Metoder för sekretess, integritet och autenticering

Kryptering

- Att dölja (grekiska)
- Sekretess
- Algoritmen

Att dölja
Ordet kryptering kommer från grekiskan och betyder dölja. Rent historiskt sett har man alltid velat gömma information, men delta har framfor allt gällt militären. Anledningen till att kryptering blivit så stort de senaste åren beror på att att fler företag och privatpersoner väljer att utbyta information på osäkra medium så som Internet.

Sekretess
Del man vill åstadkomma ar att erhålla sekretess av information. Delta löses genom att dölja Martext genom att omvandla den till kryptotext, Mottagaren omvandlar tillbaka kryptotexten till den ursprungliga klartexten.

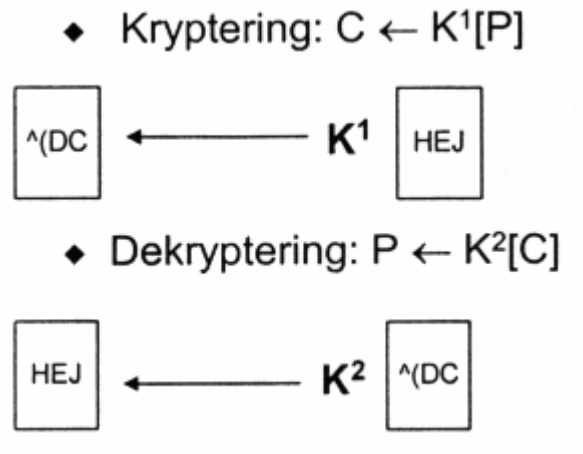
Algoritmen
Det som har skett det senaste århundradet är att själva algoritmen i sig är känd, del som ar unikt är nycklarna som används. Redan på Caesars tid krypterade man meddelanden med då var det själva algoritmen som var hemlig. Caesars chiffer bygger på att man krypterar text genom att flytta varje bokstav för sig tre steg åt höger i alfabetet. För att dekryptera det flyttar man tillbaka tre steg åt vänster. Liknade algoritmer finns t.ex. ROT13 som används i UNIX-världen.

Ex Caesar

abcdefghijklmnopqrstuvxyz defghijklmnopqrstuvxyzabc
klartext: hej alla glada
kryptotext: khm doodjodgd

Krypteringsalgoritmer

Definition



Genom att ta klartexten (p=plantext) tillsammans med en algoritm och en nyckel (K') far man kryptotext (chiphertext). Mottagaren tar kryptotexten tillsammans med samma algoritm och nyckel (K1) och erhåller den ursprungliga klartexten.

Symmetriska algoritmer

- K1=K2
- Samma *nyckel for kryptering som dekryptering*
- *Om obehörig* erhåller *nyckel kan den bads läsa krypterad data samt manipulera data.*

Nyckeln skall inte *överföras på ett osäkert medium*
Exempel: DES, IDEA, Blowfish

Asymmetriska algoritmer

- K1*K2
- *Olika nycklar för kryptering som för dekryptering*
- Kan *överföras på ett osäkert medium*
- *kräver mer CPU-kraft beräkning*

Exempel: RSA, El-Gamma

DES

- *DES (Data Encryption Standard)*
- *Utvecklat av IBM*
- *Krypterar block om 64 bits data*
- *Nyckel: 56 bitar plus 8 bitar paritet*

Finns som:

ECB
CBC
CFB
OFB
Triple DES

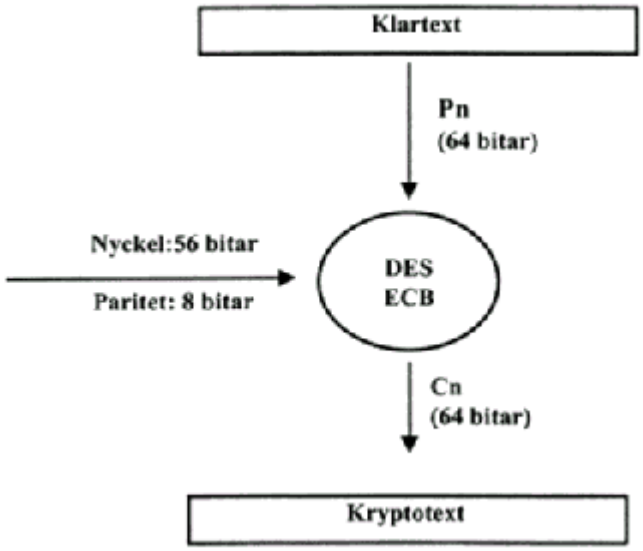
DES

Behovet av att kunna kryptera data ledde till att NBA (National Bureau of Standards) 1972 gav IBM i uppdrag att utveckla en algoritm. Algoritmen skulle följa vissa fördefenierade krav:

- Skall leverera en hög säkerhet
- Skall vara lätt att första
- Säkerheten i algoritmen skall utgöras av nyckeln
- Ekonomiskt bärande. osv

IBM arbetade på uppdraget och den färdiga standarden publicerades av NBS i Mars 1975. Algoritmen är helt öppen och baseras på vanliga skiftregister och XOR-operationer vilket samtliga processorer kan hantera. Den ar fast knuten till en nyckellängd av 56 bitar plus 8 bitar paritet. Data matas in i algoritmen om 64 bitar. Algoritmen har blivit den *mest* använda i datorvärlden.

DES - ECB Electronic Code Book



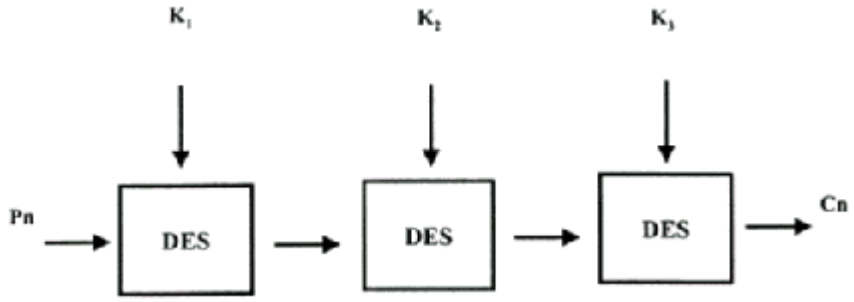
DES-ECB (Electronic Code Book)

Den enklaste formen av DES och bygger på att man matar in datablock om 64 bitar, Den nyckel man använder består av 56 bitar nyckel och 8 bitar paritet. Man använder samma nyckel for kryptering som för dekryptering eftersom algoritmen är symmetrisk.

Kryptering:	Cn <- K[Pn]
Dekryptering:	Pn <- K[Cn]
Cn:	Kryptotext block nr n
Pn:	Klartext block nr n

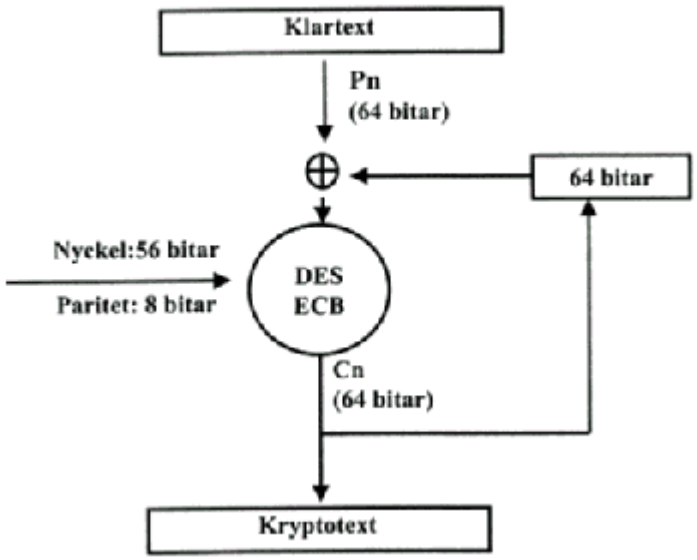
Problemen med den formen av DES är att block inte har någon inbördes relation med varandra, vilket leder till att man kan manipulera med data. Ett annat problem är att långa sekvenser av indata leder till samma karaktär av utdata. Delta får som följd att man kan minimera antalet tänkbara nycklar som används.

Trippel DES



Trippel DES
Behovet av längre nyckellängder gjorde alt IBM tog fram en ny variant av DES. Problemet ligger i att DES har en fix nyckellängd på 56 bitar. I trippel DES körs krypteringen i tre steg. En vanlig missuppfattning är att nyckellängden blir 56 bitar * 3 = 168 bitar. I själva verket används 56bitar * 2 = 1 1 2 bitar.

DES - CBC Cipher Block Chaining



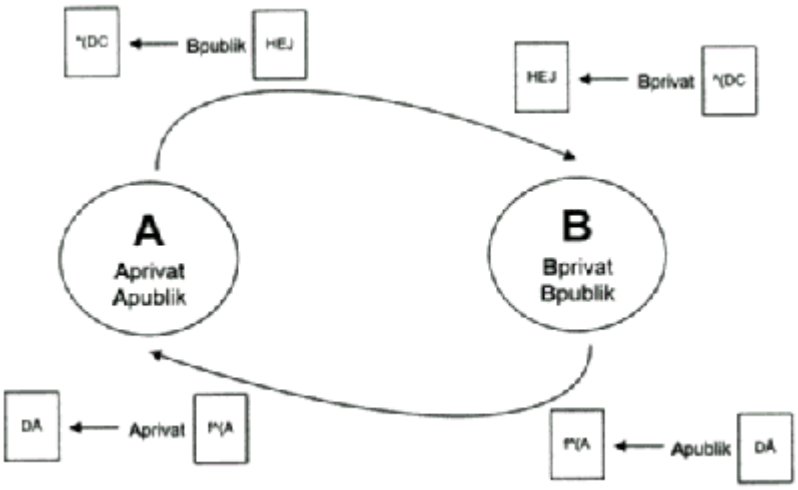
DES-CBC (Cipher Block Chaining)

För att lösa problemen som uppstår i ECB tog IBM fram en ny variant av DES. Grundtaken är alt göra blocken beroende av varandra (chaining) genom utnyttja utdata i nästa block av indata. Detta utförs mha av en vanlig XOR-funktion.
Kryptering: Cn <- K[Pn xor Cn-1]
Dekryptering: Pn <- K[Cn xor n-1]
Cn: Kryptotext block nr n
Pn: Klartext block nr n

Resultatet blir att blocken har relation till varandra samt att man inte kan minimera antalet tänkbara nycklar.

Asymmetrisk algoritm

Varje användare genererar ett nyckelpar:



Asymmetriska algoritmer
Bygger på att varje användare genererar ett nyckelpar mha av en asymmetrisk algoritm. Detta leder till att användaren får en privatnyckel och en publiknyckel. Den privata nyckel är hemlig och förvaras på ett säkert ställe eller gömd på dator i form av ett skyddande lösenord (Passphrase).
Den publika nyckeln är offentlig och kan distribueras på Internet.

1. Om användare A vill skicka ett krypterat meddelande till B. Letar denne upp B's publika nyckel som är offentlig. A tar dokumenten och krypterar den med B's publika nyckel. Där efter skickar A över dokumenten.

2. Användare B som skall läsa dokumentet använder sin privata nyckel för att dekryptera. Det är endast B's privat nyckel som kan dekryptera eftersom den genererades tillsammans med den. B’s publika nyckel användes av A vid kryptering.

3. Om användare B vill skicka ett krypterat meddelande till A. Letar han upp A's publika nyckel som är offentlig. B tar dokumenten och krypterar den med A's publika nyckel. Därefter skickar han över dokumenten.

4. Användare A som skall läsa dokumentet använder sin privata nyckel för att dekryptera. Det är endast A's privata nyckel som kan dekryptera eftersom den genererades tillsammans med den. A’s publika nyckel av B vid kryptering.

RSA

- Uppfunnet av Roland Rivest, Adi Shamir och Leonard Adleman
- Publicerades 1978
- Asymmetrisk algoritm
- Långsammare att kalkyler an DES

RSA
Är den mest kända och nyttjade asymmetriska algoritmen. Algoritmen är oberoende av nyckelängden och använder vanliga matematiska funktioner som finns implementerade i processorer. Problemet är att algoritmen är långsammare att kalkylera än DES vilket leder till att RSA inte går att använda i realtids tillämpningar.

Tag två primtal p och q n=pq
Tag talet d, relativt primtalet med (p-1) (q-1) och e sa att ed=1 (mod (p-1) (q-1))

(e,n) blir den ena nyckel och (d,n) den andra

Kryptering: $C = P^e \pmod n$
Dekryptering: $P = C^d \pmod n$

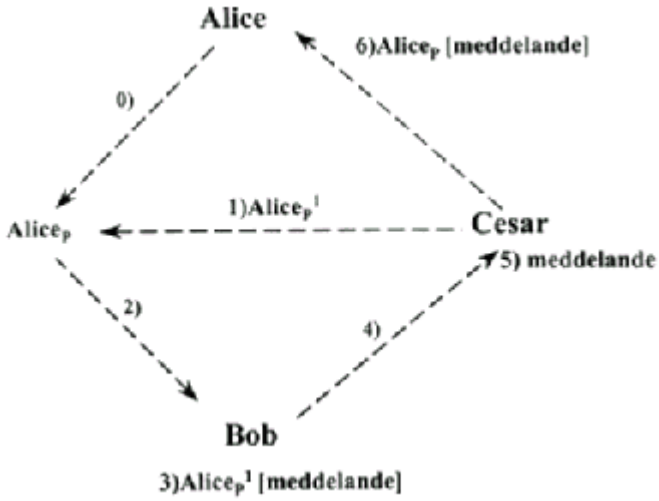
Ex = 2773, e = 17, d =157

Kodning av indata: mellanslag - 00, A = 01, B = 02, ..., 2 = 26

Indata: ITS ALL GREEK TO ME = 0920 1900 0112 1200 0718 0505 1100 2075 0013 0500

Kryptering av första blocket: 0920 mod 2773 = 0948
Kryptering av hela meddelandet => 0948 2342 1084 1444 2663 2390 0778 0774 0219 1655
Dekryptering av första blocket: 0948^157mod 2773 = 0920

Svaghet hos asymmetriska



1) Caesar byter ut Alice publika nyckel
2) Bob hämtar den förfalskade publika nyckeln
3) Bob krypterar med den falska nyckeln
4) Caesar fångar upp meddelandet
5) Caesar dekrypterar med den falska privata nyckeln
6) Caesar krypterar med Alice riktiga publika nyckel

Nycklar

- Kräver mer processorkraft
- Går ej matematiskt att bevisa styrkan ("brute force"-attack)

Processorkraft

Det största problemet med asymmetriska algoritmer är att de tar mycket längre tid att beräkna. Detta medför att asymmetriska algoritmer inte kan användas vid realtidssessioner. I stället nyttjas symmetriska algoritmer, problemet där är att nyckeln inte kan utbytas på att osäkert nät. En kombination av bada ger en lösning av problemen.

Ett exempel är Diffie-Hellman.

Styrka

Ett annat problem är att det inte matematiskt går att visa hur bra en algoritm är. Det enda man kan göra är en sa kallad "Brute force "-attac. Vilket innebär att man gissa på alla tänkbara nycklar. Styrkan ligger i hur lång tid det tar att hitta rätt nyckel. När det gäller styrkan kan man heller inte veta om det i framtiden kommer fram matematiska slutsatser som gör att algoritmen kan forceras.

Säkerhet och kryptering

- Nyckeladministration
- Pålitligheten i programvaruimplementationen
- Privata nyckeln:

  Passphrase

  Smart cards

Nyckeladministration/programvaruimplementation,

- Var finns den private nyckeln ?
- Hur skyddas den privata nyckeln ?
- Hur genereras nycklarna?
- Ar nyckellängden verkligen den som företaget påstår?
- Finns del en bakdörr i programmet ?

Meddelande integritet

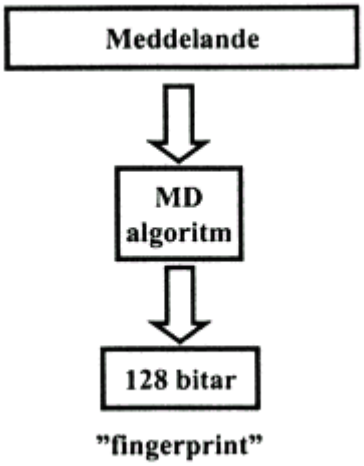
- Integritet = informationen har inte manipulerats

  Message Digests (MD5)

- Äkthet = informationen ar bunden till ex en person

  Digital signatur

Message Digest

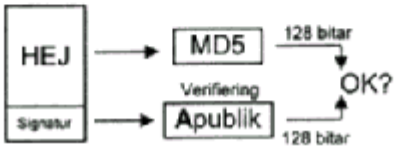


- Checksumma
- Utvecklat av RSA m RFC-standardiserad
- Finns som:
  MD2 (RFC 1319)
  MD4 (RFC 1320)
  MD5 (RFC 1321)

Digital signatur

- *Avsändaren (A)*

- *Mottagaren (B)*



Digital signatur

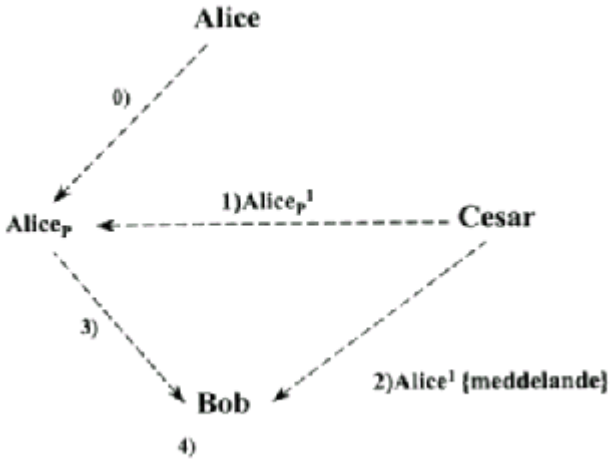
För att utföra digital signatur används både en asymmetrisk och en hash funktion som MD5. Anledningen är att man vill ha ett läsbart dokument trots krypteringen. I stället för att utnyttja mottagarens publika nyckel som vid kryptering, nyttjas den egna privata. Det ger en signering, dvs. ett bevis på att det är användaren som skickat filen eftersom det bara är han som har tillgång till nyckeln.

Användare A vill skicka ett dokument till B som är signerat dvs. ett bevis på att det är A som är avsändare:

1. Användare A kör dokumentet genom MD5-algoritmen och erhåller en checksumma om 128 bitar. Efter det signeras de 128 bitarna med sin den privata nyckeln, lägger ihop den med orginaldokumentet och skickar över informationen till B.
2. Användare B för dokumentet läsbart men måste verifiera att det kommer från A. Detta görs dels genom att köra dokumentet genom MD5-algoritmen samt genom att ta signaturen och verifiera med A's publika nyckel. Stämmer de 128 bitarna från MD5-algoritmen med utdata från verifieringen betyder det att dokumentet kommer från A och ej manipulerats på vägen.

Man kan kombinera med kryptering också, men då måste man se till att ordningsföljden är korrekt annars kommer inte signeringen att stämma, datan är manipulerad.

Svaghet hos signering



1) Caesar byter ut Alice publika nyckel
2) Caesar sänder ut meddelande som signerats med Alice falska privata nyckel
3) Bob hämtar den förfalskade publika nyckeln
4) Bob tror att han har fått ett signerat meddelande från Alice

CA

CA (Certification Authority) Signerar publika nycklar

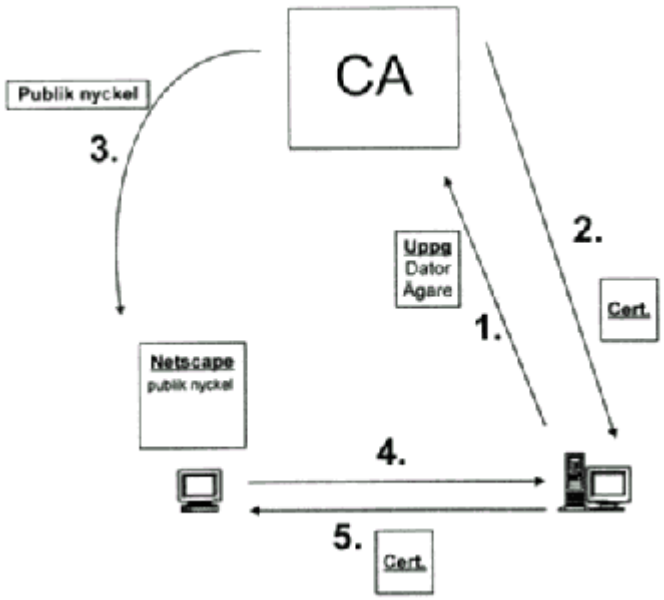
- *banker,* Staten, *privata företag*
- *AT&T. CertiSign, IBM, MCI,*
- *VeriSign*

CA (Certification Authority)
Har som uppgift att "säkra" andras publika nycklar genom att signera dem med sin privata nyckel.

Vanligt förekommande när man behöver kunna verifiera sig för olika servrar. Man kopplar upp sig mot CA som certifierar nyckeln.

Många siter motverifierar din nyckel via CA.

Exempel på CA



Certifikat

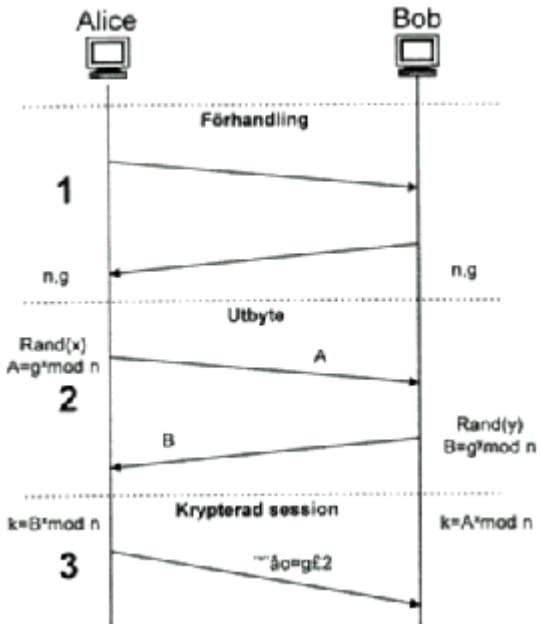
1. Företaget önskar att CA skall gå i god för deras certifikat och sänder över uppgifter om Företaget, typ adress, telefonnummer med mera. Detta kan låta sig göras via fax eller on-line på webben.
2. CA verifierar uppgifterna och beslutar om de vill gå i god för Företaget. CA skapar ett certifikat med de översända verifierade uppgifterna, dvs en checksumma som de signerar med sin privata nyckel.
3. De CA som finns på marknaden ser till att deras certifikat finns med i så många applikationer som möjligt på marknaden
4. En användare väljer att nyttja Företagets serverdator, en applikation kräver identifiering.
5. Företagets serverdator sänder över certifikatet. Klienten/användaren verifierar certifikatet med de publika nycklar som finns med i applikationen (netscape), och litar på Företagets server.

Här litar klienten på en server (Företagets), det omvända förhållandet kan lösas på ett liknande sätt med inloggning.

Exempel

This Certificate belongs to: www.omicron.se
This Certificate was issued by: Secure Server Certification RSA Data Security, Inc.
OCC AB
Borgarfjordsgatan 13 ,Kista STOCKHOLM,STOCKHOLM,SE
Serial Number: 36:74:8A:76:9F:C1:F3:8F:35:35:6C:BO:B9:76:2A:A5
This Certificate is valid from Thu Dec 28, 2001 to Sun Dec 30,2010
Certificate Fingerprint: 75:37:EZ:74:00:33:97:E7:D4:OD:CO:OE:83:75:24:88

Diffie • Hellman



Diffie -Hellman

Är en algoritm för att fastställa en sessionsnycke mellan två parter över ett publikt nät.

Diffie-Hellman nyttjar en asymmetrisk algoritm för att komma fram till en symmetrisk nyckel som begagnas under den krypterade sessionen. Detta kan ske helt öppet.

1. Alice och Bob förhandlar om talen n och g helt öppet, talen är primtal och uppfyller en viss ekvation

2. Alice slumpar talet X och och beräknar A enligt A = g^x mod n. Därefter sänds talet A till B. Bob slumpar talet y och beräknar B på enligt samma formel som Alice, B = g^y mod n.

3. Alice beräknar k enligt k = B^x mod n och Bob k = A^y mod n, talet k kommer att bli Alice och Bobs gemensamma nyckel under sessionen

Steganografi

GIF BILD => TRIPPEL DES + Passphrase <= Hemligt dokument = GIF BILD & Hemligt dokument.

Med denna typ av kryptering döljer man vilket meddelande som helst i en bild eller en film men även vanliga dokument. Metoden är ytterst kvalificerad.
Det går inte att bevisa att kryptering är utförd eller att det finns ett meddelande dolt i ett annat meddelande. Detta medför juridiska problem.

Metoden användes av Ryska agenter under kalla kriget och även under andra världskriget genom ”teaterspel” mitt framför ögonen på Britter och Amerikanare för information till Tyska agenter.