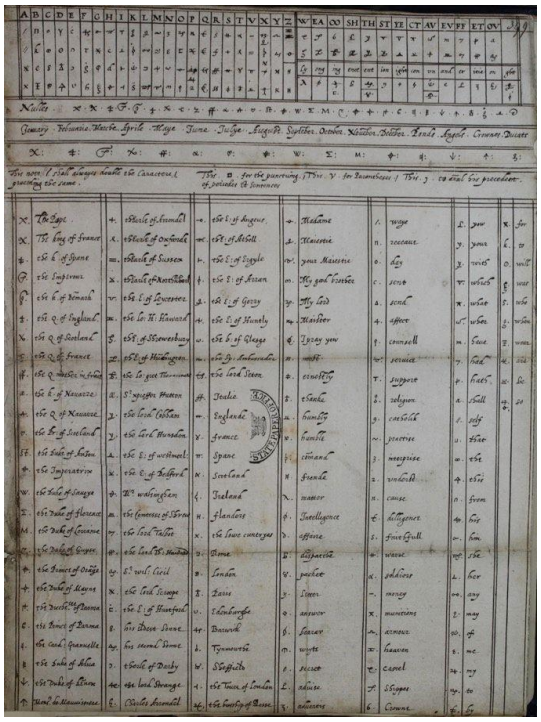
## 2.3 Kodböcker

*"Ingenting bör man anse så fördelaktigt som underrättelser;*
*ingenting bör belönas så rikligt som underrättelser;*
*ingenting bör vara så konfidentiellt som underrättelseverksamhet."*

*Sun Tsu, Krigskonsten (300-t f.Kr.)*

På onsdagsmorgonen den femtonde oktober 1586 står drottning Maria av Skottland inför rätta för högförräderi. Anklagelsen gäller anstiftan till mord på drottning Elisabet i syfte att själv erövra den engelska kronan.
Sir Francis Walsingham, förste sekreterare hos Elisabet, har redan expedierat de andra kuppmakarna.
Historikern William Camden berättar: "De höggs ner allihop, deras könsdelar skars av, inälvorna skars ut medan de ännu levde och var vid sina sinnens fulla bruk; sedan blev de styckade."
Maria nekas alla former av rättshjälp och är helt ensam vid sin bänk. Situationen var däremot inte helt hopplös eftersom hon sett till att all korrespondens med upprorsmännen skrivits med chifferskrift. Även om Walsingham hade lagt beslag på själva breven så borde han inte kunna tyda dem och därför inte använda dem som bevis.
Nationens främste expert på att knäcka koder hette Thomas Phelippes. Kunde han tolka vad som stod i breven så var hon så gott som död. Var chiffret kraftfullt nog att dölja vad hon skrivit så hade hon en chans att överleva.
Det var inte första gången som liv och död hängde på styrkan i ett chiffer.

För att kryptera och dekryptera hemliga meddelanden måste båda parterna känna till nyckeln, och under hundratals år tryckte militärer och diplomater kodböcker som distribuerades och skulle hållas i säkert förvar. Kom fienden över en kodbok kunde de inte bara avkoda utan dessutom skicka nya falska meddelanden.



Figur 1: Maria Stuarts kodbok – den skotska drottningen avrättades sedan drottning Elisabets spionmästare knäckte koden och slöjade Babingtonsammansvärjningen.

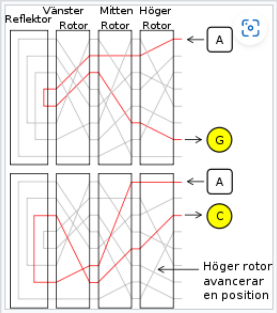
## 2.4 Engångskryptot – det enda obrytbara kryptot

Det enda krypteringssystemet som har bevisats vara omöjligt att knäcka är ***engångskrypton***. Det går ut på att använda en serie slumpmässigt valda bokstäver (eller andra tecken) som visar hur varje tecken i klartexten ska bytas ut. Längden på nyckeln måste vara minst lika lång som klartexten och en viss nyckel får bara användas till ett enda meddelande. Av praktiska skäl används detta väldigt sällan.

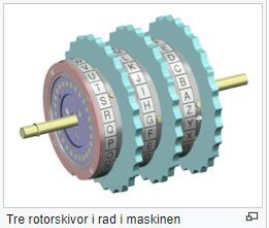
## 2.5 Kryptomaskiner

Under 1900-talets första hälft utvecklade flera länders underrättelsetjänster kryptografiska maskiner där ett antal roterande skivor utför upprepade förskjutningar på varje tecken. Tyskarnas ***Enigma*** är mest känd, eftersom britternas kodbrytare med Alan Turing i spetsen knäckte den och förkortade kriget.

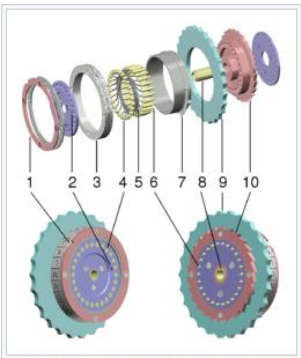
En tysk militär tre-rotors Enigma med (nedifrån och upp) insticksskort, tangentbord, lampor och fingerskivor som sticker ut ur höljet




Kryptering av två lika tecken i följd med Enigma – spänning appliceras på rotorerna och fortsätter runt reflektorerna och tillbaka till rotorerna. Bokstaven A krypteras annorlunda för var gång den används, först till G, sedan till C. Detta beror på att den högra rotorn har stegats fram och nu skickar strömmen en helt ny väg.



Tre rotorskivor i rad i maskinen




Sprängskiss av en rotorskiva. 1) ring med markeringar 2) kontakt för "A" 3) alfabetiskt hjul 4) anslutningsytor 5) ledningar 6) bleck 7) hävarm med fjäderbelastning 8) nav 9) fingerskiva 10) kuggkrans för spärrhake

## 2.6 Matematisk hjälp

Efter datorernas intåg har utvecklingen gått snabbt och kryptering blivit vanligt även civilt. Olika krypton har utvecklats som förlitar sig på datorer för att snabbt omvandla ettorna och nollorna som utgör ett meddelande. I dag är AES-algoritmen den mest använda. Den går ut på att utföra en serie operationer på matriser i flera rundor, och anses allmänt vara säker för all framtid så länge nycklarna hålls hemliga.

## 2.7 Steganografi

Att medvetet dölja ett meddelandes existens kallas steganografi.
Att tatuera in budskapet i svålen och sen låta håret växa ut var ett sätt som grekerna använde. Kineserna skrev på tunt silke som virades ihop till en boll och beströs med vax innan den intogs med några klunkar vatten. (Bägge metoderna kräver dock att det inte är helt bråttom.) På 1400-t beskriver italienaren Giovanni Porta hur osynligt bläck kan tillverkas av alun och vinäger. Lösningen tränger igenom äggskalet på ett kokt ägg och lämnar läsbar text i äggvitan. Mikropunkten, att förminska en text till en enda millimeterstor punkt, användes av tyska agenter under andra världskriget. Den första mikropunkten upptäcktes av FBI 1941.
Steganografi lider dock av en inneboende svaghet - lägger något beslag på meddelandet så är det avslöjat.

## 2.8 Kryptografi (är inte samma som Kryptologi)

Parallellt med steganografi utvecklades kryptografin. Den innebär att innehållet i meddelandet döljs istället för själva meddelandet.
Kryptografin kan delas upp i två grenar, transposition och substitution.

**Transposition** utgör i praktiken ett anagram där bokstäverna bara har kastats om. Ett ord på tre bokstäver kan bara skrivas på sex sätt. T.ex. kam, kma, akm, amk, mka, mak. En slumpmässig transposition av ett längre budskap blir i praktiken ofattbart svårtydbart, det krävs ett system som både sändare och mottagare är överens om.

Till exempel kan man ta varannan bokstav och dela upp meddelandet.

T l e e p l a m n a a a n n o s a o h e a p m d e a d t +
i l x m e k n a t v r n a b k t v c d l u p e d l n e =

T l e e p l a m n a a a n n o s a o h e a p m d e a d t i l x m e k n a t v r n a b k t v c d l u p e d l n e

Naturligtvis är tre rader bättre än två och bokstäverna kan byta plats parvis, ettan med tvåan, trean med fyran osv.

## 2.9 Substitution (Förskjutningschiffer)

Alternativet till transposition är substitution d.v.s. att byta ut en bokstav mot en annan.
En mycket tidig beskrivning finns i Kama Sutra ,300-t f.Kr., där nummer 45 av 64 nyttiga konster att lära sig för en kvinna är konsten att skriva hemliga meddelanden.
Man föreslår att slumpvis para ihop varje bokstav med en annan och sedan ersätta varje bokstav i det ursprungliga meddelandet med den nya bokstaven.
Väldigt avancerat för den tiden….

## 2.10 Scytale

Världens äldsta militära kryptografiska uppfinning är Spartanernas Scytale (400- t f.Kr.) En remsa av läder eller pergament lindas runt en stav av en viss diameter. Avsändaren skriver på längden och lindar sedan av remsan som då bara blir en rad meningslösa bokstäver.
Budbäraren kan använda remsan som t.ex. bälte eller armband tills mottagaren lindar upp remsan på en likadan scytale.

Den första dokumenterade militära användningen av substitutionschiffer står Julius Caesar för. En text från 100-t e Kr. berättar att kejsaren helt enkelt ersatte varje bokstav med bokstaven tre steg

Sedan dess kallas den här typen av substitution för Caesarrullning eller Caesarkrypto.

## 2.11 Krypto eller Chiffer

Definitionen av Krypto eller Chiffer är det namn man använder för alla former av kryptografisk substitution där varje bokstav ersätts av en annan bokstav eller symbol.

Naturligtvis kan man använda vilken förskjutning som helst mellan ett och tjugoåtta och begränsar vi oss inte till det utan gör vilka omplaceringar som helst så får vi ett ganska stort antal krypton. Något mer än 4 X 10 upphöjt till 26 tänkbara möjligheter. Om en fiendeagent kontrollerar en nyckel

i sekunden behöver han i runda slängar en miljard gånger universums ålder för att komma någon vart.

Varje chiffer är en kombination av en särskild metod, också kallad algoritm och en nyckel.
I Caesarkryptot är alltså algoritmen att förskjuta alfabetet och nyckeln talar om hur många steg det ska förskjutas.
En variant är att ersätta den här enkla "plus eller minus" -nyckeln med en nyckelfras eller ett nyckelord.

**Figur:**

a b c d e f g h i j k l m n o p q r s t u v w x y z å ä ö
R E V O L U T I N P Q S W X Y Z Å Ä Ö A B C D F G H J K M

Dubbelteckningar stryks och vid nyckelordets sista bokstav fortsätter alfabetet som vanligt fast utan nyckelordets bokstäver.

Fördelen är att det är lätt att memorera, vilket är viktigt då allt som är nedskrivet på papper kan falla i orätta händer.
På den här vägen var det under hela det första årtusendet av vår tideräkning. Alla lärde var överens om att det var omöjligt att forcera det här kryptot tack vare det väldiga antalet nycklar.

### 2.12 De första kryptoanalytikerna

Koranen består av 114 kapitel som vart och ett baserar sig på en uppenbarelse som profeten Muhammed haft. Koranen är inte skriven av Muhammed utan har nedtecknats av olika skrivare under lång tid, på lösblad och dessutom överförts från ett manuskript till ett annat,
Nu uppstår genast ett par frågor;
I vilken ordning fick Muhammed de här uppenbarelserna?
Är alla texterna verkligen profetens egna ord?
Det krävs en kultur som uppnått en ganska avancerad vetenskaplig nivå för att reda ut det här.
Matematiker, statistiker och lingvister löste problemet genom att analysera syntax, fonetik och ordfrekvenser i texterna.
Som en liten extra bonus råkade man på en genväg att knäcka ett krypto som ansetts oforcerbart i nästan 1000 år.

På 800-t skrev den arabiske vetenskapsmannen Al-Kindi en avhandling som bär titeln "Manuskript om dechiffrering av kryptografiska budskap".
Texten som löser en tusenårig gåta ryms på en enda sida och kan sammanfattas på följande vis:
1. Ta reda på vilket språk det är skrivet på.
2. Leta rätt på en vanlig klartext, gärna ganska lång.
3. Analysera vilka de vanligaste bokstäverna är och ordna dessa i en frekvenstabell.
4. Gör samma frekvensanalys av kryptotexten.
5. Matcha klartexten och kryptotexten med hjälp av frekvenstabellen.

Al-Kindis metod kallas frekvensanalys och visar att det är helt onödigt att prova alla nycklar.

**Figur:** Tabell över relativ bokstavsfrekvens i svenskan. Siffror i %.

| | |
|---|---|
| a | 9,3 |
| b | 1,3 |
| c | 1,3 |
| d | 4,5 |
| e | 9,9 |
| f | 2,0 |
| g | 3,3 |
| h | 2,1 |
| i | 5,5 |
| j | 0,7 |
| k | 3,2 |
| l | 5,2 |
| m | 3,5 |
| n | 8,8 |
| o | 4,1 |
| p | 0,7 |
| q | 0,007 |
| r | 8,3 |
| s | 6,3 |
| t | 8,7 |
| u | 1,8 |
| v | 2,4 |
| w | 0,03 |
| x | 0,1 |
| y | 0,6 |
| z | 0,02 |
| å | 1,6 |
| ä | 2,1 |
| ö | 1,5 |

I svenskan är e den vanligaste bokstaven, följt av a, n och t. Om den vanligaste bokstaven i chiffret är P så bör den således vara ett e eller ett a. Detta blir naturligtvis vanskligare att säga ju kortare text vi har.

### 2.13 Atbash

När araberna redan lärt sig montera ner chiffer kämpade européerna fortfarande med att lära sig skriva chiffrerad text. Det enda ställe där studier av hemliga skrivsätt uppmuntrades var i klostren. I gamla testamentet kan man hitta textstycken som krypterats med atbash, en hebreisk form av substitutionskrypto där man räknar hur många steg in i alfabetet en bokstav har och sen ersätter den med en bokstav som står lika många steg från slutet. "A" blir "ö" och "B" blir "ä" o.s.v.

På senare tid har böcker som "Bibelkoden" av M. Drosnin sålt i stora upplagor. Men innehållet får nog ses som mer spekulativt än vetenskapligt. Vilken tillräckligt lång text som helst kan tolkas som att den förutspått det ena eller det andra. Till exempel innehåller "Moby Dick" förutsägelser om mord på inte mindre än tretton berömda personer.

## 2.14 Övergångstiden

Mot slutet av 1500-t inlämnade kung Filip den andra av Spanien en petition till påven i Rom om att den kände franska kryptoanalytikern Francois Viete borde rannsakas då han måste vara "i förbund med djävulen". Dessbättre hade påvens egna män kunnat knäcka det spanska kryptot sen flera år tillbaka så det avslogs omgående.

De länder som uppmärksammat svagheten hos det monoalfabetiska substitutionskryptot sökte efter vägar att göra det säkrare. Man fann i huvudsak tre sätt.

### 2.14.1 Att strö in "nollor".

Nollor är tecken eller bokstäver som inte betyder något. Vi tänker oss att vi ersätter varje bokstav med siffror men att vi använder 1 till 99. Det ges oss 70 tal som inte betyder något och således försvårar analys.

### 2.14.2 At såmm än krata staaua.

Att avsiktligt använda alternativ stavning och syntax blandar också bort korten rejält.

### 2.14.3 Att använda en kod.

Ett tecken eller en bokstav kan också ersätta ett helt ord. "H" kan vara ett substitut för "mörda". HFG kan betyda "mörda kungen inatt". En stor mängd kodord kräver en bok som vore förödande i fiendens händer så därför litade man till mesta delen på en s.k. nomenklatur som är ett chifferalfabet och en kort lista med koder.

De skickligaste kryptoanalytikerna var kapabla att besegra alla dessa svårigheter.

## 2.15 Vigenérekryptot

Redan på 1460-t. Hade Leon Alberti (målare, kompositör, arkitekt) skrivit en essä om ämnet krypton. Han föreslog bruket av flera olika kryptoalfabet vilka man kunde växla mellan under krypteringens gång. (En enkel och självklar idé men likväl den bästa på på 1000 år.)

***Exempel***

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Å Ä Ö
N O P Q R S T U V W X Y Z Å Ä Ö A B C D E F G H I J K L M

*ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖ*
*NOPQRSTUVWXYZÅÄÖABCDEFGHIJKLM*
*a b c de f gh i j k l mno pq r s t uv w x yzåäö*

Man hoppar alltså mellan det övre och det nedre kryptoalfabet enligt ett överenskommet system.

Den som utvecklade det här konceptet till fulländning var Blaise de Vigenére som 1586 publicerade en avhandling där han fulländat Albertis tanke.

Vigenére föreslog bruket av ett stort antal kryptoalfabet ordnade i en tabell.
Även om den tabellen skulle bli känd av fienden kan den ändå användas i kombination med en nyckel i form av ett kodord som berättar exakt vilka alfabet som ska användas. (Se artikel tre för utförligare beskrivning.)
Det här innebär att Vigenérekryptot är oåtkomligt via vanlig frekvensanalys.
Krypterarna hade äntligen fått det vapen de behövde.

#### 2.15.1 Användes Vigenérekryptot?

Nej, det ansågs för komplicerat.
Istället för det svåranvända poly alfabetiska substitutionskryptot användes ett enklare homofoniskt dito. Det innebär att varje bokstav kan representeras av flera olika tecken. Svenskans näst vanligaste bokstav "a" har en relativ frekvens av ca. 9 %, således bör den ha åtta eller nio olika beteckningar medan bokstaven "b" (1,3 %) bara borde få max två. Det här är ett utmärkt tillfälle att ge varje bokstav ett eller flera värden mellan 00–99. Det här kryptot är också svåråtkomligt för vanlig frekvensanalys. Men...

Varje språk har sina egenheter. I engelskan är "q" en relativt ovanlig bokstav och borde således inte betecknas med mer än ett tecken. Det särskilda med bokstaven q är att den i engelskan aldrig följs av något annat tecken än u. Vidare vet vi att u utgör ca. 3% av engelsk text. Om vi hittar något tecken i kryptot som alltid följs av samma tre symboler kan vi anta att vi har hittat q och u.

### 2.15.2 Det stora chiffret

Ett imponerande exempel är "Le Grand Chiffré" som uppfanns av far och son Rossignol som uppfann ett förbättrat monoalfabetiskt chiffer på 1600-t. (Fyrahundra år senare är "Rossignol" fortfarande slang för dyrk på franska.)
Tyvärr föll det ur bruk och detaljerna glömdes snart bort vilket fick till följd att ingen kunde läsa Ludvig XIV:s hemliga meddelanden.

Det skulle dröja till slutet av 1800-t innan en kryptoexpert i franska armén lyckades lösa det. Chiffret innehöll tusentalssiffror - men bara i 587 olika kombinationer. Det slog då experten Bazeriés att det kunde röra sig om bokstavspar s.k. bigram.
Det finns 676 möjligheter att para ihop franskans 26 bokstäver.
Han visste vilka de vanligaste stavelserna i franskan var och kunde nu börja tillämpa frekvensanalys på bokstavs-parsnivå. Det tog honom tre år, men han löste det.

#### 2.15.3 Mannen med järnmasken

På Bastiljen satt en fånge med en järnmask. Denna fånge har kittlat mångens fantasi och enligt en teori skulle det t.o.m. vara kungens egen bror som han fängslat för att undvika stridigheter om tronen.
I ett brev till Ludvig XIV fanns det ett stycke där det hänvisas till en man med en mask. En viss general Bulonde har visat prov på en feghet gränsande till förräderi och förvaras nu inlåst och iförd en järnmask.
Brevet är äkta och såväl tiden som en del andra omständigheter stämmer.
Kom nu ihåg en sak kära vänner; Bara för att det är skrivet med chifferskrift så behöver det inte vara sant. Kanske är det en fälla i en fälla gillrad över 200 år tidigare.

### 2.15.4 Hur gick det för Maria Stuart, drottningen av Skottland?

Thomas Phelippes var mästerlig när det gällde frekvensanalys och undan för undan fann han nollorna och kunde börja vaska fram vad kodorden betydde. Han var också en otroligt bra

förfalskare och förfalskade ett tillägg i Marias brev där hon bad upprorsmakarna tala om sina namn så att hon kunde belöna dem efter upproret.
Maria befanns skyldig mot sitt nekande och dömd till döden.
Historikern Rickard Wingfield skriver:
Sedan lade hon sig på stupstocken helt lugnt och stilla och ropade ut "In manus tuas Domine!" ("I dina händer Herre" - Jesu sista ord på korset.) trenne eller fyra gånger under det att hon sträckte ut armar och ben, och i det sista ögonblicket höll en av bödlarna henne lätt med ena handen, den andra gav henne två slag med en yxa innan han högg av huvudet.

### 2.16 Mekaniserad säkerhet:

Churchill var efter sitt besök på Bletchley Park skakad över den bisarra blandning av människor som dagligen gav honom så värdefull information. Förutom matematiker och språkvetare fanns där en porslinsexpert, en museiintendent från Prag, en korsordsfanatiker, ett flertal bridgeexperter och en brittisk schackmästare.
Trots detta kände han stor tillgivenhet för den brokiga samlingen och kallade dem "gässen som värper guldägg utan att kackla".

# 3 Djupdykning i kryptovärden

## 3.1 Vuxenutbildning (hands-on)

I detta kapitel kommer vi att gå ner på detaljerna och prova oss fram med olika metoder men innan vi börjar på allvar, så är nog en liten ordlista på plats med kryptografiska termer. Då är vi på samma nivå när vi börjar fördjupa oss i själva algoritmerna. Mycket av det som vi hittar här är upprepningar från kapitel 2.

**Alice** – den som vill skicka ett meddelande
**Bob** – den som tar emot ett meddelande
**Eva** eller **Eve** – Den som Alice och Bob tror vill snappa upp meddelandet

Hela syftet med kryptografi är att göra det möjligt för **person A** (vanligtvis kallad **Alice** i litteraturen) att skicka ett meddelande till **person B** (som kallas **Bob**). Dessutom måste Alice skicka detta meddelande till Bob med förväntan att **person E** (*tjuvlyssnaren eller fienden*, vanligtvis kallad **Eva**) kan och kommer att fånga upp meddelandet på något sätt.

Kravet på kryptografi är att ändra meddelandet med hjälp av någon algoritm för att göra det extremt svårt för Eva att först och främst läsa meddelandet och för det andra att ersätta ett annat meddelande som Bob skulle anta vara ett riktigt meddelande från Alice.

Algoritmen som översätter meddelandet kallas **chiffer eller krypteringsalgoritm**.

Det ursprungliga meddelandet som Alice skriver är i klartext. När hon tillämpar en krypteringsalgoritm kommer denna att producera ett annat s.k. krypterat meddelande som kallas chiffertext.
Bobs jobb, när han tar emot chiffertexten, är att tillämpa den inversa **krypteringsalgoritmen**, kallad **dekrypteringsalgoritm**, för att komma åt den ursprungliga klartexten.

Det kommer kanske inte som någon överraskning att krypterings- och dekrypterings-algoritmerna är mycket nära besläktade. I själva verket är de kända som krypteringsalgoritm.

## 3.2 DES (Data Encryption Standard)

Moderna algoritmer är i allmänhet väldefinierade, välkända och välstuderade. För att ge ytterligare säkerhet använder algoritmerna oftast en nyckel (eller ibland lösenord) dvs de är i sig själva ”krypterade”.

Denna nyckel är vanligtvis ett stort binärt tal och måste förbli hemligt mellan Alice och Bob. Jag är faktiskt säker på att ni har hört talas om 40-bitars eller 56-bitars krypteringsmetoder som ***D**ata **E**ncryption **S**tandard* (**DES**), där "bitvärdet" bara är längden på nyckeln i bitar. I standard DES är till exempel nyckelns längd 56 bitar lång eller 7 byte. (1 byte = 8 bitar)

För att kryptera ett meddelande ansluter Alice nyckeln till krypteringsalgoritmen, och för att dekryptera ansluter Bob samma nyckel till dekrypteringsalgoritmen.

Dessa krypteringsalgoritmer är kända som *symmetriska algoritmer*. Systemets säkerhet bestäms av nyckelns hemlighet.

Observera att systemets säkerhet inte avskräcks av algoritmens hemlighet. Detta är en vanlig felaktighet. Om ett system använder en hemlig algoritm betyder det vanligtvis att den lätt blir föremål för hacking[1]. I vårt tidevarv, med överflödet av offentliga och kommersiella algoritmer, är det helt enkelt inte värt det att använda en krypteringsalgoritm som är hemlig.

Algoritmer med **offentlig nyckel** fungerar på ett helt annat sätt. Dessa algoritmer använder två nycklar: **en offentlig nyckel och en privat nyckel**. Med detta system publicerar Bob sin offentliga nyckel till världen.

När Alice vill skicka ett meddelande till honom krypterar hon sin klartext med den här offentliga nyckeln (dvs Bobs publika nyckel) och skickar chiffertexten. När Bob får meddelandet dekrypterar han det med sin egen personliga privata nyckel. Det mest kända programmet för att göra denna typ av kryptering är **Pretty Good Privacy (PGP).**

Ibland fungerar algoritmer för offentlig nyckel på motsatt sätt, Alice krypterar ett meddelande med sin privata nyckel och Bob dekrypterar det med Alice offentliga nyckel.

Dessa typer av algoritmer är vanligtvis kända som **digitala signaturer**: förutsättningen är att om Bob kan dekryptera ett meddelande med Alice offentliga nyckel, kan meddelandet bara ha kommit från Alice.
Meddelandet har inget som helst hemligt innehåll. Liksom riktiga signaturer läggs digitala signaturer till "dokumentet" och genom en enkel tillämpning av den offentliga nyckeln kan Bob verifiera att dokumentet verkligen kommer från Alice.

Dokument i det här sammanhanget kan betyda text, program eller ActiveX-komponenter eller vad du vill. På de flesta programmeringsspråk signerar vi till exempel paketen för våra produkter digitalt för att visa att de kommer från oss. Vi använder ett t.ex. företag som heter VeriSign, EuroSign eller Addtrust för att hålla vår offentliga nyckel och med hjälp av ett t.ex. Windows-program kan du automatiskt kontrollera paketens härkomst.

### 3.3 Lede Fi (Eve eller Eva)

I all den här diskussionen har vi mest pratat om Alice eller Bob. Eva då? Vad ska hon göra för att försöka läsa Alices chiffertext?

Här antar vi att Eve känner till krypteringsalgoritmen som används av Alice och Bob, men hon känner inte till nyckeln. Hon skulle vilja knäcka ett meddelande så att hon får härleda nyckeln och sedan läsa alla ytterligare chiffertextmeddelanden ostraffat. Varje försök att knäcka ett meddelande är känt som en attack och den enklaste attacken att förstå är *brute-force attack*. Med den här attacken försöker hon bara nyckel efter nyckel efter nyckel tills hon så småningom återställer klartexten. Ju längre nyckeln, desto längre tid tar det henne.

Faktum är att förhållandet är exponentiellt. För en 8-bitars nyckel måste Eve prova upp till 28 eller 256 olika nycklar. För en 16-bitars nyckel, dubbelt så lång, är det 65 536 olika nycklar.

För DES-algoritmen, sju gånger så lång som en 8-bitars nyckel, är det 7 * 1016 olika nycklar. Om hon kunde kontrollera en miljard nycklar per sekund skulle det ta henne upp till 2,28 år att hitta nyckeln.

[1] I grunden handlar det om att med hjälp av kunskap och programmering skaffa sig information om, ta sig in i och i vissa fall manipulera ett datorsystem – eller rent av en enskild dator

Om Eve visste att Alice och Bob hade för vana att använda 7 ASCII-tecken för sin DES-nyckel, skulle hon inte behöva prova nästan lika många nycklar för att knäcka ett meddelande och verkligen kunna knäcka det på cirka 8 sekunder i samma takt.

Om hon är värd sin vikt anställer hon en kryptoanalytiker[2] för att försöka knäcka budskapet.

Kryptoanalys är vetenskapen (eller konsten?) att analysera krypteringsalgoritmer tillsammans med någon chiffertext eller klartext eller båda för att identifiera nyckeln som används för att kryptera klartexten.

## 3.4 Gissningslekar

Klassiskt (med andra ord i tider före datorer) var krypteringsalgoritmer mycket enkla. I allmänhet agerade de alltid på bokstäver i alfabetet och genererade chiffertext som också bestod av alfabetiska bokstäver. Det finns två huvudtyper av klassiska krypteringsalgoritmer:

- Substitutionschiffer
- Transponeringschiffer.

### 3.4.1 Substitutionschiffer

Ett substitutionschiffer är den både den enklaste som äldsta krypteringsalgoritmen. Varje tecken i klartexten ersätts av ett annat tecken för att producera chiffertexten. Den mest kända av dessa är känd som Caesar-chiffer. Med denna algoritm ersätts varje tecken med en **n** bokstäver längre bort längs alfabetet (med ett omslag i slutet, uppenbarligen). Det ryktades att det chiffer som Julius Ceasar var **n** var 3, dvs A ersätts av D, B av E, X av A, Y av B och så vidare. Dekryptering är enkel: ersätt varje tecken i chiffertexten genom att räkna baklänges

Till exempel kan du enkelt dekryptera FDHVDU som CAESAR, med vetskap om att n var 3. Faktum är att även om du inte visste värdet på n är chiffret enkelhet att bryta: det finns bara 25 möjliga värden på n (n = 0 är ingen mening som en krypteringsalgoritm!) Att skriva ett cracker-program för att bryta Caesar-chiffret är trivialt. Före datoråldern var det enklaste sättet att knäcka ett Caesar-chiffer att skriva ner chiffertexten och sedan utöka alfabetet ner från varje chiffertextbokstav:

F D H V D U
— — — — —
G E I W E V
H F J X F W
I G K Y G X
. . . . . .
B Z D R Z Q
C A E S A R

Så småningom skulle du få en rad som var vettig; detta skulle vara meddelandet i klartext.

Ett annat exempel på ett Caesar-chiffer finns på UNIX-system, särskilt för meddelanden i nyhetsgrupper. ROT13 är ett Caesar-chiffer med n=13, och är lätt att använda genom att ett meddelande krypterat med ROT13 dekrypteras genom att ROT13 appliceras på chiffertexten.

[2] Kryptanalysprocessen syftar till att studera kryptografiska system för att identifiera svagheter och informationsläckor.

Följaktligen är dess huvudsakliga användning inte för att kryptera meddelanden säkert, utan för att tillfälligt dölja information så att tillfälliga tittare inte blir upprörda av det (till exempel risqué-skämt) eller som de bara vill se efter att de hade gjort ett val att göra det (till exempel spoilers för ett äventyrsspel eller en film).

Lista 1 visar kod som implementerar alla Caesar-chiffer: N-parametern definierar hur många bokstäver algoritmen ska gå vidare genom alfabetet för att kryptera varje tecken i klartexten. För att dekryptera kör rutinen samma kod internt, men tvingar N till 26-N. Så, till exempel, om du skulle kryptera med N = 3, skulle rutinen dekrypteras med N = 23. ROT13 är enkel att implementera: N = 13.

```
procedure AACaesarCipher(aEncrypt : boolean; N : integer; aInStream : TStream;
  aOutStream : TStream);
var
  BytesRead : longint;
  i         : integer;
  Ch        : byte;
  Buf       : array [0..255] of byte;
begin
  {force N in range 0..25}
  N := N mod 26;
  if (N < 0) then
    inc(N, 26);
  if not aEncrypt then
    N := 26 - N;
  {read through the input stream in blocks, encrypt the block, and
   write it to the output stream--only convert A-Z and a-z}
  BytesRead := aInStream.Read(Buf, sizeof(Buf));
  while (BytesRead > 0) do begin
    for i := 0 to pred(BytesRead) do begin
      Ch := Buf[i];
      if ((ord('A') <= Ch) and (Ch <= ord('Z'))) then
        Buf[i] := ((Ch - ord('A') + N) mod 26) + ord('A')
      else if ((ord('a') <= Ch) and (Ch <= ord('z'))) then
        Buf[i] := ((Ch - ord('a') + N) mod 26) + ord('a')
    end;
    aOutStream.Write(Buf, BytesRead);
    BytesRead := aInStream.Read(Buf, sizeof(Buf));
  end;
end;
```

Caesar-chiffret är ett exempel på ett mer generaliserat monoalfabetiskt chiffer. Här ersätter vi fortfarande varje klartextbokstav med samma chiffertextbokstav, men nu roterar vi inte alfabetet för att göra det, vi genererar bara slumpmässigt översättningstabellen. Alla monoalfabetiska chiffer är mycket osäkra eftersom de alla kan attackeras av bokstavsfrekvensmetoden: beräkning av frekvenserna för chiffertextbokstäver.

På engelska dominerar bokstäverna E, T, A, I och N, så det första du ska försöka skulle vara att översätta den vanligaste chiffertextbokstaven till ett E och se vad vi fick. Vi fortsätter i den här venen, experimenterar och försöker känna igen enskilda ord tills vi har avkodat chiffertexten helt. Det måste noteras att om vi kan upptäcka enskilda ord i chiffertexten, gör det dechiffreringen mycket lättare så i allmänhet skickas chiffertexten utan mellanslag eller skiljetecken.

### 3.5 Obevakad minut

En variant av detta är Vigenère-chiffret. Denna algoritm är ett poly alfabetiskt substitutionschiffer där vi använder flera enkla substitutionschiffer sekventiellt. Detta chiffer kräver också ett lösenord eller en nyckel, ett ord som kommer att användas för att bestämma de ersättningar som krävs.

Antag att Alice ville kryptera strängen 'Detta är ett meddelande' med lösenordet 'SECRET'. Först gör hon allt till versaler och tar sedan bort alla icke-alfabetiska tecken. Hon skriver ner detta komprimerade meddelande med lösenordet (upprepat, om det behövs) under:

THISISAMESSAGE SECRETSECRETSE

Ersättningen beror på den aktuella bokstaven i lösenordsraden. Den första bokstaven i lösenordsraden är S. Hon skriver ner alfabetet och sedan roteras under alfabetet så att S visas under A:

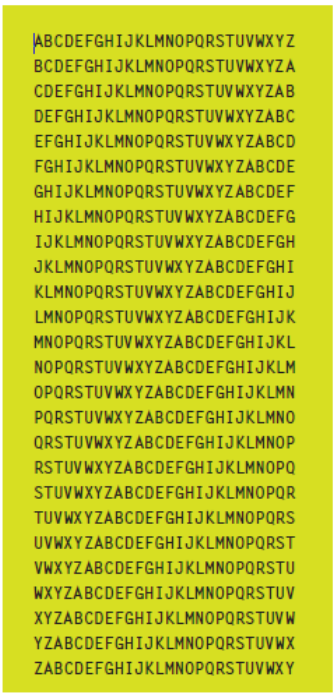
ABCDEF... TUVWXYZ
STUVWX... LMNOPQR

Hon kan nu koda T i klartext som ett L. Nästa bokstav i lösenordsraden är E, så hon genererar en ny konverteringstabell på samma sätt:

ABCDEFGH... VWXYZ
EFGHIJKL... ZABCD

Av detta ser vi att nästa chiffertextbokstav är L igen. Hon fortsätter så här tills hela klartexten är kodad. Bob, när han tar emot meddelandet, utför samma typ av metod för att dekryptera meddelandet.

För att göra det enklare för både Alice och Bob använder Vigenère-chiffer vanligtvis en 26 * 26-tabell med tecken där varje rad har alfabetet flyttat åt vänster med 1 från det föregående.




Figur 1: Komplett Vigenère utbytes tabell

Figur 1 visar denna tabell. Kom ihåg att detta är i dagarna före datorer: även om detta system heter Vigenère chiffer, utformades det först av Giovan Batista Belaso 1553; Blaise de Vigenère föreslog en liten ändring där klartexten följer lösenordet, istället för att lösenordet upprepas i all oändlighet.
För att koda hittar du lösenordordbokstaven längs vänster och klartextbokstaven längst upp och den punkt där den angivna raden och kolumnkorset är chiffertextbokstaven. För att avkoda, hitta lösenordsbokstaven längs sidan, följ raden tills du når chiffertextbokstaven och leta sedan upp kolumnen högst upp för att hitta klartextbokstaven.

Och Eva? Vad gör hon? Hon vet att Alice och Bob använder ett Vigenère-chiffer, så hon sätter igång. Det första att notera är att hon måste ha en chiffertext som är mycket längre än nyckeln, eftersom det första hon försöker göra är att räkna ut längden på nyckeln som användes för kryptering.
För att göra detta tar hon chiffertexten och jämför den med chiffertexten som har roterats med 1, 2, 3, etc., bokstäver. Hon räknar sedan antalet sammanfallande bokstäver vid varje position för varje rotation.
På grund av det faktum att klartexten är skriven på engelska, och vissa klartextbokstäver är vanligare än andra, kommer hon att upptäcka att rotationer som är en multipel av nyckellängden kommer att ha fler sammanfall än rotationer som inte är det. Detta ger en bra indikation på nyckelns längd. Av detta har hon en bra grund för att knäcka resten av meddelandet och ta reda på nyckeln.

Antag att hon finner att nyckellängden var 6. Detta innebär att bokstäverna 1, 7, 13, etc., alla kodades med samma Caesar-chiffer. Bokstäverna 2, 8, 14, etc., kodades med en annan Caesar-chiffer, och så vidare. Med hjälp av en tabell över bokstävernas frekvenser på engelska språket [3]kan hon nu göra antaganden om chiffertextbokstäverna genom att matcha frekvenser av chiffertext mot engelska språkfrekvenser. Utan alltför mycket besvär borde hon kunna räkna ut varje Caesar-chiffer, och därmed lösenordet, och därmed klartexten. Uppenbarligen för att detta ska ha någon chans att lyckas måste chiffertexten vara flera gånger längden på nyckeln eller lösenordet som används för att producera den.

[3] Naturligtvis har olika språk olika frekvenser på bokstäverna. Sverige har bl.a: A,E,N,T osv

```
procedure AAVigenereCipher(aEncrypt : boolean;
  aKey : string; aInStream : TStream; aOutStream : TStream);
var
  BytesRead : longint;
  i, j      : integer;
  Ch        : byte;
  Buf       : array [0..255] of byte;
  OutBuf    : array [0..255] of byte;
  KeyValues : array [0..255] of byte;
  KeyLen    : integer;
  KeyInx    : integer;
begin
  {the Vigenere cipher is for uppercase alphabetic letters
    only; in calculating the key values assume the key is in
    such a state}
  KeyLen := 0;
  for i := 1 to length(aKey) do
    if ('a' <= aKey[i]) and (aKey[i] <= 'z') then begin
      KeyValues[KeyLen] := ord(aKey[i]) - ord('a');
      inc(KeyLen);
    end
    else if ('A' <= aKey[i]) and (aKey[i] <= 'Z') then begin
      KeyValues[KeyLen] := ord(aKey[i]) - ord('A');
      inc(KeyLen);
    end;
  if not aEncrypt then
    for i := 0 to pred(KeyLen) do
      KeyValues[i] := 26 - KeyValues[i];
  {read through the input stream in blocks, encrypt the
    block, and write it to the output stream--only convert
    and write A-Z and a-z}
  KeyInx := 0;
  BytesRead := aInStream.Read(Buf, sizeof(Buf));
  j := 0;
  while (BytesRead > 0) do begin
    for i := 0 to pred(BytesRead) do begin
      Ch := Buf[i];
      if ((ord('A') <= Ch) and (Ch <= ord('Z'))) then begin
        OutBuf[j] := ((Ch - ord('A') + KeyValues[KeyInx])
          mod 26) + ord('A');
        inc(j);
        KeyInx := (KeyInx + 1) mod KeyLen;
      end
      else if ((ord('a') <= Ch) and (Ch <= ord('z'))) then
        begin
        OutBuf[j] := ((Ch - ord('a') + KeyValues[KeyInx])
          mod 26) + ord('A');
        inc(j);
        KeyInx := (KeyInx + 1) mod KeyLen;
      end;
    end;
    aOutStream.Write(OutBuf, j);
    BytesRead := aInStream.Read(Buf, sizeof(Buf));
    j := 0;
  end;
end;
```

Listning 2 är en implementering av Vigenère-chiffret. Det finns fyra parametrar i rutinen: om man ska kryptera eller dekryptera, nyckeln, klartexten som en ström och slutligen chiffertexten, även som en ström.

### 3.6 Jag kan inte gå för det

Det finns en variant av Vigenère-chiffret som används i hela datorindustrin. Det är lätt att genomföra, och inte särskilt svårt att bryta heller. Jag pratar om XOR-chiffer.

Alice och Bob kommer överens om en nyckel. Den här nyckeln behöver inte vara rent alfabetiska tecken, det är faktiskt bättre om det inte är det, och ju längre den här nyckeln är, desto bättre. För att kryptera klartext börjar Alice med den första byten i meddelandet och den första byten i nyckeln. Hon XOR dem tillsammans och matar ut den resulterande byten som den första byten i chiffertexten. Hon avancerar en karaktär och gör detsamma. När hon får slut på byte i nyckeln börjar hon om från början av nyckeln igen. För att dekryptera gör Bob exakt samma sak, och klartexten kommer att återställas, eftersom XOR av ett byte två gånger med samma värde resulterar i den ursprungliga byten.

```
procedure AAXORCipher(aKey : PByteArray; aKeyLen : integer; aInStream : TStream;
  aOutStream : TStream);
var
  Buf    : array [0..1023] of byte;
  KeyInx : integer;
  i      : integer;
  BytesRead : longint;
begin
  {read through the input stream in blocks, XOR the block with the key
   and write it to the output stream}
  if (aKey = nil) or (aKeyLen = 0) then
    raise Exception.Create('Cannot encrypt with XOR: the key is missing');
  KeyInx := 0;
  BytesRead := aInStream.Read(Buf, sizeof(Buf));
  while (BytesRead > 0) do begin
    for i := 0 to pred(BytesRead) do begin
      Buf[i] := Buf[i] xor aKey^[KeyInx];
      KeyInx := (KeyInx + 1) mod aKeyLen;
    end;
    aOutStream.Write(Buf, BytesRead);
    BytesRead := aInStream.Read(Buf, sizeof(Buf));
  end;
end;
```

Listing 3 har detta enkla XOR-chiffer. Återigen passerar vi in en nyckel och två strömmar, en ström för klartext och en för chiffertexten.

För att knäcka detta chiffer fortsätter Eva på samma sätt som för Vigenère-chiffret. Hon försöker först hitta tillfälligheter mellan chiffertexten och chiffertexten som förskjuts av olika förskjutningar. De skift som är multiplar av nyckellängden kommer att ha fler tillfälligheter än de som inte är det. Från detta kan vi härleda nyckellängden. Nu tar vi chiffertexten och XOR den med sig själv förskjuten av nyckellängden. Detta kommer i princip XOR texten med sig själv och tar bort nyckeln helt. Från detta kan vi sedan börja tillämpa några bokstavsfördelningar för att försöka knäcka XOR-meddelandet.

Denna typ av chiffer används i olika applikationer överallt. Den kanske mest allvarliga användningen av XOR-chiffer är med ett Windows CE-system: du ställer in ett lösenord som används för att skydda din bärbara enhet. När du slår på enheten uppmanas du att ange lösenordet. Enkelt nog. Hur som

helst, ActiveSync, programmet du kör så att du kan synkronisera data på din bärbara enhet med skrivbordet, använder samma lösenord. När du startar ActiveSync uppmanas du att ange ditt Windows CE-lösenord. Du får möjlighet att spara lösenordet på din stationära maskin så att du inte behöver ange det varje gång du vill synkronisera dina maskiner. Det sparar lösenordet, krypterat, i ditt register.
Krypteringen den använder är dock en enkel XOR-chiffer med 'susageP' som nyckel. Varför den här nyckeln? Tja, kodnamnet för Windows CE var 'Pegasus' och om du vänder på det får du 'susageP'

Egentligen, efter att ha insett hur dåligt XOR-chiffret är, finns det ett sätt på vilket det kan användas för att producera ett obrytbart chiffer.

Detta obrytbara chiffer är känd som engångskoden. (Sparbanken använder detta i s.k. kodkort) Klassiskt var engångskoden ett Vigenère-chiffer, men med en nyckel som var helt slumpmässig och lika lång som klartexten.

### 3.6.1 Vad som händer är detta:

Alice och Bob träffas och skapar ett block med sidor (tänk ”post-it block), där varje sida har en sekvens av bokstäver på den genereras slumpmässigt. Blocket dupliceras med Alice som tar den ena och Bob som tar den andra. Nu när Alice vill skicka ett meddelande till Bob använder hon standard Vigenère-chiffer, men där varje bokstav i klartexten är gift med varje bokstav i följd från den övre sidan av post-it blocket. Hon fortsätter att kryptera klartexten och använder så många sidor i post-it blocket som krävs. När hon är klar skickar hon chiffertexten och förstör sedan de sidor hon har använt. Bob använder sitt block på samma sätt för att dekryptera chiffertexten och när han är klar förstör han samma sidor. Eftersom nyckeln är helt slumpmässig och dessutom aldrig används igen (därav engångskoder), det finns inget sätt att Eva kan få grepp om den ursprungliga klartexten. Hon kan inte använda bokstavsfrekvenser eftersom samma bokstav slumpmässigt krypteras med en annan nyckelbokstav varje gång. Det finns ingen attack för att försöka härleda nyckellängden, eftersom nyckelns längd är densamma som längden på klartexten. Helt okrossbart.
I XOR-fallet händer samma sak. En mycket stor uppsättning slumpmässiga byte genereras och Alice och Bob får båda en kopia. Alice kodar sin klartext genom att extrahera ut samma antal byte från sin slumptalstabell som det finns byte i klartexten och sedan XORs dem tillsammans. Chiffertexten skickas och Alice förstör de slumpmässiga byte hon har använt. Bob får chiffertexten och använder samma antal slumpmässiga byte som det finns byte i chiffertexten och förstör de använda slumpmässiga bytena på samma sätt som Alice. Återigen har Eve ingen chans att dekryptera chiffertexten av samma skäl som i det klassiska fallet.

Tänk på att om Alice och Bob skapade sina slumpmässiga byte genom att seeda en slumptalsgenerator och sedan ringa den för att generera slumpmässiga byte, har Eve en chans igen. Eftersom hela sekvensen av slumpmässiga byte som genereras av denna metod är deterministisk (med andra ord, om Eva började med samma frö, skulle hon få samma slumpmässiga byte), har hon en springa som hon kan öppna. Allt hon behöver göra är att prova varje frö i följd, tills chiffertexten är knäckt. Om du vill blir engångsplattan plötsligt en krypteringsmetod med en 32-bitars nyckel och om Eve har en tillräckligt snabb maskin skulle hon kunna knäcka chiffertexten ganska snabbt. Ändå är sådana randomiserade krypteringsalgoritmer ganska populära och det finns en hel delmängd av slumptalsgeneratorn som är utformade för att vara kryptografiskt säkra.

## 3.7 Din fantasi

Efter all denna utställning om substitutionschiffer bör vi ta en titt på transpositionschiffer, den andra klassiska metoden för kryptering av klartext. Ett transponeringschiffer är ett där bokstäverna i

klartexten inte konverteras på något sätt; det är bara deras ordning som är hopblandad. Det enklaste av dessa chiffer är det kolumnära transponeringschiffret. Skriv klartexten på kvadratiskt papper, med en bokstav per ruta. Begränsa bredden på det kvadrerade papperet till ett visst antal celler. Läs av chiffertexten en kolumn i taget. Så, till exempel, om klartexten var " ONCE MORE UNTO THE BREACH DEAR FRIENDS ", och vårt fyrkantiga papper var 10 lådor brett, skulle vi få
ONCEMOREUN
TOTHEBREAC
HDEARFRIEN
DS
Genom att läsa bokstäverna vertikalt får vi chiffertexten 'OTHDNODS ...' För att dekryptera detta meddelande måste vi veta bredden på det kvadrerade papperet. Räkna bokstäverna i chiffertexten och dela detta med bredden för att ge dig antalet bokstäver att skriva vertikalt och så möjligheten att avkoda chiffertexten.
Under första världskriget utformade tyskarna ADFGVX-chiffret, en blandning mellan ett transpositionschiffer och ett substitutionschiffer. Det var en sofistikerad krypteringsalgoritm för sin tid. Det är också en "ordig" algoritm, eftersom varje bokstav i klartext omvandlas till två bokstäver i den slutliga chiffertexten. Skapa en 6 * 6 kvadrat, namnge varje rad och kolumn efter bokstäverna A, D, F, G, V, X, som i figur 2.

Slumpmässigt sätta varje bokstav i alfabetet i denna kvadrat, tillsammans med de tio siffrorna (totalt 36 tecken). Det första steget är ett substitutionschiffer: vi ersätter varje bokstav i klartexten med rad- och kolumnidentifieraren beroende på var bokstaven hittades. Till exempel, med hjälp av figur 2, skulle klartextbokstaven J ersättas med DF, eftersom J ligger vid korsningen av rad D och kolumn F.

| | A | D | F | G | V | X |
|---|---|---|---|---|---|---|
| A | U | H | N | A | X | O |
| D | 2 | B | J | V | D | 4 |
| F | P | T | 3 | K | 5 | G |
| G | E | 1 | I | S | 9 | 7 |
| V | 0 | F | 8 | C | W | Y |
| X | Z | Q | M | R | 6 | L |

Nu måste vi bestämma ett lösenord. Låt oss använda ordet SECRET. Kassera alla upprepade bokstäver i lösenordet, så vi har SECRT. Skriv ut den mellanliggande chiffertexten (den som bara består av ADFGVX-bokstavspar) under ordet SECRT som i standardkolumntransponeringschiffret. Här har vi naturligtvis bara fem kolumner. Nu läser vi av bokstäverna i kolumnerna, inte från första till sista, utan enligt kolumnrubrikens position i alfabetet. Så för SECRT läser vi C-kolumnen först, sedan E-kolumnen, R-kolumnen, S-kolumnen och slutligen T-kolumnen. Resultatet är ett hopplock av As, Ds, Fs, Gs, Vs och Xs. För att dekryptera måste vi känna till lösenordet och även ersättnings-tabellen.

För att demonstrera algoritmen bestämmer sig Alice för att skicka meddelandet

**THEYAREONTOYOU.**

Hon utför det första substitutionschiffret och ersätter T med FD, H med AD och så vidare. Hon placerar de resulterande bokstavsparen i en 5-kolumns införlivandetabell under ordet SECRT, som i figur 3. Hon läser sedan C-kolumnen först: AXGFV, E-kolumnen: DVGAX och så vidare, för att göra chiffertexten AXGFVDVGAX ...

Bob kan rekonstruera transpositionstabellen eftersom han känner till lösenordet, och eftersom han också känner till ersättningstabellen kan han räkna ut den ursprungliga klartexten. Och stackars Eva? Tja, det verkar inte finnas något rim eller anledning till samlingen av dessa 6 bokstäver, upprepade och blandade till synes slumpmässigt. Men överraskande bröts detta chiffer faktiskt med det faktum att de två stegen är helt frånkopplade från varandra och att överföringslösenordet inte används i substitutionen.

Koden? Tja, det är väldigt rörigt, som du kan föreställa dig. Lista 4 visar hela ADFGVX-chifferkoden och som du kan se kan vi inte dela mycket kod mellan kryptering och dekryptering som i de tre föregående chiffren. Den här månadens disk har också en rutin för att generera en slumpmässig blandning av bokstäverna och siffrorna för ersättningstabellen.

```
procedure AAADFGVXCipher(aEncrypt : boolean; aKey : string;
  const aSubstTable: TaaADFGVXTable; aInStream : TStream;
  aOutStream : TStream);
const
  ADFGVX : array [0..5] of char = 'ADFGVX';
var
  BytesRead : longint;
  i, j      : integer;
  Ch        : char;
  Buf       : array [0..255] of char;
  DblBuf    : array [0..511] of char;
  CleanKey  : string;
  KeyLen    : integer;
  KeyInx    : integer;
  PosCh     : integer;
  MinInx    : integer;
  ColLen    : integer;
  Row, Col  : integer;
  SubstTextSize : longint;
  InStreamSize  : longint;
  MemStream     : TMemoryStream;
begin
  {clean up the key so that it consists only of unique
    uppercase characters}
  CleanKey := '';
  for i := 1 to length(aKey) do begin
    Ch := upcase(aKey[i]);
    if ('A' <= Ch) and (Ch <= 'Z') then
      if (Pos(Ch, CleanKey) = 0) then
        CleanKey := CleanKey + Ch;
  end;
  {===ENCRYPTION===}
  if aEncrypt then begin
    {read the entire input stream, converting into
      letterpairs into a temporary stream}
    MemStream := TMemoryStream.Create;
    try
      BytesRead := aInStream.Read(Buf, sizeof(Buf));
      while (BytesRead > 0) do begin
        j := 0;
        for i := 0 to pred(BytesRead) do begin
          Ch := upcase(Buf[i]);
          if (('A' <= Ch) and (Ch <= 'Z')) or
             (('0' <= Ch) and (Ch <= '9')) then begin
            PosCh := Pos(Ch, aSubstTable) - 1;
            DblBuf[j] := ADFGVX[PosCh div 6];
            DblBuf[j+1] := ADFGVX[PosCh mod 6];
            inc(j, 2);
          end;
        end;
        MemStream.Write(DblBuf, j);
        BytesRead := aInStream.Read(Buf, sizeof(Buf));
      end;
      {now read the letters in each column according to the
        order of the letters in the cleaned key}
      KeyLen := length(CleanKey);
      for KeyInx := 1 to KeyLen do begin
        {find the smallest letter in the key, this is the
          column we'll be reading next}
        MinInx := 1;
        for i := 2 to KeyLen do
          if (CleanKey[i] < CleanKey[MinInx]) then
            MinInx := i;
        CleanKey[MinInx] := #127; {so we don't see it again}
        dec(MinInx); {it's easier with a 0-based number}
        {starting off with the MinInx'th letter in the
          temporary stream, copy it and every KeyLen'th
          letter after that to the output stream}
        SubstTextSize := MemStream.Size;
        PosCh := MinInx;
        j := 0;
        while (PosCh < SubstTextSize) do begin
          MemStream.Position := PosCh;
          inc(PosCh, KeyLen);
          MemStream.Read(DblBuf[j], 1);
          inc(j);
          if (j = sizeof(DblBuf)) then begin
            aOutStream.Write(DblBuf, sizeof(DblBuf));
            j := 0;
          end;
        end;
        if (j > 0) then
          aOutStream.Write(DblBuf, j);
      end;
    finally
      MemStream.Free;
    end;
  end else begin
  {===DECRYPTION===}
    {first create memory stream to use as an intermediary,
      set its size to the size of the input stream}
    InStreamSize := aInStream.Size;
    MemStream := TMemoryStream.Create;
    try
      MemStream.SetSize(InStreamSize);
      {now read the letters in each column according to the
        order of the letters in the cleaned key}
      KeyLen := length(CleanKey);
      for KeyInx := 1 to KeyLen do begin
        {find the smallest letter in the key, this is the
          column we'll be reading next}
        MinInx := 1;
        for i := 2 to KeyLen do
          if (CleanKey[i] < CleanKey[MinInx]) then
            MinInx := i;
        CleanKey[MinInx] := #127; {so we don't see it again}
        dec(MinInx); {it's easier with a 0-based number}
        {calculate the length of this column}
        ColLen := InStreamSize div KeyLen;
        if ((InStreamSize-(ColLen*KeyLen)) > MinInx) then
          inc(ColLen);
        {copy the column from the input stream to the
          temporary stream, starting off by copying to the
          MinInx'th letter in the temporary stream, and
          every KeyLen'th letter after that; we stop at the
          end of the column}
        PosCh := MinInx;
        while (ColLen > 0) do begin
          if (ColLen > sizeof(Buf)) then
            BytesRead := aInStream.Read(Buf, sizeof(Buf))
          else
            BytesRead := aInStream.Read(Buf, ColLen);
          dec(ColLen, BytesRead);
          for i := 0 to pred(BytesRead) do begin
            MemStream.Position := PosCh;
            inc(PosCh, KeyLen);
            MemStream.Write(Buf[i], 1);
          end;
        end;
      end;
      {now read the temporary stream as letter pairs,
        converting them into the original characters for the
        output stream}
      MemStream.Position := 0;
      BytesRead := MemStream.Read(DblBuf, sizeof(DblBuf));
      j := 0;
      Row := 0;
      Col := 0;
      while (BytesRead > 0) do begin
        for i := 0 to pred(BytesRead) do begin
          if not Odd(i) then begin
            case DblBuf[i] of
              'A' : Row := 0;
              'D' : Row := 1;
              'F' : Row := 2;
              'G' : Row := 3;
              'V' : Row := 4;
              'X' : Row := 5;
            end;
            case DblBuf[i+1] of
              'A' : Col := 0;
              'D' : Col := 1;
              'F' : Col := 2;
              'G' : Col := 3;
              'V' : Col := 4;
              'X' : Col := 5;
            end;
            Buf[j] := aSubstTable[Row * 6 + Col + 1];
            inc(j);
            if (j = sizeof(Buf)) then begin
              aOutStream.Write(Buf, sizeof(Buf));
              j := 0;
            end;
          end;
        end;
        if (j > 0) then
          aOutStream.Write(Buf, j);
        BytesRead := MemStream.Read(DblBuf, sizeof(DblBuf));
      end;
    finally
      MemStream.Free;
    end;
  end;
end;
```

Kod listning 2

### 3.8 Letar du efter ett gott tecken

Som en sista anmärkning i den här artikeln om klassiska chiffer vill jag nämna ENIGMA-maskinen. Detta var den krypteringsenhet som användes av den tyska armén och marinen under andra världskriget. Jag skulle snabbt vilja diskutera det av två skäl. För det första var det en mycket sofistikerad krypteringsmaskin. Och för det andra föll brytningen av dess kryptering på ett team av matematiker i Bletchley Park i Buckinghamshire, inklusive en av fäderna till modern databehandling, Alan Turing. ENIGMA-maskinen var i huvudsak en rotormaskin med tre rotorer. En rotor är ett hjul som är kopplat för att utföra en monoalfabetisk substitution. Hjulet var uppdelat i 26 segment, ett för varje bokstav, och var utformat på ett sådant sätt att varje segment på ansiktet var kopplat till

ett annat segment på baksidan. Om varje segment på baksidan var anslutet till ett ljus, skulle applicering av elektrisk ström på ett segment i ansiktet få ljuset för den krypterade bokstaven att tändas. ENIGMA-maskinen kopplade sina tre rotorer i serie, så varje rotor applicerade en substitution. För att säkerställa att samma substitution inte användes om och om igen skulle rotorerna roteras av ett antal segment efter att en bokstav krypterats. Dessutom använde ENIGMA-maskinen konceptet med en reflekterande rotor som fick kretsen att reflekteras genom de tre rotorerna igen, bakåt, dubbelt krypterande varje klartextbokstav. Det fanns ett tangentbord genom vilket klartexten kunde skrivas för kryptering (eller chiffertexten för dekryptering), åtgärden att trycka på en tangent skulle utföra rotationerna och den resulterande chiffertextbokstaven lästes av en uppsättning med 26 lampor när ström applicerades genom rotorerna. För att göra det ännu svårare att spricka kan rotorernas ursprungliga position definieras innan meddelandet krypteras eller dekrypteras. Det fanns också en ett kopplingsbord som om mappades med bokstavspar före krypteringen.  Sammantaget en skräckinjagande krypteringsmaskin.

Att bryta ENIGMA-krypteringen förlitade sig på komplex matematik från de mest lysande matematikerna i England, och också en hel del tur. De allierade hade lyckats få en ENIGMA-maskin och några av kodböckerna (dessa böcker detaljerade vilka rotorer som skulle användas på vilka dagar, rotorernas initiala positioner och så vidare), och de hade ett lyckokast genom att operatörerna ogillade att byta rotorer (tydligen en tråkig process), så en av krypteringsmöjligheterna förnekades. Ändå lyckades det brittiska teamet, inklusive Alan Turing, bryta krypteringen genom att bygga en maskin som heter The Bombe, som hjälpte till att analysera chiffertexten. Med tanke på vikten av Turings arbete under kriget var hans efterföljande behandling av den brittiska regeringen 1952, när han arresterades för en homosexuell förbindelse och fråntogs sitt säkerhetsgodkännande, inget annat än skandalöst och ledde förmodligen till hans död genom självmord 1954.