THRESHOLD ENCRYPTION VOOR HET PROJECT KLUIS

JULIEN CATHALO

Dit document stelt het concept threshold encryption voor. Na een algemene omschrijving wordt uitgelegd hoe en waarom threshold encryption gebruikt wordt in de architectuur die de sectie Onderzoek van Smals voorgesteld heeft in het kader van het project "Kluis". Tot slot worden enkele voorbeelden gegeven van bestaande toepassingen die gebruik maken van threshold encryption.

In maart 2011 vroeg Smals bij het Europees octrooibureau een octrooi aan voor dit systeem.

1. Inleiding

In het kader van het project Kluis ontwierp de sectie Onderzoek van Smals een systeem waarmee men van op afstand gegevens op een sterk beveiligde manier kan opslaan, schrijven en consulteren in een database. Het systeem maakt het mogelijk het vertrouwen voor de verwerking van de gegevens te verdelen over twee (of meer) entiteiten. Door de gegevens te vercijferen en een beroep te doen op entiteiten met verschillende belangen, krijgen gebruikers een betere garantie qua confidentialiteit van de gegevens, want als één van die entiteiten toegang zou proberen te krijgen tot de gegevens, dan zal het systeem dat verhinderen. Dit systeem kan bijvoorbeeld gebruikt worden door actoren van de gezondheidszorg om de medische dossiers van patiënten te consulteren en bij te werken.

De betrokken partijen van het systeem zijn:

- Een actor (persoon die gegevens wenst te lezen of te schrijven)
- Twee servers voor gedeeltelijke ontcijfering (SDP1 en SDP2)
- Een databaseserver (BDD)

Wat de veiligheid betreft, is het zo dat de door de actor geraadpleegde en gewijzigde gegevens vertrouwelijk zijn; zij mogen dus niet consulteerbaar zijn voor onbevoegden. Zij moeten vercijferd opgeslagen worden in de database. Laten we ons het ergst mogelijke scenario voorstellen qua confidentialiteit: het systeem wordt aangevallen door een administrator (van één van de servers voor gedeeltelijke vercijfering) met slechte bedoelingen, die toegang heeft tot de fysieke gegevensdrager en over een ontcijferingssleutel beschikt. Het gebruikte vercijferingssysteem moet ons beschermen tegen dit scenario door te beletten dat een dergelijke aanvaller de gegevens zomaar kan lezen.

Wat de performantie betreft, moeten de gegevens snel verwerkt kunnen worden en moet de actor deze bewerkingen kunnen uitvoeren op een gewone pc.

Met een eenvoudig klassiek vercijferingssysteem kan dit probleem a priori niet worden opgelost zonder de architectuur veel ingewikkelder te maken en het veiligheidsniveau te verlagen. Gelukkig biedt een innoverende vercijferingsaanpak wel een oplossing. Hij steunt op een vercijferingsmethode die in 1989 werd voorgesteld door de onderzoekers Yvo Desmedt en Yair Frankel en die de naam "threshold encryption" kreeg. De architectuur die wij voorstellen gebruikt deze vercijferingsmethode om ervoor te zorgen dat:

- om het even welke actor makkelijk gegevens kan vercijferen;
- om het even welke actor de gegevens kan ontcijferen op voorwaarde dat de twee ontcijferingsservers samenwerken bij de ontcijfering;
- geen enkele server voor gedeeltelijke ontcijfering de gegevens *alleen* kan ontcijferen.

Dat betekent met andere woorden dat de gegevens en/of één van de servers voor gedeeltelijke ontcijfering gehost en beheerd kunnen worden door een entiteit waarin de actor geen absoluut vertrouwen heeft. Alleen een samenzwering tussen de verschillende servers voor gedeeltelijke ontcijfering zou de gegevens in gevaar kunnen brengen.

Samengevat is een systeem voor threshold encryption een systeem voor vercijfering met publieke sleutel waarbij de ontcijfering een samenwerking van verschillende entiteiten vereist. Verder in dit document zullen we dit concept in detail toelichten en zullen we het vergelijken met andere aanpakken.

2. Threshold Encryption

We kunnen een systeem voor threshold encryption zien als een klassiek vercijferingssysteem met publieke sleutel waarbij de private sleutel die dient voor de ontcijfering verdeeld zou worden over verschillende gebruikers met een systeem voor "secret sharing". Om te beginnen, zullen we dus uitleggen wat een systeem voor secret sharing en een vercijferingssysteem met publieke sleutel is.

2.1. Secret Sharing

We nemen het voorbeeld van een bank die uitgerust is met een kluis die geopend kan worden met een wachtwoord. Alleen de bankdirecteur kent het wachtwoord. Hij heeft vier bedienden en hij zou willen dat zij in geval van nood de kluis kunnen openen als hij afwezig is, maar onder één voorwaarde: er moeten minstens twee bedienden aanwezig zijn (om te vermijden dat één oneerlijke bediende de inhoud van de kluis zou kunnen stelen). De directeur zoekt dus een manier om zijn wachtwoord in vier stukken te "verdelen" (hij zal aan elke bediende een stuk geven) zodat om het even welke groep van twee bedienden zijn twee stukjes zou kunnen samenleggen om het wachtwoord te bekomen.

Een eerste idee zou zijn om de letters van het wachtwoord te verdelen onder zijn bedienden. Als het wachtwoord bijvoorbeeld "azertyui" is, dan kan hij "azerty__" aan de eerste bediende geven, "azer__ui" aan de tweede, "az__tyui" aan de derde en "__ertyui" aan de laatste. Op het eerste gezicht lijkt het alsof deze oplossing werkt, want twee bedienden kunnen effectief het wachtwoord vinden: met bijvoorbeeld "azerty__" en "az__tyui" vindt men "azertyui". Maar het nadeel is dat er te veel informatie over het wachtwoord wordt onthuld aan elke bediende. Eén bediende kan immers in zijn eentje alle combinaties van twee letters testen op zijn share om het geheim te ontdekken. Een dergelijk systeem gebruiken is dus geen goed idee, het biedt geen goed veiligheidsniveau aan.

De oplossing voor het probleem van de directeur heet "secret sharing".

Met een systeem voor "secret sharing" is het mogelijk om op basis van een geheime waarde (bijvoorbeeld een wachtwoord) stukken of "shares" van een geheim gegeven zodanig aan te maken dat een vastgelegd aantal shares nodig is om het geheim te onthullen.

Een dergelijk systeem heeft meer bepaald twee parameters:

- n, het totale aantal shares;
- t, de drempel, het minimale aantal shares dat nodig is om het geheim te onthullen.

Op basis van het geheim S worden n shares gegenereerd en verdeeld onder n gebruikers. Wanneer t gebruikers hun shares samenleggen, kunnen zij het geheim S onthullen. Maar als er minder dan t gebruikers aanwezig zijn, dan kunnen zij het niet vinden. Eigenlijk gaat het verder dan dat: als t-1 gebruikers hun shares samenleggen, dan kunnen zij niet de minste informatie over het geheim raden (en vermijdt men dus het probleem uit het voorbeeld van de directeur die een aantal letters van zijn wachtwoord verdeelde).

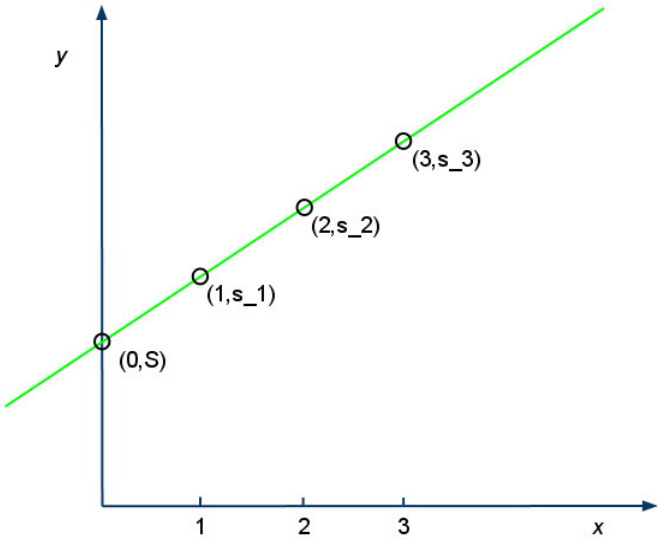
Het is ook mogelijk om gewichten toe te kennen aan de gebruikers door aan bepaalde gebruikers verschillende shares toe te vertrouwen.

Er bestaan talrijke systemen voor secret sharing. Wij citeren de eerste twee die uitgevonden werden (alle twee in 1979, los van elkaar): het schema van Shamir en dat van Blakley. We bespreken hieronder het schema van Shamir in detail omdat het één van de makkelijkst te begrijpen systemen is.

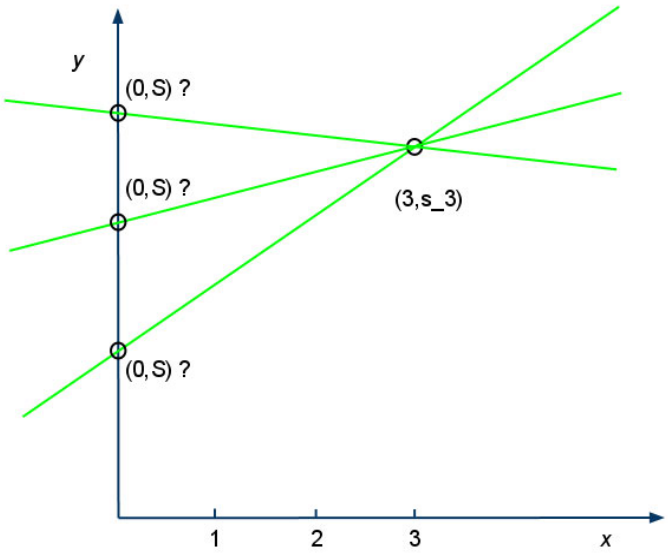
Dat schema is gebaseerd op veeltermen. Om te beginnen, leggen we het eenvoudige geval uit waarbij t = 2. Het aantal gebruikers n kan om het even welke waarde aannemen die gelijk is aan 2 of groter is dan 2. We maken gebruik van het feit dat:

- door 2 punten van het schema slechts één rechte loopt;
- door 1 punt van het schema oneindig veel rechten lopen.

De entiteit die de shares moet genereren (in ons voorbeeld de directeur), gaat dus een rechte aanmaken (met de vergelijking y = ax + S, waarbij a willekeurig wordt gekozen en S het geheim is) en gaat aan elke gebruiker de coördinaten van een punt van de rechte meedelen. Die coördinaten zullen de share van de gebruiker zijn. Uiteraard krijgt niemand de coördinaten van het punt (0,S) aangezien dit het geheim S onthult! En het is ook noodzakelijk dat alle gebruikers verschillende punten krijgen.

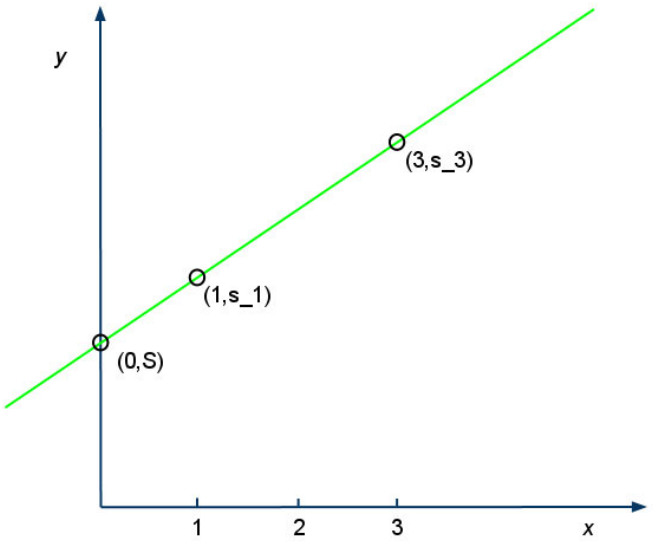


In dit voorbeeld geeft men aan gebruiker 1 de coördinaten (1,s_1), aan gebruiker 2 de coördinaten (2_s_2) en aan gebruiker 3 de coördinaten (3,s_3). Geen enkele gebruiker kan in zijn eentje de rechte bepalen. Het punt gekend door gebruiker 3 komt bijvoorbeeld overeen met een oneindig aantal mogelijke rechten, dus met een oneindig aantal mogelijke waarden voor S:



Maar zodra twee gebruikers (om het even dewelke) hun punten samenleggen, kunnen zij met een beetje rekenwerk de enige rechte vinden die door deze twee

punten loopt en dus ook de waarde van het geheim S. Als de gebruikers 1 en 3 bijvoorbeeld hun punten samenleggen, kunnen zij de rechte terugvinden:



Zo werkt het systeem dus met een drempel t = 2.

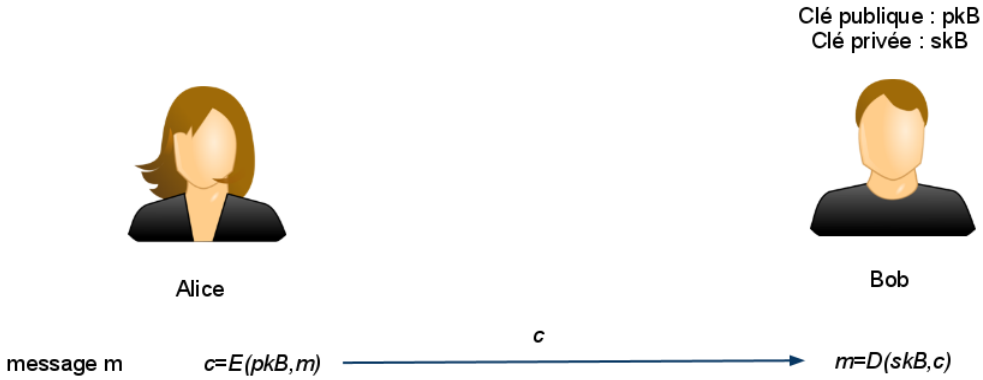
In het geval van t > 2 gebruikt men veeltermen met graad t-1 maar het toegepaste principe is hetzelfde: men genereert een veelterm met graad t-1 en elke share is de waarde van deze veelterm op een verschillend punt. Als t gebruikers hun geheimen samenleggen, vinden zij de veelterm terug (via een interpolatie van Lagrange), maar als slechts t-1 of minder gebruikers dat doen, dan zullen zij niet de minste informatie over het geheim vinden.

Andere ideeën kunnen gebruikt worden. Het schema van Blakley gebruikt bijvoorbeeld snijpunten.

Opmerking: secret sharing mag niet verward worden met een ander concept uit de cryptografie, de "key exchange" (of sleuteluitwisseling). Een systeem voor secret sharing biedt aan één persoon de mogelijkheid om een geheim gegeven te nemen en dit op te splitsen in verschillende shares. Een systeem voor key exchange (waarvan dat van Diffie-Hellman het bekendste is) biedt aan twee personen de mogelijkheid om samen een geheim gegeven tot stand te brengen.

2.2. Public Key Encryption

Met een klassiek systeem voor vercijfering met publieke sleutel of asymmetrische vercijfering kan Alice een bericht vercijferen dat bestemd is voor Bob door gewoon de publieke sleutel van Bob te gebruiken (publiek gegeven dat iedereen kent). Bob zal het bericht kunnen ontcijferen aan de hand van de bijbehorende private sleutel (geheim gegeven dat alleen Bob kent).



Afbeelding 1

Op deze figuur vercijfert Alice het bericht m met de publieke sleutel van Bob pkB; zij bekomt een vercijferd bericht c dat ze naar Bob verstuurt. Bob ontcijfert c aan de hand van zijn private sleutel skB.

2.3. Secret Sharing + Public Key Encryption = Threshold Encryption

Een systeem voor threshold encryption (concept voorgesteld in 1989 door Yvo Desmedt en Yair Frankel) is in zekere zin een kruising tussen een systeem voor klassieke vercijfering met publieke sleutel en een systeem voor secret sharing.

Met een systeem voor threshold encryption is er altijd slechts één vercijferingssleutel; die komt overeen met een groep gebruikers en elke gebruiker bezit zijn eigen ontcijferingssleutel. Bij de start definieert men twee parameters t en n zoals in een systeem voor secret sharing: n is het totale aantal gebruikers en t is de drempel.

Het idee is dat ten minste t gebruikers moeten samenwerken om een bericht bestemd voor de groep te ontcijferen. Om het even welke groep van t gebruikers kan gaan samenzitten om een bericht te ontcijferen. Als er minder gebruikers meewerken, dan zullen zij echter niet in staat zijn het bericht terug te vinden.

In dit systeem zijn de parameters de volgende:

- Het totale aantal gebruikers: n
- De drempel: t
- Een publieke sleutel: pk
- Een private sleutel per gebruiker: sk_1, sk_2, …, sk_n

Om het even wie kan een bericht m vercijferen met de publieke sleutel pk:

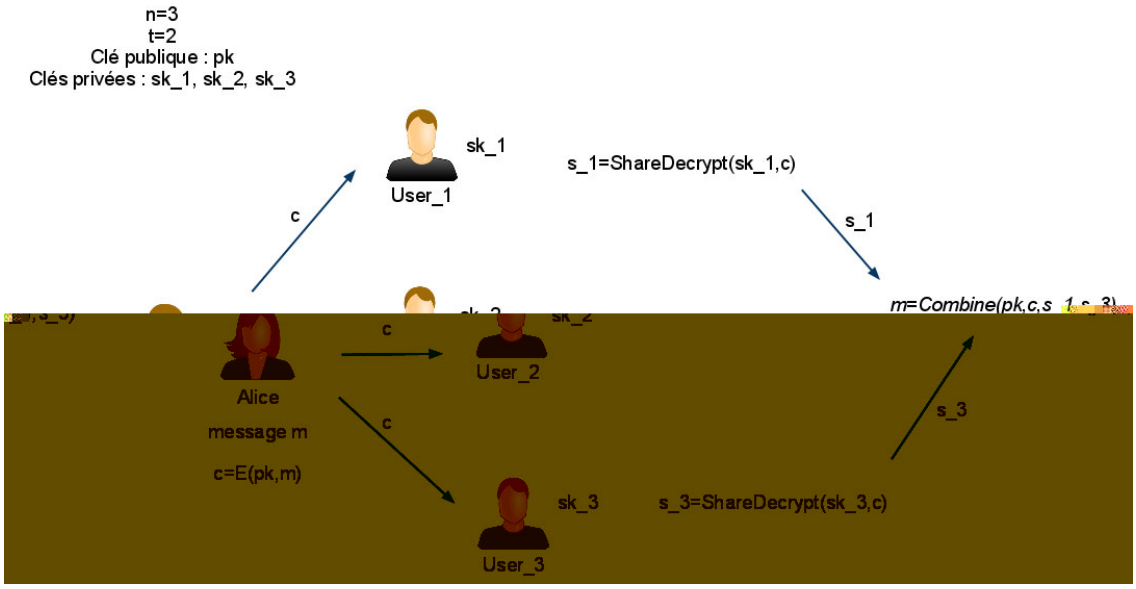
c = E(pk,m)

Een gebruiker i kan een "ontcijferingsshare" berekenen aan de hand van het vercijferde bericht c en zijn eigen private sleutel sk_i.

s_i = ShareDecrypt(sk_i,c)

Men kan t shares van de ontcijferingen s_j1, …, s_jt combineren om het bericht m opnieuw samen te stellen:

m=Combine(pk,c,s_j1,…,s_jt)



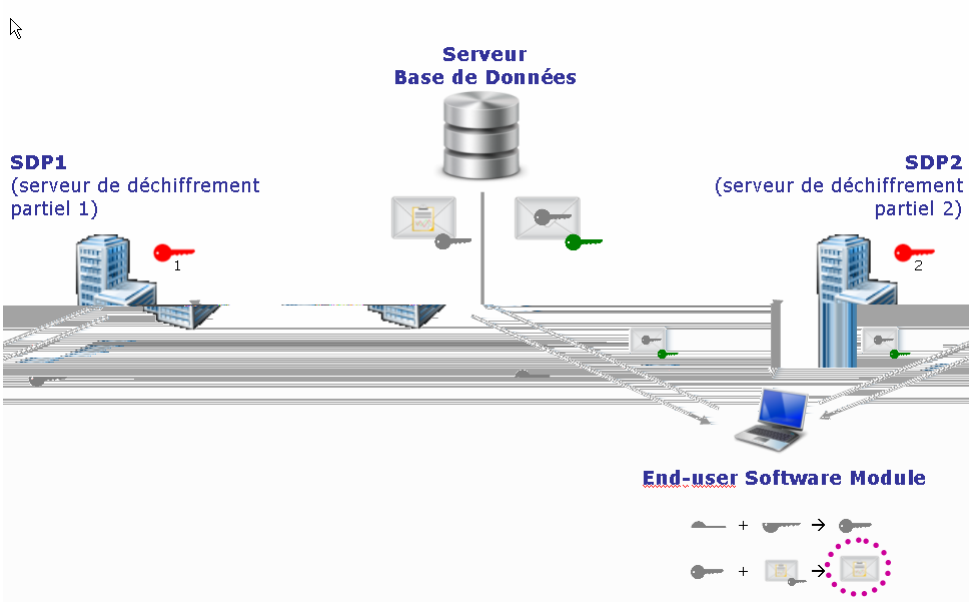
Afbeelding 2

In dit voorbeeld met 3 gebruikers en een drempel 2 vercijfert Alice een bericht m met de publieke sleutel pk en stuurt het vercijferde bericht c naar de gebruikers. De gebruikers 1 en 3 berekenen de ontcijferingsshares, respectievelijk s_1 en s_3 (hier werkt gebruiker 2 niet mee). Met de twee shares s_1 en s_3 kan het bericht opnieuw samengesteld worden in leesbare tekst. Met s_1 alleen of s_3 alleen is het onmogelijk om ook maar de minste informatie over het bericht m terug te vinden. Hier zijn het de gebruikers 1 en 3 die meewerken maar elke combinatie van ten minste twee gebruikers maakt het mogelijk het bericht terug te vinden.

3. Threshold Encryption in het kader van de Kluis

3.1. Werking

In ons voorbeeld gebruiken we een eenvoudige versie van threshold encryption waarbij t = n = 2. Maar we kunnen uiteraard servers voor gedeeltelijke ontcijfering toevoegen (dus n verhogen) en de drempel verhogen (dus t verhogen, op voorwaarde dat t kleiner is dan n of gelijk is aan n). We hebben dus een publieke sleutel pk waarmee twee private sleutels sk_1 en sk_2 overeenkomen. De server voor gedeeltelijke ontcijfering 1 (SDP1) bezit sk_1 en SDP 2 bezit sk_2. De databaseserver BDD bewaart de vercijferde gegevens.



3.2. Voordelen en vergelijking met andere aanpakken

De oplossing zorgt ervoor:

- dat om het even welke actor makkelijk gegevens kan vercijferen (hij gebruikt daarvoor de publieke sleutel pk);
- dat om het even welke actor gegevens kan ontcijferen op voorwaarde dat elke server voor gedeeltelijke ontcijfering meewerkt aan de ontcijfering;
- dat geen enkele server voor gedeeltelijke ontcijfering *alleen* de gegevens kan ontcijferen (dit wordt gegarandeerd door de eigenschappen van het gebruikte schema voor threshold encryption).

Het is zeer belangrijk op te merken dat de gegevens zelf nooit naar de SDP1 en SDP2 gestuurd worden. **Zij hebben op geen enkel moment toegang tot de gegevens**, zelfs niet in vercijferde vorm. Deze servers nemen alleen deel aan de ontcijfering van de sleutel die de gegevens beschermt.

Bovendien moet de actor geen geheim opslaan: hij ontcijfert het bericht alleen dankzij de ontcijferingsshares die de SDP's versturen. Dit is wenselijk op het vlak van de veiligheid omdat de klant a priori een eenvoudige pc zal gebruiken.

We kunnen deze oplossing vergelijken met andere mogelijke aanpakken:

Systeem met alleen een symmetrische sleutel:

We zouden een systeem kunnen bedenken dat alleen symmetrische sleutels gebruikt. De actor zou de gegevens vercijferen met een symmetrische sessiesleutel voor elk veld; vervolgens zou die symmetrische sleutel op gedeelde wijze opgeslagen worden (in de zin van "secret sharing") door de SDP's. Deze oplossing zou een bewaring van sleutels op het niveau van de SDP's vergen en zou de interacties sterk bemoeilijken, onder meer bij de vercijfering.

Systeem met asymmetrische vercijfering, private sleutel toevertrouwd aan een server:

Men zou ook aan de volgende oplossing kunnen denken: de actor vercijfert de gegevens gewoon met de publieke sleutel van een server en zij worden dan opgeslagen in de database. Wanneer de actor een gegeven wil lezen, stuurt de database hem het vercijferde gegeven en de actor stuurt het door

naar de server die het ontcijfert. Deze oplossing is eenvoudig en performant maar de kwetsbaarheid is te groot: als de database in handen valt van een server-administrator met slechte bedoelingen, dan kan hij alle gegevens lezen.

We zien dus dat de voorgestelde oplossing met threshold encryption talrijke voordelen biedt ten opzichte van de alternatieven.

4. Real stories: reële gevallen waarbij threshold encryption en secret sharing gebruikt wordt

Zoals reeds uitgelegd, steunt een algoritme voor threshold encryption op het idee om de private ontcijferingssleutel te verdelen via een algoritme voor secret sharing. Deze concepten dateren van eind de jaren 70 en er werden ondertussen honderden wetenschappelijke publicaties aan gewijd. Er zijn op de markt verschillende producten beschikbaar die deze concepten gebruiken om gegevens en sleutels te beschermen. In sommige gevallen wordt een symmetrische sleutel voor vercijfering/ontcijfering gebruikt via een systeem voor secret sharing. De volgende lijst somt enkele reële gevallen op waarbij secret sharing en threshold encryption gebruikt worden; hij dient als voorbeeld en is niet volledig.

4.1. Gebruik van secret sharing in de HSM's

Een HSM (Hardware Security Module) is een cryptoprocessor voor de uitvoering van cryptografische bewerkingen. Een HSM heeft meer bepaald drie belangrijke functies:

- Sleutels opslaan op een beveiligde manier, waarbij weerstand wordt geboden bij software- en hardwareaanvallen.
- Op performante wijze cryptografische bewerkingen uitvoeren (vercijfering, ontcijfering, handtekening, enz.) en de toepassingsserver dus ontlasten van deze taken die (in het geval van asymmetrische vercijfering) te veel rekentijd zouden vergen.
- De toegang tot de sleutels controleren.

4.1.1. Sophos Utimaco[1]

Het bedrijf Utimaco (eigendom van Sophos sinds 2008) biedt een gamma van HSM's aan onder de naam SafeGuard CryptoServer. Die producten gebruiken het systeem voor secret sharing van Shamir (gebaseerd op veeltermen en hierboven toegelicht) om de back-up van de master key te beschermen. Zo wordt de master key niet opgeslagen op één enkele drager maar verdeeld over n back-updragers die bewaard worden door n administrators; er zijn t administrators nodig om de sleutel opnieuw samen te stellen. Dat systeem biedt dus een goed compromis tussen:

- De noodzaak om gegevensverlies te vermijden: de master key mag in geen geval verloren gaan (als het een ontcijferingssleutel is, zouden de gegevens dan verloren zijn).
- De veiligheid: geen enkele administrator kan in zijn eentje de master key terugvinden (en zelfs een verbond van t-1 administrators kan hem niet vinden). De vertrouwelijkheid van de gegevens is dus verzekerd.

[1] http://hsm.utimaco.com/products/hsm-series/

4.1.2. SafeNet[2]

Secret sharing wordt ook gebruikt in bepaalde HSM's van het merk SafeNet, bijvoorbeeld in het product Luna CA4. Deze HSM is specifiek voor PKI's en meer bepaald voor de rol van CA (Certification Authority). In een PKI moet de CA certificaten uitgeven en dus handtekeningen genereren aan de hand van een private sleutel ("PKI root key" genaamd). Deze sleutel is bijzonder gevoelig in die zin dat de compromittering ervan de veiligheid van de volledige PKI in gevaar zou brengen: een aanvaller die over deze sleutel beschikt, zou valse certificaten kunnen genereren.

Luna CA4 vercijfert deze sleutel met het 3DES-algoritme en de 3DES-sleutel wordt verdeeld onder verschillende administrators aan de hand van een systeem voor secret sharing. Deze bescherming lijkt dus op die van Utimaco en biedt dezelfde voordelen (compromis tussen de noodzaak om de root key niet te verliezen en de veiligheid).

4.2. Gebruik van secret sharing in een open source PKI-software

De open source PKI-software OpenXPKI[3] gebruikt secret sharing voor dezelfde toepassing als het product SafeNet Luna CA4. De private sleutels van de root CA en de tussenliggende CA's worden beschermd door middel van het systeem voor secret sharing van Shamir.

4.3. Gebruik van secret sharing voor een veilingsysteem in Denemarken

Secret sharing is ook een basiscomponent van het systeem voor "secure multi-party computation" dat op grote schaal getest werd in Denemarken voor een complex veilingsysteem[4].

In Denemarken koopt één enkele onderneming suikerbieten op bij de producenten. De producenten hebben met die onderneming contracten voor de levering van bieten maar die contracten zijn uitwisselbaar. Om het proces voor de uitwisseling van die contracten te automatiseren, was een oplossing nodig om een correcte uitwisselingsprijs te bepalen in functie van de vraag en het aanbod. Het gaat meer bepaald om de "Market Clearing Price", de prijs waarvoor er evenveel contracten te koop worden aangeboden als er gevraagd worden. Maar dit stelde problemen op het vlak van de confidentialiteit (want de prijs waarvoor een landbouwer akkoord gaat om zijn contracten te verkopen, onthult vertrouwelijke informatie over zijn economische situatie en zijn productiviteit). In januari 2008 heeft een team van onderzoekers uit Denemarken en Nederland een systeem uitgewerkt om dit probleem aan te pakken. Het systeem is gebaseerd op een protocol voor secure multi-party computation. De verschillende deelnemers plaatsen hun aangeboden en gevraagde contracten via een webserver en de

[2] http://www.safenet-inc.com/products/data-protection/hardware-security-modules-hsms/

[3] http://www.openxpki.org/

[4] "Secure Multiparty Computation Goes Live": http://eprint.iacr.org/2008/068.pdf

server stuurt dan de correcte prijs terug. Tijdens het experiment veranderde 25.000 ton bieten op die manier van eigenaar.

4.4. Gebruik van threshold encryption voor elektronische stemsystemen

Het principe van threshold encryption wordt gebruikt in elektronische stemsystemen om de confidentialiteit van de stemming te garanderen. In een elektronisch stemsysteem wordt elk stembiljet vercijferd met een publieke sleutel. Verschillende vertrouwde derden (ook "trustees" genoemd) komen tussen in de ontcijfering maar geen van hen mag de inhoud van het stembiljet te zien krijgen. Dat principe werd geïmplementeerd in het stemsysteem "Helios Voting[5], ontwikkeld door Ben Adida, een professor van Harvard. In dit systeem is het algoritme "Threshold ElGamal Encryption" geïmplementeerd voor de vercijfering van de stembiljetten. Het systeem gebruikt t=3 en n=3 met 6 trustees. Er worden dus drie ontcijferingssleutels gebruikt en elke sleutel is gekend door twee verschillende trustees.

Dit systeem werd al gebruikt bij diverse verkiezingen, onder meer in België voor de verkiezing van de rector van de Université catholique de Louvain in 2009[6].

4.5. Andere voorbeelden

Secret sharing wordt ook gebruikt in:

- de PKI-producten van Verisign;
- het product Keon Desktop van RSA Security;
- de producten van Atos Wordline.

Tot slot is secret sharing ook geïmplementeerd in verschillende library's, bijvoorbeeld:

- Shamir Secret Sharing in Java

  http://sourceforge.net/projects/secretsharejava/

- ssss

  http://freshmeat.net/projects/ssss/

[5] http://heliosvoting.org

[6] http://www.uclouvain.be/270428.html