Nationaal Cyber Security Centrum
Ministerie van Justitie en Veiligheid

Transport Layer Security (TLS)

Beveiligingsrichtlijnen versie 2025-05

Inleiding

Transport Layer Security (TLS) is een gestandaardiseerd protocol voor het opzetten en gebruiken van een versleutelde verbinding tussen twee computersystemen of applicaties: een client en een server. De beveiligde verbinding garandeert de vertrouwelijkheid en integriteit van data die uitgewisseld wordt tussen client en server. Dankzij TLS kan je veilig op het internet browsen; vertrouwelijk e-mails uitwisselen; en op afstand verbinden met jouw digitale werkplek middels een TLS-gebaseerde VPN-oplossing. TLS wordt ook gebruikt voor het beveiligen van OT-netwerken en om de systemen en applicaties in jouw eigen netwerkomgeving veilig onderling te laten communiceren.

TLS (in sommige contexten nog SSL genoemd, naar zijn voorganger Secure Socket Layer) wordt al meer dan 30 jaar (door-) ontwikkeld en biedt ruimte om specifieke opties en instellingen te configureren. Je kan bijvoorbeeld kiezen welke TLS-versie(s) je wilt ondersteunen, welke algoritme(s) je wilt gebruiken voor bulkversleuteling en/of je TLS-compressie wilt toepassen. Deze aanpasbaarheid zorgt ervoor dat je TLS op een passende manier kan gebruiken voor jouw toepassing. Helaas zijn niet alle opties even veilig.

Deze publicatie geeft je advies over het opstellen van een TLS-configuratie die jouw toepassing op een passende manier beschermt. In de komende hoofdstukken vind je hiertoe achtereenvolgens:

- Achtergrondinformatie die relevant is in het kader van deze publicatie. We beschrijven in dit hoofdstuk onder andere de algemene werking van TLS, de opties en instellingen die relevant zijn voor het realiseren van een veilige TLS verbinding, en veelgebruikte termen (hoofdstuk 1);
- Een stappenplan dat je kunt gebruiken om tot een veilige TLS-configuratie te komen (hoofdstuk 2);
- De TLS-instellingen voor een veilige TLS-configuratie (hoofdstuk 3); en
- Randvoorwaarden die je moet invullen voor een algehele veilige werking van TLS (hoofdstuk 4).

De beveiligingsrichtlijnen voor TLS zijn in 2014 voor het eerst door het NCSC gepubliceerd en worden sindsdien onderhouden. Ben je al goed bekend met de vorige versie van deze richtlijnen, die dateert uit 2021, en ben je benieuwd wat er veranderd is? In Bijlage A: Wijziging ten opzichte van de vorige versie vind je de belangrijkste wijzigingen. Daarnaast verwijzen we je naar Bijlage B: Lijst met cipher suites voor een lijst met veelgebruikte cipher suites die je kan gebruiken als onderdeel van jouw TLS-configuratie.

Doelgroep

Deze richtlijnen zijn geschreven voor beveiligingsprofessionals die een rol hebben in het configureren of beheren van een TLS implementatie en handvatten zoeken om deze passend veilig te maken.

Deze publicatie is opgesteld in samenwerking met de volgende organisaties:

- Algemene Inlichtingen- en Veiligheidsdienst (AIVD);
- Ministerie van Volksgezondheid, Welzijn en Sport;
- Forum Standaardisatie; en
- Cryptography in Context.

Daarnaast danken we de volgende personen en organisaties voor hun kritische blik en waardevolle bijdragen aan deze hierziende publicatie:

- ABN AMRO
- Arne Padmos, Adyen
- ASN Bank
- Autoriteit Persoonsgegevens
- Centric
- Cloudflare
- KPN
- Logius
- Ministerie van Algemene Zaken
- Nederlandse Vereniging van Banken
- Northwave Cyber Security
- Pi Rogaar, zelfstandig expert
- Peter-Jan Kortlever, KNAB
- SURF
- Informatiebeveiligingsdienst voor gemeenten (IBD)
- Z-CERT

Inhoudsopgave

1 Achtergrond

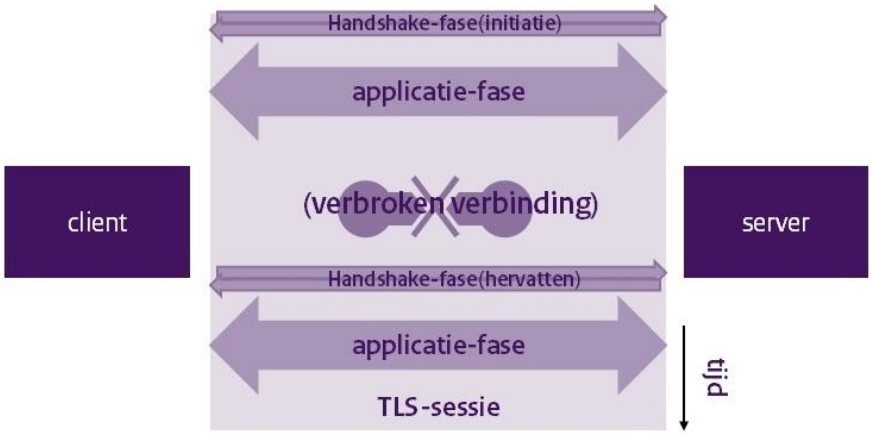
In dit hoofdstuk geven we een algemene beschrijving van het TLS protocol en de belangrijkste instellingen, opties en termen die relevant zijn vanuit veiligheidsperspectief. Gebruik dit hoofdstuk naar eigen inzicht om het advies in de rest van deze publicatie te begrijpen.

1.1 Wat is TLS?

Transport Layer Security (TLS) is een protocol waarmee een beveiligde verbinding wordt opgezet tussen twee computersystemen of applicaties: een *client* en een *server*. De beveiligde verbinding garandeert de vertrouwelijkheid en integriteit van data die uitgewisseld wordt tussen client en server. Het protocol wordt voor veel bekende toepassingen gebruikt, zoals het beschermen van webverkeer via https (het 'slotje in de browser'), veilig e-mailverkeer (IMAP en SMTP na STARTTLS) en bepaalde typen Virtual Private Networks (VPN). Een verbinding tussen een client en server die met TLS beveiligd wordt, heet een *TLS-sessie*.

Figuur 1: Illustratie van TLS-sessie

Een TLS-sessie komt alleen tot stand indien client en server overeenstemmende instellingen ondersteunen. Zodra de client een verbinding initieert met de server, start een TLS-sessie in de *handshake-fase*. Tijdens deze fase kunnen client en server elkaar authenticeren en stemmen zij af welke instellingen er gebruikt worden gedurende de TLS-sessie. De gekozen instellingen worden vervolgens gebruikt in de *applicatie-fase* om data veilig uit te wisselen. Client en server hoeven niet voor elke verbindingspoging een nieuwe TLS-sessie te starten. TLS biedt namelijk ook de mogelijkheid om een TLS-sessie te *hervatten*. Mocht een client een sessie willen hervatten, dan kan dit middels een (nieuwe) handshake, waarna de (applicatie-fase van de) TLS-sessie wordt hervat met de eerder afgestemde instellingen.



Figuur 2: Illustratie van de verschillende fases in een TLS-sessie.

1.1.1 Handshake-fase

In de handshake-fase zetten client en server een *nieuwe TLS-sessie op* of *hervatten* deze een eerder opgezette TLS-sessie.

a) **Nieuwe TLS-sessie:** Is er nog geen TLS-sessie opgezet tussen client en server? Of is er de noodzaak om een nieuwe sessie te starten, bijvoorbeeld omdat een eerdere sessie verlopen is? Dan wisselen client en server meerdere berichten uit om overeen te komen welke instellingen er gebruikt worden gedurende de TLS-sessie. Een aantal belangrijke elementen uit deze berichtenuitwisseling, die later in dit hoofdstuk verder worden toegelicht, zijn:

   a. **Authenticatie:** Bij het opzetten van een nieuwe verbinding kunnen server en/of client elkaar authenticeren om er zeker van te zijn dat zij communiceren met de beoogde partij.

   b. **Afstemming:** Client en server stemmen af welke instellingen zij zullen gebruiken tijdens de applicatiefase. Deze afstemming gebeurt ruwweg als volgt:

      1) De client stuurt een lijst met instellingen die deze ondersteunt en geeft aan welke instellingen zijn voorkeur genieten;
      2) Indien client en server dezelfde instellingen ondersteunen, dan kiest de server welke set instellingen er gebruikt wordt tijdens TLS-sessie en informeert de client over de gemaakte keuze.

   c. **Sleuteluitwisseling:** Client en server bepalen samen een geheime sleutel die gebruikt wordt tijdens de applicatie-fase.

   Indien bovenstaande succesvol is doorlopen, dan komt een TLS-sessie tot stand en kan de applicatiefase beginnen. De gebruikte instellingen worden opgeslagen en gekoppeld aan een *sessie-ID* of *sessieticket*, waarmee de sessie op een later moment hervat kan worden.

b) **Hervatten TLS-sessie:** Bij veel toepassingen is het gebruikelijk dat de client en server meer verbindingen maken of opnieuw verbindingen nadat er een eerste TLS verbinding tot stand is gebracht. Wil de client een sessie hervatten, dan stuurt hij de eerdergenoemde *sessie-ID* of *sessieticket* mee tijdens de handshake. Als de sessie nog valide is (d.w.z. de instellingen zijn nog ondersteund en de sessie heeft nog geen time-out gehad), dan wordt deze hervat met de eerder afgestemde instellingen. Doordat er minder afstemming nodig is verloopt de handshake daarbij sneller.

1.1.2 Applicatie-fase

Tijdens de applicatie-fase worden de instellingen en geheime sleutel (die overeengekomen zijn tijdens de handshake-fase) gebruikt om data op een veilige manier uit te wisselen tussen client en server. Client en server versleutelen hun data met een versleutelingsalgoritme (in deze publicatie aangeduid als *bulkversleuteling*) en de uitgewisselde geheime sleutel.

1.2 Instellingen

Het TLS protocol wordt al meer dan 30 jaar (door-)ontwikkeld. Er zijn in die jaren verschillende versies van TLS uitgebracht. Nieuwere versies zijn doorgaans veiliger, maar onderscheiden zich ook doordat deze sneller zijn en andere instellingen ondersteunen. Hieronder beschrijven we de instellingen die relevant zijn voor de beveiligingswaarde van een TLS-sessie. In Hoofdstuk 3 wordt, voor elk van deze instellingen, aangegeven in hoeverre deze zich lenen voor het opzetten van een veilige TLS-configuratie.

1.2.1 Versies

Om de compatibiliteit en interoperabiliteit tussen client en server te waarborgen worden er, bij grote doorontwikkelingen, nieuwe versies van het TLS protocol uitgebracht. Op dit moment zijn er zeven verschillende TLS-versies:

- Secure Sockets Layer (SSL) 1.0, 2.0 en 3.0: SSL is ontwikkeld door Netscape Communications Corporation en is de voorloper van TLS.
- TLS 1.0, 1.1, 1.2 en 1.3: Ontwikkeld door de Internet Engineering Task Force (IETF). De IETF biedt TLS aan als open standaard.

De meest recente TLS-versie is TLS 1.3 [1]. De verschillen met eerdere versies van TLS worden later in dit hoofdstuk verder toegelicht, maar kortgezegd onderscheidt TLS 1.3 zich door:

- Het verplichten van serverauthenticatie;
- Het gebruik van een efficiëntere handshake die sneller wordt doorlopen;
- Alleen algoritmes te ondersteunen die naar huidige inzichten als veilig gezien worden;
- Een andere definitie van het woord *cipher suite*;
- Het laten vallen van de opties "TLS-compressie" en "renegotiation"; en
- Het toevoegen van de optie "Zero Round-Trip Time (0-RTT)"

1.2.2 Cryptografische algoritmes

De beveiliging van een TLS-sessie baseert zich voor een groot deel op het gebruik van een aantal cryptografische algoritmes. Deze algoritmes in TLS worden gebruikt voor:

- **Authenticatie:** (handshake-fase): het controleren van de digitale identiteit van de server en/of client;
- **Sleuteluitwisseling** (handshake-fase): het uitwisselen van een geheime sleutel die gebruikt wordt tijdens de applicatie-fase voor bulkversleuteling;
- **Bulkversleuteling** (applicatie-fase): het versleutelen van data tijdens een TLS-sessie; en
- **Hashing** (handshake- en applicatie-fase): het waarborgen van de integriteit (en authenticiteit) van de data die wordt uitgewisseld tussen client en server.

1.2.2.1 *Authenticatie*

Authenticatie is het controleren van de digitale identiteit van client en/of server. Afhankelijk van de gebruikte TLS versie is het optioneel voor client en/of server om zich te authenticeren:

- In TLS 1.2 en eerder is het optioneel voor zowel de client als server om zichzelf te authenticeren.
- In TLS 1.3 moet de server zich altijd authenticeren; authenticatie van de client is optioneel.

Omdat client- en serverauthenticatie op dezelfde wijze verlopen in TLS spreken we in de uitleg hieronder niet over client en server. In plaats daarvan beschrijven we hoe een *bewijzende partij* (de partij die zich moet authenticeren) zich authenticeert bij de *controlerende partij* (de partij die de authenticiteit wil controleren).

Authenticatie in TLS vindt vrijwel altijd plaats op basis van digitale (X.509) certificaten[1]. In deze sectie geven we achtergrondinformatie die relevant is voor het begrijpen van de instellingen die inherent onderdeel zijn van TLS: het optioneel gebruik van OCSP stapling en het algoritme voor TLS-authenticatie. Het beheren en gebruiken van certificaten in bredere zin is een complex en breder onderwerp dat buiten de scope van deze publicatie valt.[2]

Naast authenticatie middels digitale certificaten is het overigens ook mogelijk client en server buiten de TLS-sessie om te authenticeren middels een *pre-shared key*. Meer informatie hierover is te vinden in sectie 1.4.4.

Om gebruik te maken van authenticatie met certificaten laat de bewijzende partij vooraf een certificaat maken door een certificaatautoriteit (CA). Dit certificaat bevat verschillende gegevens die relevant zijn in het kader van authenticatie, waaronder de publieke naam van de bewijzende partij en zijn (publieke) sleutels. De CA ondertekent dit certificaat (met een digitale handtekening) om de integriteit en authenticiteit hiervan te waarborgen. Betrouwbare authenticatie vereist dat zowel de controlerende partij als de bewijzende partij de CA vertrouwen.

---

[1] *X.509 is een standaard die het formaat van publieke digitale certificaten specificeert [25].*
[2] *Raadpleeg de NCSC publicatie Zorgeloos certificaatbeheer voor meer informatie over (het beheren van) certificaten.*

Tijdens de handshake stuurt de bewijzende partij het certificaat naar de controlerende partij. Authenticatie vindt vervolgens plaats op basis van de volgende stappen:

1) **Certificaatcontrole:** De controlerende partij controleert het aangeleverde certificaat door:
   a. **De digitale handtekening** van het certificaat te controleren. Met deze controle weet de controlerende partij zeker dat de CA dit certificaat heeft uitgegeven en dat het certificaat authentiek is.
   b. Te controleren of het certificaat nog geldig is en niet voortijdig is **herroepen**. De controlerende partij kan dit doen middels:

      1. **Certificate revocation list (CRL):** Elke CA houdt een lijst bij van herroepen certificaten (de CRL). De controlerende partij downloadt (periodiek) de CRL en kijkt of het certificaat hierin is opgenomen.

      2. **Online Certificate Status Protocol (OCSP):** De controlerende partij vraagt de status van het certificaat op bij een *OCSP-responder*, een dienst die vaak geleverd wordt door een CA. De OCSP-responder geeft de herroepingsstatus van het opgevraagde certificaat aan de controlerende partij.

      3. **OCSP-stapling:** Met OCSP-stapling vraagt de *bewijzende* partij de status op van zijn eigen certificaat aan een OCSP-responder. Deze beantwoordt het verzoek met een digitaal-ondertekende herroepingsstatus. De *bewijzende* partij kan deze *stapled* OCSP-response meesturen tijdens de handshake-fase (wanneer deze zijn certificaat stuurt naar de controlerende partij). De controlerende partij kan hiermee de certificaatstatus verifiëren.

      In TLS heeft de bewijzende partij de optie om OCSP-stapling toe te passen. Het staat de controlerende partij vrij om te bepalen hoe deze de herroepingsstatus van het certificaat wil controleren.

2) **TLS-authenticatie:** De controlerende en bewijzende partij maken tijdens de handshake gebruik van een digitale handtekening om de authenticatie te voltooien. Hierbij wordt de publieke sleutel (die onderdeel is van het digitaal certificaat) gebruikt om garanties te krijgen dat de controlerende partij met de beoogde bewijzende partij communiceert.

1.2.2.2 *Sleuteluitwisseling*

Het algoritme voor sleuteluitwisseling specificeert de wijze waarop client en server de sleutel overeenkomen die gebruikt zal worden voor bulkversleuteling.

In het kader van sleuteluitwisseling is de term *forward secrecy* relevant. Bij sommige algoritmes wordt de uitgewisselde sleutel gevormd met behulp van de publieke sleutel (*public key*) van de server. Deze sleutel is onderdeel van het certificaat en wordt doorgaans niet vaak verwisseld. Mocht een aanvaller de geheime privésleutel (*private key*) van de server bemachtigen die gekoppeld is aan deze publieke sleutel, dan kan deze *alle* huidige en vroegere communicatie tussen server en zijn client(s) ontsleutelen en manipuleren. Algoritmes die *forward secrecy* bieden hebben dit risico niet door gebruik te maken van een tijdelijke (*ephemeral*) sleutel die alleen voor die sessie geldt.

1.2.2.3 *Bulkversleuteling*

Bulkversleuteling vormt het hart van de applicatiefase. Met bulkversleuteling wordt de data versleuteld door een symmetrisch versleutelingsalgoritme (*cipher*). Dit algoritme versleutelt de data met de (eerder uitgewisselde) sleutel. Nagenoeg alle bulkversleutelingsalgoritmes in deze publicatie zijn *block ciphers*. Deze worden in TLS veelal beschreven in de vorm:

`Cipher-sleutellengte-operatiemodus`

Hierin is:

- `Cipher`: De naam van het versleutelingsalgoritme, bijvoorbeeld AES of CAMELLIA
- `Sleutellengte`: De lengte van de sleutel gebruikt door het cipher, bijvoorbeeld 128 of 256 (bit).
- `Operatiemodus`: De modus waarop het block cipher gebruikt wordt, bijvoorbeeld CBC (*cipher-block chaining*), CCM (*counter with cipher block chaining message authentication code*) en GCM (*Galois/Counter Mode*).

Block ciphers in CCM en GCM modus, als ook het algoritme ChaCha20-Poly1305, zijn *AEAD*-algoritmes (*authenticated-encryption with associated data*). Deze algoritmes waarborgen zowel de vertrouwelijkheid, integriteit en authenticiteit van de versleutelde data. Block ciphers in CBC-modus waarborgen enkel de vertrouwelijkheid van data. Zij maken gebruik van een hashfunctie (specifiek een *HMAC, hash-based message authentication code*) om ook integriteit en authenticiteit te waarborgen.

1.2.2.4 *Hashing*

Een (cryptografische) hashfunctie is een wiskundige functie die een unieke digitale vingerafdruk berekent op basis van een gegeven input. Hashfuncties zijn eenrichtingsfuncties, waarbij de inputgegevens niet te herleiden zijn vanuit de berekende digitale vingerafdruk. Hashfuncties worden op verschillende manieren gebruikt in TLS:
Als onderdeel van (de ondertekening en controle van) digitale certificaten;

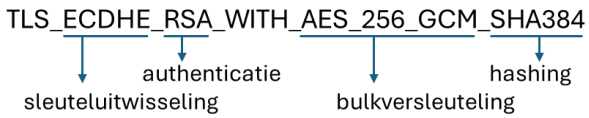
1) Als onderdeel van TLS-authenticatie op basis van digitale certificaten;
2) Het genereren van pseudo-willekeurige toevalsgetallen (PRNG), bijvoorbeeld voor de functie waarmee een geheime sleutel wordt afgeleid (KDF);
3) Als integriteitsbescherming voor berichten die uitgewisseld worden tijdens de handshake-fase; en
4) Als integriteitsbescherming tijdens de applicatie-fase indien er gebruik gemaakt wordt van HMAC.

1.2.3 Cipher suite

De combinatie van de algoritmes die gebruikt (kunnen) worden tijdens een TLS-sessie wordt een *cipher suite* genoemd. Cipher suites worden gebruikt bij zowel de configuratie van client en server als tijdens de handshake fase (voor afstemming). Elke cipher suite beschrijft één combinatie van algoritmes volgens een gestandaardiseerd format. In TLS 1.3 wordt een andere definitie en format van een cipher suite gehanteerd ten opzichte van oudere versies. Omdat het advies in deze publicatie betrekking heeft op zowel TLS 1.3 als oudere versies beschrijven we hieronder beide definities en formats.

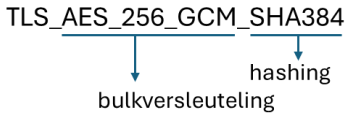
Cipher suites vóór TLS 1.3

Tot en met TLS 1.2 bestond een cipher suite, zoals eerder beschreven, uit de algoritmes voor sleuteluitwisseling, authenticatie, bulkversleuteling en hashing. Het figuur hieronder illustreert de notatie van een cipher suite zoals gebruikt vóór TLS 1.3.



Figuur 3: Voorbeeld van een cipher suite in TLS versies vóór TLS 1.3

Met de aanpassing in de handshake in TLS 1.3 is ook de definitie van cipher suite aangepast. In TLS 1.3 beschrijft een cipher suite enkel de algoritmes voor bulkversleuteling en hashing. Het figuur hieronder illustreert de notatie van een cipher suite zoals gebruikt in TLS 1.3.



Figuur 4: Voorbeeld van een cipher suite in TLS 1.3

1.3 Overige instellingen

Naast de keuze voor cryptografische algoritmes zijn er nog een aantal andere TLS-opties die relevant zijn vanuit veiligheidsperspectief. Deze worden hieronder kort toegelicht.

1.3.1 Compressie

Compressie is een techniek waardoor informatie met minder data gerepresenteerd wordt, wat leidt tot minder (communicatie) overhead. Een applicatie kan zelf compressie implementeren, of gebruik maken van TLS-compressie. Tijdens de handshake mogen clients aangeven of zij voorkeur hebben voor TLS-compressie, maar zij moeten altijd TLS zonder compressie ondersteunen. Compressie kan een negatief effect hebben op de beveiliging van een TLS-sessie [2] [3]. In praktijk wordt TLS-compressie weinig gebruikt. Ondersteuning voor TLS-compressie is zelfs geheel uitgeschakeld in TLS 1.3.

1.3.2 Renegotiation

Renegotiation houdt in dat client en server *tijdens de applicatie-fase* een nieuwe *handshake-fase* opstarten. De client zou bijvoorbeeld de voorkeur kunnen geven aan het gebruik van een ander bulkversleutelingsalgoritme, of om een sessie (die bijna verloopt) een doorstart te geven.

Oude versies van renegotiation bevatte kwetsbaarheden, die verholpen worden door het toepassen van een *renegotiation indication extension* [4]. Sinds TLS 1.3 wordt renegotiation niet meer ondersteund.

1.3.3 Sessies hervatten (sessie-ID, sessietickets en 0-RTT)

Zoals eerder beschreven kunnen client en server (tijdens de handshake-fase) een reeds opgezette TLS-sessie hervatten. In TLS versies tot TLS 1.2 werd het hervatten van een sessie ondersteund door gebruik te maken van *sessie-IDs* of *sessietickets* [5]. In TLS 1.3 is enkel het gebruik van sessietickets mogelijk.

- **Sessie-identifiers (Sessie-IDs):** Tijdens het opzetten van een TLS-sessie slaan client en server de gekozen instellingen op. Zij refereren naar deze instellingen met een sessie-ID.

  Wil een client een sessie hervatten, dan stuurt hij het sessie-ID mee tijdens de handshake. Server en client gebruiken dit Sessie-ID om de bijbehorende instellingen te laden uit hun opslag en hervatten hiermee de sessie.

- **Sessieticket:** Het gebruik van sessie-IDs heeft als nadeel dat een server de gekozen instellingen van *alle* geldige TLS-sessies moet opslaan. Om dit te voorkomen kan er gebruik gemaakt worden van sessietickets. De server maakt een sessieticket aan als de instellingen voor de sessie zijn gekozen. Dit ticket is effectief de lijst van gekozen instellingen, versleuteld door de server. De server stuurt dit ticket naar de client aan het einde van de handshake. Wil een client de sessie hervatten, dan stuurt hij het sessieticket mee tijdens een (opvolgende) handshake. De server ontsleutelt het sessieticket en hervat de sessie met de daarin beschreven instellingen.

***Zero Round-Trip Time* (0-RTT)** is een optie die geïntroduceerd is in TLS 1.3. Het doel van 0-RTT is om client en server (nog) sneller hun sessie te laten hervatten. Bij het gebruik van 0-RTT wordt, al tijdens de handshake-fase, een stuk applicatiedata meegestuurd door de client (dus nog *voordat* de applicatie-fase is begonnen). Aan het gebruik van 0-RTT kleven beveiligingsrisico's. Het biedt geen bescherming biedt tegen replay-aanvallen op de TLS-laag en veilige toepassing is afhankelijk van de applicatie die gebruik maakt van TLS.

1.4 Overige relevante termen

In de vorige secties zijn de belangrijkste instellingen beschreven die relevant zijn in het kader van een veilige TLS configuratie. In deze sectie beschrijven we nog een laatste aantal termen die relevant zijn om het advies uit deze publicatie te volgen.

1.4.1 TLS-bibliotheek

TLS wordt in veel verschillende applicaties gebruikt. Het programmeren van alle functionaliteiten van TLS is veel werk en vergt specialistische kennis. Daarom bevat de meeste software geen eigen code voor TLS, maar wordt er gebruik gemaakt van een TLS-(software)bibliotheek die een implementatie van het TLS protocol bevat.

Er zijn enkele tientallen TLS-bibliotheken beschikbaar. Van sommige bibliotheken is de broncode beschikbaar, anderen worden als gesloten product beschikbaar gesteld. Ze kunnen in besturingssystemen geïntegreerd worden, maar kunnen ook afzonderlijk worden geleverd. Niet elke bibliotheek implementeert alle beschikbare TLS-instellingen of wordt even goed onderhouden door hun ontwikkelaar. Allicht de meest gebruikte TLS-bibliotheek is OpenSSL.

Door gebruik te maken van een TLS-bibliotheek voorkom je dat je zelf fouten maakt in een TLS-implementatie. Let wel op: alle software bevat bugs, dus ook TLS-bibliotheken. Bugs kunnen op hun beurt weer tot kwetsbaarheden leiden[3]. Het gebruiken van een TLS-bibliotheek is geen garantie voor een TLS implementatie zonder kwetsbaarheden.

1.4.2 Unieke en toevalsgetallen

In veel cryptografische toepassingen, waaronder TLS, spelen unieke en (pseudo-) cryptografisch-veilige toevalsgetallen (cryptographically secure random numbers) een belangrijke rol. Alle besturingssystemen en TLS-bibliotheken bevatten

[3] *Een voorbeeld is de in 2014 verholpen kwetsbaarheid "Heartbleed Bug" in OpenSSL software; zie ook https://heartbleed.com/*

methoden om unieke en cryptografisch veilige toevalsgetallen te genereren. Daarnaast zijn er hardwaremodules verkrijgbaar voor het genereren van dergelijke toevalsgetallen.

1.4.3 TLS proxy

Tot dusver hebben we TLS beschreven als een protocol dat zorgt voor een beveiligde verbinding tussen client en server. In praktijk kan een TLS-sessie ook worden opgezet tussen een client en een *proxy*: een systeem dat zich bevindt tussen client en server, zoals geschetst in Figuur 5. Optioneel wordt er ook een beveiligde verbinding opgezet tussen proxy en server.

Een proxy kan voor verschillende doeleindes gebruikt worden. Zo kunnen organisaties al het inkomende netwerkverkeer op één punt (de proxy) inspecteren op malware (TLS inspectie). Ook kan een proxy onderdeel zijn van anti-DDoS diensten en inbraakdetectie- en inbraakpreventiesystemen (IDS/IPS). Daarnaast kunnen TLS-proxies gebruikt worden om oudere systemen van de buitenwereld af te schermen en (indirect) te voorzien van een TLS verbinding.

Er kleven enkele beveiligingsrisico's aan het gebruik van een TLS proxy. De grote hoeveelheden (onversleuteld) netwerkverkeer die een TLS proxy verwerkt maken het een aantrekkelijk doelwit voor digitale aanvallen. Een aanvaller die toegang krijgt tot de proxy kan gevoelige informatie inzien, zoals financiële gegevens, persoonsgegevens en wachtwoorden. Ook kan een TLS proxy gebruikt worden voor attacker-in-the-middle aanvallen. Een proxy voorkomt daarnaast het gebruik van *certificate pinning*, een beveiligingsmechanisme waarmee een client enkel een TLS verbinding opzet met een specifieke server, en maakt het voor de gebruiker onmogelijk de certificaathouder en gerelateerde organisatie te controleren (in geval van OV-, EV- of QWAC-certificaten)[4].

Figuur 5: Een TLS proxy.

1.4.4 Pre-shared key (PSK)

In de beschrijving tot dusver beginnen client en server hun communicatie vanaf een eerste handshake. Tijdens deze handshake gebruiken ze verschillende algoritmes en geheime parameters om elkaar te authenticeren en af te spreken op welke manier zij met elkaar zullen communiceren.

TLS biedt ook de mogelijkheid om gebruik te maken van een *(out-of-band established) pre-shared key* (PSK). In dit geval worden (specifieke) clients en server op een alternatieve manier met elkaar geauthenticeerd en/of voorzien van een sleutel voor bulkversleuteling. Zo zou de geheime sleutel voor bulkversleuteling bijvoorbeeld op voorhand uitgewisseld kunnen worden tussen client en server door gebruik te maken van een USB-stick (met daarop de sleutel) en een koerier (die de USB-stick van server naar client brengt).

Met een *out-of-band* PSK vindt een deel van de handshake effectief plaats buiten de TLS-sessie om. Het gebruik van een *out-of-band* PSK kent zijn eigen uitdagingen en risico's en is onderhevig aan verschillende randvoorwaarden voor veilig gebruik. Denk aan de eisen die gesteld moeten worden aan de wijze waarop sleutelmateriaal wordt vervoerd en uitgewisseld tussen server en client. Een *out-of-band* PSK kan alleen voor specifieke toepassingen gebruikt worden en is niet schaalbaar voor algemeen gebruik.

Sessietickets, zoals besproken in paragraaf 1.3.3, zijn in de TLS 1.3 handshake gedefinieerd als onderdeel van een PSK. In dit geval betreft het een *in-band* PSK die tijdens de TLS-sessie zijn verstuurd. Dit leent zich wel voor een bredere set toepassingen.

1.4.5 Cryptografische sterkte

De *cryptografische sterkte* is een maat om de beveiligingswaarde van een cryptografisch algoritme in uit te drukken. De cryptografische sterkte wordt uitgedrukt in bit. Ruwweg kan gesteld worden dat een cryptografische sterkte van 128-bit inhoudt dat een aanvaller ongeveer 2^{128} (2 tot de macht 128) pogingen nodig heeft om vercijferde informatie succesvol te ontcijferen. De cryptografische sterkte is een inschatting gebaseerd op het (type) algoritme, de gebruikte sleutellengte (indien het een versleutelingsalgoritme betreft) en het effect van (bekende) kwetsbaarheden in de algoritmiek. De sterkte van een

[4] *Raadpleeg de NCSC publicatie Zorgeloos certificaatbeheer voor meer informatie over EV-certificaten.*

versleutelingsalgoritme is daarbij *niet* altijd hetzelfde als de sleutellengte die een algoritme gebruikt. Verschillende algoritmes kunnen met andere sleutellengtes dezelfde cryptografische sterkte behalen.

Een cryptografisch-relevante quantumcomputer (CRQC) is een quantumcomputer die krachtig genoeg is om de cryptografische sterkte van sommige algoritmes ernstig te verlagen. In deze publicatie drukken we, uit praktische overwegingen, de cryptografische sterkte enkel uit zoals bepaald voor conventionele computertechnologie. Wel nemen we het effect van een CRQC op deze cryptografische sterkte mee in het advies. Meer informatie over een CRQC en de quantumdreiging volgt later in deze publicatie.

2 Stappenplan: Hoe kom ik tot een passende TLS configuratie?

Dit hoofdstuk biedt een algemeen stappenplan waarmee je een passende, veilige TLS configuratie realiseert.

De richtlijnen in deze publicatie zijn primair bedoeld om jou te helpen te komen tot een zo veilig mogelijke TLS-configuratie. In de praktijk speelt bij de keuze van een configuratie niet alleen de veiligheid een rol. Zo moet jouw toepassing mogelijk ook toegankelijk zijn voor verouderde clients die geen ondersteuning bieden voor de laatste (veiligere) versies van TLS. De keuze voor een passende TLS configuratie is dus (ook) een bedrijfsafweging. Doorloop de stappen in dit stappenplan daarom samen met diegene die verantwoordelijk is voor de toepassing waarvoor de TLS configuratie opgesteld wordt (meestal de productverantwoordelijke/producteigenaar).

1. **Inventarisatie.** Begin jouw stappenplan door zicht te krijgen op de mogelijkheden van jouw client(s) en servers. Hiermee zorg je ervoor dat je weet welke instellingen je kan en moet ondersteunen voor jouw toepassing.
   Stel jezelf de volgende vragen:

   a. Zal jouw server gebruik maken van bestaande applicaties of systemen? Inventariseer dan welke TLS-instellingen hierdoor ondersteund worden.
      Wordt er voor de server een nieuwe applicatie of systeem aangeschaft? Zorg er dan voor dat deze passende (en veilige) TLS-instellingen ondersteunt.

   b. Heb je een goed beeld bij de clients die jouw TLS configuratie moet ondersteunen?
      i. Schets met de productverantwoordelijke een algemeen profiel van de doelgroep. Moet de toepassing bijvoorbeeld toegankelijk zijn voor een brede groep eindgebruikers? Of is de toepassing gericht op een selectievere groep clients, zoals systemen op jouw interne (bedrijfs-)netwerk?

      ii. Maak een inschatting van de (type) clients waar de doelgroep gebruik van zal maken. Inventariseer onder andere:

         1. Vanaf welke apparaten jouw toepassing toegankelijk moet zijn (bijvoorbeeld een systeem op het eigen bedrijfsnetwerk of een mobiele telefoon);
         2. Welke applicaties jouw doelgroep zal gebruiken (bijvoorbeeld een algemene webbrowser of een applicatie in eigen beheer); en
         3. In hoeverre jouw configuratie specifieke ondersteuning moet bieden voor bepaalde operating systems.

         Neem hierbij ook mee hoe oud een client mag zijn. Houd er rekening mee dat een brede doelgroep (zoals burgers) typisch gebruik zal maken van een diverse set clients, die in sommige gevallen ook sterk verouderd kunnen zijn.

      iii. Bepaal welke instellingen ondersteund worden door de eerder geïdentificeerde types clients. Voor de meeste gangbare apparaten, operating systems, browsers en TLS-bibliotheken is dit eenvoudig op te zoeken.

2. **Selectie:** Kies de instellingen voor jouw TLS configuratie. Maak hiervoor gebruik van:

   a. De resultaten van stap 1, die een beeld geven van bij de instellingen die je kan en moet ondersteunen voor jouw toepassing;
   b. Het advies rondom veilige TLS-instellingen en randvoorwaarden uit Hoofdstuk 3 en 4.

   Beperk jouw configuratie tot veilige instellingen, corresponderende met de beveiligingsniveaus **Voldoende** of **Goed** zoals aangegeven in Hoofdstuk 4. Moet jouw configuratie, gegeven de resultaten van stappen 1-3, instellingen ondersteunen die minder veilig zijn? Ondersteunt jouw server bijvoorbeeld enkel verouderde TLS versies, of moet jouw toepassing toegankelijk zijn voor clients die enkel oudere (onveilige) instellingen ondersteunen?

   Bespreek dan met de productverantwoordelijke:
   - In welke mate een aanpassing/update van de server kan leiden tot een veiligere configuratie;
   - In welke mate jouw organisatie in staat is om de clients aan te passen, bijvoorbeeld door een software-update;
   - Of het echt nodig is om clients te ondersteunen die enkel onveiligere instellingen ondersteunen, of dat de beveiligingsrisico's zwaarder wegen dan toegankelijkheid; en
   - Of je met andere (aanvullende) beveiligingsmaatregelen de risico's kan beperken.

Win, waar mogelijk, specialistische kennis en ervaring in om hier een goed antwoord op te formuleren. Blijft de noodzaak bestaan om minder veilige instellingen te ondersteunen? Zie dit nooit als een reden om helemaal geen TLS te gebruiken. Zelfs een minder veilige TLS-configuratie kan voor sommige aanvallers ondoordringbaar zijn. Documenteer de afwijkingen en de resultaten van bovenstaande afwegingen en laat deze bekrachtigen door de productverantwoordelijke.

3. **Configuratie:** Configureer de TLS instellingen die je wilt ondersteunen. De TLS-configuratie maakt deel uit van de software die de TLS-verbinding gebruikt. Wil je bijvoorbeeld https aanbieden, configureer TLS dan als onderdeel van de webserversoftware.

   Tijdens de handshake-fase stuurt een client een lijst van ondersteunde instellingen naar de server, inclusief de instellingen die zijn voorkeur genieten. Zo kan een client de voorkeur geven om bepaalde bulkversleutelingsalgoritmes te gebruiken, bijvoorbeeld omdat de client draait op hardware die zo'n algoritme goed ondersteunt. De server kiest uiteindelijk de instellingen die gebruikt worden voor de TLS-sessie. Configureer de server op zo'n manier dat:
   - Deze de voorkeur geeft aan instellingen die een (zo veilig mogelijke) verbinding realiseren voor jouw toepassing (zie paragraaf 3.2);
   - Kies, uit deze selectie, de instellingen die de voorkeur genieten vanuit de client.

   Configureer, naast de TLS instellingen, het certificaat dat jouw server gebruikt voor authenticatie. Is jouw certificaat niet direct door de root CA ondertekend? Configureer dan ook de tussenliggende (*intermediate*) certificaten die gebruikt worden om het pad tussen de root CA en jouw certificaat te authenticeren[2]. Overweeg ook het configureren van TLSA-records ter ondersteuning van het DANE protocol.

4. **Controle:** Controleer in hoeverre de TLS-configuratie correct is geconfigureerd. Het is mogelijk dat er per abuis instellingen zijn geconfigureerd die voor jou niet acceptabel zijn, of dat bepaalde instellingen niet zijn geconfigureerd. Maak hiervoor gebruik van tools en websites waarmee je dergelijke controle kan uitvoeren, inclusief hoe deze relateren aan de beveiligingsniveaus uit deze publicatie.[5]

5. **(Openbare) testen:** Test of de server correct werkt met de verschillende beoogde clients. Is jouw server niet goed toegankelijk voor sommige clients? Ga dan terug naar de vorige stappen en overweeg hoe deze opgelost kunnen worden.
   Stel jezelf de volgende vragen:
   - Zijn alle geselecteerde TLS-instellingen correct geconfigureerd op de server en clients?
   - Heeft de configuratie (acceptabele) impact op de performance van de applicatie?
   - Kunnen eventuele problemen opgelost worden door aanvullende (veilige) instellingen te ondersteunen?
   - Is het mogelijk om de client aanvullende (veilige) instellingen te laten ondersteunen, bijvoorbeeld door een software-update?

De veiligheid van een TLS verbinding is, naast een correcte instelling, onderhevig aan een correcte implementatie. Test de implementatie regelmatig en maak het controleren en beheren van TLS-configuraties onderdeel van jouw reguliere beheerprocessen, zoals kwetsbaarhedenbeheer.

[5] *Voorbeelden van dergelijke tools zijn testssl.sh (https://testssl.sh/) en sslyze (https://github.com/nabla-c0d3/sslyze). Op https://www.internet.nl/ kan je testen in hoeverre jouw webservers en e-mailservers voldoen aan de richtlijnen zoals genoemd in deze publicatie. Via Qualys SSL labs kan je een soortgelijke controle uitvoeren (https://www.ssllabs.com/ssltest/).*

3 Veilige TLS-instellingen

Een veilige TLS-sessie vereist dat jouw server geconfigureerd wordt met veilige instellingen. In dit hoofdstuk beschrijven we welke TLS instellingen als veilig of minder veilig worden gezien. We maken hiervoor gebruik van vier beveiligingsniveaus: **Goed, Voldoende, Uit te faseren** en **Onvoldoende.**

Hieronder beschrijven we eerst de verschillende beveiligingsniveaus en hoe je deze kan gebruiken om tot een veilige TLS-configuratie te komen. Daarna wordt het beveiligingsniveau beschreven per instelling.

3.1 Beveiligingsniveaus gehanteerd in dit document

In deze publicatie hanteren we vier beveiligingsniveaus om de beveiligingswaarde van TLS-instellingen uit te drukken:

- **Goed:** Het label **Goed** wordt toegekend aan instellingen die gezien worden als het meest veilig en toekomstbestendig.
- **Voldoende:** Deze instelling biedt voldoende beveiligingswaarde naar huidige inzichten.
- **Uit te faseren:** Instellingen worden als **Uit te faseren** beschouwd indien de verwachting is dat deze (op termijn) **Onvoldoende** zullen worden, bijvoorbeeld met oog op de doorontwikkeling van aanvalstechnieken. Dergelijke instellingen bieden slechts een geringe veiligheidsmarge. Faseer het gebruik van deze instellingen uit.
- **Onvoldoende:** Dit zijn instellingen die niet veilig zijn, bijvoorbeeld omdat deze bekende en makkelijk te misbruiken kwetsbaarheden bevatten. Gebruik deze instellingen niet.

De beveiligingsniveaus zijn tot stand gekomen op basis van expert opinion en het raadplegen van relevante analyses, standaarden en richtlijnen van gerenommeerde instituten en organisaties. Hierbij is ook een inschatting van de cryptografische sterkte meegenomen indien de betreffende instelling zich hierin laat uitdrukken. We hanteren hiervoor dat een cryptografische sterkte van 128-bit nodig is voor het beveiligingsniveau **Voldoende** of **Goed**[6]. Een **Uit te faseren** instelling heeft een minimale cryptografische sterkte van 112-bit. Waar van toepassing worden referenties aangeleverd die gebruikt zijn in de beveiligingswaardering van de instellingen.

De beveiligingsniveaus in deze publicatie richten zich op algemeen, breed gebruik van TLS en zijn *toepassing-agnostisch.* Sommige TLS instellingen zijn bijvoorbeeld niet veilig te gebruiken voor publieke netwerken, maar kunnen wel gebruikt worden in een specifieke, eigen netwerkomgeving. Dergelijke situaties worden niet meegenomen in de bepaling van de beveiligingsniveaus.

De woorden **'Onvoldoende', 'Uit te faseren', 'Voldoende'** en **'Goed'** hebben ook een betekenis in het dagelijkse taalgebruik. Om eventuele verwarring te voorkomen, worden deze woorden in de richtlijnen in een **vet lettertype** weergegeven wanneer dit verwijst naar een beveiligingsniveaus.

De dreiging van cryptografisch-relevante quantumcomputers

Een cryptografisch-relevante quantumcomputer (CRQC) is een quantumcomputer die krachtig genoeg is om de cryptografische sterkte van verschillende algoritmes ernstig te verlagen. Hiermee is een CRQC een bedreiging voor cryptografie. Met name veelgebruikte vormen van asymmetrische cryptografie – die worden gebruikt voor authenticatie en sleuteluitwisseling – zijn kwetsbaar en is effectief te 'kraken' met een CRQC. Ook verlaagt een CRQC de cryptografische sterkte van algoritmes voor bulkversleuteling en hashing.

We verwachten dat de eerste CRQC tussen 2030 en 2040 het licht zal zien. Dat betekent niet dat jouw organisatie op dit moment geen risico loopt. Zo kunnen aanvallers met een *store-now-decrypt-later* aanval jouw (met TLS) versleutelde data nu verzamelen en deze op een later moment ontsleutelen (wanneer zij beschikken over een CRQC). Ook leert de geschiedenis dat het migreren naar veiligere algoritmes complexe uitdagingen kent die veel tijd, planning en voorbereiding vragen.

In deze publicatie geven we advies over hoe je rekening kan houden met de dreiging die voortkomt uit een CRQC. Instellingen waarvan bekend is dat deze kwetsbaar zijn voor aanvallen met een CRQC zijn niet toekomstbestendig en krijgen in deze publicatie hooguit het beveiligingsniveau **Voldoende** toegewezen. Uit praktische overwegingen drukken we de cryptografische sterkte enkel uit zoals bepaald voor conventionele computertechnologie. Wel nemen we het effect van een CRQC op deze cryptografische sterkte mee in het advies. In paragraaf 4.5 beschrijven we daarnaast een aantal acties die je nu al kan uitvoeren om jouw TLS configuratie voor te bereiden op de komst van een CRQC.

[6] *Dit is in lijn met o.a. het Tsjechische securityautoriteit NÚKIB (128-bit) [13] en het Duitse BSI (120-bit) [11]. Het Amerikaanse NIST hanteert op dit moment nog 112-bit, maar beoogt dit ook te brengen naar 128-bit in hun hernieuwde richtlijnen [24].*

3.2 Richtlijn voor een veilige TLS configuratie

Gebruik de volgende richtlijnen om tot een veilige TLS configuratie te komen.

- Geef de voorkeur voor het gebruik van instellingen met de status **Goed**, aangevuld met **Voldoende** instellingen ter ondersteuning van oudere software indien noodzakelijk.
- Gebruik uitsluitend **Uit te faseren** instellingen waar dat nodig is met het oog op client-server interoperabiliteit. Formuleer hierbij duidelijke en meetbare criteria voor de uitfasering hiervan, zoals “deze instellingen worden uitgefaseerd op <datum>, <Na de release van <webbrowser> versie X in het 'stable release channel>, of <Wanneer het relatieve aantal gebruikers minder bedraagt dan Y%>. Laat deze bekrachtigen door de verantwoordelijke van de toepassing waarvoor TLS wordt geconfigureerd.
- Maak geen gebruik van **Onvoldoende** instellingen.
- Maak gebruik van de beschreven veilige parameters om het beschreven beveiligingsniveau te bereiken.
- Zorg er daarnaast voor dat je ook de randvoorwaarden voor een veilige TLS configuratie in ogenschouw neemt, zoals beschreven in Hoofdstuk 4.

Ben je niet in staat om deze richtlijnen te volgen? Bijvoorbeeld omdat jouw toepassing ook toegankelijk moet zijn voor verouderde clients die enkel **Onvoldoende** instellingen ondersteunen? Zie dan het stappenplan in Hoofdstuk 2 voor handelingsperspectief.

3.3 Veilige instellingen

Gebruik het advies uit deze sectie om veilige TLS-instellingen te configureren.

3.3.1 Versies

Gebruik TLS 1.3. In deze versie van TLS zijn verschillende kwetsbaarheden verholpen, wat leidt tot een hoger beveiligingsniveau. De verwachting is daarnaast dat quantumveilige cryptografie (cryptografie die bestand is tegen een aanval met een CRQC) als eerst (en als enige) wordt gestandaardiseerd voor TLS 1.3. Door nu gebruik te maken van TLS 1.3 zorg je ervoor dat een noodzakelijke, toekomstige migratie naar quantumveilige cryptografie soepeler zal verlopen.
Niet alle clients bieden al ondersteuning voor TLS 1.3. Configureer het gebruik van TLS 1.2 ter ondersteuning van oudere clients, maar geef de voorkeur aan TLS 1.3 waar mogelijk [6]. Andere versies zijn niet veilig in het gebruik.

Versie	Beveiligingsniveau
TLS 1.3	Goed
TLS 1.2	Voldoende
TLS 1.1	Onvoldoende
TLS 1.0	
SSL 3.0	
SSL 2.0	
SSL 1.0	

Tabel 1: TLS versies hun beveiligingsstatus [6].

3.3.2 Authenticatie

Maak gebruik van authenticatie op basis van X509 certificaten[7].
Gebruik voor TLS-authenticatie een van de algoritmes uit Tabel 2.

Algoritme	Beveiligingsniveau
ECDSA	Voldoende
RSA	
EdDSA	
DSS	Onvoldoende
EXPORT-*	
PSK[8]	
Anon	
NULL	
KRB5[8]	

Tabel 2: Algoritmes voor TLS-authenticatie en hun beveiligingsniveau [6] [7].

Overweeg, bij het kiezen van een algoritme, het volgende:

- Al de algoritmes in Tabel 2 zijn kwetsbaar voor de quantumdreiging. Zij bieden vooralsnog **Voldoende** bescherming. Quantumveilige alternatieven zijn (nog) geen onderdeel van de TLS standaarden.
- Om hetzelfde beveiligingsniveau te bereiken vereisen ECDSA en EdDSA een kortere (publieke) sleutel dan RSA. Hierdoor hoeft er minder data te worden uitgewisseld tijdens de handshake. Ook is het ondertekenen met ECDSA vele malen sneller. De verificatie is bij RSA sneller. RSA is daarnaast (licht) beter bestand tegen de dreiging van een CRQC door het gebruik van een langere geheime parameter (sleutellengte).
- Het algoritme EdDSA biedt een goede beveiligingswaarde, maar is minder breed ondersteund dan ECDSA en RSA. Overweeg of het gebruik maken van EdDSA echt noodzakelijk is voor jouw toepassing, of dat je gebruik kunt maken van ECDSA of RSA.

[7] *Het advies in deze publicatie beperkt zich tot de instellingen die direct onderdeel zijn van een TLS configuratie. Het inrichten van goed certificaatbeheer is een belangrijke randvoorwaarde voor TLS-authenticatie, maar valt buiten scope van deze publicatie. Zie paragraaf 4.4 voor meer informatie.*
[8] *Het gebruik van een PSK en KRB5 leent zich alleen voor niche toepassing en vereisen specialistische aandacht om beveiligingsrisico's te mitigeren. Deze algoritmes zijn daarom zijn* ***Onvoldoende*** *voor algemeen gebruik. Lees paragraaf 1.4.4 voor meer informatie.*

3.3.2.1 *Veilige parameters*

De beveiliging van ECDSA, RSA en EdDSA zijn mede afhankelijk van de gekozen parameters. Kies de parameters zoals hieronder aangegeven om het eerder aangegeven beveiligingsniveau te behalen.

Gebruik je ECDSA voor TLS-authenticatie? Gebruik een gestandaardiseerde elliptic curve zoals beschreven in Tabel 3. Overweeg hierbij het volgende:

- Grotere elliptic curves kennen een grotere cryptografische sterkte, maar kennen ook een hogere (performance) overhead van client en server. Gebruik sterkere curves enkel indien noodzakelijk voor jouw toepassing.
- In praktijk worden de brainpool*-curves significant minder gebruikt dan de overige curves. Onderzoek of het gebruik maken van deze curves echt noodzakelijk is voor jouw toepassing, of dat je gebruik kunt maken van andere curves.

Elliptic curve	Beveiligingsniveau	Cryptografische sterkte (bit)
secp521r1	Voldoende	260
secp384r1		192
secp256r1		128
Curve448*		224
Curve25519*		128
brainpoolP512r1		256
brainpoolP384r1		224
brainpoolP256r1		128
secp224r1	Uit te faseren	112
Andere curves	Onvoldoende	—

Tabel 3: Elliptic curves voor ECDSA voor TLS-authenticatie (en ECDHE voor sleuteluitwisseling), hun beveiligingsniveau en cryptografische sterkte [8] [9] [10] [11].
**Enkel te gebruiken voor ECDHE.*

Gebruik je RSA voor TLS-authenticatie? Gebruik een private/public sleutelpaar met een modulus volgens Tabel 4. Ondersteunt jouw TLS configuratie daarnaast TLS 1.2? Controleer dan welk *paddingmechanisme(s)* jouw TLS-bibliotheek ondersteunt (RSA-PSS of RSA-PKCS#1 v1.5). Het modernere RSA-PSS is veiliger (en is het enige ondersteunde paddingmechanisme in TLS 1.3). Geef de voorkeur aan het gebruik van RSA-PSS en ondersteun RSA-PKCS#1 v1.5 alleen indien dit nodig is voor jouw toepassing.

Parameter	Waarde	Beveiligingsniveau	Cryptografische sterkte (bit)
Sleutellengte (modulus)	Minimaal 3072 bit	Voldoende	128+
	2048-3071 bit	Uit te faseren	112-128
	Minder dan 2048 bit	Onvoldoende	<112

Tabel 4: Parameters voor RSA voor TLS-authenticatie, hun beveiligingsniveau en cryptografische sterkte [12] [11] [13].

Gebruik je EdDSA voor TLS-authenticatie? Gebruik een gestandaardiseerde Edwards-curve zoals getoond in Tabel 5.

Elliptic curve	Beveiligingsniveau	Cryptografische sterkte (bit)
Ed448	**Voldoende**	224
Ed25519		128
Andere curves	**Onvoldoende**	—

Tabel 5: Edwards-curves voor EdDSA voor TLS-authenticatie, hun beveiligingsniveau en cryptografische sterkte.

3.3.3 Sleuteluitwisseling

Maak gebruik van een sleuteluitwisselingsalgoritme dat forward secrecy biedt door het gebruik van kortstondige sleutels. Alle **Goed, Voldoende** en **Uit te faseren** algoritmes in Tabel 6 bieden hier mogelijkheden toe.

X25519MLKEM768, SecP256r1MLKEM768 en SecP384r1MLKEM1024 zijn hybride, quantumveilige algoritmes die gebruik maken van een combinatie van de algoritmes ECDHE en ML-KEM [14]. Deze algoritmes zijn (relatief) nieuw en maken *nog* geen deel uit van de TLS standaarden. Wel zijn deze onderdeel van een lopend standaardisatietraject voor TLS [15]. Met name X25519MLKEM768 wordt in toenemende mate ondersteund door TLS-softwarebibliotheken en wordt toegepast in (de meeste) webbrowsers. Overweeg om X25519MLKEM768 nu al te configureren om weerbaar te zijn tegen de dreiging van een CRQC. Zie ook sectie 4.5.

Algoritme	Beveiligingsniveau
X25519MLKEM768	**Goed**
SecP256r1MLKEM768	
SecP384r1MLKEM1024	
ECDHE	**Voldoende**
DHE	**Uit te faseren**
RSA	**Onvoldoende**
DH	
ECDH	
KRB5[8]	
NULL	
PSK[8]	
SRP	

Tabel 6: Algoritmes voor sleuteluitwisseling en hun beveiligingsniveau [6].

3.3.3.1 *Veilige parameters*

Gebruik je X25519MLKEM768, SecP256r1MLKEM768 of SecP384r1MLKEM1024 voor jouw configuratie? De parameters zijn inherent onderdeel van deze algoritmes. Je hoeft hier dus geen aanvullende keuzes voor te maken.

Gebruik je ECDHE voor jouw configuratie? Gebruik dan de parameters zoals eerder aangegeven in Tabel 3.

Gebruikt jouw configuratie vooralsnog DHE? Faseer deze uit. Gebruik, tot de uitfasering, een van de volgende gestandaardiseerde *finite field groups* (ff). Grotere finite field groups kennen een hogere cryptografische sterkte, maar ook een hogere (performance) overhead. De meeste toepassingen zullen, tijdens de uitfasering, voldoende bescherming hebben bij het gebruik van ffdhe3072 en ffdhe4096.

Finite field group	Beveiligingsniveau	Cryptografische sterkte (bit)
ffdhe8192	Uit te faseren	192
ffdhe6144		175
ffdhe4096		150
ffdhe3072		125
ffdhe2048	Onvoldoende	103

Tabel 7: Finite field groups voor DHE, hun beveiligingsniveau en cryptografische sterkte [6] [16].

3.3.4 Algoritmes voor bulkversleuteling

Gebruik een algoritme dat authenticatie en encryptie effectief integreert in een operatiemodus (AEAD-algoritmes). Dit zijn block-ciphers in -CCM of -GCM modus, of ChaCha20-Poly1305, zoals weergegeven in Tabel 8.

Algoritme	Beveiligingsniveau	Cryptografische sterkte (bit)
AES-256-GCM	Goed	256
ChaCha20-Poly1305		256
AES-256-CCM	Voldoende	256
AES-128-GCM		128
AES-128-CCM		128
AES-256-CBC	Uit te faseren	256
AES-128-CBC		128
CAMELLIA*		128/256*
ARIA*		128/256*
3DES	Onvoldoende	—
SEED		—
AES-256-CCM_8		64†
AES-128-CCM_8		64†
IDEA		—
DES		—
RC4		—
NULL		—

Tabel 8: Algoritmes voor bulkversleuteling, hun beveiligingsniveau en cryptografische sterkte [6] [17] [13]. AES-algoritmes zijn aangegeven in het format ciphernaam - # - operatiemodus, waar # refereert naar de sleutellengte.
** Voor CAMELLIA en ARIA is het beveiligingsniveau vastgesteld ongeacht de gekozen operatiemodus. De cryptografische sterkte is identiek aan de gekozen sleutellengte (128 of 256 bit).*
*† AES-CCM_8 is een variant van AES-CCM met een ingekorte authenticatie-tag. Dit heeft geen effect op de cryptografische sterkte vanuit vertrouwelijkheidsperspectief, maar verlaagt deze vanuit integriteitsbeschermingsoogpunt (naar 64 bit). Uit deze optiek worden deze algoritmes als **Onvoldoende** beschouwd.*

De algehele voorkeur gaat uit om gebruik te maken van AES-256-GCM of ChaCha20-Poly1305. Deze bieden de sterkste beveiligingswaarde. Wil je ook andere algoritmes ondersteunen? Overweeg dan het volgende.

- AES-ciphers in GCM modus genieten de voorkeur ten opzichte van andere operatiemodi vanwege hun brede ondersteuning in bibliotheken en implementaties. AES-GCM vergt in de regel ook minder rekenkracht dan AES-CCM.

- Een CRQC verlaagt (in theorie) de cryptografische sterkte van bulkversleutelingsalgoritmes. In praktijk zijn er op dit moment verschillende factoren die een aanval met een CRQC op bulkversleutelingsalgoritmes onpraktisch maken [18] [19]. De algoritmes AES-128-GCM en AES-128-CCM bieden hiertoe nog **Voldoende** bescherming.
- De ciphers ARIA en (in mindere mate) CAMELLIA worden in de westerse wereld zelden gebruikt. De ciphers zijn (zeer) beperkt ondersteund in moderne browsers en operating systems en ook (nog) niet opgenomen in de TLS 1.3 standaard. Er zijn geen indicaties dat de algoritmes zelf onveilig zijn, maar beperkte adoptie is, naast een recept voor beperkte interoperabiliteit, een risicofactor voor onontdekte softwarekwetsbaarheden. Gebruik deze ciphers niet, tenzij er een overtuigende reden is uit interoperabiliteitsoverweging.
- De block ciphers AES, CAMELLIA en ARIA ondersteunen allen een sleutellengte van 128, 192 en 256 bit. In praktijk genieten enkel configuraties met een sleutellengte 128 en 256 bit brede ondersteuning, waardoor enkel deze configuraties zijn opgenomen in
  Tabel 8.
- AES-ciphers in CBC modus vereisen het gebruik van de “encrypt-then-MAC”-extensie. Het advies is om het gebruik hiervan **Uit te faseren** [6]**.** In praktijk wordt deze extensie namelijk beperkt ondersteund door clients en servers en is het gebruik hiervan niet afdwingbaar, wat beveiligingsrisico’s oplevert.

3.3.5 Hashfuncties voor TLS

Maak gebruik van veilige hashfuncties zoals beschreven in
Tabel 9.

Algoritme	Beveiligingsniveau	Cryptografische sterkte (bit)
SHA-512	**Goed**	256
SHA-384		192
SHA-256		128
SHA-224	**Uit te faseren**	112
SHA-1	**Onvoldoende**	< 63
MD5		< 19

Tabel 9: Algoritmes voor hashfuncties, hun beveiligingsniveau en cryptografische sterkte (o.b.v. collision resistance) [6] [17] [13] [20].

3.4 Veilige opties

Gebruik het advies uit deze sectie om veilige keuzes te maken uit overige TLS-opties.

3.4.1 Compressie

Maak geen gebruik van TLS-compressie. Dit kan jouw toepassing kwetsbaar maken voor beveiligingsaanvallen, zoals CRIME [3]. Gebruik je TLS 1.3? Dan hoef je TLS-compressie zelf niet uit te zetten, aangezien TLS 1.3 dit niet meer ondersteunt.

Compressie-optie	Beveiligingsniveau
TLS 1.3: N.v.t.	**Goed**
TLS 1.2: Geen TLS-compressie	
TLS 1.2: TLS-compressie	**Onvoldoende**

Tabel 10: Opties voor TLS-compressie en hun beveiligingsniveau.

Naast TLS-compressie kan je ervoor kiezen om compressie toe te passen op applicatieniveau. Dit valt buiten de scope van jouw TLS configuratie, maar het is goed om te weten dat dit een beveiligingsrisico’s met zich mee kan brengen. Een voorbeeld hiervan is een BREACH aanval, waarbij compressie van webverkeer een kwetsbaarheid introduceert [2]. Onderzoek of jouw toepassing kwetsbaar is voor dergelijke aanvallen, voordat je kiest voor het toepassen van compressie op applicatieniveau.

3.4.2 Renegotiation

Ondersteunt jouw configuratie TLS 1.2?

- Zorg ervoor dat jouw configuratie geen ondersteuning biedt voor *insecure* renegotiation. Controleer hiervoor of de beveiligingsfix uit RFC 5746 correct is geïmplementeerd in jouw gebruikte TLS-bibliotheek [4].

- Schakel de optie voor (secure) renegotiation bij voorkeur uit. Renegotiaton maakt jouw server namelijk kwetsbaarder voor DoS-aanvallen waarvoor mitigatie niet triviaal is. Over het algemeen is het niet nodig om clients de mogelijkheid te bieden om een renegotiation te initiëren (client-initiated renegotiation).
- Moet jouw configuratie renegotiation écht ondersteunen? Limiteer dan het aantal toegestane renegotiation pogingen op de server.

Ondersteunt jouw configuratie enkel TLS1.3? Dan hoef je niets te doen. TLS 1.3 ondersteunt geen renegotiation.

Client-initiated renegotiation	Beveiligingsniveau
TLS 1.3: N.v.t.	**Goed**
TLS 1.2: Uit	
TLS 1.2: Gelimiteerd, secure renegotiation	**Voldoende**
TLS 1.2: Onbeperkt, secure renegotiation	**Uit te faseren**
TLS 1.2: Insecure renegotiation	**Onvoldoende**

Tabel 11: Opties voor renegotiation en hun beveiligingsniveau.

3.4.3 Sessies hervatten (Sessie-IDs, sessietickets)

Wil je het voor jouw toepassing mogelijk maken om sessies snel te hervatten? Maak dan gebruik van TLS 1.3. Deze versie van TLS biedt hier de meest veilige ondersteuning voor. Eerdere versies van TLS hadden het risico dat een aanvaller informatie kon ontsleutelen indien deze de *encryptiesleutel voor sessietickets* kon bemachtigen van de server. Deze tekortkoming is in TLS 1.3 gecorrigeerd.

Maak je gebruik van sessietickets in TLS 1.2? Dan is jouw server kwetsbaar voor aanvallers die jouw encryptiesleutel voor sessietickets weten te stelen. Neem extra maatregelen om toegang tot deze sleutel te voorkomen. Maak deze zo goed als mogelijk ontoegankelijk voor derden en vervang deze regelmatig.

Maak je gebruik van Sessie-IDs in TLS 1.2? Bescherm de opgeslagen sessie-parameters op soortgelijke wijze als de *encryptiesleutel voor sessietickets* zoals hierboven beschreven. Uit bruikbaarheidsoogpunt genieten sessietickets de voorkeur, gegeven de extra overhead die het gebruik van Sessie-IDs met zich meebrengt.

Optie hervatten sessie	Beveiligingsniveau
Geen optie tot hervatting	**Goed**
TLS 1.3: Sessietickets	
TLS 1.2: Sessietickets + aanvullende beschermingsmaatregelen	**Voldoende**
TLS 1.2: Sessie-IDs + aanvullende beschermingsmaatregelen	
Overig	**Onvoldoende**

Tabel 12: Opties voor het hervatten van een sessie en hun beveiligingsniveau.

3.4.4 Sessies hervatten (0-RTT)

Gebruikt jouw configuratie TLS 1.3? Schakel dan de optie '0-RTT' standaard uit. Deze optie maakt jouw TLS configuratie kwetsbaar voor replay-aanvallen. Het risico dat hieruit voortkomt is afhankelijk van de applicatiedata die er tijdens de handshake wordt meegestuurd. 0-RTT is daarmee alleen veilig te gebruiken in een applicatie-specifieke context [6]. Het ondersteunen van 0-RTT vereist daarbij zowel expertise als het volgen van applicatie-specifieke 0-RTT profielen [6]. In toepassing-agnostische context is het gebruik van 0-RTT **Onvoldoende**.

Gebruikt jouw configuratie enkel TLS 1.2? Dan hoef je niets te doen. TLS 1.2 ondersteunt geen 0-RTT.

0-RTT	Beveiligingsniveau
TLS 1.3: Geen 0-RTT	**Goed**
TLS 1.2: N.v.t.	
TLS 1.3: Wel 0-RTT	**Onvoldoende**

Tabel 13: Opties voor 0-RTT en hun beveiligingsniveau.

3.4.5 Certificaatstatuscontrole

Configureer het gebruik van OCSP-stapling op jouw server als dit mogelijk is. Hiermee bied je jouw clients een privacy-vriendelijke manier om de herroepingsstatus van jouw certificaat te controleren. Let op dat sommige CA's geen ondersteuning bieden voor OCSP vanuit privacyoverwegingen. In de praktijk is het daardoor niet altijd mogelijk om OCSP-stapling te gebruiken.

Beheer je, naast de server, ook de client? Geef de voorkeur aan het gebruik van CRLs of OCSP-stapling boven (reguliere) OCSP voor certificaatstatuscontrole.

4 Randvoorwaarden

Een TLS configuratie is 'veilig' wanneer deze veilige instellingen ondersteunt. Met enkel een veilige configuratie ben je er echter nog niet. Er zijn verschillende aspecten die niet direct onderdeel zijn van jouw TLS configuratie, maar wel belangrijk zijn voor de veilige werking van TLS. In dit hoofdstuk bieden we advies over deze aspecten.

4.1 TLS-bibliotheken

Kies voor een betrouwbare TLS-bibliotheek die past bij jouw toepassing. Stel jezelf, de leverancier van de bibliotheek of diegene die de bibliotheek integreert in jouw systemen de volgende vragen om een inzicht te krijgen in de betrouwbaarheid van deze bibliotheek.

Beschikt de gekozen TLS-bibliotheek bijvoorbeeld:

- Over een openbaar beleid over de wijze waarop melders kwetsbaarheden kunnen rapporteren?
- Over ontwikkelaars met toegang tot adequate middelen om (veiligheids-)ondersteuning te verlenen?
- Over een goede staat van dienst wat betreft het reageren op aanvallen op TLS in het verleden en op kwetsbaarheden in de implementatie van de bibliotheek?
- Over een manier om gebruikers te informeren over veiligheidsupdates, op een manier die duidelijk te onderscheiden is van de reguliere updates?
- Wordt de bibliotheek regelmatig op onafhankelijke wijze gecontroleerd en geëvalueerd?
- Maakt de bibliotheek gebruik van constant time-implementaties om beter bestand te zijn tegen *side-channel aanvallen*?
- Heeft de bibliotheek de aanbevolen maatregelen geïmplementeerd uit RFC 7627 (extended_master_secret [21]) en RFC 5746 (secure renegotiation [4])?
- Leent de bibliotheek zich voor crypto-agility? In welke mate kunnen cryptografische algoritmes eenvoudig vervangen worden, bijvoorbeeld wanneer er kwetsbaarheden zijn aangetroffen? En bevat de bibliotheek al ondersteuning voor quantumveilige cryptografische algoritmes?

Besturingssystemen beschikken doorgaans over meer dan één TLS-bibliotheek. Zorg ervoor dat je weet welke bibliotheek de serversoftware gebruikt en dat deze up-to-date is. Kies altijd de meest recente versie van de gekozen bibliotheek waarin de ontwikkelaar eventuele kwetsbaarheden in eerder versies heeft kunnen verhelpen. Gebruik uitsluitend instellingen die voor jouw toepassing noodzakelijk zijn. Dit voorkomt dat programmeerfouten in niet-noodzakelijke functionaliteiten tot kwetsbaarheden leiden.

4.2 Toevalsgetallen

Toevalsgetallen (random numbers) worden gevormd door een bron van willekeurigheid (entropie) te koppelen aan een PRNG (pseudo-random number generator). Dergelijke getallen worden op verschillende manieren gebruikt in TLS. Zo wordt de geheime sleutel voor bulkversleuteling afgeleid uit een entropiebron gecombineerd met een hash-functie.

De meeste besturingssystemen en TLS-bibliotheken bevatten een goede, cryptografisch veilige PRNG. Identificeer welke PRNG er gebruikt wordt in jouw TLS-bibliotheek en onderzoek of deze hoge kwaliteits-toevalsgetallen genereert (d.w.z. onvoorspelbare toevalsgetallen). Consulteer hiervoor de (leverancier van) de TLS-bibliotheek die je gebruikt voor jouw toepassing. Van de volgende RNG is bekend dat hij onveilig is:

- Dual EC DRBG [22]

Het is mogelijk dat het genereren van toevalsgetallen ervoor zorgt dat jouw TLS-server zwaar wordt belast. Het toevoegen van een hardwaremodule (een hardwarematige RNG) zorgt ervoor dat er altijd voldoende entropie en toevalsgetallen beschikbaar zijn. Dergelijke functionaliteit zit reeds ook ingebakken in moderne processoren.

4.3 TLS proxy

Overweeg je om een TLS proxy in te zetten? Krijg zicht op de beveiligings- en privacyrisico's die hierbij gemoeid zijn en beheers deze op gepaste wijze.

Vraagt een commerciële partij om jouw geheime sleutels voor het opzetten van een TLS proxy, bijvoorbeeld als onderdeel van anti-DDoS maatregelen? Stel deze niet eenvoudigweg beschikbaar. Het is mogelijk om een TLS proxy op te zetten zonder de geheime sleutels uit te geven[9]. Kies je voor een leverancier die dit niet ondersteunt? Breng de risico's in kaart die voortvloeien

[9] *Zie bijvoorbeeld* https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-technical-details/

uit het bekend maken van jouw geheime sleutels. Neem contractuele maatregelen ter compensatie van de verminderde technische controle en controleer regelmatig in welke mate de leverancier de afgesproken maatregelen ook in praktijk brengt.

4.4 Certificaten

Een adequaat certificaatbeheer vormt een belangrijke voorwaarde voor het veilig gebruik van TLS. Raadpleeg de NCSC publicatie Zorgeloos certificaatbeheer voor meer informatie over (het beheren van) certificaten.

4.5 Quantumveiligheid

Bereid jouw organisatie en toepassing voor op de komst van een CRQC. Een CRQC bestaat (nog) niet, maar kan nu al een risico vormen voor jouw organisatie. Zo kunnen aanvallers met een *store-now-decrypt-later* aanval jouw versleutelde data nu verzamelen en deze op een later moment ontsleutelen (wanneer zij beschikken over een CRQC). Ook leert de geschiedenis dat het migreren naar veiligere algoritmes complexe uitdagingen kent die veel tijd, planning en voorbereiding vragen. Meer informatie hierover vind je in de handreiking "Maak je organisatie quantumveilig" [23].

Quantumveilige cryptografische algoritmes voor authenticatie en sleuteluitwisseling maken *nog* geen deel uit van de TLS standaarden. Wel lopen er trajecten om hier verandering in aan te brengen. Zo loopt er een standaardisatietraject voor het gebruik van de sleuteluitwisselingsalgoritmes X25519MLKEM768, SecP256r1MLKEM768 en SecP384r1MLKEM1024 in TLS [15]. Deze algoritmes (en met name X25519MLKEM768) worden in toenemende mate gebruikt op het internet en ondersteund door TLS-bibliotheken. De verwachting is dat quantumveilige cryptografie stapsgewijs onderdeel zal worden van de TLS-standaarden. Dat betekent niet dat je moet wachten tot het zover is. Onderneem de volgende stappen om jouw TLS-configuratie en organisatie voor te bereiden op de komst van een cryptografisch relevante quantumcomputer:

1. Ondersteun zo snel als mogelijk (een recente TLS-bibliotheek met ondersteuning voor) TLS 1.3. Quantumvelige cryptoalgoritmes zullen naar verwachting alleen beschikbaar worden gesteld voor TLS 1.3. Door nu gebruik te maken van TLS 1.3 zorg je ervoor dat een noodzakelijke, toekomstige migratie soepeler zal verlopen.
2. Quantumveilige cryptografie vereist grotere ClientHello berichten om te functioneren. Door implementatiebeperkingen kunnen niet alle servers hiermee overweg. Onderzoek in hoeverre jouw server zo'n beperking heeft en in hoeverre je deze kan wegnemen. Zie voor meer informatie https://tldr.fail/
3. Gebruik algoritmes voor hashing en bulkversleuteling met een cryptografische sterkte van 256-bit. Deze bieden, ook tegen een cryptografisch relevante quantumcomputer, de hoogste beveiligingswaarde.
4. Zorg ervoor dat je weet hoe – en waar – jouw organisatie gebruik maakt van TLS. Zo zorg je ervoor dat je snel en gericht aanpassingen kan maken.
5. Zorg ervoor dat jouw configuratie en gebruikte cryptografische algoritmes makkelijk te vervangen zijn (crypto-agility). Dit stelt je in staat om een migratie naar veilige cryptografie snel uit te voeren.
6. Voer een risicoanalyse uit om te ontdekken op welke manier de quantumdreiging een risico vormt voor jouw toepassing (en organisatie) en op welke manier je daar nu al mee aan de slag moet.
    a. Wijst jouw risicoanalyse uit dat jouw toepassing/organisatie geen urgent risico loopt ten aanzien van de quantumdreiging? Wacht dan met het implementeren van quantumveilige cryptografische algoritmes tot deze onderdeel zijn van de TLS-standaarden en -bibliotheken.
    b. Loopt jouw toepassing/organisatie (nu al) significante risico's? Win expertise in en implementeer (een van) de *hybride* sleuteluitwisselingsalgoritmes die kandidaat zijn voor TLS-standaardisatie, zoals ook beschreven in paragraaf 3.3.3.

Raadpleeg het 'PQC-migratiehandboek' voor meer handvatten en duiding van bovenstaande stappen [24]. Hierin vind je uitgebreid advies over de quantumdreiging, het uitvoeren van een risicoanalyse, een overzicht van quantumveilige cryptografische algoritmes en een beschrijving van *hybride cryptografie* (waarin 'klassieke' en 'quantumveilige' algoritmes gecombineerd worden). Het handboek biedt daarnaast handvatten voor verschillende persona's: van *early adopters* (organisaties die vooruitlopend op de standaardisatie van quantumveilige cryptografie, al maatregelen moeten of willen treffen) tot *regular adopters* (die nog even kunnen wachten op de standaarden). Lees daarnaast de handreiking 'Maak je organisatie quantumveilig' voor een algemeen handelingsperspectief om jouw organisatie voor te bereiden op de komst van een cryptografisch relevante quantumcomputer.

Bijlage A: Wijziging ten opzichte van de vorige versie

In deze bijlage vind je de belangrijkste wijzigingen van deze richtlijnen ten opzichte van de versie uit 2021 (getiteld “ICT-beveiligingsrichtlijnen voor Transport Layer Security v2.1 (TLS)”.

Algemene wijziging

- De structuur van de publicatie is herzien om duidelijker onderscheid te maken tussen achtergrondinformatie, TLS-specifieke en randvoorwaardelijke adviezen.

Aanscherping definitie beveiligingsniveaus en quantumveiligheid

- De definitie en criteria voor **Goed, Voldoende** en **Uit te faseren** instellingen zijn aangescherpt en verder geëxpliciteerd.
- De dreiging van een cryptografisch relevante quantumcomputer wordt expliciet meegenomen in de beoordeling van deze beveiligingsniveaus.

TLS Versies

- TLS1.0 en TLS 1.1 zijn, conform TLS best-current practices (BCP 195), afgewaardeerd naar **Onvoldoende** [6].

Certificaatondertekening en validatie

- De algoritmes ECDSA, RSA, EdDSA (en bijbehorende parameters) zijn afgewaardeerd van **Goed** naar **Voldoende** met oog op de quantumdreiging.
- De sleutellengte voor RSA 2048-3071 is afgewaardeerd naar **Uit te faseren**. In deze richtlijnen wordt een cryptografische sterkte nagestreefd van 128-bit, waarvoor een RSA-sleutellengte van 3072-bit wordt aangehouden.

Sleuteluitwisseling

- De algoritmes X25519MLKEM768, SecP256r1MLKEM768 en SecP384r1MLKEM1024 zijn opgenomen als **Goed**.
- Het algoritme ECDHE (en bijbehorende elliptic curves) is afgewaardeerd van **Goed** naar **Voldoende** met oog op de quantumdreiging.
- DHE voor sleuteluitwisseling is afgewaardeerd naar **Uit te faseren** conform BCP 195 [6]. De hierbij behorende finite-field groepen zijn hiertoe ook afgewaardeerd. De finite-field groep ffdhe2048 is daarbij afgewaardeerd naar **Onvoldoende**, gegeven de beperkte cryptografische sterkte die deze finite field-groep biedt.
- RSA voor sleuteluitwisseling is, conform BCP 195, afgewaardeerd naar **Onvoldoende** [6].
- De aanbeveling voor het PSS paddingschema is verplaatst van Tabel 4 naar de begeleidende tekst.

Bulkversleuteling

- Ciphers in CBC-modus zijn, conform BCP 195, afgewaardeerd naar **Uit te faseren** [6]. Dit, met oog op het feit dat block-ciphers in CBC-modus enkel veilig te gebruiken indien de encrypt_then_mac extensie wordt gebruikt uit RFC7366 [25]. Deze extensie moet zowel aan de client als server kant worden gebruikt. In praktijk wordt deze extensie beperkt ondersteund en is het gebruik hiervan niet afdwingbaar, wat beveiligingsrisico’s oplevert.
- De beveiligingsniveaus voor CAMELLIA, ARIA en SEED zijn explicieter beschreven. Voor CAMELLIA en ARIA zijn geen algoritmische kwetsbaarheden bekend, maar gegeven de beperkte adoptie is er een risico’s op onontdekte softwarekwetsbaarheden. Om deze reden zijn deze algoritmes ingeschaald als **Uit te faseren.** SEED is een verouderd algoritme en is opgenomen als **Onvoldoende**.
- AES-128-GCM is afgewaardeerd naar **Voldoende**.
- AES in CCM modus is opgenomen als **Voldoende**.
- 3DES-CBC is afgewaardeerd naar **Onvoldoende.**

Hashfuncties

- In oudere versies van deze richtlijnen werd er een verschil gemaakt tussen de specifieke toepassing van de hashfuncties en de daarvoor gehanteerde beveiligingsniveaus. Dit resulteerde in drie tabellen (voor *authenticatie, sleuteluitwisseling en bulkversleuteling*), elk met een ander format. Inhoudelijk verschilden deze tabellen nauwelijks van elkaar. In deze versie

worden hashfuncties ondergebracht in 1 tabel met bijbehorend advies. Het gebruik van SHA-1 is daarbij, ongeacht de toepassing, opgenomen als **Onvoldoende.**

- Het hashingalgoritme SHA-224 is expliciet opgenomen als **Uit te faseren**. Dit is geen inhoudelijke wijziging van de richtlijn, maar een verduidelijking.

Overige opties

- Het gebruik van client-initiated renegotiation is afgewaardeerd naar **Uit te faseren**
- Het advies voor overige opties zijn inhoudelijk niet gewijzigd, maar zijn verder geëxpliciteerd en voorzien van aanvullende onderbouwingen.

Bijlage B: Lijst met cipher suites

Als hulpmiddel voor het configureren en controleren van jouw TLS instellingen beschrijven we in deze bijlage de cipher suites die het beveiligingsniveau **Goed, Voldoende** en **Uit te faseren** krijgen. We hanteren hierbij de regel dat het beveiligingsniveau van een cipher suite zo laag is als het laagst-gewaardeerde algoritme. De lijst in deze bijlage is opgesteld op basis van de lijsten van IANA, OpenSSL en GnuTLS, opgehaald via https://ciphersuite.info/ (op 20 maart 2025).

Let op dat deze bijlage geen uitputtende lijst bevat van **Goed, Voldoende** en **Uit te faseren** cipher suites. Het is mogelijk dat jouw TLS-bibliotheek andere cipher suites ondersteunt. Controleer (in dit geval) in welke mate je deze cipher suites kan ondersteunen op basis van het beveiligingsniveau van de betreffende algoritmes. Daarnaast is enkel het configureren van deze cipher suites niet genoeg om te komen tot een veilige TLS configuratie. Raadpleeg voor meer instellingen en randvoorwaarden hoofdstukken 3 en 4.

TLS 1.3 - Goed

TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256

TLS 1.3 - Voldoende

TLS_AES_128_CCM_SHA256
TLS_AES_128_GCM_SHA256

TLS 1.2 - Voldoende

TLS_ECDHE_ECDSA_WITH_AES_128_CCM
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CCM
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

TLS 1.2 - Uit te faseren

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_CCM
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CCM
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256
TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256
TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384
TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256

TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384

Verwijzingen

[1] Internet Engineering Task Force (IETF), „RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3," 2018.

[2] A. Prado, N. Harris en Y. Gluck, „https://www.breachattack.com/," 2013. [Online].

[3] D. Fischer, „CRIME Attack Uses Compression Ratio of TLS Requests as Side Channel to Hijack Secure Sessions," ThreatPost, 2012.

[4] Internet Engineering Task Force (IETF), „RFC 5746: Transport Layer Security (TLS) Renegotiation Indication Extension," 2010. [Online].

[5] Internet Engineering Task Force (IETF), „RFC 5077: Transport Layer Security (TLS) Session Resumption without Server-Side State," 2008.

[6] Internet Engineering Task Force (IETF), „BCP 195 (RFC 7525 & RFC 8996): Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," 05 2021. [Online]. Available: https://www.rfc-editor.org/bcp/bcp195.txt.

[7] CA/Browser Forum, „Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates v1.8.4," CA/Browser Forum, 2022.

[8] Internet Engineering Task Force (IETF), „RFC 5480: Elliptic Curve Cryptography Subject Public Key Information," 2009.

[9] Thales, „Supported ECC curves," Thales, [Online]. Available: https://www.thalesdocs.com/gphsm/ptk/protectserver3/docs/ps_ptk_docs/ptkc_programming/ecc_curves/index.html. [Geopend 13 02 2025].

[10] SSL Support team, „What is Elliptic Curve Cryptography (ECC)?," SSL.com, 08 10 2024. [Online]. Available: https://www.ssl.com/article/what-is-elliptic-curve-cryptography-ecc/#ftoc-heading-6. [Geopend 13 02 2025].

[11] German Federal Office for Information Security (BSI), „TR-02102-02: "Cryptographic Mechanisms: Recommendations and Key Lengths, Part 2 – Use of Transport Layer Security (TLS)" Version: 2025-1," 04 03 2025. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-2.pdf.

[12] National Institute of Standards and Technology (NIST), „SP 800-131A: Transitioning the Use of Cryptographic Algorithms and Key Lengths, revision 2," 2019.

[13] The National Cyber and Information Security Agency of the Czech Republic (NÚKIB), „Minimum requirements for cryptographic algorithms," 2024.

[14] National Institute of Standards and Technology (NIST), „FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard," 2024.

[15] K. Kwiatkowski, P. Kampanakis, B. Westerbaan en D. Stebila, „Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3," IETF, 25 12 2024. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-kwiatkowski-tls-ecdhe-mlkem-03. [Geopend 17 03 2025].

[16] Internet Engineering Task Force (IETF), „RFC 7919 : Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)," 08 2016. [Online].

[17] German Federal Office for Information Security (BSI), „BSI TR-02102-1 "Cryptographic Mechanisms: Recommendations and Key Lengths" Version: 2025-1," 04 03 2025. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf.

[18] Sarah D., UK National Cyber Security Centre, „On the practical cost of Grover for AES key recovery," in *5th PQC Standardization Conference*, 2024.

[19] S. Mandal, M. Rahman, S. Santanu en I. Takanori, „Implementing Grover's on AES-based AEAD schemes," *Nature Scientific Reports 14,* 10 11 2024.

[20] National Institute of Standards and Technology (NIST), „Hash Functions," 09 09 2024. [Online]. Available: https://csrc.nist.gov/projects/hash-functions. [Geopend 13 02 2025].

[21] Internet Engineering Task Force (IETF), „RFC 7627: Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension," 2015.

[22] NIST, „NIST opens draft Special Publication 800-90A, Recommendation for random number generation using deterministic random bit generators, for review and comment," NIST, 2012.

[23] AIVD, NCSC, „Maak je organisatie quantumveilig," 2023. [Online]. Available: https://www.ncsc.nl/documenten/publicaties/2023/september/18/maak-je-organisatie-quantumveilig.

[24] AIVD, CWI, TNO, „Het PQC-migratie handboek," 2024. [Online]. Available: https://www.aivd.nl/documenten/publicaties/2024/12/3/het-pqc-migratie-handboek.

[25] Internet Engineering Task Force (IETF), „RFC 7366: Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," 2014.

[26] National Institute for Standards and Technology (NIST), „NIST SP 800-131Ar3 ipd: Transitioning the Use of Cryptographic Algorithms and Key Lengths; Initial Public Draft," 01 10 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar3.ipd.pdf.

[27] Internet Engineering Task Force (IETF), „RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," 05 2008. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5280.