Hogeschool Rotterdam

**Projectgroep Sidbas**

Module: MINCMISAD01-1516

Klas: DINF3

Software release

Secure Chat Applicatie

**Projectgroep Sidbas**

Sidney Prins, 0838456  
Sebastiaan van Staden, 0883388

Module: MINCMISAD01-1516  
Klas: DINF3

**zondag 8 november 2015**

**Voorwoord**

Dit softwarerelease document is geschreven in opdracht van de Hogeschool Rotterdam door Sebastiaan van Staden en Sidney Prins. Doel van het softwarerelease document is om inzicht te geven van de oplevering van de secure chat applicatie, hoe deze werkt en waarom we gekozen hebben voor de huidige opzet van de applicatie.

Projectgroep Sidbas

Rotterdam, november ’15

Inhoudsopgave

[Samenvatting 4](#_Toc434742588)

[Versie geschiedenis 4](#_Toc434742589)

[Verklarende Woordenlijst 5](#_Toc434742590)

[1. Inleiding 6](#_Toc434742591)

[1.1 Disclamer 6](#_Toc434742592)

[1.2 Doel 6](#_Toc434742593)

[2. Configuratie & Werking 7](#_Toc434742594)

[2.1 Program language Fsharp 7](#_Toc434742595)

[2.2 Client 7](#_Toc434742596)

[2.2.1 Libraries 7](#_Toc434742597)

[2.2.2 Chatroom types 8](#_Toc434742598)

[2.2.3 Websocket 8](#_Toc434742599)

[2.2.4 OTR 8](#_Toc434742600)

[2.2.5 SMP Protocol 8](#_Toc434742601)

[2.2.6 Veilig opslaan van data en verificatie 9](#_Toc434742602)

[2.2.7 DSA 9](#_Toc434742603)

[2.2.8 Diffie Hellman 9](#_Toc434742604)

[2.2.9 Upgrading 9](#_Toc434742605)

[2.3 Server 9](#_Toc434742606)

[2.3.1 Libraries 10](#_Toc434742607)

[2.3.2 TLS & Websocket 10](#_Toc434742608)

[2.3.3 Upgrading 10](#_Toc434742609)

[2.4 Conclusie 10](#_Toc434742610)

[3. Deployment 11](#_Toc434742611)

[4. Data migratie 11](#_Toc434742612)

[5. Issues & Bugs 11](#_Toc434742613)

[5.1 ISSUE 1: Client Saved history moet chatroomtype mee krijgen 11](#_Toc434742614)

[5.2 ISSUE 2: Client History moet gedeleted kunnen worden 12](#_Toc434742615)

[BIJLAGE Activity Diagram 12](#_Toc434742616)

[BIJLAGE Sleutel grootte 13](#_Toc434742617)

[AES key size 13](#_Toc434742618)

[MAC key size 13](#_Toc434742619)

[Diffie Hellman key size 13](#_Toc434742620)

[Server key size 13](#_Toc434742621)

[Bronnen 14](#_Toc434742622)

# Samenvatting

Dit document beschrijft de software release, uitleg van de secure chat applicatie en de versie geschiedenis.

# Versie geschiedenis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Versie #** | **Geimplementeerd door** | **Revisie datum** | **Goedgekeurd door** | **Goedkeurings datum** | **Reden** |
| 1.0 | *Sebastiaan* | *2015-10-29* |  |  | First draft |
| 1.1 | Sidney en Sebastiaan | 2015-11-02 |  |  | Adding document |
| 1.2 | Sidney en Sebastiaan | 2015-11-03 |  |  | Adding document |
| 1.3 | Sidney | 2015-11-04 |  |  | Corrections |
| 1.4 | Sebastiaan | 2015-11-05 |  |  | Corrections |

# Verklarende Woordenlijst

Sidbas – Drie voorletters van Sidney en bas van Sebastiaan.

Arne Padmos – leraar en begeleider van de Minor Security.

Encryptie en Decryptie - Binnen de cryptografie staat encryptie voor het coderen (versleutelen) van gegevens op basis van een bepaald algoritme. Deze versleutelde gegevens kunnen nadien weer gedecrypteerd (ontcijferd of gedecodeerd) worden zodat men de originele informatie weer terugkrijgt. Dit proces wordt decryptie genoemd.

AES - In de cryptografie is Advanced Encryption Standard (AES) een computerversleutelingstechniek (encryptie). Het is de opvolger van de "Data Encryption Standard" (DES). AES is een subset van het Rijndael-algoritme waarbij de blokgrootte 128-bits is, en de sleutel 128, 192 of 256 bits. Rijndael zelf kan alle blokgrootten en sleutels aan die een veelvoud zijn van 32-bit met een minimum van 128-bit en een maximum van 256-bit.

SHA - De SHA-familie (Secure Hash Algorithm) is een verzameling gerelateerde cryptografische hashfuncties ontworpen door de Amerikaanse National Security Agency en gepubliceerd door het Amerikaanse National Institute of Standards and Technology. Het doel van deze algoritmen is het berekenen van een verkorte versie van een reeks tekens, een "samenvatting" (Engels: digest), die met een hoge mate van waarschijnlijkheid uniek is voor een gegeven tekenreeks, de "boodschap" (Engels: message).

OTR - Off-the-Record Messaging is een Cryptografisch protocol waar encryptie gebruikt wordt voor onderlinge conversaties, OTR werkt één op één. Met OTR protocol zijn de berichten versleuteld en kunnen gebruikers controleren dat degene met wie ze chatten ook daadwerkelijk de persoon is die hij of zij beweert te zijn. Verder voegt OTR ook perfect forward secrecy toe en "deniability", zodat kan worden ontkend dat de gebruiker een bepaald chatbericht heeft verstuurd.

MPOTR - Multi-party Off-the-Record Messaging. Hetzelfde als OTR maar dan voor meerdere gebruikers.

TLS - Secure Sockets Layer (SSL) en diens opvolger Transport Layer Security (TLS), zijn encryptie-protocollen die communicatie op het internet beveiligen.

Diffie hellman - Het diffie-hellman-sleuteluitwisselingsprotocol is een cryptografisch protocol, waarmee twee deelnemers die verder niets van elkaar weten over een onbeveiligd communicatiekanaal een geheime encryptiesleutel kunnen uitwisselen, die daarna kan worden gebruikt om communicatie tussen de deelnemers te versleutelen.

DSA - Digital Signature Algorithm.

RSA - RSA is een asymmetrisch encryptiealgoritme, dat veel gebruikt wordt voor elektronische handel (beveiliging van transacties en dergelijke). Het formele algoritme werd in 1977 ontworpen door Ron Rivest, Adi Shamir en Len Adleman (vandaar de afkorting RSA).

Fsharp - F# (F sharp) is een mix van een functionele en een objectgeoriënteerde programmeertaal voor het .NET-platform van Microsoft.

Server - De server applicatie die nodig is om de clients te verwerken.

Client - De applicatie die gemaakt is om secure te chatten.

MAC - Message authentication codes.

Salt - Een salt is een hulpmiddel voor het afleiden van sleutels in cryptografie. Een salt bestaat uit willekeurig gekozen bits. Het resultaat van de sleutelafleiding wordt vaak gebruikt voor het versleuteld opslaan van wachtwoorden.

IV - Initialisatie Vector wordt gebruik in cryptografie om laatste blok deel van cryptografie op te vullen.

Hash - Willekeurig aantal karakters omzetten door een algoritme in een vaste grootte van willekeurige karakters. Gebruikt om bijvoorbeeld een woord om te zetten naar iets onleesbaar.

Fingerprint - Een hash gemaakt van een publieke sleutel om de sleutel mee te identificeren.

Websocket - Een communicatie protocol waarmee onder andere een verbinding tussen client en server kan worden opgezet, waarbij beide de server en client berichten naar elkaar kunnen versturen.

# 1. Inleiding

Voor de module SAD Minor van de Hogeschool Rotterdam hebben wij een secure chat applicatie gemaakt. In dit softwarerelease document wordt inzicht gegeven over de werking van de secure chat applicatie.

### Disclamer

De Sidbas secure chat applicatie is in geen geval aansprakelijk bij geval van schade en onjuistheden binnen dit document.

### Doel

Het doel van de release is om de nieuwe toevoegingen en veranderingen in de release en componenten van de secure chat applicatie weer te geven. Tevens worden de keuzes, problemen, opzet en bugs omschreven.

# 2. Configuratie & Werking

Het doel van de applicaties is om een zo’n veilig mogelijke chat op te zetten tussen gebruikers. De volgende twee applicaties zijn gemaakt. De client waarmee gebruikers met elkaar kunnen chatten en de server die dient om alle gegevens van de client te indexeren en berichten te verwerken.

Overzicht componenten

1. Client - Windows
2. Server - Websocket – Windows

## 2.1 Program language Fsharp

We hebben voor de applicaties gebruik gemaakt van de programmeertaal F#. De eerste lessen die gegeven werden voor de minor gingen over F#. In eerste instantie wilden we F# en security leren. Om de programmeertaal en security onder de knie te krijgen vereiste veel werk, daardoor hebben we onze focus verlegd naar de security. Hierdoor hebben we niet functioneel geprogrammeerd, maar meer een C# variant gemaakt in F#.

## 2.2 Client

De applicatie is gebouwd in het dot net framework 4.5.2 en kan op Windows vista of nieuwer worden geïnstalleerd. De applicatie word gebruikt om met een naam, chatroomnaam en een chatroomtype, een beveiligde verbinding op te zetten om met elkaar te kunnen praten. We zullen de werking van de chatroom types bespreken in hoofdstuk 2.2.2.

Wanneer de gebruiker een chatroom binnen komt, word per gebruiker een websocket verbinding opgezet die beveiligd is met TLS. Websockets en TLS wordt besproken in hoofdstuk 2.2.3. Wanneer meerdere gebruikers in de chatroom zitten, word per gebruiker een OTR verbinding gemaakt. Een OTR verbinding zorgt ervoor dat alleen gebruikers die verbonden zijn, het bericht kunnen inzien. Meer uitleg hierover in hoofdstuk 2.2.4.

Om te verifiëren of de gebruikers die verbonden zijn ook echt de gebruikers zijn met wie is afgesproken, kunnen de gebruikers een geheim wachtwoord instellen die gecontroleerd word door het SMP protocol, zie hoofdstuk 2.2.5 voor meer uitleg over het SMP protocol.

De gebruiker heeft ook de mogelijkheid om een chatroom op te slaan. Hierbij worden alle gebruikers die op dat moment zich in de chatroom bevinden, opgeslagen. Tevens wordt ook van de gebruiker een fingerprint van de publieke DSA sleutel en het geheime wachtwoord behorende bij de chatroom opgeslagen. Hiermee kan later, als de gebruiker opnieuw in de chatroom komt, geverifieerd worden of een gebruiker onder dezelfde naam dezelfde gebruiker is als voorheen. Zie hoofdstuk 2.2.6 voor meer informatie.

### 2.2.1 Libraries

**Bouncycastle.crypto**  
Bouncycastle is een library met cryptografische oplossingen waarvan gebruik van gemaakt kan worden. OTRLib maakt gebruik van BouncyCastle.

**newtonsoft.JSON**  
JSON word gebruikt om tussen server en client te communiceren en voor het opslaan van gegevens in een bestand. JSON word ook gebruikt om JSON om te zetten in een type die F# weer kan uitlezen. Alhoewel het mogelijk is om dit zonder een library te doen, maakt deze library het een stuk eenvoudiger.

**OTRLib**  
Implementatie van OTR in C#.

**websocket-sharp**  
Implementatie van websockets in C# waarvan gebruik kan worden gemaakt om een server op te zetten en om via een client naar die server te verbinden. Ondersteunt het gebruik van secure websockets (TLS).

We hebben gebruik gemaakt van deze libraries zodat we niet zelf alle beveiligings protocollen hoeven te implementeren.

### 2.2.2 Chatroom types

Bij het aanmaken van een chatroom kan gekozen worden uit meerdere soorten chatrooms.

**1 op 1 chatroom**  
Bij een 1 op 1 chatroom kunnen maar maximaal 2 gebruikers in een chatroom bevinden. Als een derde gebruiker in de chatroom wil komen, zal een bericht worden weergegeven dat de chatroom vol zit.

**Gesloten chatroom**  
Bij een gesloten chatroom moet een wachtwoord worden opgegeven. De gebruiker die als eerste in een chatroom komt, bepaalt het wachtwoord. Als het wachtwoord niet klopt, kunnen de gebruikers niet deelnemen aan de conversatie.

**Openbare chatroom**In een openbare chatroom kan iedereen deelnemen.

We hebben voor deze drie types gekozen zodat de gebruiker zelf kan kiezen hoe hij wilt chatten en met welke restricties. Wanneer bijvoorbeeld voor een publieke chatroom word gekozen is het moeilijker te controleren wie iedereen is.

### 2.2.3 Websocket

Als een client wilt verbinden met een chatroom, wordt een websocket verbinding gemaakt naar de server. De server zal de gebruikersnaam opslaan onder de chatroomnaam en de client zal luisteren naar berichten die het terug krijgt van de server. Hierbij wordt onder andere opgehaald welke mensen in de chatroom zitten, of iemand is weggegaan en wie een bericht heeft verstuurd. Met websockets is het makkelijker om de server een bericht naar de client te versturen. Hierdoor kan de client, die verbonden is met de websocket, direct de berichten ontvangen. Deze websocket is beveiligd met TLS 1.2.

### 2.2.4 OTR

Off-the-Record messaging is een cryptografisch protocol waarbij encryptie gebruikt word voor onderlinge conversaties. OTR maakt gebruikt van end-to-end encryptie, waardoor alleen de gebruiker waarmee je verbonden bent het bericht kan ontcijferen. Dit zorgt ervoor dat de server de berichten niet kan inlezen.

OTR maakt gebruik van AES symmetrische sleutel algoritme om de onderlinge berichten te versleutelen met SHA1. Deze sleutel wordt bij elk bericht veranderd. Ook maakt OTR gebruik van het Diffie Hellman protocol (zie hoofdstuk 2.2.8). Wanneer de sleutel onderschept wordt is de forward secrecy altijd gegarandeerd.

MAC sleutels worden gebruikt om tijdens de conversatie de berichten te authentiseren, zodat de gebruiker weet wie een bericht heeft verstuurd. Tevens word de laatst gebruikte MAC sleutel meegestuurd met berichten. Iemand die het bericht heeft ontsleuteld, zou deze MAC sleutel kunnen gebruiken om berichten na te maken. Maar doordat elke keer een nieuwe MAC sleutel word gemaakt is dit niet mogelijk.

We hebben voor OTR gekozen omdat als we een veilige chat applicatie willen maken, we graag een protocol willen gebruiken die ervoor zorgt dat de server nooit de verstuurde berichten kan inlezen. Tevens is OTR een algemeen geaccepteerd veilig protocol. Daarom hebben we dit protocol geïmplementeerd, en niet gekozen om alles zelf te implementeren.

### 2.2.5 SMP Protocol

In cryptografie is het socialistische miljonair probleem een probleem waarbij 2 miljonairs van elkaar willen weten of zij evenveel geld hebben zonder hierbij te zeggen wat zij hebben.

Het SMP protocol zit ingebouwd in het OTR protocol die we gebruiken. Hierbij wordt een geheime wachtwoord gebruikt die alle gebruikers met elkaar afstemmen via een andere kanaal zoals bijvoorbeeld what’s app of in-persoon. Met het geheime wachtwoord kan geverifieerd worden of de gebruiker is wie hij zegt dat hij is. We hebben deze mogelijkheid in de applicatie ingebouwd zodat SMP protocol gebruikt kan worden.

### 2.2.6 Veilig opslaan van data en verificatie

Wanneer voor het eerst de applicatie wordt opgestart, kan gekozen worden om in te loggen. Als dit voorheen al een keer is gedaan, moet het ingestelde wachtwoord ingevoerd worden, anders wordt een nieuw bestand aangemaakt met een wachtwoord naar keuze. De opgeslagen gegevens worden door middel van AES encryptie opgeslagen. We hebben voor AES gekozen, omdat AES geschikt is voor bestanden en algemeen geaccepteerd is als veilig protocol. AES staat ingesteld op 256 bits encryptie. De IV en salt, die worden gebruikt bij het versleutelen en ontsleutelen van het bestand, word gegenereerd en lokaal opgeslagen. Dit doen we zodat elke gebruiker een andere IV en salt heeft. We hoeven de IV niet elke keer anders te genereren omdat hiermee geen berichten worden versleuteld, maar alleen de lokale opgeslagen gegevens.

Wanneer ingelogd, kan de chatroom opgeslagen worden. Hierbij word de chatroomnaam, het geheime wachtwoord en een lijst van gebruikers met de bijbehorende DSA fingerprint opgeslagen. Deze fingerprint word later weer gebruikt voor verificatie. Zie hoofdstuk 2.2.7 voor meer informatie. Wanneer een chatroom is opgeslagen kan deze direct via de applicatie worden geopend.

### 2.2.7 DSA

OTR gebruikt Digital Signature Algorithm (DSA) sleutels als onderdeel voor de authenticatie en SMP processen. Voor identificatie, word gebruik gemaakt van DSA sleutels in het OTR protocol. Om ervoor te zorgen dat iedereen een andere DSA sleutels heeft en dat niet steeds nieuwe DSA sleutels worden gegenereerd, zal de applicatie maar één keer een DSA sleutel genereren en lokaal opslaan. We gebruiken een sleutel lengte van 2048 bij 256, dit zit in het OTR library verwerkt.

Een DSA fingerprint van de gebruiker, die word opgeslagen bij een chatroom, kan dan weer worden gebruikt om te verifiëren dat deze van dezelfde gebruiker komt uit een vorige chatsessie.

### 2.2.8 Diffie Hellman

Met het diffie-hellman-sleuteluitwisselingsprotocol word bij elk bericht nieuwe sleutels gegenereerd en meegestuurd. Deze sleutels worden dan gebruikt om de communicatie tussen de deelnemers te versleutelen.

### 2.2.9 Upgrading

Wanneer veranderingen in de software zal plaats vinden, zal deze via Github automatisch worden bijgewerkt.

## 

## 2.3 Server

De server is gebouwd in dot net framework 4.5.2 en draait op Windows vista en nieuwer. De server zorgt dat alle berichten van clients bij de juiste personen terechtkomen en verzorgt tevens de authenticatie voor een wachtwoord beschermde chatroom.

De server werkt met websockets. De voornaamste reden waarom wij voor websockets hebben gekozen is omdat het makkelijker is om berichten van de server naar de client te versturen. Hierdoor kan de client, die verbonden is met de websocket, direct berichten ontvangen.

### 2.3.1 Libraries

**websocket-sharp**  
Implementatie van websockets in C# waarvan gebruik kan worden gemaakt om een server op te zetten en om via een client naar die server te verbinden. Ondersteunt het gebruik van secure websockets TLS1.2.

**newtonsoft.JSON**  
JSON word gebruikt om tussen server en client te communiceren. JSON wordt ook gebruikt om JSON om te zetten in een type die F# weer kan uitlezen. Alhoewel het mogelijk is om dit zonder een library te doen, maakt deze library het een stuk eenvoudiger.

### 2.3.2 TLS & Websocket

De server staat in verbinding met de clients door middel van een websocket. De server slaat de gegevens op van de clients. De chatroomnaam, naam, type chatroom en eventuele wachtwoorden worden hierbij ook opgeslagen. Deze gegevens worden allemaal tijdelijk in het geheugen opgeslagen zodat de server weet naar wie een bericht verstuurd moet worden. Verder worden de gegevens gebruikt om verbindingen te sluiten en om de huidige gebruikers binnen een chatroom naar de clients te versturen.

De websocket is beveiligd met TLS 1.2. We hebben Windows waarop de server draait geüpgraded van TLS 1.0 naar TLS 1.2. Dit was nodig omdat TLS 1.0 kwetsbaar is voor een BEAST attack[[1]](#footnote-1).

### 2.3.3 Upgrading

Wanneer veranderingen in de software zal plaats vinden, zal deze via Github automatisch worden bijgewerkt.

## 2.4 Conclusie

De eerste lessen van de minor gingen diep in op F#, waardoor wij in het beginsel hebben gekozen om F# en security te doen. Naarmate wij verder kwamen, realiseerde wij, alhoewel F# interessant is qua functioneel programmeren, dat we hiervoor niet genoeg tijd hadden om F# te leren samen met de punten van security. Om deze reden hebben wij F# meer als een normale programmeertaal gebruikt en meer gefocust op security.

We hebben geprobeerd alles vanaf het begin zelf te programmeren. Het programmeren van de websockets bleek veel tijd te kosten We moesten alle berichten ontcijferen die via websockets binnenkomen, omdat deze berichten standaard in websockets wordt encoded. Hiervoor bestaan geen gedefinieerde functies. Zodoende zijn we op zoek gegaan naar bestaande libraries voor het opzetten van de connecties tussen client en server. Hierdoor hadden we meer tijd om een werkende chat applicatie op te leveren die gebruikt maakt van veilige protocollen.

Wat gaan we nou precies gebruiken als protocol? Het idee was om eerst alles via een web applicatie te laten lopen, maar nadat we samen met Arne Padmos hebben gezeten, kwamen we tot de conclusie dat het een stuk makkelijker is om een foutief bestand te genereren die geïnfecteerd is van een webserver dan via een Windows applicatie. Om deze reden hebben we ook uiteindelijk gekozen om met F# een Windows applicatie te bouwen en om geen javascript met HTML meer te gebruiken.

Bij het zoeken van een protocol hebben we veel protocollen onderzocht: OTR, TextSecure en die van Cryptocat zelf. Uiteindelijk was OTR het enige protocol die wij konden implementeren in F# via een C# library. Het bleek moeilijker dan gedacht, we moesten veel omzetten in F# voordat het werkte. Het hele proces was ingewikkeld, maar tijdens het implementeren hebben we geleerd hoe functies in F# werken en hoe het OTR protocol werkte.

Hoe weet de gebruiker of degene waar hij mee wilt chatten ook echt de persoon is, die nu  verbonden is met de chat? Binnen OTR zit het protocol Social Millionaire Protocol (SMP). Dit is een algoritme gemaakt zodat een antwoord dat gebruikers met elkaar afspreken, kan worden vergeleken. Zo kan gecontroleerd worden of het echt degene is waarmee gepraat wordt of niet. Dit zit standaard in de OTR library en we hebben dit in de applicatie gebouwd zodat hier gebruik van gemaakt kan worden. Het was niet moeilijk om dit te implementeren, alleen is de werking van het SMP protocol ingewikkeld om te leren.

Op dit punt hadden we een vrij goede applicatie gebouwd, echter is het opnieuw invullen van je naam, chatroomnaam en geheim wachtwoord via SMP veel werk. Om deze reden hebben wij de functionaliteit gebouwd waarmee een chatroom opgeslagen kan worden. Het bestand waarin dit word opgeslagen is tegelijkertijd beveiligd met AES. Hierdoor voert een persoon eenmalig een wachtwoord in om het bestand te ontsleutelen en kan hij al zijn chats benaderen.

Na het wekelijks overleg met Arne Padmos, kwamen we erachter, dat de gebruiker de controle van het SMP wachtwoord niet elke keer opnieuw wilt doen. Meestal is ook het wachtwoord dat bij het SMP ingevoerd wordt, relatief zwak. Om deze reden hebben we de applicatie omgebouwd. Nu heeft elke applicatie zijn eigen publieke en geheime sleutel, die word gebruikt om sessies te starten. Wanneer een gebruiker is ingelogd, worden de DSA fingerprints gebruikt om gebruikers te verifiëren. Hierdoor weet de gebruiker dat het dezelfde persoon is waarmee hij eerder heeft gepraat. Zo wordt voorkomen dat iemand snel opnieuw kan inloggen onder bijvoorbeeld de naam Bob zonder dat de gebruiker in de gaten heeft dat het ook echt Bob is. We hebben dit zo geprogrammeerd dat het automatisch gaat.

Met AES, Diffie Hellman, MAC en DSA sleutels word nu elk bericht beveiligd. Tevens wordt voor elk bericht de sleutels van MAC, Diffie Hellman en AES veranderd om forward secrecy en deniable authenticatie te garanderen. Tevens hebben we gekozen om met AES encryptie de geschiedenis op te slaan. Eigenlijk is dit niet nodig omdat onze aanname is dat de computer waarop de applicatie draait veilig is. Daarom hebben we ook de DSA privé sleutel en IV lokaal opgeslagen.

Wij hebben veel kennis opgedaan als het gaat om security en F#. We vonden het een leuk project en het heeft onze ogen geopend hoe moeilijk het is om digitale veiligheid te garanderen.

# 3. Deployment

De applicatie zal met self extracting software Installforge worden gemaakt en beschikbaar worden gesteld in Github evenals source code.

# 4. Data migratie

De server zal beschikbaar worden gesteld op Github. Elke pc met minimaal besturingssysteem Windows net framework 4.5.2 kan deze applicatie openen. Wanneer de server achter een router zit, zal deze router de ingestelde poorten moeten open zetten, om de server te kunnen bereiken.

# 5. Issues & Bugs

De meeste bugs worden automatisch gevonden door Visual studio en F# tools. Deze dienen dan opgelost te worden om de nieuwe release online te krijgen. Hieronder worden de kwesties en bekende fouten omschreven, die in deze software release V1.4 van de secure chat applicatie voorkomen. Tevens wordt geprobeerd een oplossing te vinden voor de fouten om deze in de volgende release te voorkomen.

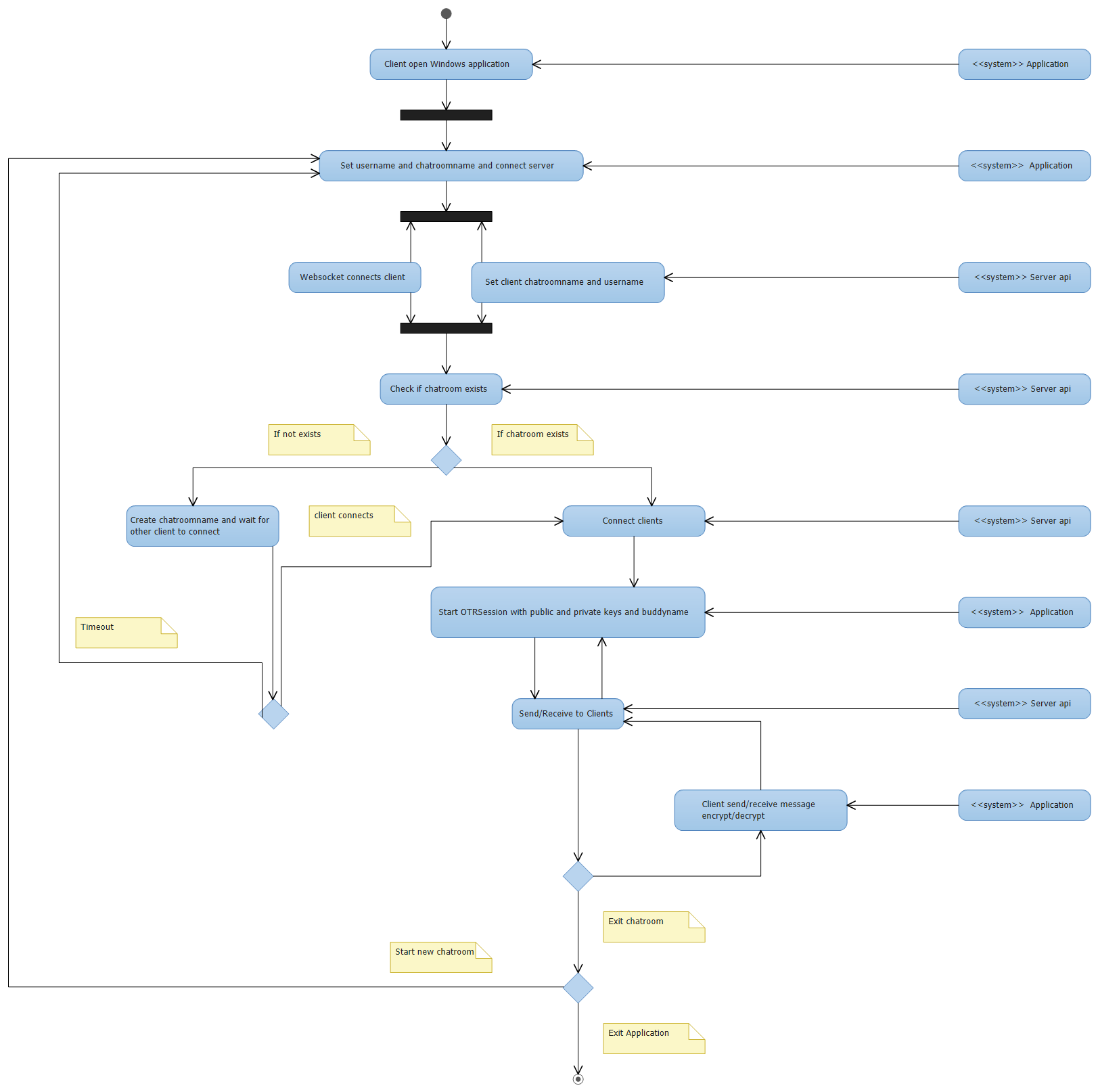
### 5.1 ISSUE 1: Client Saved history moet chatroomtype mee krijgen

Resolutie: nog te programmeren voor volgende release.

### 5.2 ISSUE 2: Client History moet gedeleted kunnen worden

Resolutie: nog te programmeren voor volgende release.

# BIJLAGE Activity Diagram



# BIJLAGE Sleutel grootte

### AES key size

public const int AES\_KEY\_LENGTH\_BITS = 128;

\_sha\_1\_bytes = Utility.SHA1GetHash(\_aes\_recv\_key);

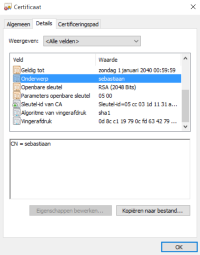
### MAC key size

public const int MAC\_KEY\_LENGTH\_BITS = 256;

### Diffie Hellman key size

public const int DH\_PRIVATE\_KEY\_MINIMUM\_LENGTH\_BITS = 320;

### Server key size

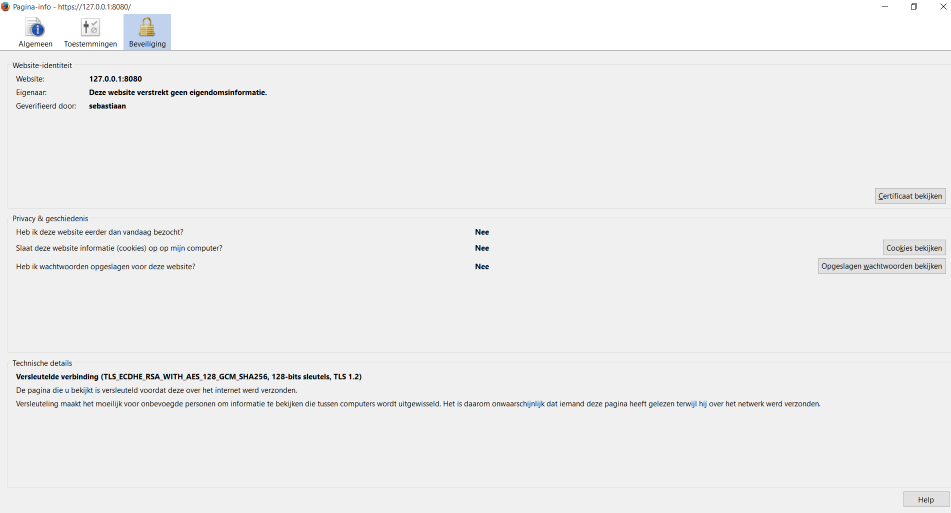


RSA 2048 bits

Windows commandline om server.pfx te genereren:

*makecert.exe -r -pe -n "CN=sebastiaan" -sky exchange -sv server.pvk server.cer*

*pvk2pfx.exe -pvk server.pvk -spc server.cer -pfx server.pfx*



Eigenschappen TLS verbinding

# Bronnen

Rijndael (cryptografie) - From Wikipedia, 2 aug 2015 om 11:53 - <https://nl.wikipedia.org/wiki/Rijndael_%28cryptografie%29>

Advanced Encryption Standard - From Wikipedia, 9 mrt 2013 om 08:20 - <https://nl.wikipedia.org/wiki/Advanced_Encryption_Standard>

SHA-familie - From Wikipedia, 27 aug 2014 om 15:28 - <https://nl.wikipedia.org/wiki/SHA-familie>

SSL niet bestand tegen afluisteren overheid - vrijdag 26 maart 2010, 13:08 door security.nl - <https://www.security.nl/posting/28072/SSL+niet+bestand+tegen+afluisteren+overheid>

Multi-party Off-the-Record Messaging - Ian Goldberg, Matthew D. Van Gundy, Hao Chen, Berkant Ustaoglu - November 9–13, 2009, Chicago, Illinois, US - <https://www.cypherpunks.ca/~iang/pubs/mpotr.pdf>

Secure Sockets Layer - From Wikipedia, 23 mrt 2015 om 21:09 - <https://nl.wikipedia.org/wiki/Secure_Sockets_Layer>

RSA (cryptografie) - From Wikipedia, 3 jan 2015 om 17:51 - <https://nl.wikipedia.org/wiki/RSA_%28cryptografie%29>

Message authentication code - From Wikipedia, 28 October 2015 om 15:24 - <https://en.wikipedia.org/wiki/Message_authentication_code>

Digital Signature Algorithm - From Wikipedia, 30 September 2015, at 13:36 - <https://en.wikipedia.org/wiki/Digital_Signature_Algorithm>

InstallForge | The Free Setup Creator for Windows - <http://installforge.net/>

A C# implementation of the WebSocket protocol client and server - <http://sta.github.io/websocket-sharp/>

Off-The-Record (OTR) Security Protocol - Don Fizachi, 29 Apr 2014 - <http://www.codeproject.com/Articles/644318/Off-The-Record-OTR-Security-Protocol>

Off-the-Record Messaging - From Wikipedia, 9 September 2015 om 14:20 - <https://en.wikipedia.org/wiki/Off-the-Record_Messaging>

Attacks on SSL and TLS - From Wikipedia, 28 aug 2015 om 14:11 - <https://wiki.mozilla.org/Security/Server_Side_TLS#BEAST_.28CVE-2011-3389.29>

Socialist millionaire - From Wikipedia, 5 aug 2015 om 10:24 - <https://en.wikipedia.org/wiki/Socialist_millionaire>

1. BEAST is een kwetsbaarheid in de Initialisatie Vector (IV) van de CBC-modus van AES, Camellia en een paar andere cijfers die CBC-modus te gebruiken. De aanval kan een aanvaller MITM plaintext waarden herstellen door het meerdere keren versleutelen van hetzelfde bericht. [↑](#footnote-ref-1)