**Vragen**

1. Signal – bedrijf: Whisper System
2. Wat is OTR en hoe werkt het
3. Hoe werkt crypto cat (sebas)
4. Wat is Open SSL en hoe werkt het (Sid)
5. Hoe gaan we de applicatie opbouwen ( OTR, OpenSSL en wat gaat erin zitten ) library (sid)
6. Van implementaties die bestaan, welke code gaan we gebruiken? (sebas)
7. Mogelijke issues met OTR en OpenSSL, wat zijn de verschillen?
8. Hoe ga je dat doen met meerdere chatrooms doen (sebas)

Antwoorden

<https://github.com/WhisperSystems/Signal-iOS>

**Wat is Signal**

Signal is een app voor de iPhone, waar het mogelijk is om te chatten en bellen met iemand anders, waarbij veiligheid voorop staat. Signal maakt gebruik van hun eigen protocollen, genaamd: Redphone en TextSecure. Deze 2 protocollen hebben ook apps voor Android onder dezelfde namen en is compatibel met Signal.

Het protocol, TextSecure, kan het beste vergeleken worden met OTR. De grootste verschillen met OTR zijn als volgt:

* Er worden ECC keys gebruikt in plaats van DSA keys
* Er word geen OTR ratchet gebruikt maar AXOLOTL

**Wat is OTR en hoe werkt het**

OTR staat voor Off the Record en is een protocol om berichten te versleutelen, met name voor instant messaging programma’s. OTR gebruikt een combinatie van AES 128 bit, Diffie hellman exchange 1536 bits group size en SHA-1 hashes.

**Main features:**

**Perfect forward secrecy**

Nieuwe berichten gebruiken een andere key waardoor het onmogelijk word om berichten in het verleden te decrypten, mocht een key gestolen worden.

**Digitale handtekeningen**

Door middel van het Diffie Hellmann principe verifieren Alice en Bob elkaar, maar tegelijkertijd worden er geen sensitieve data opgeslagen om te verifieren dat het ook daadwerkelijk Alice was.

**Message authenication codes**

Worden verder gebruikt om te bewijzen dat een bericht ook daadwerkelijk van Alice is, door te weten dat een bericht die door haar is verzonden, met een aangegeven MAC, juist is ( gegenereerd met Bob zijn publieke key )

**Forgeability**

Als Eve een encrypte bericht afluisterd en een paar bits veranderd, dan moet er geen half-werkende bericht uitkomen maar een random gegenereerde stuk tekst.

**Hoe werkt crypto cat**

Cryptocat gebruikt de Off-the-Record Messaging (OTR) protocol om chatberichten te encrypten, wanneer twee clients met elkaar berichten uitwisselen. Voor groeps berichten gebruikt Cryptocat een groeps chat protocol deploying Curve25519, AES-256, and HMAC-SHA512, allemaal industrie standaard voor cryptografische applicaties. Alle berichten in Cryptocat, inclusief de  groeps chat berichten en bestands verkeer zijn end-to-end encrypted (E2EE). Het end-to end protocal. Cryptocat maakt ook gebruik van Jabber XMMP (dit is alleen voor media files).

**Wat is Open SSL en hoe werkt het**

OpenSSL is een open source implementatie van de protocollen SSL en TLS, met de core geschreven in C. OpenSSL beschikt over veel protocollen die het ondersteund, waardoor het mogelijk is om het naar wens af te stellen.

SSL of TLS werkt in het kort door je bedrijf door een authority the laten verifieren wie dan met zijn key de data versleuteld. De client kan dan, met voorgedefineerde keys van deze authorities, deze key weer decrypten als hij met de server praat. Hiermee kan de client verifieren of de server te vertrouwen is ( als de client, de authority vertrouwd natuurlijk ).

Hierna word een key gegenereerd die word gebruikt om berichten te encrypten door de client en die alleen door de server kunnen worden gedecrypt.

**Hoe gaan we de applicatie opbouwen ( OTR, OpenSSL en wat gaat erin zitten ) library**

Voor een chat applicatie is het belangrijk dat er end-to-end encryption aanwezig is zodat de server niet mee kan luisteren. Om te zorgen dat voorgaande conversaties niet kunnen worden gestolen, is perfect forward secrecy nodig. Het OTR ( Off the Record ) protocol zorgt voor beide bovenstaande punten echter OTR ondersteunt geen beveiligde verbindingen tussen meerdere personen en er is geen library beschikbaar in F# voor OTR. OpenSSL heeft geen library in F#.

**Van implementaties die bestaan, welke code gaan we gebruiken**

**Cipher:** AES - https://msdn.microsoft.com/en-us/library/system.security.cryptography.aescryptoserviceprovider%28v=vs.110%29.aspx

**Hash:** SHA-2 - https://msdn.microsoft.com/en-us/library/system.security.cryptography.sha256%28v=vs.110%29.aspx

**Public-key:** Elliptic curve diffie hellman (TLS)

**RSA** : https://msdn.microsoft.com/en-us/library/system.security.cryptography.rsa(v=vs.110).aspx

In F#, de programmeer taal waar we de chat in moeten bouwen, zijn de bovenstaande libraries beschikbaar en zullen we in de chat gebruiken.. Het is ook mogelijk om zelf libraries te bouwen in C#. We zouden deze dan kunnen gebruiken. Maar vergt wel veel aanpassingen. Het is de vraag of we dat willen en/of moeten?

Keys genereren we met PasswordDeriveBytes Class. Dezez heeft een standaard van PBKDF2

DK = PBKDF2(PRF, Password, Salt, c, dkLen)

* *PRF* is a pseudorandom function of two parameters with output length *hLen* (e.g. a keyed [HMAC](https://en.wikipedia.org/wiki/HMAC))
* *Password* is the master password from which a derived key is generated
* *Salt* is a sequence of bits, known as a [cryptographic salt](https://en.wikipedia.org/wiki/Cryptographic_salt)
* *c* is the number of iterations desired
* *dkLen* is the desired length of the derived key
* *DK* is the generated derived key

**Hoe ga je dat doen met meerdere (gebruikers) chatrooms qua veiligheid**

Gaan we met AES (Symmetric key) doen en bij elk bericht een andere IV. Deze kan dan door sessie cookie ontcijferd worden.

**Schema van hoe we het opbouwen**

Er zal een server draaien op een server, beveiligd met HTTPS waar gebruikt zal worden gemaakt met TLS. Dit verzorgt de beveiliging tussen client en server.

Chatrooms kunnen worden benaderd via de server, door een naam in te typen. Als de chatroom nog niet bestaat, kun je een wachtwoord ingeven en word de chatroom gecreerd. Bestaat de chatroom, dan word er om een wachtwoord gevraagd. Als dit klopt, zal er een symmetrische key worden gegenereerd en deze zal worden gebruikt om de berichten te encrypten/decrypten

**Aanvullingen:**

Per bericht, een nieuw key genereren door bijvoorbeeld AES met IV te gebruiken, zodat voorgaande berichten niet kunnen worden gedecrypt, door de IV. Een nieuw IV word meegestuurd per bericht.

Een nieuwe gebruiker zal alleen berichten zien die worden verstuurd als hij op het moment is verbonden, hij krijgt geen inzicht op berichten in het verleden.

<https://msdn.microsoft.com/en-us/library/system.security.cryptography.rsa(v=vs.110).aspx>

Bronnen:

**Signal**

<https://github.com/WhisperSystems/TextSecure/wiki/ProtocolV2>

<https://en.wikipedia.org/wiki/TextSecure>

<http://codexgalactic.com/2014/01/27/axolotl-protocol-for-cryptographically-secure-messaging/>

**OTR**

<https://en.wikipedia.org/wiki/Off-the-Record_Messaging>

<https://otr.cypherpunks.ca/otr-wpes.pdf>

**Cryptocat**

<https://en.wikipedia.org/wiki/End-to-end_encryption>

<https://github.com/cryptocat/cryptocat/wiki/Multiparty-Protocol-Specification>

<https://nl.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol>

<http://xmpp.org/xmpp-software/>

**MiTLS**

<https://www.mitls.org:2443/wsgi/home>

**Netsh TLS Windows server**

<https://technet.microsoft.com/en-us/library/cc725926%28v=ws.10%29.aspx>

<http://fsprojects.github.io/FSharp.Control.AsyncSeq/library/AsyncSeq.html>

**Keys genereren**

<https://msdn.microsoft.com/en-us/library/system.security.cryptography.passwordderivebytes(v=vs.110).aspx>

**Meer agents**

<http://blogs.msdn.com/b/dsyme/archive/2010/02/15/async-and-parallel-design-patterns-in-f-part-3-agents.aspx>

**Push notifications**

<https://fwaris.wordpress.com/2012/06/06/f-push-notification-33k-concurrent-clients-1-relaxed-server/>