Kryptering i chattprogram

Mikko Myllyniemi

Referat

I denna avhandling analyseras krypteringsmetoderna och protokollen som används i de populära chattprogrammen Whatsapp och Telegram. Då ett av chattprogrammen är symmetriskt och det andra asymmetriskt kommer det också att presenteras olika styrkor och svagheter mellan de två.

Innehållsförteckning

1. Inledning

Kryptering används överallt nuförtiden, vare sig man själv märker det eller inte. Kryptering kan märkas mest i vardagen i användning av chattprogram, där varje meddelande ska krypteras då det skickas iväg och avkrypteras då det har kommit fram.

Kryptering förekom redan i forntida Egypten och senare i antikens Grekland och romarriket med ett enkelt syfte som man håller fast vid också i dagens läge, att säkerställa ”säker kommunikation” [1]. Detta innebär att bara de som kommunicerar med varandra har tillgänglighet till de meddelanden som skickas. Kryptering går ut på att tecknen i ett meddelande byts ut för att bilda en sträng av synbarligen slumpmässiga tecken som sedan återställs då meddelandet har nått dem man kommunicerar med. Detta görs med hjälp av en nyckel. I modern kryptering är nyckeln en sträng av bitar som används av en krypteringsalgoritm för att bestämma vad ett tecken skall ändras till [2]. Det finns flera olika populära sätt att generera nycklar, och de går ofta ut på att använda sig av stora primtal och moduloräkning.

Ett av de vanligaste hoten man strävar efter att skydda sig från med kryptering är en man-i-mitten attack, vilket innebär att meddelandet på ett sätt eller annat tas fast då det är påväg och läses. [3].

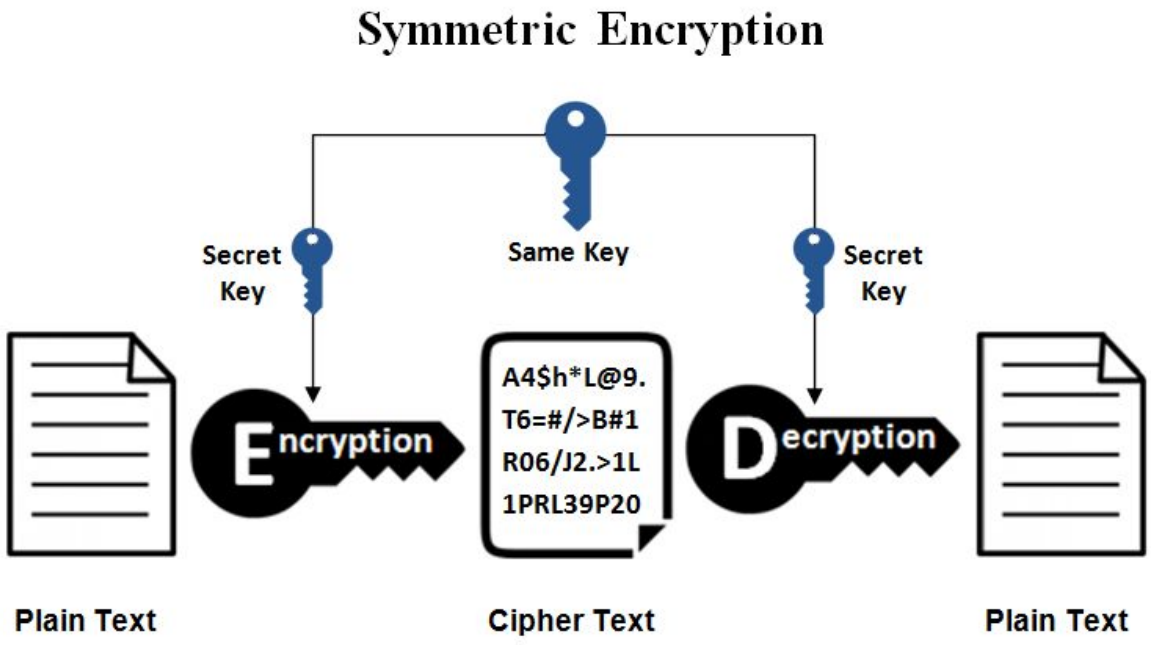
Syftet med avhandlingen är att gå igenom och analysera hur Whatsapp och Telegram sköter kryptering och gå igenom styrkor och svagheter i deras metoder.

2. Bakgrund

2.1. Symmetrisk kryptering

Symmetrisk kryptering är den äldre och förmodligen enklare typen av kryptering. I symmetrisk kryptering används bara en nyckel för att kryptera och avkryptera ett meddelande, och nyckeln delas mellan användarna. Det att man måste dela med sig nyckeln för att kunna kommunicera säkert är en av symmetriska krypteringens största svagheter.

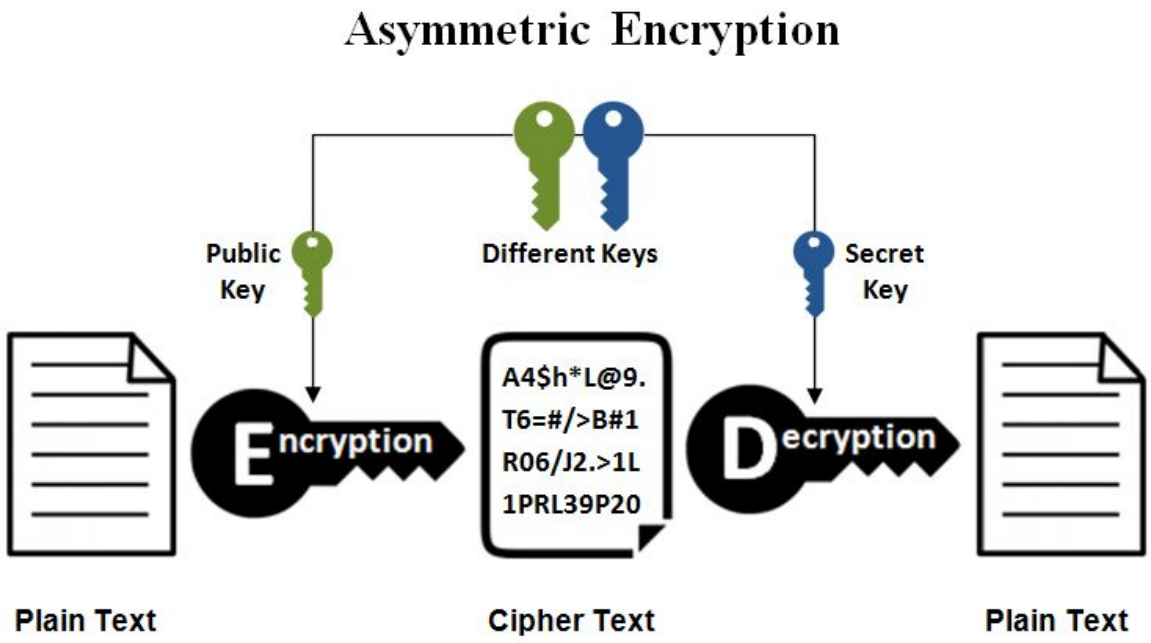
Ett par populära symmetriska krypteringsalgoritmer är Blowfish, AES, DES samt RC4, RC5 och RC6. Nuförtiden används AES mest av dessa.



Figur 1. En bild på symmetrisk kryptering [symcrypt]

2.2. Asymmetrisk kryptering

Asymmetrisk kryptering utvecklades för att bli av med problemet av att använda sig av en nyckel som måste delas ut. Asymmetrisk kryptering går ut på att varje användare har en hemlig och en öppen nyckel. Då man vill skicka ett meddelande åt någon krypterar man meddelandet med mottagarens öppna nyckel, så att mottagaren sedan kan avkryptera meddelandet med sin hemliga nyckel. En av de första asymmetriska krypteringsalgoritmerna, RSA, beskrevs 1978 [RSA orig].

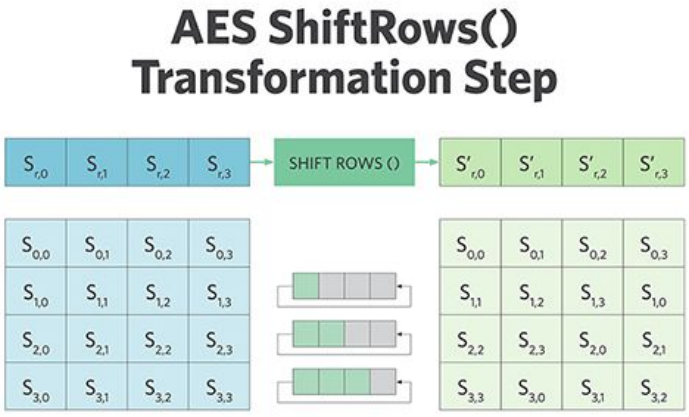


Figur 2. En bild på asymmetrisk kryptering [asymcrypt]

2.3. AES

AES (Advanced Encryption Standard) är en populär symmetrisk krypteringsalgoritm som används överallt i världen och har även valts av USA:s regering för att säkra klassificerad information. Algoritmen utvecklades av NIST (National Institute of Standards and Technology) år 1997, då det upptäcktes att DES (Data Encryption Standard) som användes då var sårbar för brute-force attacker.

I AES krypteras block av data av storleken 128 bitar med hjälp av nycklar som är antingen 128, 192 eller 256 bitar. Rijndael-chiffret som AES är baserat på kunde ta emot andra storlekar också, men den funktionaliteten togs inte med i AES. Krypteringen sker flera gånger i rundor, 10 stycken för 128-bitars nycklar, 12 gånger för 192-bitars nycklar och 14 gånger för 256-bitars nycklar. Varje runda består av flera olika steg, som börjar med att dela på data som skall krypteras och sätta det i en array. Varje tecken ändras sedan enligt en tabell, sedan skiftas raderna som data är i och slutligen blandas kolumnerna om. [4]



Figur 3. En tabell som visar ett exempel på hur rader kan skiftas. [5]

2.4. RSA

RSA är ett av de första asymmetriska krypteringsalgoritmerna och används aktivt än idag. RSA beskrevs av Ron Rivest, Adi Shamir och Len Ableman år 1977. Idén med asymmetrisk kryptering är att båda användarna av ett program som använder sig av asymmetrisk kryptering har en hemlig och en öppen nyckel. Den hemliga nyckeln kan öppna ett meddelande som är krypterat med den öppna nyckeln och vice versa. Då en användare vill skicka ett krypterat meddelande till någon krypteras meddelandet med mottagarens öppna nyckel, och avkrypteras med mottagarens hemliga nyckel. Avsändaren kan ytterligare kryptera meddelandet med sin egen hemliga nyckel, så att mottagaren kan sedan med att avkryptera meddelandet med avsändarens öppna nyckel bekräfta avsändarens identitet **bisatsordning**.

Nyckelgenereringen för RSA består av 4 delar:

1. Välj två stora primtal, p och q. Talen får inte vara samma.
2. Bestäm talet n, med att multiplicera p och q.
3. Välj talet e, så att e och (p-1)(q-1) är sinsemellan relativt prima, samt att $1 < e < (p-1)(q-1)$
4. Välj talet d, så att $ed \equiv 1 \pmod{(p-1)(q-1)}$

Den öppna nyckeln består av talen n och e, den hemliga nyckeln består av talen p,q och d. Ifall nyckeln har genereras enligt reglerna ovan kan man inte räkna ut p och q, även om e och n är kända.

För att kryptera följden av tal, x:

$y = x^e \bmod n$, där y är en krypterad följd av tal.

För att avkryptera följden av tal, y:

$x = y^d \bmod n$, där x är den ursprungliga följden av tal.

3. Whatsapp

Whatsapp är ett program för Android, iOS, Windows Phone, Blackberry OS och Symbian telefoner **telefoner som använder os osv..**, som används för skickandet och mottagandet snabbmeddelanden (instand messaging).
Whatsapp släpptes i januari 2009, och utvecklades av Brian Acton och Jan Koum som då ännu jobbade för Yahoo! [historywhapp].

Ursprungliga versionen av Whatsapp gick ut på att man kunde bestämma ett status som visades till vem än som var i ditt nätverk **undivk ditt**. Då Apple släppte ut en uppdatering som tillät användare att "ping" varandra då de inte var på någon app i juni 2009, ändrade Acton och Koum på sitt program så att då någon i ditt nätverk ändrade på sin status blev man aviserad. Efter det tillade de meddelandehantering och programmet började mer och mer likna hur den ser ut idag. I december samma år tillade de möjligheten att skicka bilder åt varandra genom programmet. Whatsapp började som ett gratis program, men Acton och Koum satt ett pris på programmet bland annat för att sakta ner det våldsamt växande antalet användare.

I februari 2017 hade Whatsapp 1.2 miljarder användare runtom hela världen.

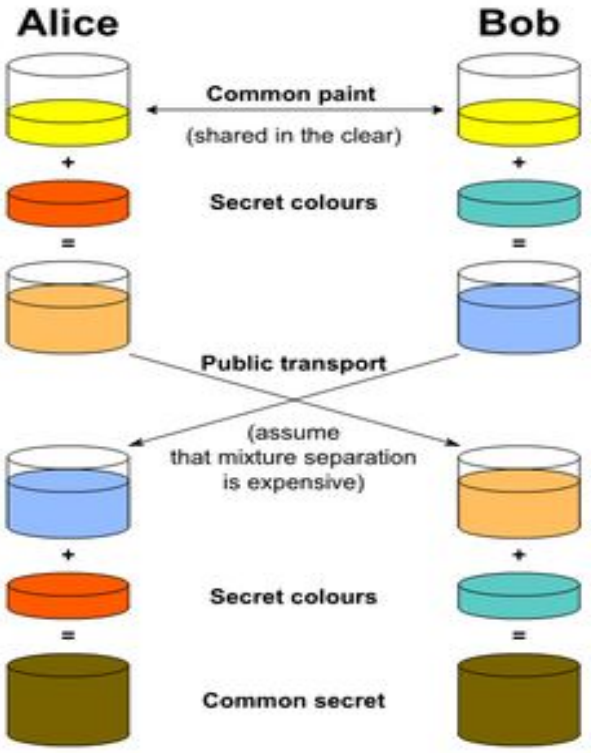
Whatsapp meddelade i november 2014 att de samarbetade med Open Whisper Systems för att säkra Whatsapp med end-to-end kryptering. Open Whisper Systems egna Signal Protocol, som redan användes för en del andra program, användes nu också i Whatsapp. Den 5 april 2016 anmälde Whatsapp tillsammans med Open Whisper System att Whatsapps alla kommunikationsfunktioner var säkrade.

3.1. Signal Protocol

Signal Protocol utvecklades av Open Whisper Systems 2013 och användes först i ett program med öppen källkod, med namnet TextSecure, som senare fick namnet Signal. Protokollet har flera steg och använder sig av Double Ratchet, X3DH (triple Diffie-Hellman), Curve25519, AES-256 och HMAC-SHA256.

3.1.1. X3DH (Triple Diffie-Hellman)

Då den krypterade förbindelsen skapas generas en hel del nycklar: par av identitetsnycklar (identity key eller IK), signerade nycklar (signed prekey eller SPK), kortlivade nycklar (ephemeral key eller EK) och engångsnycklar (one-time prekey eller OPK).
Nycklarna används enligt X3DH. Diffie-Hellman kan bra illustreras med hjälp av färger.



Figur 4. Illustration av Diffie-Hellman Key Exchange

Färgerna i illustrationen är nycklar, och de så kallade ”secret colours” är identitetsnycklar, som bara nyckelns ägare får se. Då mera nycklar (färger) används blir det svårare att få fram de ursprungliga nycklarna som användes och krypteringen blir starkare.

Signal Protocol använder sig av en server som fungerar som ett medium där de nödvändiga nycklarna för kryptering delas ut bland användarna.

Varje nyckel som nämns i exemplet är en öppen nyckel, som har en motsvarande hemlig nyckel. Till att börja med skickar person B sin identitetsnyckel IK_B till servern. Detta en gång och samma identitetsnyckel används fortsättningsvis. Person B skickar också sin signerade nyckel SPK_B och en kombination av IK_B och SPK_B som kallas Sig_B (prekey signature). Ifall dessa nycklar redan finns på servern ersätter de de äldre nycklarna, detta händer ungefär en gång i veckan.

Efter att person B har skickat nycklarna till servern skall person A hämta nycklarna som person B har skickat. Person A hämtar IK_B, SPK_B, Sig_B och, ifall person B skickade den, en engångsnyckel OPK_B. Person B har en bestämd mängd av engångsnycklar som skickas till servern då en förbindelse skapas, men ifall person B inte har engångsnycklar kvar skickas det inte en. Ifall person A lyckas ta emot alla nycklar fortsätter person A med att skapa en kortlivad nyckel EK_A och räknar ut följande värden ifall det inte finns en engångsnyckel:

$$DH1 = DH(IK_A, SPK_B)$$
$$DH2 = DH(EK_A, IK_B)$$
$$DH3 = DH(EK_A, SPK_B)$$
$$SK = KDF(DH1 \,||\, DH2 \,||\, DH3)$$

DH, Elliptic-curve Diffie-Hellman, är ett protokoll som kombinerar två nycklar för att bilda en så kallad ”shared secret”, som är data som bara person A och B har tillgänglighet till.

KDF, key derivation function, är en algoritm som används för att med derivering antingen förlänga en nyckel eller för att ändra formatet på en nyckel.

SK, secret key, är en 32-bitars nyckel som består av KDF av konkateneringen av DH1, DH2 och DH3.

Ifall det finns en engångsnyckel tilläggs:

$$DH4 = DH(EK_A, OPK_B)$$

Och SK ändras till:

$$SK = KDF(DH1 \,||\, DH2 \,||\, DH3 \,||\, DH4)$$

Person A räknar nu ut AD (associated data) av $IK_A \,||\, IK_B$, och skickar sedan iväg ett meddelande åt person B med IK_A, EK_A och AD (AD är krypterade med nyckeln SK) och en identifierare som anmäler vilka signerade och engångsnycklar som har använts. Person A kan sedan fortsätta skicka meddelanden åt person B med att använda SK eller någon nyckel som härrör från SK.

3.1.2. Double Ratchet Algorithm

Då person A och B har fått någon gemensam nyckel ur X3DH använder de Double Ratchet för att härleda två olika nycklar, en root key och en sending chain key. Double Ratchet Algorithm fungerar med hjälp av en Key Derivation Chain, som använder samma Key Derivation Function (KDF) som används i X3DH för att fortsättningsvis skapa nya nycklar.

Idén med Double Ratchet är att man har en sending chain och en receiving chain, som använder nycklar härledda från samma root key. Då person A ska kryptera sitt meddelande för att skickas iväg till person B tar hen ett steg framåt i sin sending chain för att få ut en sending chain key som hen kan kryptera meddelandet med. Då person B sedan tar emot meddelandet tar hen ett steg framåt i sin receiving chain, vilket ger person B en nyckel som hen kan avkryptera meddelandet med. Root key byts ut hjälp av KDF varje gång ett meddelande skickas.

4. Telegram

Telegram är ett molnbaserat chattprogram för Android, iOS, Windows Phone, Windows, Linux och macOS. Telegram utvecklades av Pavel Durov och släpptes i augusti 2013. Programmets klientsida har öppen källkod, men serversidan har inte.

4.1. MTProto

MTProto är ett symmetriskt krypteringsschema som utvecklades av Nikolai Durov med hjälp av andra utvecklare som jobbar för Telegram. MTProto är använder sig av 256-bitars AES kryptering, Diffie-Hellman Key Exchange och 2048-bitars RSA kryptering.

4.1.1. Cloud chats

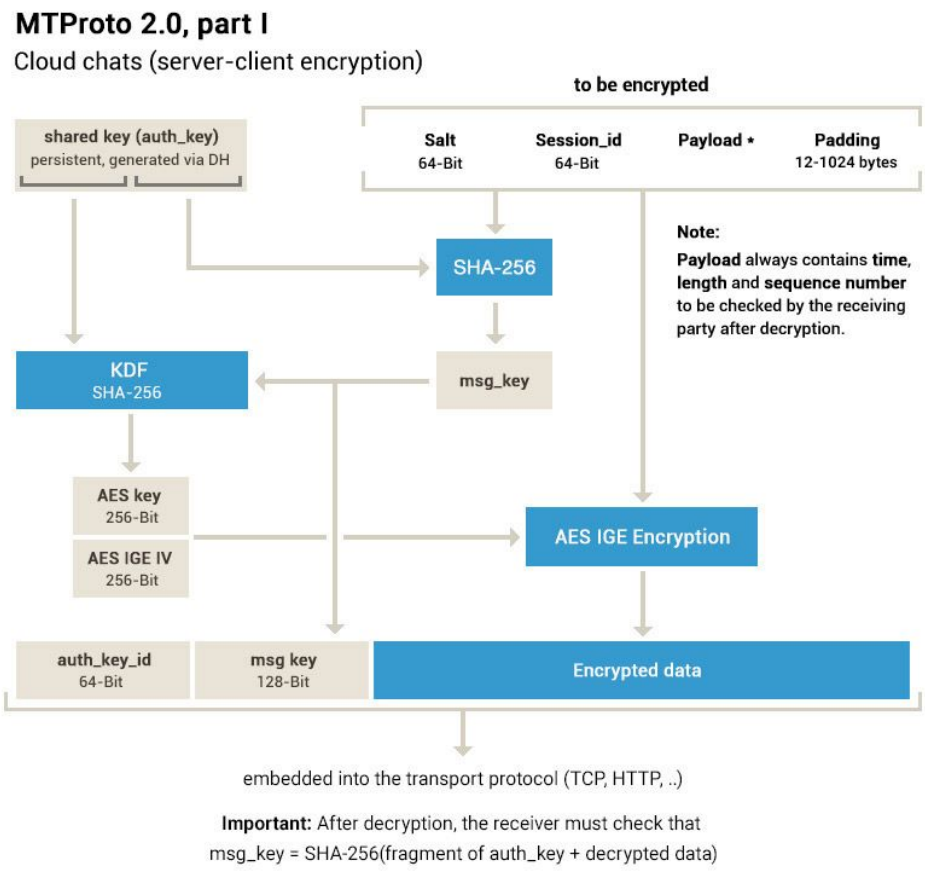
MTProto använder sig av server-client kryptering då det gäller diskussioner som inte är “secret”. Detta innebär att då ett meddelande skickas iväg till någon annan far meddelandet via en server och sedan till mottagaren.

Figur 5 visar hela krypteringsschema för molnbaserade server-client krypteringen. Auktorisationsnyckeln auth_key (authorization key) är en nyckel som ges ur Diffie-Hellman Key Exhange, och är ständig igenom hela krypteringsprocessen. En kopia av nyckeln kombineras med Salt, Session_id, Payload och Padding, och förs igenom SHA-256 för att bilda msg_key (message key).

Salt är en slumpmässig 64-bitars nummer som byts ut med jämna mellanrum.

Session_id är en slumpmässig 64-bitars nummer som används för att skilja åt sessioner ifall mera än en session använder sig av samma auth_key

Padding är 12 till 1024 överlopps bitgruppar som ser till att helheten som skall krypteras är av rätt storlek



Figur 5. Server-client krypteringsschema

Den andra kopian av auth_key kombineras med msg_key med hjälp av KDF för att bilda aes_key (en nyckel som används av AES) och aes_iv (en initialiserings vektor som också används av AES). Salt, Session_id, Payload och Padding krypteras sedan med AES-256, och de krypterade data, msg_key och auth_key_id skickas sedan iväg med ett transport protokoll.

4.1.2. Secret chats

Källor

http://people.csail.mit.edu/rivest/Rsapaper.pdf // RSA orig artic []
https://www.giac.org/paper/gsec/2604/introduction-modern-cryptosystems/104482
// modern cryptosystems []
https://www.incapsula.com/web-application-security/man-in-the-middle-mitm.html // MITM attack
[3]
https://www.techopedia.com/definition/24749/cryptographic-key // nyckel [2]
http://www.cs.technion.ac.il/~cs236506/04/slides/crypto-slides-14-pk-tutor.1x1.pdf
// RSA exempel []
https://access.redhat.com/blogs/766093/posts/1976023 // historia [1]
http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard // AES [4]
https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences
// sym vs asym
https://www.sans.org/reading-room/whitepapers/vpns/history-encryption-730 // brief history
https://www.theverge.com/2016/2/1/10889534/whats-app-1-billion-users-facebook-mark-zuckerberg // whatsapp users
https://www.forbes.com/sites/parmyolson/2014/02/19/exclusive-inside-story-how-jan-koum-built-whatsapp-into-facebooks-new-19-billion-baby/#4272672b2fa1 // history whatsapp
https://www.whatsapp.com/features/ // whatsapp
https://medium.com/@justinomora/demystifying-the-signal-protocol-for-end-to-end-encryption-e2ee-ad6a567e6cb4 // OWS signal protocol (checka deez sources)
https://signal.org/docs/specifications/x3dh/ // x3dh
https://telegram.org/faq // allmän telegram
https://core.telegram.org/mtproto // mtproto
https://www.youtube.com/watch?v=DMtFhACPnTY // sha-256

Bilder

http://cdn.ttgtmedia.com/rms/onlineImages/security-aes_shiftrows_desktop.jpg // AES shift row exempel. [5]
https://www.ssl2buy.com/wiki/wp-content/uploads/2015/12/Symmetric-Encryption.png // sym crypt
https://www.ssl2buy.com/wiki/wp-content/uploads/2015/12/Asymmetric-Encryption.png // asym crypt
https://upload.wikimedia.org/wikipedia/commons/thumb/3/35/Diffie-Hellman_Key_Exchange-modified.png/250px-Diffie-Hellman_Key_Exchange-modified.png //x3dh