

ONDERZOEK

SOFTWARELICENTIEMODEL



Datum	11-10-2022
Versie	1.0
Status	Final
Auteur	Patrick van Nieuwburg

Versie

Versie	Datum	Auteur(s)	Wijzigingen	Status
0.1	11-10-2022	PvN	Begin onderzoek	Afgerond
0.2	20-10-2022	PvN	Afronden onderzoek softwarelicentie	Afgerond
0.3	10-12-2022	PvN	Onderzoek cryptografie	Afgerond
0.4	30-12-2022	PvN	Onderzoek mailserver	Afgerond
1.0	09-01-2023	PvN	Beantwoorden hoofdvraag en deelvragen	Afgerond

Verspreiding

Versie	Datum	Aan
0.3	20-12-2022	Mark Melstroom
1.0	10-01-2023	Inleveren portfolio – Mark Melstroom
1.0	10-01-2023	Jeroen van Rijckevorsel

Inhoudsopgave

1.	Begrippenlijst	5
2.	Onderzoeksvragen	6
2.1.	Hoofdvraag	6
2.2.	Deelvragen	7
2.2.1.	Wat zijn voor TBWB geschikte softwarelicentiemodellen en hoe kunnen we deze toepassen?	7
2.2.2.	Wat is een geschikt verkoopmodel met betrekking tot dit softwarelicentiemodel?	7
2.2.3.	Hoe houden we een klant op de hoogte van zijn licentiestatus en wat doet TBWB wanneer een klant niet betaalt?	8
2.2.4.	Hoe kan een softwarelicentiemodel geïmplementeerd worden in het 'standaardproduct' van TBWB?	8
3.	Softwarelicentie	10
3.1.	Wat is een software licentie	10
3.2.	Hoe werkt een softwarelicentie	10
3.3.	Waarom zijn software licenties belangrijk	11
3.3.1.	Hoe beschermd een licentie de ontwikkelaars	11
3.3.2.	Hoe beschermd een licentie de gebruikers	11
4.	Soorten softwarelicenties	12
4.1.	Software implementatie methodes	12
4.1.1.	On-Premise	12
4.1.2.	Cloud Based	12
4.1.3.	Hybrid	12
4.2.	Softwarelicentiemodellen	12
4.2.1.	Perpetual license	12
4.2.2.	Subscription license	13
4.2.3.	Concurrent softwarelicense	13
4.3.	Bronnen gebruikt voor dit onderzoek	14
5.	Advies softwarelicenties	15
6.	Cryptografie	16
6.1.	Waarom cryptografie	16
6.2.	Hashing	16
6.3.	Encryptie & Decryptie	16
6.3.1.	Symmetric Algorithms	16
6.3.2.	Asymmetric algorithms	17

6.3.3.	Digital signatures	17
6.4.	Creëren van een private en public key	18
6.5.	Bronnen gebruikt voor dit onderzoek	18
6.6.	PoC voor dit onderzoek	18
7.	Mailserver	19
7.1.	Wat is een mailserver.....	19
7.2.	Soorten mailservers	19
7.3.	Advies	19
7.4.	Bronnen gebruikt voor dit onderzoek	19
7.5.	PoC voor dit onderzoek	19

1. Begrippenlijst

Afkorting/ begrip	Beschrijving
TBWB	Mijn stagebedrijf
Standaardproduct	De nieuwe software die we bij TBWB aan het schrijven zijn die het mogelijk maakt standaard software neer te zetten bij nieuwe klanten.
Stakeholders	Een persoon met een belang of bezorgdheid in het bedrijf. Voor mijn project zijn dit de productmanager de afdeling verkoop, de afdeling service en de afdeling software.
WCS	Warehouse controle system; is een softwaretoepassing binnen TBWB voor het aansturen van de transportbanen. Dit bepaalt hoe de goederenstroom over ons geautomatiseerd magazijn gaat.
WMS	Warehouse management system; is een softwaretoepassing binnen TBWB waarin we precies weten welke items of artikelen op welke plek ligt. Daarnaast is dit stuk software in beheer over de invoer en uitvoer van deze artikelen.
PLC	Programmable logic controller; is een elektronisch apparaat met een microprocessor, dat op basis van de informatie op zijn diverse ingangen zijn uitgangen aanstuurt.
Rowa	Een mini-magezijn van 'BD Rowa' voor het opslaan van kleine artikelen binnen een geautomatiseerd systeem
PoC	Proof of concept; is een realisatie van een bepaalde methode of idee om de haalbaarheid ervan aan te tonen.
Major update	Grote update binnen een softwareprogramma
Minor update	Kleine update binnen een softwareprogramma
Changerequest	Een aanvraag van de klant voor het aanpassen of toevoegen van functionaliteit binnen de software.
MoSCoW-Methode	De MoSCoW-methode is een wijze van prioriteiten stellen in onder meer de software engineering.
MVP	Minimum viable product; is een vroege, uitgekilde versie van het eindproduct, met het absolute minimum aan functionaliteiten

2. Onderzoeksvragen

2.1. Hoofdvraag

Hoe kan een softwarelicentiemodel TBWB helpen om een maandelijkse/jaarlijkse inkomstenstroom te genereren zonder onze klanten te belemmeren in hun dagelijkse werkzaamheden.

2.1.1.1. Antwoord

Door een softwarelicentiemodel te introduceren zie je af van een eenmalige aankoop van de software. Ik heb geadviseerd om nog steeds een eenmalige vergoeding te vragen voor de aanschaf van de software, maar dat er elk jaar kosten worden doorberekend aan de klant voor de service en onderhoud van deze software.

Daarnaast wil ik ook gebruik gaan maken van gebruikerslicenties. Deze licenties heb ik zo opgezet dat de klant op elk moment van de dag het aantal gebruikers kan aanpassen. Komen de kerstdagen eraan, en ze gaan met grotere ploegen werken, dan kunnen er binnen een paar minuten extra medewerkers meedraaien in het WMS. Zijn de kerstdagen voorbij, dan hebben ze de aantal gebruikers ook zo weer afgeschaald.

Ik wil natuurlijk niet dat we de klant gaan belemmeren in hun dagelijkse werkzaamheden, maar er zitten twee kanten aan deze eis. Aan de ene kant heb ik ervoor gezorgd dat de applicatie zo in elkaar zit dat de licentiemanager die bij de klant op de server staat, blijft werken als de verbinding wegvalt met de licentieservice. Elke vijf minuten wordt het certificaat dat bij de licentieservice opgeslagen staat, opgehaald en opgeslagen op de server van de klant. Mocht er uiteindelijk toch iets misgaan, is er een pagina ontwikkeld voor de afdeling service waar de servicemedewerker met één druk op de knop een nieuw certificaat kan genereren voor de klant. Deze kan hij daarna downloaden en bij de klant op de server zetten. Zo garandeer je dat de klant altijd door kan met zijn werkzaamheden en niet tegen het probleem aanloopt dat ze niet meer kunnen inloggen op het WMS.

Aan de andere kant willen we ook de klanten 'stimuleren' op tijd zijn facturen te betalen. Door aan het certificaat een datum toe te voegen wanneer de laatste factuur is betaald kan ik in de applicatie ervoor zorgen dat de klant kleine ongemakken zal ervaren wanneer er niet betaald wordt. Denk hierbij aan een melding die na het inloggen op het scherm getoond wordt, deze zullen ze dan elke keer moeten wegklikken. Is de klant één maand te laat met betalen, behoud TBWB het recht om de procesverwerking en het aantal gebruikerslicenties te verlagen, om zo de klant te stimuleren tot betalen. Deze verlaging doet TBWB tot op zekere hoogte, de klant moet natuurlijk ten alle tijden blijven draaien.

2.2. Deelvragen

2.2.1. Wat zijn voor TBWB geschikte softwarelicentiemodellen en hoe kunnen we deze toepassen?

Binnen TBWB hebben we meerdere stakeholders die iets vinden van de invoer van een licentiemodel, hierbij kan worden gedacht aan: de verkoopafdeling, de serviceafdeling, de producteigenaar van het standaardproduct en het hoofd van de software-afdeling. Met al deze partijen wil ik in gesprek gaan om een aantal vragen voor te leggen waarop ik graag een antwoord zou willen. Al deze vragen worden dan door mij gebundeld zodat ik een voorstel kan doen dat zo goed mogelijk tegemoetkomt aan de wensen en eisen van de stakeholders.

2.2.1.1. Antwoord

Mijn advies naar TBWB toe is om te kiezen voor een hybride model. TBWB verkoopt een WMS en een WCS. Deze modules worden geïnstalleerd op de server van de klant dus het softwarelicentiemodel dat daarbij past is het on-premise model. De klant heeft lokaal toegang tot deze applicaties, daardoor is er geen betere keus dan dit model.

De gebruikerslicenties worden online beheerd dus een cloud-based oplossing is hier het antwoord. Doordat we nu gebruik maken van een on-premise model en een cloud-based model spreek je van een hybride model.

2.2.2. Wat is een geschikt verkoopmodel met betrekking tot dit softwarelicentiemodel?

Het standaardproduct gaan we opbouwen uit verschillende micro-services. De vragen die van toepassing zijn: 'Gaan we een klant laten betalen voor het gebruik-per service of verkopen we een licentie voor de hele installatie? Gaan we het licentiemodel verkopen per gebruiker of krijgt de klant een licentie voor het hele bedrijf? Valt service onder de licentie of wordt een service contract los gezien van het licentiemodel? Hoe kan dit worden toegepast in de POC?'

2.2.2.1. Antwoord

Voor de gebruikerslicenties is mijn advies om te kiezen voor gelijktijdige licenties. Omdat TBWB in de industriële automatisering werkt komt het vaak voor dat onze klanten in ploegen werken. Het kan dus goed zijn dat de klant negentig man personeel in dienst heeft, terwijl er maar dertig man personeel gelijktijdig met het systeem bezig is.

Ik wil deze gelijktijdige licenties dan ook opzetten door gebruik te maken van een 'floating license model'. Hierdoor geef je de klant de flexibiliteit om iedereen te laten werken met het systeem en de licenties niet hoeft te koppelen aan een email of gebruikersnaam. We gaan de klant dus eenmalig laten betalen voor de aanschaf van de modules. De modules die bij de klant op de server staan. Nadat de eenmalig aanschaf is gedaan zou ik adviseren om daarna elk jaar een percentage van de verkoop door te rekenen aan de klant voor updates, service en onderhoud aan het systeem.

Vooralsnog blijft het service contract los van de licenties, in de toekomst kunnen we deze twee modellen wel koppelen maar vooralsnog staan deze twee dingen los van elkaar.

2.2.3. Hoe houden we een klant op de hoogte van zijn licentiestatus en wat doet TBWB wanneer een klant niet betaalt?

We werken met klanten die productie draaien, dus de klanten moeten vroegtijdig op de hoogte gesteld worden van het moment waarop hun licentie afloopt. Doen we dit via mail, sms of laten we de afdeling administratie bellen? De manier waarop we klanten informeren, moet inzichtelijk gemaakt worden. En wat gaan we doen wanneer een licentie is afgelopen en een klant niet betaalt? Stoppen we dan zijn service? Hoe gaan we hier als organisatie mee om?

2.2.3.1. Antwoord

De klant wordt op de hoogte gehouden door gebruik te maken van een mailserver in de licentieservice. Deze stuurt aan het eind van de maand een automatisch incasso om het maximaal aantal gebruikers van afgelopen maand door te berekenen aan de klant.

Zodra de klant deze niet betaald heeft hij een maand de tijd om dit alsnog te doen, in deze maand gaan we naarmate de maand vordert een melding in de front-end tonen waarop staat dat ze contact op moeten nemen met de afdeling administratie. Deze melding moet als irritant gezien worden zodat de klant toch overgaat op betalen.

Als de klant na 1 maand nog steeds niet heeft betaald behoud TBWB het recht om de gebruikerslicenties te verminderen naar drie gebruikers en de performance van de klant te verminderen. In plaats van vijftig orders per minuut kunnen er dan nog maar tien order per minuut uitgevoerd worden.

2.2.4. Hoe kan een softwarelicentiemodel geïmplementeerd worden in het 'standaardproduct' van TBWB?

Na afloop van mijn onderzoek wil ik een POC maken waarin ik de wensen van alle stakeholders wil laten terugkomen. Wensen van deze stakeholders zouden bijvoorbeeld kunnen gaan over: 'Overzicht van alle licentiehouders. Werking van een mail, sms-service wanneer een licentie bijna afloopt? Scherm voor de klant waarop hij zijn licentie makkelijk kan opwaarderen of een licentie kan koppelen aan een ander account? Als een klant niet betaalt, gaan bepaalde services dan offline?' Of deze wensen nodig zijn moet blijken uit de interviews die ik ga afnemen bij de stakeholders in de tweede sprint van mijn project.

2.2.4.1. Antwoord

Na het verwerken van alle interviews ben ik tot de conclusie gekomen dat elke stakeholder zijn eigen aandachtspunt heeft. De afdeling service wil er zeker van zijn dat hun afdeling geen telefoontjes krijgt na het implementeren van het softwarelicentiemodel dat er ineens niet meer gewerkt kan worden omdat de licentieservice wegvalt.

De afdeling verkoop wil dat de klant zelf zijn licenties kan beheren zodat verkoop niet continu gestoord wordt om licenties toe te voegen. Daarnaast moet er voor die afdeling een goede basis staan om het product te kunnen verkopen aan de klant.

De afdeling software wil daarentegen weer weten hoe ik de security ga oppakken. Er worden certificaten gegenereerd en die mogen niet van buitenaf bewerkt worden hoe gaan we hier mee om?

Dit zijn de belangrijkste punten waar ik aan wil werken voor mijn applicatie dus ga ik mijn applicatie in 3 grote lijnen opzetten. Ten eerste wil ik een dashboard maken. In dit dashboard kan de klant het aantal gebruikerslicenties aanpassen, de afdeling service handmatig certificaten genereren en de afdeling administratie kan hier de factureringsstatus bijwerken. Mijn visie is om van dit dashboard in de toekomst echt een 'klantenportaal' van te maken. De klant kan hier dan al zijn facturen en servicecontracten bekijken en eventuele changerequests indienen.

Ten tweede moet er op de server van de klant een licentiemanager komen. Deze licentiemanager staat offline op de server van de klant en staat in verbinding met StockControl, het standaardproduct van TBWB. StockControl controleert de geldigheid van de gebruikerslicenties en of het aantal gebruikerslicenties niet wordt overschreden. Daarnaast staat de licentiemanager ook in contact met de licentieservice om het certificaat wat deze manager gebruikt up-to-date te houden. Belangrijk is dat deze licentiemanager moet blijven werken als er geen verbinding is tussen de manager en de service. Hiermee garandeer ik dat de klant kan blijven doorwerken als er geen verbinding is. Ik wil kijken of ik door middel van een 'watchdog' de klant en de afdeling service op de hoogte kan brengen wanneer de verbinding is weggefallen.

En als laatste wil ik een licentieservice gaan bouwen. Deze service staat online en is de kern van het gehele systeem. Wanneer er in het dashboard aanpassingen worden gemaakt om het aantal gebruikerslicenties te wijzigen, worden deze doorgezet naar deze service. De service zorgt er dan voor dat er een nieuw beveiligd certificaat gegenereerd wordt die door de licentiemanager gebruikt kan worden. Doordat deze licentieservice online staat heb ik de mogelijkheid om een mailserver te introduceren. Deze mailserver stuurt automatisch een mail naar de afdeling administratie zodra er een verandering plaatsvindt in het aantal gebruikers. Zo kan de administratief medewerker de klant gelijk facturen.

3. Softwarelicentie

3.1. Wat is een software licentie

Kort gezegd is een software licentie een document of overeenstemming dat juridisch bindende richtlijnen geeft over het gebruik en distributie van software. Vaak geven software licenties de eindgebruiker de mogelijkheid om een of meerdere kopieën van de software te gebruiken zonder de auteursrechten te schenden. Deze licentie geeft richtlijnen om de verantwoordelijkheid en de beperkingen aan te geven van deze software.

Een aantal van de voorwaarden die beschreven staan in de overeenkomst zijn onder anderen hoe de software gebruikt mag worden, de beperkingen van aansprakelijkheid, garanties en disclaimers. Daarnaast wordt er ook aangegeven hoe de software beschermd wordt wanneer er inbreuk wordt gemaakt op het intellectuele eigendomsrecht van andere partijen.

Software licenties zijn doorgaans propriëtair, gratis of open source. Wat deze vormen onderscheid is hoe de gebruikers de software mogen her-distribueren en/of kopiëren voor toekomstige ontwikkeling en gebruik.

3.2. Hoe werkt een softwarelicentie

In een software licentie staat dus beschreven wat de rechten zijn van de ontwikkelaars en gebruikers over een stuk software. Een aantal voorbeelden:

- Wat de software kost;
- Welke toegang de gebruikers hebben tot de broncode;
- Hoe vaak de software gedownload mag worden;
- Hoeveel gebruikers je licentie mag bevatten;

Vaak wordt een softwarelicentie uitgegeven als een licentieovereenkomst voor ondernemingen of een licentieovereenkomst voor eindgebruikers. Deze licentieovereenkomst is een contract tussen de eindgebruiker of gebruikersorganisatie en de ontwikkelende partij. De gebruiker moet voordat hij gebruik van de software kan maken de voorwaarden die in de licentie beschreven staat accorderen. Hierdoor ontstaat dan een bindend contract tussen de gebruiker en de ontwikkelaars.

Deze software wordt vaak geleverd met een product- of licentiesleutel. Deze sleutel wordt in de software gebruikt om te controleren of deze gebruiker recht heeft om gebruik te maken van de software. Dit verifiëren of identificeren van de gebruiker kan op basis van een account gebeuren of deze kan versleuteld worden in een stuk hardware.

3.3. Waaronz zijn software licenties belangrijk

In deze licentie zijn de rechten vastgelegd voor alle partijen, dit zijn de auteur, de aanbieder en de eindgebruiker. Het definieert de relatie tussen het softwarebedrijf en gebruikers en legt uit hoe ze worden beschermd.

3.3.1. Hoe beschermd een licentie de ontwikkelaars

- Ze beschermen de ontwikkelaars door intellectueel eigendom en handelsgeheimen vast te leggen op basis van de auteursrechten;
- Er wordt bepaald wat andere partijen kunnen doen met de software;
- Ze beperken de aansprakelijkheid van de verkoper;

3.3.2. Hoe beschermd een licentie de gebruikers

- Er wordt beschreven wat de gebruiker kan doen met de software die ze zelf niet geschreven hebben;
- Er wordt vastgesteld hoe gebruikers de softwarelicenties moeten naleven, en beschermt zichzelf tegen inbreukclaims en hun wettelijke aansprakelijkheid;
- Ze helpen gebruikers een positieve relatie te onderhouden met softwareontwikkelaars en leveranciers;
- Ze voorkomen te hoge uitgaven aan licenties door duidelijke parameters vast te stellen over hoeveel licenties een organisatie nodig heeft;

4. Soorten softwarelicenties

4.1. Software implementatie methodes

4.1.1. On-Premise

On-premise licenties worden over het algemeen geïnstalleerd op de server of computer van de gebruiker. Deze licentie is lokaal toegankelijk via het apparaat en was vroeger veel voorkomend in het tijdperk van de cd-roms en diskettes.

4.1.2. Cloud Based

Waarbij on-premise licenties geïnstalleerd staan op het systeem van de gebruiker worden cloud based licenties uitgevoerd, gevold en beheerd via internet. Deze vorm is veel handiger omdat je alle systemen gebruik kunnen maken van dezelfde installatiesoftware. Daarnaast zijn de mogelijkheden om nieuwe functionaliteit toe te voegen een stuk toereikender.

Ontwikkelaars hebben door gebruik te maken van cloud based oplossingen veel betere data over hoe de gebruikers hun software gebruiken. Hiermee maakt het de ontwikkelaar makkelijker om het verkoopmodel aan te passen of ongeoorloofd gebruik te kunnen detecteren.

Het grootste voordeel is toch wel dat er wijzigingen aan de software kunnen plaatsvinden zonder dat er nieuwe hardware nodig is om deze update mogelijk te maken. De klant kan gebruikers toevoegen of aftrekken en eenvoudig hun licenties verlengen.

4.1.3. Hybrid

Een Hybrid licentiemodel wordt vaak gebruikt als tussenoplossing tussen de on-premis en cloud base oplossingen. Wanneer een bedrijf de overgang wil maken naar cloud based, maar nu met een on-premis model werkt. Of wanneer er maar een aantal functionaliteiten nodig zijn waarbij extra beveiliging nodig heeft terwijl andere teams betere flexibiliteit nodig hebben.

4.2. Softwarelicentiemodellen

4.2.1. Perpetual license

Simpel gezegd is een perpetual license een softwarelicentie die je een keer koopt en vervolgens kan gebruiken zolang je wilt. Voordat de Cloud based oplossingen kwamen was dit de meest gebruikt variant binnen de softwarelicentiemodellen.

Het makkelijke aan een perpetual license is dat je een prijs hebt die makkelijk te verantwoorden is aan de klant, aan de andere kant heb je vaak niet de mogelijkheid de software te updaten.

4.2.2. Subscription license

Een subscription license is het tegenovergestelde van een perpetual license. Bij een subscription license heb je maandelijks of jaarlijkse terugkerende kosten voor je abonnement. Wanneer de klant ervoor kiest om het abonnement op te zeggen verliest hij zijn toegang tot de software.

Het voordeel van een subscription license is dat dit abonnement veel meer flexibiliteit ondersteunt dan een perpetual license. Zo heeft de klant geen hoge opstartkosten, blijft de software up-to-date door updates en patches en de ontwikkelaars bouwen een relatie op met hun klanten.

4.2.2.1. Saas License

Software as a service, of beter gezegd SaaS is een softwarelicentie waarbij de klanten betalen om gebruik te maken van de software zonder een exemplaar van deze software te bezitten. Aan dit abonnementsmodel zitten vaak services gekoppeld zoals klantenservice, upgrades en toegang tot bepaalde hulpmiddelen. De software wordt vaak gehost in de cloud in plaats van een installatie op de server.

Je kunt vandaag de dag stellen dat deze vorm van een licentiemodel het meest gebruikt wordt bij bedrijfssoftware. De software is zo goed te onderhouden en de licentieovereenkomst kan hierdoor makkelijk worden aangepast. Doordat alles via in de cloud draait is de software voor alle klanten makkelijk te schalen.

4.2.2.2. User-Based license

In tegenstelling tot een SaaS softwarelicentie die beschikbaar is voor het hele bedrijf worden user-based licenties gekoppeld aan een gebruiker. Met een gebruikerslicentie maakt het vaak niet uit hoeveel apparaten een gebruiker heeft. Deze vorm wordt vaak gebruikt voor software die veel persoonlijke informatie of voorkeuren opslaat van de gebruiker. Zo hebben gebruikers hun eigen dashboard voor hun taken maar kan de informatie gedeeld worden met collega's.

4.2.3. Concurrent softwarelicense

Bij een concurrent softwarelicentiemodel koopt een bedrijf een x aantal licenties. Deze licenties mogen door alle gebruikers worden gebruikt tot de max van de aantal licenties is bereikt. Zo kunnen alle gebruikers op alle apparaten werken met dezelfde licentie. Dit model is zowel eenvoudig als flexibel waardoor dit een goed model is voor bedrijven die de kosten laag willen houden.

Dit softwarelicentiemodel is handig voor bedrijven die in verschillende tijdzones en/ of ploegen werken. Zo kunnen meerdere gebruikers gebruik maken van dezelfde licentie en hoeft het bedrijf zich geen zorgen te maken dat ze licenties hebben die niet gebruikt worden.

4.2.3.1. Floating license

Een floating license wil niks anders zeggen dan een softwarelicentie die op basis van het aantal gelijktijdige gebruikers wordt verleend. Bij deze softwarelicentie maakt het niet uit wie de gebruikers zijn of welk apparaat ze gebruiken. Het wordt een floating license genoemd omdat de licentie makkelijk te verplaatsen is tussen verschillende gebruikers binnen een organisatie.

4.2.3.2. Feature license

Een feature license maakt verschillende functies beschikbaar voor verschillende klanten. Hierdoor krijgen de softwareontwikkelaars de mogelijkheid de software voor verschillende klanten anders aan te bieden. Denk hierbij aan een softwarelicentie voor een 'basispakket' of een 'premiumpakket' dat veel meer gebruikers toelaat. Hiermee wordt een combinatie gemaakt van een user-based license en een feature license.

4.3. Bronnen gebruikt voor dit onderzoek

Plourde, R. (2021, 27 januari). *8 Best Practices for Successful Software License Management*. USU Software. <https://blog.usu.com/en-us/8-best-practices-for-successful-software-license-management>

Software Contract Solutions. (2018, 14 juni). *Important Questions to Ask When Negotiating Software License Agreements*. <https://softwarecontractsolutions.com/important-questions-to-ask-when-negotiating-software-license-agreements/>

Absolute Guide to Software Licensing Types | Licensing Models. (z.d.-b). <https://cpl.thalesgroup.com/software-monetization/software-licensing-models-guide>

5. Advies softwarelicenties

Mijn advies is gebaseerd op dit onderzoek en de interviews die ik heb afgenomen in mijn [analyse](#). Ik heb alle aandachtspunten die ik heb binnengekregen van mijn stakeholders en mijn eigen research gebundeld om zo tot een advies te komen welk licentiemodel geschikt is voor TBWB en hoe ik deze wil gaan implementeren.

Mijn advies naar TBWB toe is om een te kiezen voor een hybride model. Verschillende modules (WMS, WCS en PLC) die bij de klant op de server staan (on-premise) en de gebruikerslicenties die beheerd worden door een onlinesysteem.

Voor de gebruikerslicenties is mijn advies om te kiezen voor gelijktijdige licenties. Omdat TBWB in de industriële automatisering werkt komt het vaak voor dat onze klanten in ploegen werken. Het kan dus goed zijn dat de klant negentig man in dienst heeft, terwijl er maar dertig man gelijktijdig met het systeem bezig is.

Ik wil deze gelijktijdige licenties dan ook opzetten door gebruik te maken van een 'floating license model'. Hierdoor geef je de klant de flexibiliteit om iedereen te laten werken met het systeem en de licenties niet hoeft te koppelen aan een email of gebruikersnaam.

Ik wil de gebruikerslicenties zo opzetten dat de klant ten alle tijden de mogelijkheid heeft de aantal gebruikerslicenties te verhogen. Veel van onze klanten hebben het een stuk drukker in de weken voorafgaande de kerstdagen. Door een flexibel systeem te leveren zorg je ervoor dat de klant in de drukke periodes een aantal extra gebruikslicenties kan aan schaffen en deze weer te verlagen zodra de drukte weer gezakt is.

Deze gebruikerslicenties beginnen te lopen zodra de klant deze heeft aangevraagd. Zodra deze licenties worden aangemaakt wil ik deze bijhouden in de database en aan het eind van de maand ervoor zorgen dat alle klanten een automatische incasso krijgen over het maximaal aantal gebruikers dat de klant heeft gebruikt die maand.

6. Cryptografie

6.1. Waarom cryptografie

De definitie van cryptografie is de wetenschap van het beveiligen van informatie door deze om te zetten in en beveiligde vorm zodat alleen de juiste partijen deze data kunnen lezen en verwerken.

Het geeft vertrouwen, want de data die verstuurd wordt moet beschermd worden tegen het lezen van die data. Het zorgt voor integriteit omdat ervoor gezorgd wordt dat de data niet aangepast kan worden. En het geeft een verhoogde veiligheid op het gebied van authenticatie. Het identificeren en valideren van een gebruiker wordt gedaan door middel van publieke en privé sleutels.

C# package: System.Security.Cryptography

6.2. Hashing

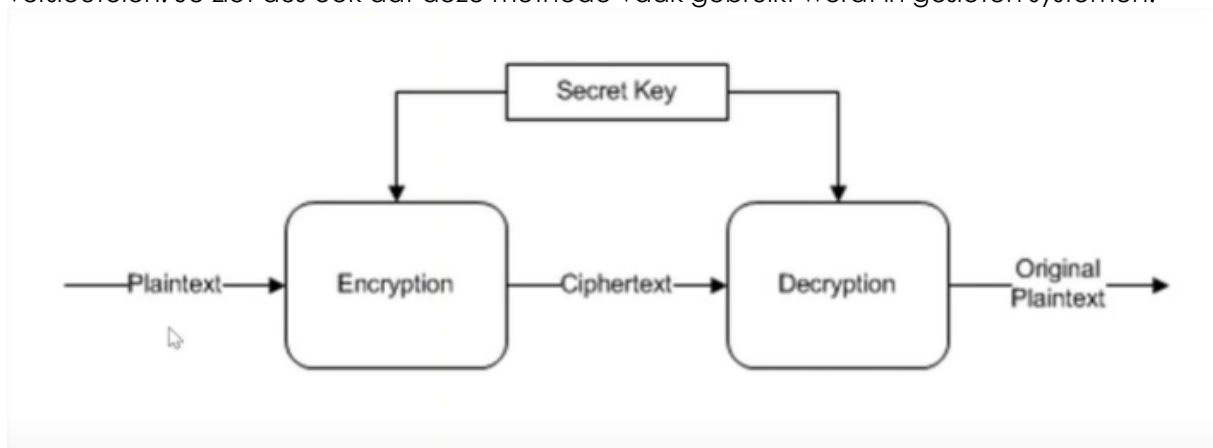
Hashen is een eenrichtingsfunctie. Het is zo heel makkelijk om gegevens te versleutelen maar maakt het heel moeilijk om deze gegevens te kraken zonder de juiste sleutels.

Een hash-functie is het omzetten van een variabele lengte naar een vaste lengte. Deze functie genereert een zogeheten 'data vingerafdruk (digest)'. Deze vingerafdruk mag door iedereen gezien worden maar er kan niet meer geknoeid worden.

6.3. Encryptie & Decryptie

6.3.1. Symmetric Algorithms

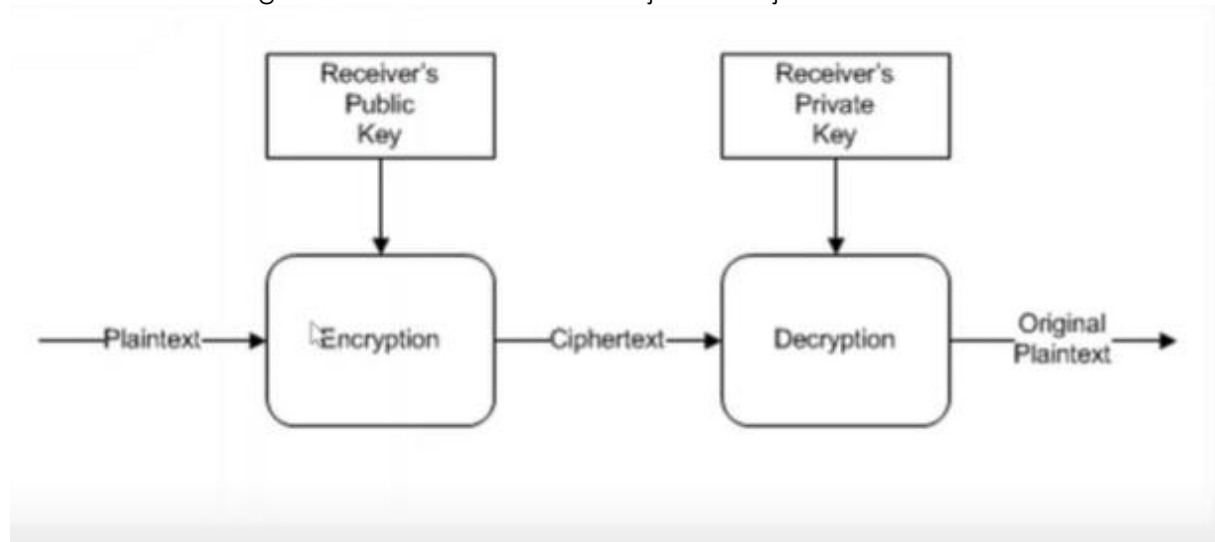
Bij symmetrische algoritmes gebruikt de codering en decodering van een bestand dezelfde sleutel. De meest voorkomende aanval om deze encryptie te ontcijferen is door het 'brute forcen' van de sleutel. Het nadeel van deze methode is dat het moeilijk is om de sleutel uit te geven aan derde partijen. Deze partijen kunnen dan door middel van de sleutel zelf de data versleutelen. Je ziet dus ook dat deze methode vaak gebruikt wordt in gesloten systemen.



C# class: SymmetricAlgorithm, deze maakt bijvoorbeeld gebruik van AES en TripleDES om de bestanden te versleutelen.

6.3.2. Asymmetric algorithms

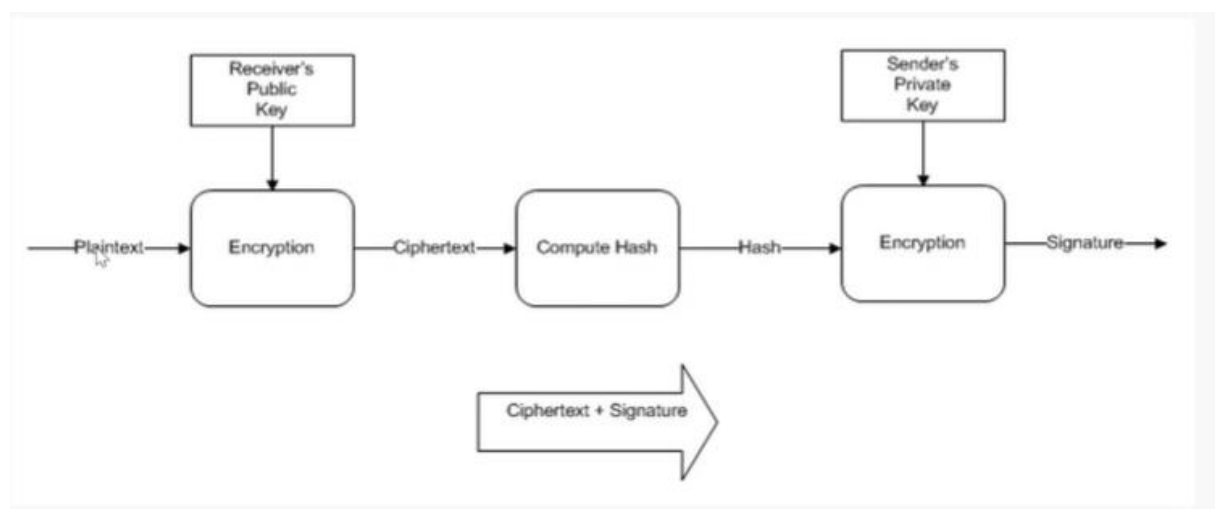
Bij Asymmetrische algoritmes worden er 2 sleutels gebruikt. Een publieke sleutel voor het coderen van het bestand en een prive sleutel voor het decoderen van het bestand. Het nadeel is dat deze methode tot wel duizend keer trager is dan symmetrische algoritmes. Je ziet dat deze methode vaak gebruik wordt voor het versleutelen van bestanden die via het internet gedownload worden. Degene die het bestand wil downloaden kan zo achterhalen dat het bestand dat wordt gedownload ook daadwerkelijk van de juiste afzender afkomt.



C# class: AsymmetricAlgorithm – RSA, DSA, ECDiffieHellman

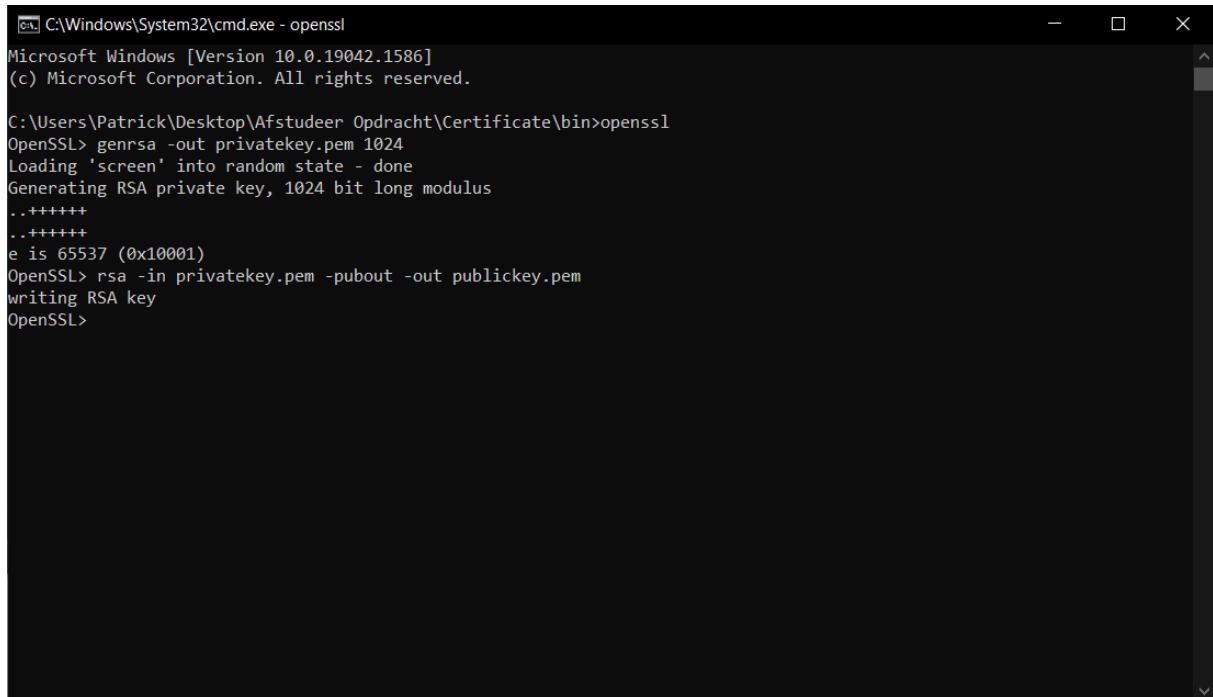
6.3.3. Digital signatures

Digitale handtekeningen zijn bedoeld om bestanden tussen twee systemen te versturen. De tekst dat verstuurd moet worden, wordt door een publieke sleutel versleuteld en genereert een digitale handtekening. De tekst is nog te lezen maar kan niet veranderd worden door de eindgebruiker. Zodra er aanpassingen zijn gemaakt in de tekst klopt de handtekening niet meer en kan de ontvanger hem niet meer uitlezen.



6.4. Creëren van een private en public key

Door gebruik te maken van OpenSSL kun je heel gemakkelijk publieke en privé sleutels aanmaken. Ik heb voor mijn demo gekozen voor een 1024bit RSA-sleutel. Je maakt eerst een privé sleutel aan, deze sleutel gebruik je daarna om een publieke sleutel te genereren. Zo zit je privé sleutel gekoppeld aan je publieke sleutel.



```
C:\Windows\System32\cmd.exe - openssl
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Patrick\Desktop\Afstudeer Opdracht\Certificate\bin>openssl
OpenSSL> genrsa -out privatekey.pem 1024
Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
..+++++
..+++++
e is 65537 (0x10001)
OpenSSL> rsa -in privatekey.pem -pubout -out publickey.pem
writing RSA key
OpenSSL>
```

6.5. Bronnen gebruikt voor dit onderzoek

Microsoft Visual Studio. (2019, 3 oktober). *Cryptography 101 with .NET Core*. YouTube.
<https://www.youtube.com/watch?v=rLEJLuA3hd0>

Generating keys using OpenSSL. (z.d.).
https://developers.yubico.com/PIV/Guides/Generating_keys_using_OpenSSL.html

6.6. PoC voor dit onderzoek

Ik heb voor dit onderzoek een [proof of concept](#) gemaakt om te kijken wat de werking is voordat ik het aan mijn eindproduct ga toevoegen.

7. Mailserver

Voor het maken van mijn applicatie heb ik een manier nodig om aan het eind van de maand de klanten een mail te sturen voor het betalen van hun factuur. Deze mail wordt uiteindelijk een melding van een automatische incasso maar voor mijn PoC wil ik de klanten een mail sturen dat ze moeten betalen.

7.1. Wat is een mailserver

Een mailserver, ook wel Mail Transfer Agent genoemd, is een server die dient voor de verwerking van e-mails. Dit wordt niet alleen gedaan voor internet e-mails maar is ook bedoeld voor interne emails. Een mailserver heeft twee taken. Het afleveren van een mail naar een computer, dit wordt ook wel de client genoemd en een mail afleveren bij een andere mailserver.

7.2. Soorten mailservers

1. **POP**; De POP server zorgt ervoor dat je jouw mail kunt ophalen bij een mailserver en opslaat op je eigen systeem. Deze mail wordt daarna verwijderd van de server.
2. **SMTP**; De SMTP server zorgt ervoor dat je mail verstuurd wordt naar het emailadres dat je hebt ingevoerd. De mail blijft dan op de computer staan onder 'verzonden e-mails'.
3. **IMAP**; De IMAP server lijkt heel veel op de POP server omdat je hier ook mails van de server ophaalt en opslaat op je eigen systeem. Het verschil is dat hier de mail blijft bestaan op de server.
4. **Exchange**; De exchange server is een mailserver waarbij de inkomende en uitgaande mails gesynchroniseerd worden en dat de agenda en adressenboek worden bijgehouden.

7.3. Advies

Ik heb gekozen om van in mijn applicatie gebruik te maken van een SMTP-mailserver. Het enige wat mijn applicatie moet doen, is naar de verschillen email adressen die geregistreerd staan in de database, een mail te sturen voor een betaalverzoek. Dit is precies de werking van een SMTP mailserver.

7.4. Bronnen gebruikt voor dit onderzoek

Gillis, A. S. (2021, 1 september). *mail server (mail transfer/transport agent, MTA, mail router, internet mailer)*. WhatIs.com. <https://www.techtarget.com/whatis/definition/mail-server-mail-transfer-transport-agent-MTA-mail-router-Internet-mailer>

7.5. PoC voor dit onderzoek

Ik heb voor dit onderzoek een [proof of concept](#) gemaakt om te kijken wat de werking is voordat ik het aan mijn eindproduct ga toevoegen.