Internet Banking/Shopping: De gevaren achter het hangslot

Bachelorscriptie informatiekunde

Philipp van Bebber (0608785)

Begeleider: Engelbert Hubbers
Radboud Universiteit Nijmegen

15 juni 2009

Inhoudsopgave

1 Inleiding **4**
1.1 TLS informatiebeveiliging 4

2 TLS **5**
2.1 Record Layer Protocol 6
2.2 Handshake Protocol 8
2.3 Versleutelingsalgoritmen & key exchange 11
2.3.1 Key exchange methoden 11
2.3.2 Versleutelingsalgoritmen 12

3 Veiligheid van versleutelingsmethoden **16**
3.1 OpenSSL 17

4 Browsers **18**
4.1 Mozilla Firefox 19
4.2 Microsoft Internet Explorer, Google Chrome & Apple Safari 20

5 Methode **23**
5.1 Software 23
5.2 Webservers 23
5.3 Cipher suite testsets 23

6 Resultaten **24**
6.1 Testset 1 24
6.2 Testset 2 24
6.3 Testset 3 25
6.4 Testset 4a/b 25
6.5 Testset 5a/b 25
6.6 Testset 6 25
6.7 Testset 7 26
6.8 Testset 8 26

7 Conclusie **28**
7.1 Gevolgen 28
7.1.1 Banken 28
7.1.2 Webwinkels 29

8 Appendix A **32**

9 Appendix B **34**

1 Inleiding

Tegenwoordig is TLS (Transport Layer Security) het standaardprotocol dat gebruikt wordt om een client – server verbinding te beveiligen. De client is in dit geval de op een lokale computer geïnstalleerde browser. Banken en webwinkels maken altijd gebruik van TLS om de klant zo goed mogelijk tegen fraudeurs te kunnen beschermen. In de laatste jaren zijn hoop versleutelingsmethoden gekraakt [TWP07], [KPP+06] en om die reden moet vooral gekeken worden in hoeverre TLS nog veiligheid kan garanderen. TLS kan in het kort beschreven worden als het elkaar identificeren (1), het uitwisselen van sleutels (2), het kiezen van een versleutelingsmethode (3) en het ten slotte opbouwen van de verbinding (4). Van welk versleutelingsmethode uiteindelijk gebruikt wordt gemaakt bepaalt de server. De client heeft dus geen mogelijkheid om zelf een keuze te maken. Het probleem dat hier boven water komt is dat er sterke en zwakke versleutelingsmethoden zijn. De server kiest een van de door de client aangeboden versleutelingsmethoden. De vraag die zich dan van zelf stelt is: Wordt er voor de meest veilige versleutelingsmethode gekozen? Vooral omdat er geen directe interactie met de gebruiker plaatsvindt en de gebruiker dan ook weer niet over de gemaakte keuze geïnformeerd wordt is het van belang om dit te onderzoeken.

Onderzoeksvragen

Veilig communiceren via TLS: Kiezen webservers van webwinkels en banken altijd de sterkste versleutelingsmethode die wordt aangeboden door de browser? Indien dat niet het geval is, is de veiligheid van de communicatie door de zwakkere versleutelingsmethode alsnog gegarandeerd?

1.1 TLS informatiebeveiliging

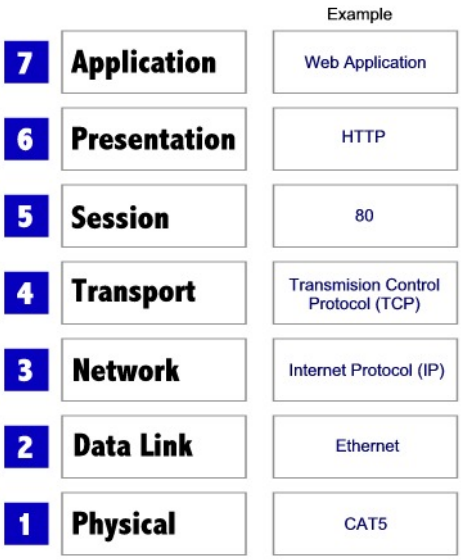
Wanneer het gaat om het beveiligen van webapplicaties en de informatie die deze bevatten zijn de volgende criteria van belang: Confidentiality, integrity en authentication (CIA). Daarnaast nemen ook availability en non-repudiation nog een belangrijke rol in. TLS verzorgt de volgende aspecten van informatiebeveiliging:

1. **Vertrouwelijkheid (confidentiality):** De data is versleuteld en kan dus niet door anderen gelezen worden.
2. **Authenticiteit (authentication):** De server laat door middel van certificaten zien dat hij ook daadwerkelijk de server is die hij beweert te zijn. Alleen in het geval van DH anonymous (zie hiervoor sectie 2.3.1) wordt deze eigenschap niet verzorgd. Soms is het voor de cliënt verplicht om zich ook te identificeren, maar dit is alleen een optie. In de meeste gevallen is de applicatie op de cliënt machine, die gebruikmaakt van TLS, verantwoordelijk voor de identificatie van de cliënt.
3. **Integriteit (integrity):** Verstuurde berichten kunnen niet door anderen veranderd worden omdat van hash functies en MAC (Message Authentication Code) gebruik wordt gemaakt.
4. **Beschikbaarheid (availability):** De benodigde informatie moet altijd toegankelijk zijn. TLS garandeert een beveiligde communicatie en geeft toegang. Echter, wanneer de server niet verbonden is met het internet kan TLS deze eigenschap niet verzorgen.

TLS verzorgt niet de eigenschap van non-repudiation (Onweerlegbaarheid) omdat het alleen verantwoordelijk is voor het opbouwen van een beveiligde verbinding. Deze wordt meestal door de applicatielaag verzorgd.

2 TLS

In dit hoofdstuk wordt de structuur en werkwijze van TLS 1.2 zoals beschreven in [DR08] uitgelegd. De eerste vraag die gesteld moet worden is: In welke laag van het OSI-model [fS96] bevindt zich het TLS protocol? Alle lagen van het OSI-model leveren hun bijdrage aan de totale security en moeten dus beveiligd zijn. Elke laag is dus verantwoordelijk voor de eigen beveiliging. TLS wordt in de vierde laag (transport layer) van het OSI-model (Figuur 1) geïmplementeerd. Dit wordt ook door de naam **T**ransport**L**ayer**S**ecurity duidelijk.

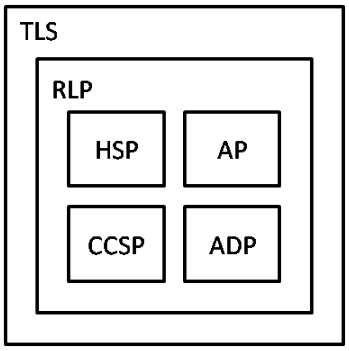


Figuur 1: OSI Model

Om een veilige communicatie tussen cliënt en server te kunnen waarborgen is ervoor gekozen om binnen TLS verder nog vijf protocollen te implementeren. Deze zijn het Record Layer Protocol (RLP), het Handshake Protocol (HSP), het Alert Protocol (AP), het Change Cipher Spec Protocol (CCSP) en het Application Data Protocol (ADP). Het RLP is het hoofdprotocol dat binnen TLS wordt gebruikt en de overige vier protocollen maken gebruik van het RLP.
Het RLP is verantwoordelijk voor het versturen van data. Elk record wordt verstuurd met een aantal parameters die van tevoren voor de betreffende verbinding worden vastgelegd. Deze parameters worden in het begin van een sessie afgesproken.

TLS bestaat dus uit drie fasen:

1. De onderhandeling van de parameters (HSP)
2. Identificatie en sleuteluitwisseling (HSP)



Figuur 2: Protocol specificatie

3. Encryptie en berichtidentificatie (RLP/ADP)

In (1) worden protocolversie, versleutelingsalgoritme en compressiemethode bepaald. In (2) identificeert zich de server middels een certificaat waarop sleutels uitgewisseld worden. Uiteindelijk kan in (3) de data op een veilige manier uitgewisseld worden.

2.1 Record Layer Protocol

Zoals eerder al beschreven verzorgt het RLP het veilig versturen van de data. In [DR08] wordt die functie van het RLP beschreven als:

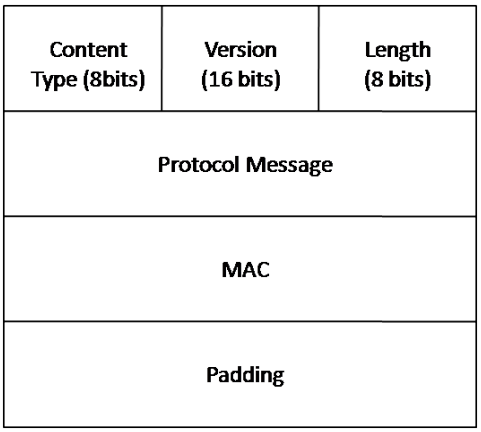
> "The Record Protocol takes messages to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, and transmits the result. Received data is decrypted, verified, decompressed, reassembled, and then delivered to higher-level clients."

Het HSP is verantwoordelijk voor de onderhandeling van de parameters en wordt aan het begin van een sessie aangeroepen. Het AP verstuurt voornamelijk berichten wanneer fouten zijn opgetreden die als gevolg hebben dat de verbinding niet meer veilig is en onderbroken zal worden, zoals certificate_revoked, certificate_expired, illegal_parameter, handshake_failure etc. Daarnaast verstuurt het AP ook berichten wanneer geen fouten zijn opgetreden, namelijk wanneer de datatransactie succesvol is afgelopen en de verbinding beëindigd wordt (close_notify). De data die verstuurd moet worden wordt door het ADP in fragmenten opgedeeld en gecomprimeerd. Het CCSP wordt in sectie 2.2 nog nader toegelicht. De structuur van een record bevat de volgende componenten:

De getallen tussen haakjes geven de expliciete tags (decimaal) aan.

ContentType

- Change Cipher Spec Protocol (20)
- Alert Protocol (21)



Figuur 3: Record layer protocol

- Handshake Protocol (22)
- Application Protocol (23)

Versie

- SSL versie 3 (30)
- TLS 1.0 (31)
- TLS 1.1 (32)
- TLS 1.2 (33)

Length

- De lengte van de record (inclusief header).

Protocol Message

- Een of meerdere berichten (afhankelijk van Content Type).

MAC

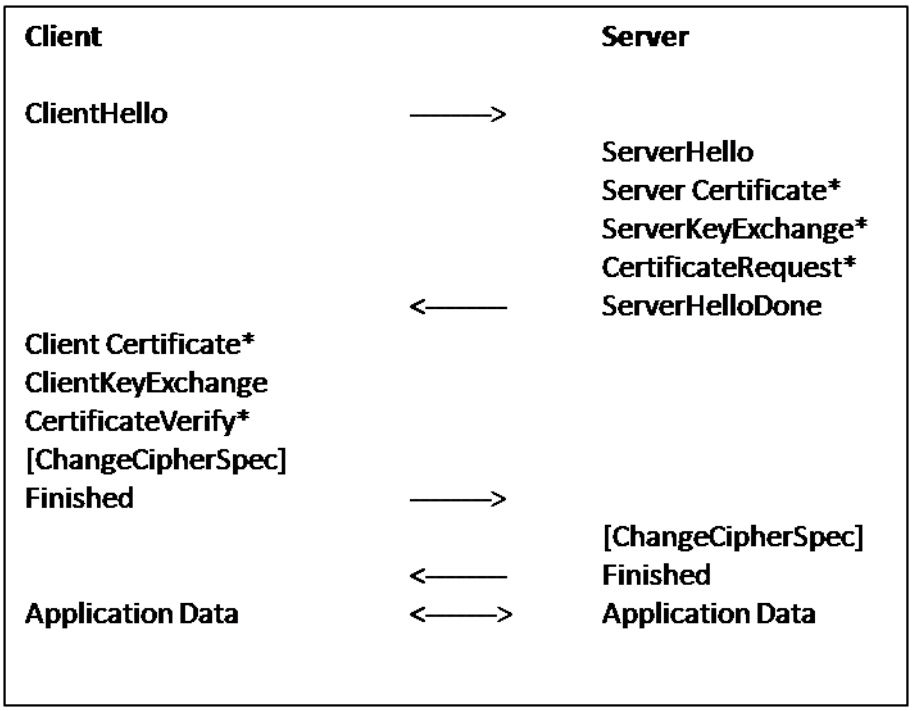
- Message Authentication Code (optioneel). Wanneer een MAC wordt gebruikt, wordt deze over het hele record (inclusief header) berekend. Of een MAC wordt gebruikt hangt af van het stadium waarin de communicatie zich bevindt.

Padding

- In het geval dat het bericht niet de maximale grootte heeft wordt data toegevoegd. Deze is nooit relevant en van padding wordt alleen gebruik gemaakt wanneer voor een block cipher met vaste bloklengte is gekozen.

2.2 Handshake Protocol

Het handshake protocol geeft de structuur aan in welke volgorde berichten worden verstuurd en op welke manier sleutels worden uitgewisseld en afspraken gemaakt. Aan het einde van een succesvolle handshake hebben server en cliënt een afspraak gemaakt over de TLS/SSL versie, de sessie ID, compressiemethoden, versleutelingsalgoritme en de master key. De communicatie tijdens een handshake verloopt op de volgende wijze (berichten met * worden situatieafhankelijk wel of niet verstuurd; [ChangeCipherSpec] betekent dat het bericht geen belangrijke informatie bevat, het is alleen bedoeld om de andere mee te delen dat een andere key gebruikt zal worden):



Figuur 4: Handshake protocol: communicatieverloop

1. ClientHello

Het ClientHello bericht bevat de volgende velden: protocol versie, random getal, sessionID*, versleutelingsalgoritmen en compressiemethoden. De sessionID hoeft niet ingevuld te worden. In het geval dat de server het herstellen van een verbinding accepteert, kan de cliënt een ID van een eerdere sessie opsturen. Het veld bevat een NULL waarde wanneer er geen eerdere sessie ID bestaat. De door de cliënt ondersteunde versleutelingsalgoritmen en compressiemethoden worden in twee lijsten gezet waarbij geldt dat de voorkeur aan het begin van de lijst wordt gezet.

2. ServerHello

Het ServerHello bericht wordt alleen verstuurd na het ontvangen van een ClientHello bericht, het bevat de volgende velden: protocol versie, random getal (deze is niet afhankelijk van het

random getal in ClientHello), sessionID, versleutelingsalgoritme en compressiemethode. In het veld protocol versie geeft de server aan welke versie gebruikt zal worden. De random getallen in ClientHello en ServerHello worden later gebruikt tijdens het uitwisselen van sleutels. De sessieID kan of een al eerder gebruikte ID zijn om aan de cliënt mee te delen dat de sessie wordt hersteld, of een nieuwe ID zijn of het veld bevat een NULL waarde (de server ondersteunt niet het herstellen van sessies). Daarnaast kiest de server een van de aangeboden versleutelingsalgoritmen en compressiemethoden.

3. ServerCertificate*
De server moet een certificaatbericht naar de cliënt sturen wanneer gebruik wordt gemaakt van certificaten om de server te kunnen identificeren. Het bericht bevat een certificate chain en wordt altijd verzonden behalve wanneer de server voor het key exchange algoritme Diffie-Hellman anonymous in ServerHello heeft gekozen.

4. ServerKeyExchange
Het ServerKeyExchange bericht wordt alleen verstuurd wanneer het om een verbinding zonder authenticatie gaat (DH anonymous) of het certificaat in (3) alleen wordt gebruikt voor het ondertekenen van berichten en niet voor het versleutelen van berichten (zoals bij Diffie-Hellman emphemeral waar de parameters ondertekend worden met behulp van het certificaat). Het bericht bevat een Diffie-Hellman public key waarmee de cliënt het uitwisselen van de sleutel kan afronden.

5. CertificateRequest*
De server kan nadat hij zijn certificaat aan de cliënt heeft verstuurd ook zelf een certificaat van de cliënt aanvragen zodat de cliënt zich moet identificeren.

6. ServerHelloDone
Met dit bericht laat de server aan de cliënt weten dat hij klaar is met het verzenden van het ServerHello bericht en de optionele berichten. Hij wacht nu op een reactie van de cliënt.

7. ClientCertificate*
Het bericht wordt alleen door de cliënt verstuurd als antwoord op een CertificateRequest bericht van de server. De inhoud van het bericht is dezelfde als in een ServerCertificate bericht.

8. ClientKeyExchange
Dit bericht verstuurt de cliënt altijd. Het bevat de premaster key versleuteld met RSA, of in het geval van Diffie-Hellman, parameters die het voor server en cliënt mogelijk maken om de premaster key te kunnen berekenen.

9. CertificateVerify*
Wanneer de server een CertificateRequest aan de cliënt heeft gestuurd, stuurt de client een certification chain op. Aan de hand daarvan kan de server de echtheid van het certificaat uit 7 controleren. Dit geldt alleen voor certificaten die voor het ondertekenen van berichten gebruikt kunnen worden, dus alle behalve certificaten met vaste Diffie-Hellman parameters.

10. [ChangeCipherSpec]
Dit bericht verstuurt de cliënt om de server mee te delen dat vanaf nu de master key gebruikt zal worden.

11. Finished
Het Finished bericht van de cliënt is het eerste bericht dat met de master key is versleuteld. De inhoud van het bericht is een hash en een MAC van de hele handshake conversatie. De server kan het bericht ontsleutelen en verifiëren.

12. [ChangeCipherSpec]
De server laat de cliënt weten dat hij vanaf nu de master key zal gebruiken.

13. Finished
De inhoud van dit bericht is hetzelfde als in (11).

Nadat het handshake proces succesvol is verlopen wordt het applicatie protocol geactiveerd dat de data versleutelt voordat deze via het record protocol wordt verstuurd.

De volgorde van de berichten mag niet afwijken van de net genoemde (optionele berichten kunnen ontbreken). Wanneer server of cliënt een bericht ontvangt dat niet in de volgorde van de eerder verstuurde/ontvangen berichten past, is het resultaat een foutmelding die door het AlertProtocol wordt gegeven en de verbinding wordt onderbroken. Het ontbreken van optionele berichten leidt uiteraard niet tot een foutmelding.

In het geval van het herstellen van een oude verbinding stuurt de server meteen na het ServerHello bericht het ChangeCipherSpec bericht gevolgd door het Finished bericht. De cliënt antwoordt met dezelfde berichten en de verbinding met de al eerder afgesproken parameters kan opnieuw gebruikt worden.

ChangeCipherSpec
Onafhankelijk van het gekozen algoritme om de sleutels te delen, wordt altijd van hetzelfde algoritme gebruik gemaakt om de master key te berekenen.

$$\text{Master_secret} = \text{PRF}(\text{pre_master_secret}, \text{“master secret”}, \text{ClientHello.random}, \text{ServerHello.random})[0..47];$$

De master sleutel is altijd 48 bytes lang. De premaster key verschilt qua lengte afhankelijk van het feit of RSA of Diffie-Hellman wordt gebruikt. PRF staat voor *PseudoRandomFunction* en deelt de invoerwaarde in twee gelijke delen op. Van het eerste deel wordt de MD5 hash waarde berekend en van het tweede deel de SHA-1 hash waarde. De MAC is het resultaat van de XOR van de twee hash waarden. Het gebruiken van twee onafhankelijke hash functies vergroot de veiligheid van de master key.

2.3 Versleutelingsalgoritmen & key exchange

TLS 1.2 ondersteunt drie versleutelingsalgoritmen. Deze algoritmen verschillen per versleutelingssterkte en key exchange methode. De key exchange methode levert uiteindelijk de masterkey op die gebruikt zal worden bij het versleutelingsalgoritme. Omdat beide key exchange methoden voor alle vier versleutelingsalgoritmen gebruikt kunnen worden valt er toch uit een groot aantal combinaties te kiezen. In deze sectie worden eerst de mogelijke key exchange methoden uitgelegd. Daarna volgt een beschrijving van de versleutelingsalgoritmen.

2.3.1 Key exchange methoden

RSA

De naam RSA staat voor de namen van de ontwikkelaars Rivest, Shamir en Adleman. Deze hebben in 1978 een methode ontwikkeld om via een onveilig kanaal op een veilige manier sleutels uit te kunnen wisselen. In [Tan03] wordt ervan uitgegaan dat RSA veilig is wanneer tenminste een sleutel van 1024 bits wordt gebruikt. Kleinere sleutels hebben een negatief effect op de veiligheid van RSA. Twee personen kunnen een gemeenschappelijke sleutel via een onveilig kanaal op de volgende manier delen [Tan03]:

1. B (server) kiest twee grote priemgetallen p en q (meestal 1024 bits)
2. B berekent $n = p * q$ en $z = (p - 1) * (q - 1)$
3. B kiest een getal d dat relatief priem is met z. De getallen d en z hebben dus geen gemeenschappelijke deler behalve het getal 1.
4. B berekent e zodat geldt: $e * d = 1(mod\ z)$.

De ciphertext C van een bericht P wordt berekend door $C = P^e$ (mod n) te berekenen. Een versleuteld bericht wordt ontsleuteld door $P = C^d(mod\ n)$ te berekenen. Om een bericht te kunnen versleutelen, heeft A (cliënt) e en n nodig. Voor het ontsleutelen heeft B d en n nodig. Samenvatting:

- $Ciphertext = Plaintext^e(mod\ n)$
- $Plaintext = Ciphertext^d(mod\ n)$
- $Publickeypair(e, n)$
- $Privatekeypair(d, n)$

De geheime waardes zijn: p, q, d en z
De publieke waardes zijn: e en n.

RSA is een asymmetrisch encryptiealgoritme omdat het gebruik maakt van een publieke en private sleutel. De veiligheid van RSA berust op een wiskundig probleem, namelijk het ontbinden van heel grote getallen in factoren. De priemgetallen p en q worden nooit publiek gemaakt en zonder een van de twee te weten is het niet mogelijk het getal $p * q$ te ontbinden zolang p en q groot genoeg zijn. In [Tan03] wordt beweerd dat je voor het ontbinden van een 500-digit getal 10^{25} jaar nodig hebt met een brute force attack.

Diffie Hellman

Diffie Hellman key exchange (DH) werd in 1976 ontwikkeld door Whitfield Diffie and Martin Hellman. Het is een symmetrisch encryptiealgoritme omdat zender en ontvanger een gemeenschappelijke sleutel delen. Twee partijen A en B kunnen op de volgende manier een gemeenschappelijke sleutel delen [Tan03]:

1. A en B spreken af dat het priemgetal p en de basis g (*) gebruikt zullen worden.
2. A kiest een geheim getal a en stuurt $(g^a \ mod \ p)$ naar B
3. B doet hetzelfde met zijn eigen geheim getal b en stuurt $(g^b \ mod \ p)$ naar A.
4. A berekent $\left(g^b \ mod \ p\right)^a \ mod \ p$
5. B berekent $(g^a \ mod \ p)^b \ mod \ p$

(*) g is primitive root (mod p), voor alle integers a met grootste gemeenschappelijke deler $GGD(a, p) = 1$ geldt, dat een integer q bestaat met $g^q \equiv a \ (mod \ p)$ Het resultaat van bericht 4 en 5 is hetzelfde en is tegelijkertijd de gemeenschappelijke sleutel voor het versleutelen van berichten.

De geheime waardes zijn: a, b en $g^{ab} = g^{ba}$
De publieke waardes zijn: p, g, $(g^a \ mod \ p)$ en $(g^b \ mod \ p)$

De veiligheid van DH berust op het wiskundig probleem van de discrete logaritme. Wanneer sprake is van DHE (Diffie Hellman ephemeral) betekent dit dat de sleutel van de ontvanger statisch en geverifieerd is [Res99]. Dit betekent dat de sessieparameters per sessie niet veranderen. Daarentegen genereert de zender voor elke sessie een nieuwe sleutel waardoor de gemeenschappelijke sleutel altijd weer verandert.

In beide gevallen (RSA of DH) is het voor de twee partijen niet mogelijk om te verifiëren dat ze ook daadwerkelijk met de gewenste persoon aan het praten zijn. Dus moeten de berichten in beide gevallen op een of andere manier geverifieerd worden. Dit wordt binnen TLS indien nodig via certificaten afgehandeld.

2.3.2 Versleutelingsalgoritmen

RC4

RC4 of Arcfour is een stream cipher versleutelingsalgoritme. Stream cipher betekent dat een plaintext bitstream met een keystream van dezelfde lengte gecombineerd wordt. Dit gebeurt meestal met een XOR functie. De sleutellengte varieert tussen 40 bits en 256 bits.

RC4 in pseudocode [KT97]:

```
1.              S [0] .. S [255];

2.              S [0] = 0; S [1] = 1; etc. up to S [255] = 255;
```

```
3.  for (i = 0; i < 256; i = i + 1)
                  S2 [i] = key [i % keylen]; // key: sleutel
                        // keylen: sleutellengte
4. j = 0;
for (i = 0; i < 256; i = i + 1)
                 {
                 j = (j + S [i] + S2 [i]) % 256;
                 temp = S [i];
                 S [i] = S [j]; // swap([si],s[j])
                 S [j] = temp;
                 }
        i = 0; j = 0;
```

In (1) en (2) wordt een rij aangemaakt (256 elementen) en de waarde van de elementen wordt gelijk gezet aan hun index. In (3) wordt een tweede rij van dezelfde lengte aangemaakt waarin de sleutel is opgeslagen. Indien de lengte van de sleutel korter is dan de rij wordt deze herhaald totdat alle elementen van de rij een waarde hebben. In (4) worden de waarden van de twee rijen met elkaar verwisseld zodat gewaarborgd is dat de uiteindelijke keystream niet meer makkelijk achterhaald kan worden.

```
5.  i = (i+1) % 256;
               j = (j + S[i]) % 256;
               temp = S [i];
               S [i] = S [j];
               S [j] = temp;
               t = (S [i] + S [j]) % 256;
               K = S [t];
```

In (5) wordt een pseudorandom byte K berekend. K XOR ciphertextbyte levert een plaintextbyte op en K XOR plaintextbyte levert een ciphertextbyte op. De operaties in (5) worden voor alle bytes herhaald om deze te encrypten/decrypten waarbij de index steeds verhoogd wordt.

Triple DES (3DES)

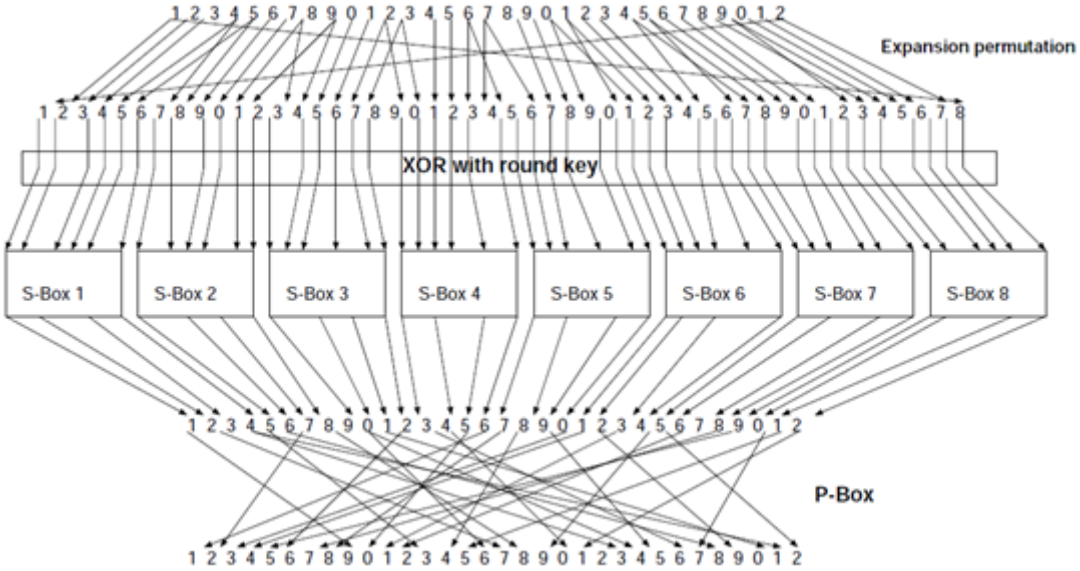
3DES is gebaseerd op de versleutelingstechniek DES (data encryption standard). Het woord Triple betekent dat het algoritme DES drie keer wordt aangeroepen om een bepaalde plaintext te versleutelen. Hierbij wordt gebruik gemaakt van twee of drie verschillende sleutels voor elke aanroep van het algoritme. Gebruikelijk zijn twee sleutels waarbij een sleutel twee keer wordt gebruikt [Tan03].

DES maakt gebruik van een 56 bit sleutel. Uiteindelijk lijkt de sleutel wel op een 64 bit sleutel maar per byte wordt één bit gebruikt voor parity [KPS02] en zo'n bit hoort dus niet bij de sleutel zelf. DES hoort bij de *symmetric-key block ciphers*. Dat wil zeggen dat dezelfde sleutel voor encryptie en decryptie wordt gebruikt en dat voordat een plaintext wordt versleuteld deze eerst wordt opgedeeld in blokken van dezelfde lengte.

DES zoals beschreven in [oST99]:

1. Deel de plaintext op in 64 bits blokken.
2. Bereken 16 x 48bits roundkeys van de master key (*)
3. Voor elk blok:
   a. Deel het blok in twee delen (2x32bits) en maak er twee 48bits blokken van door op een vastgestelde manier elementen te dupliceren.
   b. XOR de twee blokken met elke roundkey en gebruik het resultaat steeds weer als input voor de volgende XOR operatie (dus in totaal 16 XORs).
   c. Deel elk blok op in 8 x 6 bits s-boxes (substitution boxes). Elk blok wordt met behulp van een tabel (niet lineair) vertaald naar een 4 bit blok.
   d. De P-box (permutation box) verandert de structuur van de 32 bits (8 x 4 bits) en het resultaat is een 32 bit ciphertextblock.
   e. Voeg de twee blokken weer samen tot een 64 bit blok.

(*) De zestien roundkeys worden van de master key afgeleid. Eerst wordt de 56 bit masterkey in 2 gelijke delen (2x 28 bits) opgedeeld. Daarna worden de bits van de twee blokken een of twee naar links verplaatst en van beide delen worden 24 bits afgepakt die samengevoegd een roundkey zijn. De rotatie van de bits in de blokken wordt zodanig herhaald totdat alle 16 roundkeys gegenereerd zijn.



Figuur 5: TDES versleutelingsprocedure

AES

Net als DES is ook AES een symmetric-key block cipher. De lengte van de blokken is vastgelegd op 128 bits en mogelijke sleutellengten zijn 128, 192 en 256 bits waarbij de 192 bit variant door de meeste systemen niet wordt ondersteund [Tan03]. AES maakt gebruik van arrays voor het opslaan van plaintext. Voor elk blok wordt een 4x4 byte array (16 bytes = 128 bits), de state, aangemaakt.

Het principe van AES is ook zoals bij DES op roundtransformations gebaseerd waarbij het aantal ronden waar processen worden uitgevoerd afhankelijk is van de sleutellengte (128 bits = 10 rounds, 192 bits = 12 rounds, 256 bits = 14 rounds) [oST01]. Een aantal processen wordt herhaaldelijk uitgevoerd dat ten slotte als resultaat een cipherblock oplevert.

AES zoals beschreven in [oST01]:

Voor elke round geldt:

1. **SubBytes():** Elk byte van een blok wordt getransformeerd (S-Box). De nieuwe waarde wordt in de substitutie tabel van de S-Box opgezocht.
2. **ShiftRows():** De elementen van de tweede, derde en vierde rij worden onder elkaar op een vastgestelde manier uitgewisseld (2e rij: 1xshiftLeft, 3e rij: 2x shiftLeft, 4e rij: 3xshiftLeft).
3. **MixColumns():** De kolommen worden verwisseld. Eerst worden alle 16 velden van de state met een constante vermenigvuldigd. De constanten liggen van tevoren vast en zijn altijd dezelfde en worden in een tabel opgezocht. Daarna wordt de XOR functie op de resultaten toegepast.
4. **AddRoundKey():** De op dit moment geldige roundkey wordt met de state middels XOR gecombineerd.

Dit zijn de standaardprocessen die voor elke round gelden, maar er zijn twee uitzonderingen:

1. In de eerste ronde wordt alleen de AddRoundKey() functie toegepast.
2. In de laatste ronde wordt de MixColumns() funtie niet toegepast, dus alleen SubBytes, ShiftRows en AddRoundKey.

De SubKeys worden voor alle ronden van de MainKey afgeleid. Daarbij wordt Rijndael's Key-Schedule methode gebruikt. Gedetailleerde uitleg over hoe een subkey wordt berekend is terug te vinden in [oST01].

Camellia

De symmetric-key Block cipher Camellia gebruikt zoals AES blokken van lengte 128bits [AIK+00]. Mogelijke keylengten zijn 128, 196 en 256 bits. Ook hier wordt weer door middel van round-transformations en een S-Box een cipherblock gegenereerd. Het aantal ronden waar processen worden uitgevoerd komt niet overeen met AES. Voor de sleutellengte 128bit zijn het 18 ronden

en voor 196/256 bit sleutels is voor 24 ronden gekozen. Camellia werd in 2005 aan het TLS protocol toegevoegd [MKK05].

Camellia zoals beschreven in [AIK+01]:

1. Bereken de subkeys $kw_{t(64)}, k_{u(64)}, kl_{v(64)}$. (t=1,2,3,4), (u=1,2,...,18) en (v=1,2,3,4) voor 128bit keys. Voor 192/256 bit keys loop u door tot 24 en v tot 6.
2. XOR de plaintext P met $kw_{1(64)} \| kw_{2(64)}$ en deel het resultaat op in twee 64 bit blokken L en R: $P_{(128)} \oplus \left(kw_{1(64)} \| kw_{2(64)}\right) = L_{0(64)} \| R_{0(64)}$
3. Voor elk round (behalve r= 6 en 12) geldt:
   $L_r = R_{r-1} \oplus F(L_{r-1}, k_r)$,
   $R_r = L_{r-1}$
4. Voor round 6 en 12 geldt:
   $L'_r = R_{r-1} \oplus F(L_{r-1}, k_r)$ (*)
   $R'_r = L_{r-1}$
   $L'_r = FL(L'_r, kl_{2r/6-1})$ (*)
   $R'_r = FL^{-1}(R'_r, kl_{2r/6})$
5. De ciphertext is $C_{128} = L_{18(64)} \| R_{18(64)} \oplus kw_{3(64)} \| kw_{4(64)}$

(*) F en FL zijn functies die een subkey k gebruiken om een 64 bit input-blok naar een 64 bit output-blok te transformeren.

De block cipher heeft de afgelopen jaren een aantal standaardiseringprocessen doorlopen en steeds meer applicaties en protocollen zijn compatibel met Camellia [Lab].

3 Veiligheid van versleutelingsmethoden

Het *National Institute of Standards and Technology* (NIST) meet de veiligheid van een algoritme aan de hand van de tijd die het kost om alle mogelijke keys te testen [oST06]. Dit heeft alleen waarde als een brute force attack de meest efficiënte methode is om de informatie te achterhalen. Het NIST geeft de veiligheid van een algoritme aan de hand van bits of security aan. Wanneer een algoritme met een key van Y bits vergelijkbaar is met een ander symmetrisch algoritme dat gebruik maakt van een key met X bits dan is X het aantal bits of security. Twee algoritmen zijn vergelijkbaar wanneer de tijd die het kost om de algoritmen te kraken dezelfde is (attack methode en computercapaciteit zijn hetzelfde). Het zou 2^{X-1} T kosten om een brute force attack uit te voeren. T is de tijd in milliseconden die benodigd is om een plaintext value te versleutelen en met de corresponderende ciphertext value te vergelijken. De eerste kolom van onderstaande tabel geeft de bits of security weer. In de tweede kolom wordt het bijbehorende versleutelingsalgoritme genoemd waarbij 2DES een variant van 3DES is waar de eerste en de derde key dezelfde zijn. In het geval van 3DES zijn alle drie keys onafhankelijk. De derde kolom geeft de minimum lengte van de public key (L) en de private key (N) aan wanneer een key exchange methode gebruikt wordt die gebaseerd is op finite field cryptography (FFC) zoals DSA of DH. In de

vierde kolom is de key lengte aangegeven waarvoor gekozen moet worden in het geval van RSA key exchange (integer factorization cryptography, IFC). De vijfde kolom geeft de range van F aan in het geval van elliptic curve cryptography (ECC). De waarde van F komt overeen met de benodigde key lengte. De zesde kolom geeft de tijd in jaren aan die benodigd wordt om het algoritme te kraken. Wanneer voor een sterk versleutelingsalgoritme wordt gekozen moet ook de key exchange methode sterker beveiligd zijn. Om die redenen neemt de key lengte van de key exchange methode in onderstaande tabel toe. Een versleutelingsalgoritme kan alleen de benodigde maat aan veiligheid garanderen wanneer ook de key exchange methode dit kan garanderen.

Bits of security	**Symmetric key algorithms**	**FFC (DSA, DH)**	**IFC (RSA)**	**ECC (ECDSA)**	2^{X-1} **(*) (in jaren)**
80	2DES	L=1024 N=160	K=1024	F=150-223	$\approx 1,916^{13}$
112	3DES	L=2048 N=224	K=2048	F=224-255	$\approx 8,232^{22}$
128	AES-128	L=3072 N=256	K=3072	F=256-383	$\approx 5,395^{27}$
192	AES-192	L=7680 N=384	K=7680	F=384-511	$\approx 9,995^{46}$
256	AES-256	L=15360 N=512	K=15360	F=512+	$\approx 1,835^{66}$

Tabel 1: Veiligheid van DES en AES varianten [oST06]

Aanname: Het versleutelen en het vergelijken met het corresponderende ciphertext value duurt 1 milliseconde.

De cipher suite Camellia gebruikt dezelfde blocklengte als AES (128, 192 en 256 bits) [AIK+00]. De onderliggende rekenoperaties hebben veel gemeenschappelijk met AES. Camellia en AES gebruiken allebei een S-Box. De bits of security van Camellia liggen waarschijnlijk in de beurt van de AES bits of security.

Naast het vergelijken van DES en AES varianten geeft het NIST de aanbeveling dat de 2DES variant alleen maar nog tot 2010 gebruikt mag worden [oST06]. Daarna kan deze niet meer als veilig gezien worden. 3DES is volgens het NIST nog tot 2030 veilig en de drie AES varianten worden zelfs na 2030 nog als veilig aangezien. De resultaten zijn op statistische analyses gebaseerd en hoeven niet correct te zijn.

RC4 wordt door het NIST niet in de lijst opgenomen. Er is in 2005 al een attack op een met RC4 beveiligde verbinding succesvol geweest waarbij de key (128bit) in minder dan een minuut met een normale thuiscomputer achterhaald werd [TWP07]. Het probleem ligt vooral in het key scheduling algoritme dat niet altijd random keys genereert.

3.1 OpenSSL

OpenSSL (www.openssl.org) is een tool waarmee een http verbinding tussen cliënt en webserver beveiligd kan worden. Daarnaast kun je met OpenSSL ook de configuratie van webservers testen door bepaalde requests aan deze te sturen. Wanneer voor OpenSSL als testtool is gekozen is er de mogelijkheid om zelf een testset van cipher suites te definiëren of een van de meegeleverde sets te gebruiken. OpenSSL doet de volgende uitspraken over de veiligheid van cipher suites:

1. **High encryption:** Voor de key lengte K geldt: K>128bits (inclusief AES-128)

2. **Medium encryption:** Voor de key lengte K geldt: K = 128bits (exclusief AES-128)

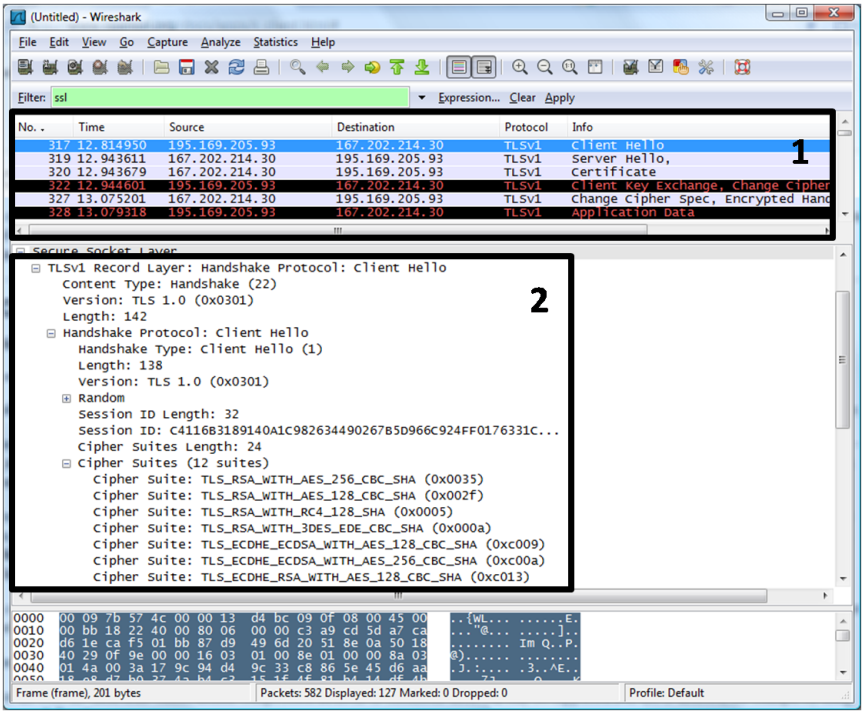
3. **Low encryption:** Voor de key lengte K geldt: K<128bits

Het valt op dat in alle drie gevallen ook de cipher suites zonder authenticatie geactiveerd zijn. Dit heeft het gevaar dat wanneer zonder certificaten (authenticatie) wordt gecommuniceerd, de andere partij niet de partij hoeft te zijn waar je denkt mee te praten.

In de categorie high encryption komt AES (128 & 256) en 3DES met drie onafhankelijke keys terecht. De 3DES variant met 128bit, waarbij de eerste en de derde key dezelfde zijn, en RC4 komen in de medium encryption categorie terecht. Alle single DES varianten zijn van type low encryption.

4 Browsers

In hoofdstuk 2 is op de specificaties van TLS ingegaan en gedetailleerd uitgelegd hoe de communicatie verloopt. In dit hoofdstuk wordt de implementatie van TLS in browsers beschreven. Het is vooral interessant om te kijken welke cipher suites een browser aan de server in het ClientHello bericht aanbiedt. Dit wordt achterhaald met de tool Wireshark (www.wireshark.org) dat de in- en uitgaande berichten van een netwerkkaart scant en de structuur van de berichten analyseert. In de onderstaande figuur is een TLS ClientHello bericht afgebeeld dat met Wireshark is opgevangen. Het programma biedt de mogelijkheid om alle ontvangen berichten te filteren. In het plaatje is het SSL/TLS filter toegepast zodat alleen berichten van het type SSL/TLS getoond worden.

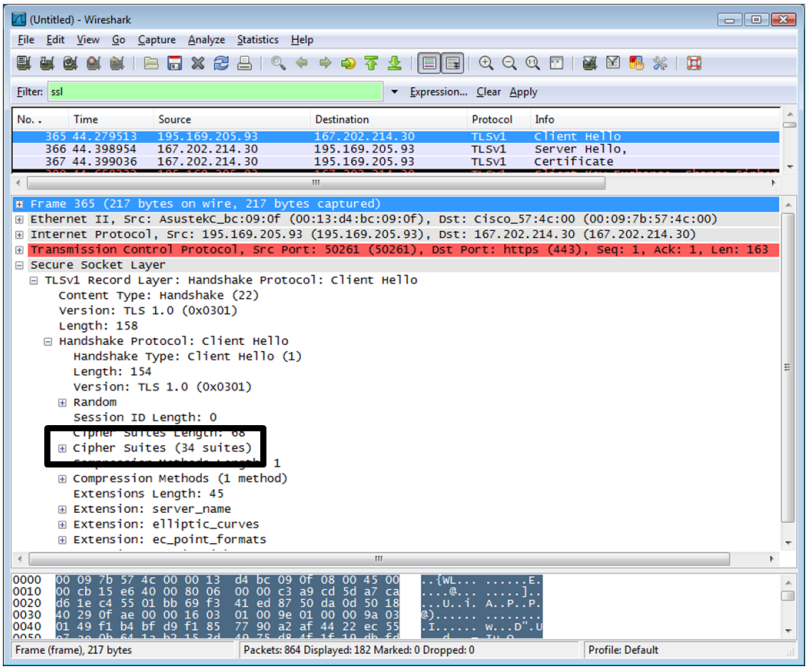


Figuur 6: Wireshark scherm

In 1 is de volgorde van verstuurde en ontvangen berichten te zien. Deze komt overeen met de theorie uit hoofdstuk 2. In 2 is een typisch ClientHello bericht afgebeeld. Het bericht houdt twaalf cipher suites in waarvan één uiteindelijk wordt gekozen (niet alle twaalf cipher suites zijn afgebeeld).
In de volgende secties wordt de configuratie van TLS in vier verschillende browsers beschreven. Deze zijn Mozilla Firefox, Microsoft Internet Explorer, Google Chrome en Apple's Safari. Alle browsers draaien op het besturingssysteem Windows Vista Business SP1 van Microsoft. Daarnaast beschikt het besturingssysteem over het protocol Schannel (secure channel) [Net]. Schannel bevat veertien cipher suites die met TLS compatibel zijn. In de standaard configuratie zijn er twaalf van de veertien cipher suites geactiveerd, alleen TLS_RSA_WITH_NULL_MD5 en TLS_RSA_WITH_NULL_SHA zijn niet geactiveerd omdat deze geen versleuteling bieden maar alleen authenticatie. De twaalf geactiveerde cipher suites worden in de onderstaande tabel onder het kopje "Internet Explorer, Chrome & Safari" weergegeven. De Business versie van Vista beschikt over een Group Policy Editor (Start → Run → "gpedit.msc") waarmee de volgorde en de beschikbaarheid van de cipher suites gewijzigd kan worden. De voorkeur voor bepaalde cipher suites kan dus veranderd worden en cipher suites kunnen van de lijst verwijderd worden.

4.1 Mozilla Firefox

TLS is in de huidige versie (3.0.8) van Firefox geïmplementeerd. Naast TLS ondersteunt Firefox ook de voorganger SSL v2 en v3 waarbij SSL v2 in de standaardconfiguratie niet geactiveerd is omdat deze versie niet meer als veilig gezien wordt [Cen].



Figuur 7: Wireshark: FireFox cipher suites

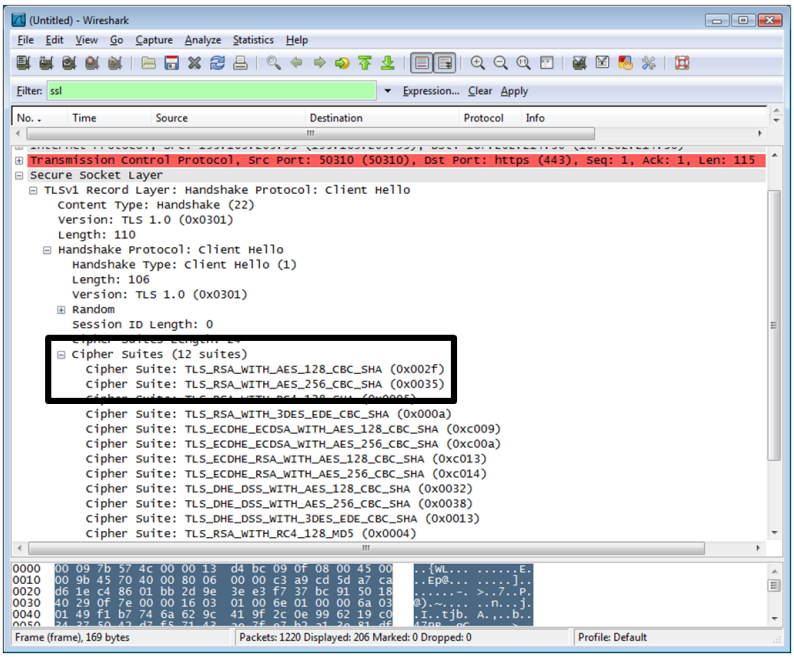
Firefox biedt de eindgebruiker de keuze om zelf te bepalen of een cipher suite wel of niet gebruikt mag worden. Dat betekent dat je zelf deze instellingen kunt maken. De volgorde van

de cipher suites in de lijst kan echter niet zelf bepaald worden omdat het configuratiescherm van Firefox het alleen toestaat om een cipher suite wel of niet aan te vinken. Firefox stuurt in de standaardconfiguratie een lijst met 34 cipher suites naar de server (zie figuur 7).

De ontwikkelaars van Firefox hebben ervoor gekozen om alle cipher suites zonder versleutelingsalgoritme te deactiveren, maar het is wel mogelijk om deze met de hand aan te zetten. Daarnaast zijn alle single DES varianten niet geactiveerd omdat deze niet meer als veilig gezien worden, maar ook deze cipher suites kunnen aangezet worden. Firefox is de enige browser die over een eigen cipher suite lijst beschikt. De volgorde van de aangeboden cipher suites is in tabel 2 beschreven.

4.2 Microsoft Internet Explorer, Google Chrome & Apple Safari

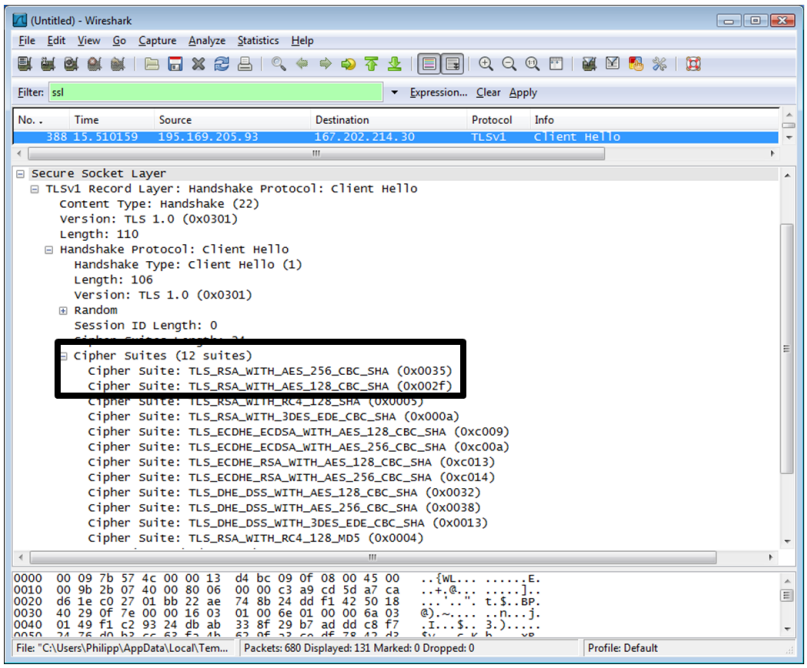
Microsoft Internet Explorer 8, Google Chrome 1.0.1 & Apple Safari 3.2.2 beschikken niet over een eigen cipher suite lijst. Ze gebruiken de door het Schannel protocol aangeboden cipher suites [Net]. De configuratie van de browsers zelf kan niet veranderd worden zodanig dat de cipher suites veranderen. Het onderstaande ClientHello bericht is voor alle drie browsers hetzelfde, alleen de random numbers en timestamps verschillen uiteraard. In alle drie gevallen worden precies dezelfde twaalf cipher suites aangeboden.



Figuur 8: Wireshark: Schannel cipher suites

Om echter aan te kunnen tonen dat IE, Chrome & Safari daadwerkelijk de Schannel cipher suites gebruiken moet een test uitgevoerd worden. De eerste twee cipher suites in Schannel worden met elkaar verwisseld. Het resultaat is voor alle drie browsers hetzelfde (zie figuur 9).

AES 256bit staat in alle drie gevallen nu aan het begin van de cipher suite lijst. De bewering, dat de Schannel cipher suite gebruikt wordt, is dus juist.



Figuur 9: Wireshark: Schannel cipher suites (veranderde volgorde)

Het valt op dat de voorkeur wordt gegeven aan cipher suites met een kortere key. Bijvoorbeeld is de meest wenselijke cipher suite in de standaardconfiguratie TLS_RSA_WITH_AES_128_CBC_SHA en daarna volgt TLS_RSA_WITH_AES_256_CBC_SHA.
In tabel 2 worden de cipher suites weergegeven, die door de browsers in het ClientHello bericht worden aangeboden. De cijfers onder de browserkopjes staan voor de volgorde in de cipher suite lijst. Zoals in hoofdstuk 2 uitgelegd is de eerste cipher suite het meest wenselijk en de laatste cipher suite het minst wenselijk. Wanneer geen cijfer bij een bepaalde cipher suite staat betekent dit dat deze niet door de browser wordt aangeboden.

Cipher suite	**Firefox**	**Internet Explorer, Chrome & Safari**
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	1	6
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	2	8
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	3	
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA	4	
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	5	
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	6	10
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	7	
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	8	
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	9	
TLS_RSA_WITH_AES_256_CBC_SHA	10	2
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	11	
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	12	5
TLS_ECDHE_RSA_WITH_RC4_128_SHA	13	
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	14	7
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	15	
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA	16	
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	17	
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	18	9
TLS_ECDH_RSA_WITH_RC4_128_SHA	19	
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	20	
TLS_ECDH_ECDSA_WITH_RC4_128_SHA	21	
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	22	
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	23	
TLS_RSA_WITH_RC4_128_MD5	24	12
TLS_RSA_WITH_RC4_128_SHA	25	3
TLS_RSA_WITH_AES_128_CBC_SHA	26	1
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	27	
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	28	
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	29	
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	30	11
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	31	
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	32	
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA	33	
TLS_RSA_WITH_3DES_EDE_CBC_SHA	34	4

Tabel 2: Ondersteunde cipher suites per browser

Een opmerkelijk verschil tussen de kolom van Firefox en die van IE, Chrome en Safari is dat Firefox een lijst met 34 cipher suites in het ClientHello bericht verstuurt. Daarentegen verstuurt Schannel van Microsoft slechts een lijst met twaalf cipher suites. Ook de prioriteiten die Microsoft en Firefox ofwel Mozilla aan de cipher suites geven zijn enorm verschillend. De meest geprefereerde cipher suite van Schannel staat in de cipher suite lijst van Firefox slechts op plek 26. Verder komen maar slechts vier cipher suites van de Top10 cipher suites van Schannel in de Top10 van cipher suites van Firefox terug. De conclusie is vooral dat de opinies over bepaalde cipher suites per ontwikkelaargroep zeer verschillend zijn.

5 Methode

In dit hoofdstuk wordt de opbouw van het experimentele gedeelte van dit onderzoek beschreven. Ten eerste wordt uitgelegd welke software gebruikt wordt. Ten tweede volgt een lijst met organisaties die over een met TLS beveiligde webserver beschikken. Deze vormen de testgroep. Ten slotte worden alle testsets in een lijst weergegeven die aan de webservers worden aangeboden.

5.1 Software

Alle software die de opties ondersteunt dat de gebruiker zelf de cipher suite lijst en de volgorde van de cipher suites kan bepalen, kan gebruikt worden. Hier is gekozen voor OpenSSL. Schannel had ook voor AES, 3DES en RC4 testsets gebruikt kunnen worden, maar Firefox niet omdat de volgorde van de cipher suites niet aangepast kan worden.

Met OpenSSL kun je zelf het ClientHello bericht in elkaar zetten en dus ook de cipher suites en de volgorde zelf bepalen. Een voorbeeld is:

OpenSSL: s_client -connect SERVER:443 -tls1 -cipher CIPHER

Dit is het commando om een OpenSSL een ClientHello bericht naar de server te laten sturen. In plaats van SERVER komt het webadres van de server te staan. CIPHER staat voor een cipher suite of een lijst van cipher suites die door dubbele punten van elkaar gescheiden worden (bv. DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA). De volgorde in deze lijst is tevens de voorkeursvolgorde. Merk op: OpenSSL gebruikt het "TLS_" prefix niet, maar voor de rest zijn de namen gelijk aan die uit tabel 2. Het besturingssysteem waarmee de tests worden uitgevoerd is Windows Vista Business SP1. Dat impliceert dat de Windows versie van OpenSSL (0.9.8k) gebruikt wordt.

5.2 Webservers

De webservers worden onderverdeeld in twee testgroepen. In de eerste testgroep komen alleen webservers terecht die een hoge mate aan beveiliging moeten bieden. Dit zijn voornamelijk webservers van banken. De tweede testgroep bevat webservers van webwinkels die een normale mate aan beveiliging moeten bieden.

De volgende webservers worden getest:

1. **Banken (high security):** Rabobank, ABN AMRO, ING, SNS Bank en Friesland Bank
2. **Webwinkels (normal security):** BOL, Amazon, Ebay, Wehkamp en TakeItNow

Aan alle webservers zullen dezelfde testsets van cipher suites aangeboden worden. Deze worden in sectie 5.3 gedefinieerd. In Appendix B zijn de "https" adressen van de webservers vermeld.

De Friesland Bank is een van de onbekendste banken van Nederland. De vraag die hier voor de hand ligt, is of onbekende banken dezelfde veiligheid bieden als bekende banken. Hetzelfde geldt voor de onbekende webwinkel TakeItNow.

5.3 Cipher suite testsets

In deze sectie worden de testsets gedefinieerd die aan de webservers worden aangeboden.

Testsets

De cipher suites zijn afkomstig uit OpenSSL.

1. Alle cipher suites zonder versleutelingsalgoritme.
2. Alle cipher suites met versleutelingsalgoritme RC4.
3. Alle cipher suites met versleutelingsalgoritme Single DES.
4. Alleen AES128 en AES256 cipher suites.
5. Alleen AES128, AES256 en 3DES cipher suites.
6. Alleen AES256,AES128 en Single DES, maar Single DES staat op de eerste plek.
7. Alleen Camellia cipher suites.
8. Alleen cipher suites met Diffie-Hellman emphemeral (DHE/EDH)key exchange

De verwachting is dat de eerste drie testsets niet door webservers geaccepteerd worden omdat deze niet meer als veilig gezien worden. Voor de vierde en vijfde testset is gekozen om te achterhalen wat de voorkeuren van webservers zijn indien alleen high security cipher suites worden aangeboden. Vragen die gesteld moeten worden zijn: Is er een voorkeur voor 256bit sleutels in het geval van AES? Wordt er een voorkeur aan AES of aan TDES gegeven? Met de zesde testset wordt achterhaald of een zwak algoritme op de eerste plek de keuze van de server zodanig beïnvloedt dat de verbinding niet veilig is. Testset 7 bevat alleen Camellia cipher suites. Zien de webservers de relatief nieuwe Camellia cipher suites als een optie of worden deze afgekeurd? Met testset 8 wordt achterhaald of er webservers zijn die geen DHE cipher suites ondersteunen. De volledige lijst met alle testsets en cipher suites staat in Appendix A.

6 Resultaten

In deze sectie worden de met OpenSSL verkregen resultaten weergegeven. Voor het uitvoeren van het experiment werd de hypothese opgesteld dat de webservers de eerste drie testsets niet zullen accepteren. Dit blijkt alleen te gelden voor de eerste testset.

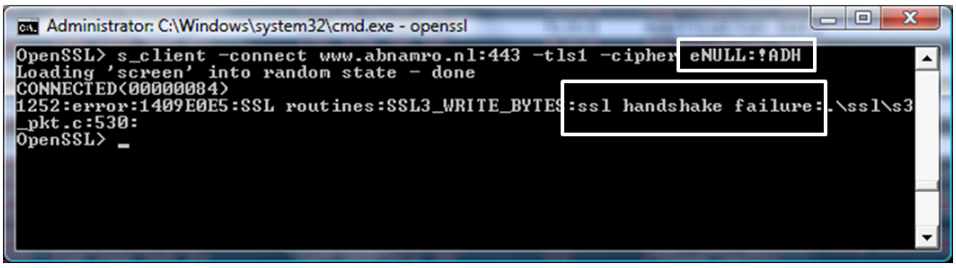
6.1 Testset 1

- Alle webservers weigeren en geven een handshake failure notificatie.

De ABN AMRO server weigert cipher suites zonder versleuteling (figuur 10). Het prefix eNULL:!ADH betekent dat alleen NULL cipher suites met authenticatie worden gebruikt (testset 1), dus geen anonymous Diffie-Hellman.

6.2 Testset 2

- Alle webservers behalve BOL accepteren het.



Figuur 10: OpenSSL: handshake failure

6.3 Testset 3

- Friesland Bank, Amazon en TakeItNow accepteren single DES cipher suites.
- De rest accepteert het niet.

6.4 Testset 4a/b

- ING, SNS Bank en BOL accepteren geen AES cipher suites en geven een handshake error.
- Rabobank, ABN AMRO, Amazon en Ebay geven altijd de voorkeur aan AES256 cipher suites. (Alle vier ondersteunen wel AES128 wanneer AES 256 niet wordt aangeboden.)
- Friesland Bank en Wehkamp geven altijd de voorkeur aan AES128 cipher suites. (Voor de zekerheid is er een extra test gedaan. Beide ondersteunen ook AES256 cipher suites.)
- TakeItNow let op de volgorde en kiest de AES variant waaraan de gebruiker de voorkeur geeft.

6.5 Testset 5a/b

- ING accepteert de testset niet.
- ABN AMRO, SNS Bank, Friesland Bank, BOL, Amazon en Ebay geven altijd de voorkeur aan 3DES cipher suites, onafhankelijk van de cipher suite volgorde.
- De Rabobank geeft altijd de voorkeur aan AES256 en Wehkamp aan AES128 cipher suites.
- Ook hier kijkt TakeItNow weer naar de voorkeur van de client en maakt de keuze hiervan afhankelijk.

6.6 Testset 6

- Webservers die al eerder geen AES en single DES cipher suites accepteerden, zoals ING, SNS Bank en BOL, geven een foutmelding.
- Rabobank, ABN AMRO, Amazon en Ebay kiezen voor de 256 bit variant van AES en Wehkamp kiest voor een AES128 cipher suite. Een zwak algoritme op de eerste plek beïnvloedt dus niet de keuze die gemaakt wordt bij deze webservers.

- De webservers van Friesland Bank en TakeItNow laten zich verleiden en kiezen de single DES cipher suite. Beide maken hun keuze afhankelijk van de voorkeur van de client terwijl ze ook voor 3DES of AES hadden kunnen kiezen.

6.7 Testset 7

- Alle webservers weigeren en geven een handshake failure notificatie.

6.8 Testset 8

- Rabobank, ABN AMRO en Friesland Bank accepteren DHE/EDH cipher suites.
- De Friesland Bank let weer op de gebruikersvoorkeur.
- Rabobank en ABN AMRO geven altijd de voorkeur aan RSA cipher suites wanneer deze worden aangeboden.
- De rest accepteert het niet.

Gekozen cipher suite, webservers (banken)

Testset #	Rabo Bank	ABN Amro Bank	ING Bank	SNS Bank	Friesland Bank
1	na	na	na	na	na
2	RSA_WITH_RC4_128_MD5	RSA_WITH_RC4_128_MD5	RSA_WITH_RC4_128_SHA	RSA_WITH_RC4_128_MD5	RSA_WITH_RC4_128_MD5
3	na	na	na	na	RSA_WITH_DES_CBC_SHA
4a	RSA_WITH_AES_256_SHA	RSA_WITH_AES_256_SHA	na	na	RSA_WITH_AES_128_SHA
4b	RSA_WITH_AES_256_SHA	RSA_WITH_AES_256_SHA	na	na	RSA_WITH_AES_128_SHA
5a	RSA_WITH_AES_256_SHA	RSA_WITH_3DES_EDE_CBC_SHA	na	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA
5b	RSA_WITH_AES_256_SHA	RSA_WITH_3DES_EDE_CBC_SHA	na	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA
6	RSA_WITH_AES_256_SHA	RSA_WITH_AES_256_SHA	na	na	RSA_WITH_DES_CBC_SHA
7	na	na	na	na	na
8	EDH_RSA_WITH_AES256_SHA	EDH_RSA_WITH_AES256_SHA	na	na	na

Tabel 3: na = niet geaccepteerd, EDH: De OpenSSL prefix voor Diffie Hellman emphemeral (EDH=DHE)

Gekozen cipher suite, webservers (webwinkels)

Testset #	BOL	Amazon	Ebay	Wehkamp	TakeItNow
1	na	na	na	na	na
2	na	RSA_WITH_RC4_128_MD5	RSA_WITH_RC4_128_MD5	RSA_WITH_RC4_128_MD5	RSA_WITH_RC4_128_SHA
3	na	RSA_WITH_DES_CBC_SHA	na	na	EDH_RSA_WITH_DES_CBC_SHA
4a	na	RSA_WITH_AES_256_SHA	RSA_WITH_AES_256_SHA	RSA_WITH_AES_128_SHA	RSA_WITH_AES_128_SHA
4b	na	RSA_WITH_AES_256_SHA	RSA_WITH_AES_256_SHA	RSA_WITH_AES_128_SHA	RSA_WITH_AES_256_SHA
5a	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_AES_128_SHA	RSA_WITH_AES_128_SHA
5b	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_3DES_EDE_CBC_SHA	RSA_WITH_AES_128_SHA	EDH_RSA_WITH_3DES_EDE_CBC_SHA
6	na	RSA_WITH_AES_256_SHA	RSA_WITH_AES_256_SHA	RSA_WITH_AES_128_SHA	EDH_RSA_WITH_DES_CBC_SHA
7	na	na	na	na	na
8	na	na	na	na	EDH_RSA_WITH_AES256_SHA

Tabel 4: na = niet geaccepteerd, EDH: De OpenSSL prefix voor Diffie Hellman emphemeral (EDH=DHE)

7 Conclusie

Het valt op dat alle webservers behalve de webserver van TakeItNow steeds RSA cipher suites kiezen wanneer deze worden aangeboden. Daarnaast zijn er enkele webservers die alleen RSA cipher suites accepteren en een handshake failure bericht sturen wanneer alleen DHE cipher suites (test 8) worden aangeboden. Verder is op te merken dat de ING Bank slechts één testset accepteerde, namelijk testset 2. Er worden dus alleen RC4 cipher suites door de ING webserver goedgekeurd. De Camellia cipher suites worden door geen enkele webserver geaccepteerd. Een reden hiervoor zou kunnen zijn dat Camellia pas in de laatste jaren gestandaardiseerd werd en nog steeds niet met alle protocollen en applicaties compatibel is [Lab]. Webservers die consequent gebruik maken van AES of 3DES bieden voldoende veiligheid omdat deze cipher suites nog tot 2030 of zelfs daarna als veilig gezien worden [oST06].

De RC4 cipher suites bieden niet onder alle omstandigheden een voldoende maat aan security bieden omdat het key scheduling algoritme in bepaalde implementaties [FMS01] geen echte random keys genereert. Dat maakt het mogelijk om met een bepaalde hoeveelheid data de sessiekey te achterhalen. Dit probleem treedt bijvoorbeeld op met het wireless LAN protocol WEP [TWP07] dat van zo'n zwakke RC4 implementatie gebruik maakt. Merk op dat zelfs de verbinding nog met TLS beveiligd is indien de WEP versleuteling is gekraakt. Alle geteste webservers behalve BOL ondersteunen RC4 cipher suites en zien deze dus als voldoende veilig. Maximov en Khovratovich [MK08] hebben een algoritme ontwikkeld waarmee RC4 gekraakt kan worden. Hun aanval heeft een complexiteit van 2^{241} op de oorspronkelijke RC4-256 bit variant. Omdat de webservers en de clients in dit onderzoek alleen gebruik maken van RC4-128 bit zal de complexiteit nog aanzienlijk lager zijn en een attack zou misschien in een aannemelijke tijd uitgevoerd kunnen worden. Single DES wordt al jaren niet meer als veilig gezien en in 2006 achterhaalde de computer Copacobana een single DES key in 6,4 dagen [KPP+06].

7.1 Gevolgen

Zijn de onderzochte webservers van banken en webwinkels veilig? Hoe makkelijk kan een gebruikerswachtwoord achterhaald worden? Dit zijn vragen die zeker beantwoord moeten worden.

7.1.1 Banken

Klanten van banken moeten altijd inloggen wanneer ze gebruik willen maken van internet bankieren. De inlogprocedure berust op een challenge-response request waar (uit dit onderzoek blijkt dat dat niet altijd het geval is) de webserver van de bank een code via de beveiligde verbinding naar de klant stuurt. Vervolgens berekent de klant met behulp van de code een unieke response code en stuurt deze weer op naar de bank. Banken hebben dus twee beveiligingsmaatregelen getroffen: de met SSL/TLS beveiligde verbinding en het challenge-response systeem dat elke keer een nieuwe code genereert.

Het is dus alleen mogelijk om informatie in een acceptabele tijd te verkrijgen wanneer een zwak algoritme voor de beveiliging van de verbinding wordt gebruikt. Zelfs wanneer het lukt om de sessiesleutel te achterhalen is het nog steeds niet mogelijk om met behulp van de verkregen gegevens in te kunnen loggen. De sessie is al lang niet meer geldig en de challenge wordt door de bank bepaald. Het is wel mogelijk om informatie over de toen door de klant uitgevoerde

transacties te verkrijgen. Het feit dat ING alleen RC4 gebruikt, heeft dus geen direct gevolg voor de integrity van het systeem maar wel voor de privacy.

Merk op dat er in dit onderzoek alleen is gekeken naar de standaard betaalfunctionaliteit van internet bankieren. Als banken extra services aanbieden die geen extra challenge-response beveiliging hebben, zoals vermoedelijk ING rentepunten, komt de integriteit van het systeem wel in het geding.

De bekende banken doen het beter dan de onbekende Friesland Bank die wel een met single DES beveiligde verbinding toestaat. Dat wil niet zeggen dat alle onbekende banken hun webservers slecht beveiligen maar in dit onderzoek is dit wel het geval.

7.1.2 Webwinkels

In het geval van webwinkels verloopt de inlogprocedure niet via challenge-response. Klanten loggen zich met behulp van hun gebruikersnaam/email en het bijhorende wachtwoord in. Een zwak beveiligde verbinding kan ernstige gevolgen voor de klant opleveren. Het is bekend dat de meeste klanten hun wachtwoord niet maandelijks wijzigen. Meestal doen ze dat alleen wanneer ze niet meer weten wat hun wachtwoord ook al weer was. Een mogelijke aanval voor een met Single DES beveiligde TLS verbinding is:

1. Maak zelf een account aan en voer een attack op je eigen transactie uit om de structuur van het bericht te achterhalen.
2. Scan alle data die tussen klant en webwinkel server uitgewisseld worden en sla deze op.
3. Voer een attack op de Single DES versleuteling uit.
4. In een van de in (2) verkregen berichten is het wachtwoord (of een hash van het wachtwoord) en de gebruikersnaam te vinden.
5. Gebruik deze gegevens om in te loggen.
6. Bestel bv. een nieuw plasmatelevisie voor 2000 euro en laat deze naar een door jou opgegeven adres leveren.

De echte klant zou het pas door hebben wanneer het bedrag van zijn rekening is afgeschreven. De fraudeur heeft al lang zijn nieuwe televisie in ontvangst genomen. Indien de hash van een wachtwoord wordt verstuurd, kan van tevoren de hash van het eigen wachtwoord (middels javascript berekend) met de values van de berichten vergeleken worden om de positie van het wachtwoord in het bericht toch te weten te krijgen.

Het is dus vooral belangrijk servers die een statisch wachtwoord gebruiken met sterke versleutelingsalgoritmen te voorzien om hun klantgegevens voldoende te beveiligen. Zwakke algoritmen mogen niet ondersteund worden. De server van de onbekende webwinkel TakeItNow let op de voorkeur van de client. De server kiest voor een single DES versleuteling wannneer deze op de eerste plek van de cipher suite lijst staat terwijl ook AES of TDES cipher suites in de lijst staan. In dit onderzoek wordt ook in het geval van webwinkels aangetoond dat onbekende webservers slechter beveiligd zijn dan bekende webservers.

Referenties

[AIK+00] K. Aoki, T. Ichikawa, M. Kanda, M. Mitsuru, S. Moriai, J. Najajima, and T. Tokita, *Camellia: A 128–bit block cipher suitable for multiple platforms – design and analysis*, Selected Areas in Cryptography (2000), 39–56.

[AIK+01] ———, *Specifications of camellia – a 128-bit block cipher*, September 2001.

[Cen] Mozilla Developer Center, *Security in firefox 2*, `https://developer.mozilla.org/En/Security_in_Firefox_2`.

[DR08] T. Dierks and E. Rescorla, *The transport layer security (tls) protocol version 1.2*, Tech. Report RFC5246, Network Working Group, August 2008.

[FMS01] S. Fluhrer, I. Mantin, and A. Shamir, *Weaknesses in the key scheduling algorithm of rc4*, Lecture notes in computer science **2259** (2001), 1–24.

[fS96] International Organization for Standardization, *Open systems interconnection – basic reference model*, 1996.

[KPP+06] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, A. Rupp, and M. Schimmler, *How to break des for euro 8,980*, 2006.

[KPS02] C. Kaufman, R. Perlman, and M. Speciner, *Network security*, 2 ed., Prentice Hall PTR, 2002.

[KT97] K. Kaukonen and R. Thayer, *A stream cipher encryption algorithm arcfour*, Tech. Report Internet Draft, Network Working Group, July 1997.

[Lab] NTT Information Sharing Platform Laboratories, *Standardization related information*, `http://info.isl.ntt.co.jp/crypt/eng/camellia/standard.html#camellia_2009`.

[MK08] A. Maximov and D. Khovratovich, *New state recovery attack on rc4*, Lecture notes in computer science **5157** (2008), 297–316.

[MKK05] S. Moriai, A. Kato, and M. Kanda, *Addition of camellia cipher suites to transport layer security (tls)*, Tech. Report RFC4132, Network Working Group, July 2005.

[Net] Microsoft Developer Network, *Cipher suites in schannel*, `http://msdn.microsoft.com/en-us/library/aa374757(VS.85).aspx`.

[oST99] National Institute of Standards and Technology, *Data encryption standard (des)*, October 1999.

[oST01] ———, *Advanced encryption standard (aes)*, November 2001.

[oST06] ———, *Recommendation for key management – part 1: General*, NIST Special Publication SP800-57 (2006).

[Res99] E. Rescorla, *Diffie-hellman key agreement method*, Tech. Report RFC2631, Network Working Group, June 1999.

[Tan03] A. S. Tanenbaum, *Computer networks*, 4 ed., PTR, 2003.

[TWP07] E. Tews, R.P. Weinmann, and A. Pyshkin, *Breaking 104 bit wep in less than 60 seconds*, Lecture notes in computer science **4867** (2007), 188–202.

8 Appendix A

In deze appendix zijn de testsets gedefinieerd. De cipher suites worden in onderstaande volgorde aan de webservers aangeboden.

Testsets

```
1. Alle cipher suites zonder versleutelingsalgoritme:

TLS_RSA_WITH_NULL_SHA
TLS_RSA_WITH_NULL_MD5

2. Alle cipher suites met versleutelingsalgoritme RC4:

TLS_RSA_WITH_RC4_SHA
TLS_RSA_WITH_RC4_MD5

3. Alle cipher suites met versleutelingsalgoritme Single DES:

TLS_DHE_RSA_WITH_DES_CBC_SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_DES_CBC_MD5

4. Alleen AES128 en AES256 cipher suites:

a.
TLS_DHE_RSA_WITH_AES_128_SHA
TLS_DHE_DSS_WITH_AES_128_SHA
TLS_RSA_WITH_AES_128_SHA
TLS_DHE_RSA_WITH_AES_256_SHA
TLS_DHE_DSS_WITH_AES_256_SHA
TLS_RSA_WITH_AES_256_SHA

b.
TLS_DHE_RSA_WITH_AES_256_SHA
TLS_DHE_DSS_WITH_AES_256_SHA
TLS_RSA_WITH_AES_256_SHA
TLS_DHE_RSA_WITH_AES_128_SHA
TLS_DHE_DSS_WITH_AES_128_SHA
TLS_RSA_WITH_AES_128_SHA

5. Alleen AES128, AES256 en TDES cipher suites:

a.
```

TLS_DHE_RSA_WITH_AES_128_SHA
TLS_DHE_DSS_WITH_AES_128_SHA
TLS_RSA_WITH_AES_128_SHA
TLS_DHE_RSA_WITH_AES_256_SHA
TLS_DHE_DSS_WITH_AES_256_SHA
TLS_RSA_WITH_AES_256_SHA
TLS_DHE_RSA_3DES_EDE_CBC_SHA
TLS_DHE_DSS_3DES_EDE_CBC_SHA
TLS_RSA_3DES_EDE_CBC_SHA

b.
TLS_DHE_RSA_3DES_EDE_CBC_SHA
TLS_DHE_DSS_3DES_EDE_CBC_SHA
TLS_RSA_3DES_EDE_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_SHA
TLS_DHE_DSS_WITH_AES_128_SHA
TLS_RSA_WITH_AES_128_SHA
TLS_DHE_RSA_WITH_AES_256_SHA
TLS_DHE_DSS_WITH_AES_256_SHA
TLS_RSA_WITH_AES_256_SHA

6. Alleen AES256, AES128 en Single DES, maar Single DES staat op de eerste plek:

TLS_DHE_RSA_WITH_DES_CBC_SHA
TLS_DHE_DSS_WITH_DES_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_SHA
TLS_DHE_DSS_WITH_AES_128_SHA
TLS_RSA_WITH_AES_128_SHA
TLS_DHE_RSA_WITH_AES_256_SHA
TLS_DHE_DSS_WITH_AES_256_SHA
TLS_RSA_WITH_AES_256_SHA

7. Alleen Camellia cipher suites:

TLS_DHE_RSA_WITH_CAMELLIA256_SHA
TLS_DHE_DSS_WITH_CAMELLIA256_SHA
TLS_RSA_WITH_CAMELLIA256_SHA
TLS_DHE_RSA_WITH_CAMELLIA128_SHA
TLS_DHE_DSS_WITH_CAMELLIA128_SHA
TLS_RSA_WITH_CAMELLIA128_SHA

8. Alleen cipher suites met Diffie-Hellman emphemeral (DHE/EDH) key exchange:

Dit zijn alle cipher suites van testset 1 t/m 7 die "DHE_RSA"

```
of "DHE_DSS" als key exchange methode hebben.
```

9 Appendix B

Webadressen (https) van de banken/webwinkels:

Bank/Webwinkel	**Webadres**
Rabo Bank	https://bankieren.rabobank.nl
ABN Amro Bank	https://www.abnamro.nl
ING Bank	https://mijn.ing.nl
SNS Bank	https://www.snsbank.nl
Friesland Bank	https://internetbankieren.frieslandbank.nl
BOL	https://www.bol.com
Amazon	https://www.amazon.co.uk
Ebay	https://signin.ebay.nl
Wehkamp	https://www.wehkamp.nl
TakeItNow	https://www.takeitnow.nl

Tabel 5: HTTPS adressen van de webservers