Dankwoord

Een thesis is het resultaat van een jaar noeste arbeid, en je laat je altijd weer adviseren door een heleboel mensen, zonder wiens steun je de opdracht niet tot een goed einde had kunnen brengen.

Vooreerst waren er onze begeleiders, Karel Wouters en Danny De Cock. In hun e-mails die ons met lichtsnelheid bereikten, wisten zij ons steeds opnieuw te verrassen met een alternatieve kijk op de zaken. Ze waren er ook vooral wanneer we geen licht meer zagen aan het eind van de tunnel, om ons een duwtje in de juiste richting te geven. Uiteraard was onze thesis ook nooit mogelijk geweest zonder onze promotoren, Prof. Bart Preneel en Prof. Joos Vandewalle. Verder mag William Ahern, de bezieler van AnonNet, natuurlijk niet ontbreken in dit lijstje. Zijn project is voor ons het cruciale startpunt geweest, en hij heeft ook steeds de tijd genomen om constructieve kritiek te geven op onze ideeën en nieuwe elementen aan te bieden.

Last but not least, volgen natuurlijk onze vrienden en familie. Nico is ervan overtuigd dat de ESAT-bende in Leuven en de Chiro-troep er steeds voor gezorgd hebben dat er ontspanning en vertier was bij momenten van oververhitting. Bij jullie kon ik steeds terecht om wat stoom af te laten! Jo en Jan, ik hoop dat jullie maar goed beseffen dat jullie schuldig zijn aan mijn keuze voor dit thesisonderwerp. Bedankt iedereen, na mijn jaartje in Grenoble weet ik dat dit niet het einde betekent van onze kameraadschap, want uit het oog bleek voor jullie niet uit het hart! De momenten thuis, bij mama, papa en de zusjes, waren de beste manier om even tussen de bits en de bytes uit te vluchten en te ontsnappen aan de valserikken op het internet. Het is ongelooflijk hoe snel je je zinnen verzet, als je aan je jongere zusje gewoon eens mag uitleggen hoe een integraal in elkaar zit, of als je aan de weekendtafel gewoon kan luisteren naar ieders belevenissen en avonturen van de week. Dries, na ons avontuur in tweede kan, kenden we weer een explosief jaartje samenwerking. Hevige maar gezonde discussies alom, daarom ben ik zeker dat onze vriendschap niet meer stuk kan!

Dries heeft de voorbije vijf jaar ook heel wat plezier beleefd met zijn Leuvense vrienden. De etentjes in de Alma, de nachtelijke kebaps en frietjes die samen verorberd werden en de pintjes in de fakbar zullen volgend jaar serieus gemist worden. Bedankt Molle, Stefaan en co. Daarnaast zijn er natuurlijk ook nog de vrienden van de scouts die in het weekend voor de nodige ontspanning zorgen. Dan komt deze nerd van achter zijn computers vandaan. De Vosselaarse nacht zijn lang! Bedankt Jef, Stijn, Pieter, Tim en de rest van de bende. Natuurlijk ben ik ook een dankjewel verschuldigd aan mijn ouders. Zonder ons ma zou er in het weekend van werken niet veel in huis gekomen zijn en zouden er heel wat uurtjes meer in mijn bed zijn doorgebracht. De interesse die onze pa toonde, deed ook echt wel deugd. Nico, bedankt dat je het werk op je genomen hebt, toen ik enkele weken voor de eindmeet een tijdje in het ziekenhuis opgenomen werd. Als laatste zou ik mijn vriendin Sofie willen bedanken voor de fijne tijd die wij samen hebben en hopelijk nog vele jaren zullen hebben.

Bedankt allemaal, zonder jullie was deze thesis nooit geworden wat ze nu is!

Nico en Dries

Samenvatting

In deze tekst wordt een uitgebreide analyse aangebracht van de beveiligingsaspecten in peer-to-peer systemen. De nadruk wordt gelegd op bestandsuitwisseling, maar de meeste bestudeerde onderwerpen gelden ook voor andere peer-to-peer systemen. Beveiligde en redundante opslag van informatie, beveiligde communicatie tussen peers, betrouwbaarheid van peers en anonimiteit van de gebruikers worden bestudeerd. De bestudeerde technieken worden eerst theoretisch geanalyseerd, maar telkens ook toegepast op bestaande peer-to-peer systemen.

Het anonimiteitsaspect wordt aandachtig bestudeerd. Eerst wordt een algemene analyse gegeven van de anonimiteit bij informatie-uitwisseling in peer-to-peer systemen. De informatie-uitwisseling kan via een routeringsalgoritme gebeuren of via een communicatiekanaal. Vervolgens wordt het concept van anonieme communicatiekanalen in detail bekeken. Er worden bouwblokken geïntroduceerd om anonieme kanalen te realiseren. Deze bouwblokken kunnen weerstand bieden tegen twee duidelijk onderscheidbare bedreigingsmodellen: passieve verkeersanalyse door een krachtige globale afluisteraar of actieve samenwerking van valse peers. Na de analyse van de bouwblokken wordt besproken welke bouwblokken gebruikt worden in een aantal concrete protocols voor anonieme kanalen.

We ontwerpen AnonNet, een peer-to-peer systeem dat anonieme kanalen implementeert. Het is een algemeen anonimizeringssysteem dat gebruikt kan worden om internetverkeer anoniem te maken voor de gebruiker. De bestudeerde peer-to-peer technieken worden gecombineerd in het ontwerp, zodat de oplossing een gedecentraliseerde structuur meegegeven wordt. Bovendien is AnonNet transparant voor de toepassing: het kan de gebruikte toepassing zonder aanpassing anonimiteit bieden. AnonNet biedt sterke weerstand tegen verkeersanalyse.

Inhoudsopgave

Hoofdstuk 1

Algemene inleiding

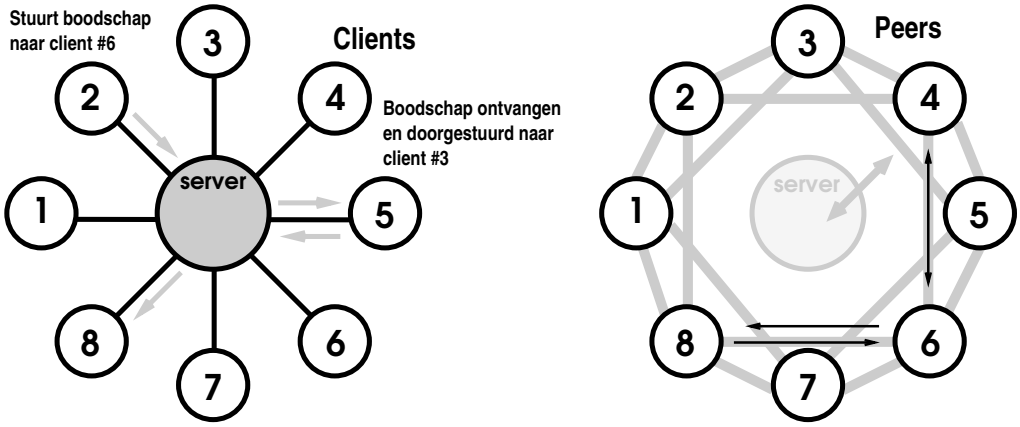
Evolutie van client-server naar peer-to-peer Tot voor kort was het grootste deel van het internetverkeer gericht op de informatie-afnemer. Dit noemt men de client-server architectuur van het huidige internet. Servers hebben onderling snelle verbindingen, en de eindgebruikers verbinden zich via trage computers met de servers. De beperkte kracht van de computers liet de gebruikers enkel toe informatie af te nemen. De explosieve groei in rekenkracht bij de eindgebruiker heeft ertoe geleid dat ze directe verbindingen tussen elkaar tot stand kunnen brengen. Deze evolutie leidt tot een verhoogde interactiviteit tussen de eindgebruikers. Men gaat over van zuivere informatieconsument naar informatieproducent. Enkele voorbeelden van peer-to-peer toepassingen zijn: informatieopslag en -uitwisseling (*file sharing*), gedistribueerde berekeningen, *messaging* en informatieverspreiding of -publicatie.

Het peer-to-peer netwerkmodel gaat uit van deze interactie tussen de gebruikers. Alle spelers in het netwerk zijn evenwaardig en bevinden zich in een 'wolk' van medegebruikers. Gebruikers kunnen op elk ogenblik toetreden tot deze wolk, of eruit verdwijnen. Bovendien biedt de peer-to-peer architectuur een oplossing om de krachten te bundelen zodat de rekenkracht aan de rand van het internet nuttig gebruikt wordt. Uiteraard wordt de bestaande – gecentraliseerde – internetinfrastructuur nog steeds gebruikt om het peer-to-peer model tot stand te brengen. Het is hier echter eerder een technisch middel, het virtuele model (peer-to-peer) komt niet meer overeen met de fysische implementatie.

Het verschil tussen de client-server architectuur en het peer-to-peer model wordt geïllustreerd in figuur 1.1. In het client-server model gaat alle communicatie tussen twee gebruikers langs een centrale server. In het peer-to-peer model wisselen de *peers* onderling data uit via rechtstreekse verbindingen. Sommige peer-to-peer systemen gebruiken soms ook servers. Er wordt vaak een onderscheid gemaakt tussen 'ware' peer-to-peer en 'hybride' peer-to-peer. 'Ware' peer-to-peer verwijst naar een model waar alle deelnemende computers peers zijn en er geen sprake is van een centrale server, die de informatie-uitwisseling coördineert, controleert of beheert. 'Hybride' peer-to-peer vertrouwt op een centrale server om enkele van die functies te verwezenlijken.

Vier pijlers Grofweg steunt het peer-to-peer principe op vier pijlers: *decentralisatie*, verhoogde graad van *samenwerking*, diensten aan de *rand van het netwerk*, en grote *dynamiek* van de netwerkinfrastructuur.

Peer-to-peer is niet nieuw In de startdagen van het internet waren er enkel nog maar servers. In die dagen werden de pijlers van onze internettoepassingen uitgedacht, en ze hadden een sterk peer-to-peer karakter. De sterk hiërarchische modellen van Usenet en DNS zijn hier getuige



(a) Client-server.

(b) Peer-to-peer. Centrale server enkel in het hybride peer-to-peer model.

Figuur 1.1: Client-server versus peer-to-peer

van. Als we naar de huidige peer-to-peer toepassingen kijken, zien we dat er getracht wordt de beste elementen uit beide structuren te combineren. Kijk maar naar Napster: zijn gecentraliseerde zoekalgoritmen passen zeker niet in een peer-to-peer kader, maar het behaalde effect is zonder twijfel aan niemand voorbij gegaan.

Nieuwe uitdagingen en hindernissen Het nieuwe netwerkmodel brengt nieuwe uitdagingen en hindernissen met zich mee, zowel op technisch als op maatschappelijk gebied.

Technische ingrepen Vooreerst kunnen we niet rekenen op een symmetrische bandbreedteverdeling tussen de peers van het netwerk. Het gebruik van ADSL- en kabelmodems resulteert in een grotendeels asymmetrische bandbreedteverdeling aan de eindpunten. Bovendien stapt men over van statische naar dynamische netwerkadressen, tenminste tot een nieuwe adressenruimte (IPv6) beschikbaar wordt, die voldoende plaats biedt aan het exponentieel stijgend aantal componenten.

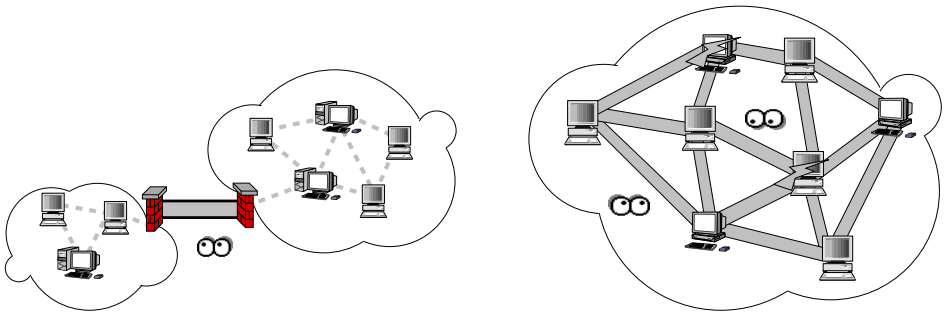
Maatschappelijke impact Daarnaast mogen we de sociologische impact niet uit het oog verliezen. Er is nood aan een systeem dat onze sociale standaarden in ere houdt. De basisprincipes waarop we deze standaarden bouwen, moeten we voorzien in het model. In sommige situaties is het nodig mensen verantwoordelijk te stellen voor hun acties. Het is hier dat bijvoorbeeld Usenet gefaald heeft. Bovendien willen we antisociaal gedrag ontmoedigen doordat het systeem personen kan identificeren, desnoods aan de hand van pseudoniemen, om de privacy te behouden.

Tegenover de bovenstaande vraag naar authentisering, vertrouwen en verantwoordelijkheid staan begrippen als privacy en anonimiteit. Op het eerste zicht lijken deze begrippen elkaar tegen te spreken. Toch hoeven ze elkaar niet uit te schakelen, het zal nodig zijn een compromis te vinden in de graad van privacy en anonimiteit enerzijds, en authentisering anderzijds. Al is de gedachte van bescherming van informatie niet nieuw, toch moeten nieuwe strategieën en protocols uitgedacht worden om te voldoen aan de structuur van peer-to-peer netwerken.

Voorbeeld In wat volgt bekijken we het voorbeeld van *groupware* toepassingen. Deze toepassingen helpen een groep collega's in een lokaal netwerk bij de organisatie van hun activiteiten en het uitwisselen van informatie.

Figuur 1.2(a) toont een voorbeeld van de huidige situatie. Nu draaien de groupware toepassingen lokaal, zonder fysische verbinding met het internet, of in een beveiligde zone achter een beveiligde server. Geografisch uiteengelegen gemeenschappen kunnen eventueel met een vaste, beveiligde verbinding met elkaar communiceren.

In figuur 1.2(b) wordt een peer-to-peer structuur voorgesteld. Er is nood aan een meer transparante aanpak waarbij we niet allemaal verbinding moeten maken met een zelfde, beveiligde server, maar waarbij we zelf deel uitmaken van de tussenliggende verbindingen tussen de verschillende gebruikers, en waarbij de informatie niet meer centraal geplaatst is. Op deze manier wordt het bijvoorbeeld mogelijk vertrouwelijke documenten lokaal bij te houden en toch beschikbaar te stellen voor collega's. De onderliggende, fysische infrastructuur is verborgen voor de eindgebruiker, hij bevindt zich in een virtueel afgeschermde omgeving.



(a) Huidige situatie van groupware toepassingen. De lokale netwerken worden afgeschermd, een afluisteraar kan enkel de beveiligde verbinding observeren.

(b) Groupware toepassing in een peer-to-peer kader. Geen enkele peer mag zomaar vertrouwd worden, en overal kunnen zich afluisteraars bevinden.

Figuur 1.2: Groupware

Verdere indeling van de thesistekst In hoofdstuk 2 wordt een uitgebreide analyse aangebracht van de beveiligingsaspecten in peer-to-peer systemen. Vervolgens wordt het anonimiteitsaspect van peer-to-peer systemen aandachtig bestudeerd in hoofdstuk 3. In deze hoofdstukken worden de bestudeerde technieken eerst theoretisch geanalyseerd, maar telkens ook toegepast op bestaande peer-to-peer systemen. In hoofdstuk 4 worden anonieme communicatiesystemen in detail bekeken. Er worden bouwblokken geïntroduceerd die anonieme kanalen realiseren. In hoofdstuk 5 wordt het gebruik van deze bouwblokken in een aantal concrete protocols voor anonieme kanalen in kaart gebracht. Hoofdstuk 6 is gewijd aan het ontwerp van een peer-to-peer systeem dat anonieme kanalen implementeert. Decentralisatie en toepassingsonafhankelijkheid staan centraal in dit ontwerp. In hoofdstuk 7 wordt de implementatie van het ontworpen systeem toegelicht. Er volgt ook een evaluatie van het uiteindelijke resultaat. De tekst wordt in hoofdstuk 8 afgesloten men een algemeen besluit van het verrichte werk.

Hoofdstuk 2

Beveiliging van peer-to-peer systemen

In dit hoofdstuk wordt besproken wat er allemaal op het gebied van beveiliging bestaat bij peer-to-peer systemen. In paragraaf 2.1 worden een aantal basisprincipes omschreven. Telkens wordt een probleem aangebracht en mogelijke oplossingen hiervoor. In paragraaf 2.2 wordt bekeken hoe in de praktijk beveiliging gerealiseerd wordt. De werking van een aantal bestaande peer-to-peer systemen wordt uit de doeken gedaan. Er wordt zo veel mogelijk getracht te illustreren welke basisprincipes gebruikt worden.

2.1 Basisprincipes

Er zijn een aantal basisprincipes die deel uit maken van een goed beveiligd peer-to-peer opslagsysteem. Alle principes zijn van toepassing op alle peer-to-peer toepassingen, behalve beveiliging van opslag, wat specifiek is voor informatie-opslag.

2.1.1 Opslag van data

Beveiliging van opslag

In opslagsystemen slaan peers informatie op. De gebruikers van het systeem kunnen deze gegevens opvragen en nieuwe data aan het systeem toevoegen. Het is wenselijk dat het systeem de peers oplegt informatie op een beveiligde manier op te slaan. Hiervoor bestaan twee hoofdredenen.

- Peers mogen niet aansprakelijk zijn voor de data die zij opslaan. Ze moeten kunnen ontkennen dat ze weten welke data op hun harde schijf staat.
- Slechte peers zullen proberen de data die ze opslaan, te wijzigen of te vernietigen. Dit moet onmogelijk gemaakt worden.

Het zal dus nodig zijn om de opgeslagen informatie te vercijferen zodat peers de exacte inhoud van de opgeslagen informatie niet kennen. Er bestaat een heel aantal cryptografische algoritmen, zowel symmetrisch als asymmetrisch, om geheimhouding van gegevens te verzekeren. De sleutel die gebruikt wordt voor vercijfering, moet ook opgeslagen worden. Er moet een systeem bestaan om vercijferde data en de gebruikte sleutel aan elkaar te koppelen.

Authenticiteit van informatie wordt bekomen door cryptografische technieken, waarmee men verandering van gegevens kan detecteren. Indien hierbij ook een sleutel gebruikt wordt (bv. bij

berekening van een MAC of digitale handtekening), dient deze sleutel ook in het opslagsysteem bewaard te worden.

Als laatste probeert men te verhinderen dat slechte peers informatie vernietigen. Er bestaan twee technieken om dit te voorkomen.

- Data kan op verscheidene peers bewaard worden. Deze *replicatie* maakt het moeilijker om alle exemplaren van bepaalde informatie te verwijderen.
- Informatie kan *opgesplitst* worden in n delen, terwijl het mogelijk is de informatie te reconstrueren met slechts $k < n$ delen. Een aanvaller moet meer dan $n - k$ delen vernietigen voordat informatie uit het systeem verdwijnt. Deze n delen worden vervolgens op verschillende peers bewaard.

Het opsplitsen van informatie kan op twee manieren gebeuren.

- Er bestaan cryptografische technieken om informatie over peers te verdelen.

  Het secret sharing schema van Shamir [1], bijvoorbeeld, wordt gebruikt om een sleutel te verdelen over meerdere peers. Het algoritme maakt gebruikt van veelterminterpolatie om dit te realiseren. Met $k = 3$ en $n = 5$ werkt het systeem als volgt:

  - je neemt het snijpunt van een parabool met de y-as als waarde voor de sleutel
  - je kiest dan 5 punten op de parabool; de punten op de parabool bepalen de delen van de sleutel
  - de parabool is volledig bepaald door 3 punten; het is dus mogelijk de sleutel te construeren met 3 van de 5 punten

  Het secret sharing algoritme kan enkel gebruikt worden om een kleine hoeveelheid informatie te splitsen en dus niet om bestanden te splitsen.

  Een ander voorbeeld is het informatieverstrooiingsalgoritme van Rabin [2]. Dit kan wel gebruikt worden om bestanden te splitsen.

- Bestanden worden meestal gesplitst met behulp van codeertechnieken. Foutverbeterende codes zorgen er voor dat je minder delen nodig hebt om het origineel te reconstrueren. In RAID-5 wordt bijvoorbeeld één schijf gebruikt als pariteit, zodat bij faling van een schijf deze vervangen kan worden waardoor de data toch niet verloren gaat ($n - k$ is hier 1).

Deze verdeling van informatie heeft als extra voordeel dat de belasting over de peers verdeeld wordt.

Gebrek aan opslagruimte

Er moet ook beslist worden wat er gebeurt bij gebrek aan opslagruimte. Het doel van de toepassing bepaalt dit.

- *Gedistribueerde informatie-opslag* is gericht op het efficiënt opvragen en opslaan van informatie. Bij gebrek aan opslagruimte op een peer wordt onpopulaire informatie verwijderd om plaats te maken voor nieuwe informatie. Men kan kiezen om de gegevens die het minst frequent opgevraagd zijn (*least commonly used*) of het langst niet geraadpleegd zijn (*least recently used*), te verwijderen.

- *Publicatiesystemen* richten zich vooral op oncensureerbare en persistente opslag van documenten. Wanneer een document in het publicatiesysteem geplaatst wordt, wordt een *levensduur* opgegeven. De levensduur bepaalt de tijdsduur waarin de informatie niet verwijderd mag worden. Het systeem garandeert dat informatie niet verwijderd zal worden tijdens de levensduur ervan: bij gebrek aan opslagruimte kunnen gewoon geen nieuwe documenten meer opgeslagen worden op een peer.

- Het is ook mogelijk om een hybride systeem te maken. Onpopulaire documenten met een korte levensduur zullen dan als eerste verwijderd worden.

2.1.2 Veilige communicatie

Zoals voor alle netwerkdiensten geldt, is het aan te raden om netwerkcommunicatie te beveiligen. Internet is totaal onveilig: netwerkcommunicatie kan afgeluisterd en gewijzigd worden. Zoals bij de beveiliging van informatie-opslag worden hiervoor standaard cryptografische technieken aangewend.

- Een cryptografische protocol [3] bepaalt een sessiesleutel. Een aantal mogelijkheden zijn uit den boze:

  – Een vertrouwde derde partij kan niet gebruikt worden omdat deze moeilijk te implementeren is op een gedecentraliseerde wijze.

  – Peers kunnen geen symmetrische lange-termijn-sleutels met elkaar delen. Dit is niet schaalbaar. Een publieke-sleutel-infrastructuur (PKI) is echter wel mogelijk.

  Hierdoor blijven maar twee scenario's over:

  – Er wordt geen publieke-sleutel-infrastructuur gebruikt. De sessiesleutel wordt dan bekomen via – anonieme – Diffie-Hellman-sleutelovereenkomst of via Shamir's no-key protocol. Het nadeel van dit scenario is dat het cryptografisch protocol altijd gevoelig zal zijn voor *tussenpersoonaanvallen*.

  – Er bestaat wel een publieke-sleutel-infrastructuur. Een protocol van het vorige scenario kan nu geauthentiseerd worden, waardoor tussenpersoonaanvallen niet langer mogelijk zijn, ofwel men spreekt een sessiesleutel af via sleuteltransport. Het nadeel van dit scenario is echter dat peers de publieke sleutels van de andere peers moeten te weten komen (dit probleem is analoog aan het probleem van informatielokalisatie dat in paragraaf 3.2.1 besproken wordt).

- Geheimhouding van communicatie kan gerealiseerd worden door een stroomcijfer of een blokcijfer in een bepaalde mode te gebruiken. Zowel Cipher Block Chaining, Output Feedback als Cipher Feedback kunnen gebruikt worden als mode.

  Geheimhouding kan eventueel bereikt worden door steganografie of via 'chaffing en winnowing' [4].

- Authenticiteit van informatie wordt verzekerd door het gebruik van een manipulatiedetectiecode (MDC of hashfunctie) of boodschap-authentiseringscode (MAC). Een manipulatie-detectiecode kan enkel gebruikt worden als de hashwaarde nadien nog vercijferd wordt, terwijl een MAC zowel voor als na encryptie berekend kan worden.

2.1.3 Betrouwbaarheid van peers

In een peer-to-peer netwerk kan men geen enkele gebruiker echt vertrouwen. De dienst die een peer-to-peer netwerk aanbiedt, moet door de gebruikers zelf uitgevoerd worden. Een aanvaller zal dus zelf peers draaien in zijn poging om de normale werking te verstoren. Hoe groter het netwerk is, hoe moeilijker het voor de aanvaller is het peer-to-peer systeem te saboteren, daar hij over meer peers zal moeten beschikken. Een correcte werking moet mogelijk blijven, ook als een deel van de gebruikers kwade bedoelingen heeft en de dienstverlening probeert te ondermijnen.

Een meer uitgebreide analyse van dit onderwerp is te vinden in het boek Peer-to-peer – Harnessing the Benefits of a Disruptive Technology: het hoofdstuk Accountability [5] handelt over betrouwbaarheid van peers en het hoofdstuk Reputation [6] over reputatiesystemen.

Aanvallen

Een aanvaller zal proberen de drie belangrijkste middelen waarover peers beschikken, aan te vallen. De meeste aanvallen worden in het jargon *denial-of-service*-aanvallen (verhinderen van dienstverlening en vaak afgekort tot DoS) of *flooding* (overbelasten door te veel operaties uit te voeren) genoemd.

Bandbreedte Een krachtige tegenstander kan proberen de communicatieverbinding van peers te verzadigen met een massa netwerkpakketten. Deze aanval is mogelijk op elke netwerkdienst, dus niet specifiek op peer-to-peer diensten. Er zijn twee bekende voorbeelden van aanvallen op bandbreedte.

- In februari 2000 zijn enkele van de grootste websites op internet, zoals Yahoo!, Amazon.com en CNN.com, tijdelijk ontoegankelijk geworden door gedistribueerde denial-of-service-aanvallen (DDoS).
- Het Slashdot effect [7] heeft al een heel aantal websites tijdelijk – totaal ongewild – offline gebracht door een plotse massale toename in het aantal bezoekers van de website.

In een peer-to-peer netwerk zal een bericht meestal een aantal keren verdergestuurd worden. Het effect van een bandbreedteaanval is dan nog erger.

Het is bijna onmogelijk om tegen bandbreedte aanvallen te verdedigen. Er bestaan echter wel oplossing voor specifieke aanvallen. Het gebruik van SYN cookies bijvoorbeeld maakt TCP SYN flooding onmogelijk [8]. Bij TCP SYN flooding wordt een grote hoeveelheid half open TCP connecties geopend, waardoor het slachtoffer geen andere TCP connecties meer kan maken.

Opslagcapaciteit Bij een publicatiesysteem kan een aanvaller proberen de opslagruimte van een peer te vullen met nutteloze data zodat er nadien geen nuttige informatie meer bewaard kan worden. Bij een gedistribueerd opslagsysteem zullen echter bij gebrek aan opslagcapaciteit bepaalde gegevens verwijderd worden. Dus is de aanvaller in staat bestaande nuttige gegevens te verwijderen.

Rekenkracht Het aanbieden van de peer-to-peer dienstverlening vraagt processortijd en geheugen. Publieke-sleutel-operaties bijvoorbeeld zijn traag en een aanvaller kan proberen een peer te overbelasten met dit soort bewerkingen totdat er geen rekenkracht meer overblijft. Dit soort aanval is minder waarschijnlijk dan de vorige twee.

Verdedigingsmechanismen

Er is dus nood aan een systeem om te weten te komen welke peers men kan vertrouwen en vooral welke juist niet. Aangezien een peer-to-peer systeem liefst volledig gedecentraliseerd is, kan men geen beroep doen op een vertrouwde derde partij in het proces om onbetrouwbare peers te identificeren. Hierbij komt dan nog de vereiste dat de peers anoniem willen zijn. Er zijn twee mechanismen om toch een correcte werking van het peer-to-peer netwerk te verzekeren.

Elektronische betalingen Er wordt een elektronische betaling gevraagd voordat men toegang krijgt tot een middel (bandbreedte of opslag). Dit moet het overgebruik van middelen verhinderen. Toegang tot middelen wordt dus beperkt. Er bestaan twee soorten elektronische betalingen: bewijs van werk en digitaal geld.

- Voordat toegang tot een middel verleend wordt, moet een bewijs van werk (*proof of work*) geleverd worden: er moet bewezen worden dat men een rekenintensieve bewerking heeft uitgevoerd. De elektronische betaling is niet herbruikbaar. Het doel van dit soort betaling is de potentiële aanvaller te vertragen in zijn aanval. Dit biedt echter geen afdoende bescherming tegen een aanvaller met veel rekenkracht. Dit principe is voor het eerst voorgesteld in 1992 als beschermingsmiddel tegen spam, ongeadresseerde e-mail [9].

  Hash Cash [10] en client puzzels [11] zijn systemen gebaseerd op het vinden van k-bit gedeeltelijke hashfunctie botsingen, wat een rekenintensieve taak is. Hashfuncties zijn ontworpen om botsingsbestendig te zijn: het vinden van een paar (x, x') zodat $hash(x) = hash(x')$ moet moeilijk zijn. Bij een gedeeltelijke hashfunctiebotsing moeten de laagste k bits van de hashwaarde overeenkomen: laagste k bits van $hash(x)$ zijn hetzelfde als die van $hash(x')$.

- De betaling heeft een intrinsieke waarde en is herbruikbaar. Een peer ontvangt *digitaal geld* en kan dit dan later zelf gaan uitgeven. Dit soort betaling verhindert het overgebruik van middelen: een aanvaller zal zonder digitaal geld vallen als hij geen geld verdient door zelf middelen aan te bieden. Een nadeel is echter dat er altijd een *bank* nodig zal zijn en dit is moeilijk te implementeren op een gedecentraliseerde wijze.

Zowel identificeerbaar als anoniem digitaal geld bestaan.

- *Identificeerbaar* geld is vergelijkbaar met een kredietkaart. De bank zal een bepaald bedrag van een rekening verplaatsen naar een andere. De bank weet hierdoor wie welke betalingen doet.

  Ronald Rivest en Adi Shamir stelden twee eenvoudige digitaal geld schema's, PayWord en MicroMint, voor in 1996 [12]. PayWord is een krediet gebaseerd schema op basis van kettingen van hashwaarden, terwijl MicroMint een muntstuk voorstelt door een k-voudige hashfunctiebotsing ($hash(x_1) = hash(x_2) = \ldots = hash(x_k)$).

- *Anoniem* geld is vergelijkbaar met muntstukken. Het geld zal uitgegeven worden door een bank, maar de betalingen kunnen niet aan personen gekoppeld worden.

  David Chaum was de pionier op het gebied van anoniem digitaal geld. Hij heeft een aantal patenten op zijn digitaal-geld-systeem en richtte in 1990 DigiCash op om zijn ideeën te commercialiseren. In zijn systeem eCash worden elektronische muntstukken gerealiseerd

door blinde handtekeningen [13]: muntstukken worden in een enveloppe van carbonpapier aan de bank gegeven, de bank handtekent de enveloppe en deze handtekening wordt ook doorgedrukt op de muntstukken in de enveloppe; het eindresultaat is dat de muntstukken door de bank gehandtekend zijn, maar de bank de muntstukken zelf nooit gezien heeft en dus muntstukken niet aan een persoon kan koppelen.

Een probleem bij eCash is dat de handtekening van de bank *online* geverifieerd moet worden om dubbele uitgave van muntstukken te verhinderen. Online verificatie maakt de betaling traag: je kan dit vergelijken met het verschil tussen een betaling met bancontact (online) en één met proton (offline). Het digitaal geld systeem van Stefan Brands laat echter toe om dubbele uitgaven *offline* te detecteren [14].

Men hoeft niet per se een digitaal geld schema te gebruiken. Men kan een andere vorm van 'voor wat, hoort wat' gebruiken, bijvoorbeeld *handel drijven* in opslagruimte: als een gebruiker informatie wil opslaan, moet hij daarvoor in de plaats zelf opslagruimte ter beschikking stellen. Een gebruiker kan dan niet meer documenten opslaan in het systeem dan de hoeveelheid die hij zelf aan opslagcapaciteit aanbiedt.

Reputatiesysteem In een *reputatiesysteem* gaat men een reputatie voor elke peer bijhouden. Peers met een lage reputatie krijgen toegang tot minder middelen. Peers die gewantrouwd worden, worden genegeerd door de andere peers en krijgen geen toegang meer tot middelen. Een reputatiesysteem kan zowel gemeenschappelijk als individueel zijn.

- In een *gemeenschappelijk* reputatiesysteem zal de reputatie van een peer bepaald worden door alle andere peers samen. Het eenvoudigste is om dit gecentraliseerd te implementeren, maar de uitdaging is juist om alle peers gedecentraliseerd samen te laten werken in de bepaling van de reputatie van peers.

- In een *individueel* systeem zal elke peer zelf de reputatie van de andere peers bepalen. Dit systeem is dus per definitie gedecentraliseerd.

Gecentraliseerde reputatiesystemen worden al geruime tijd op internet gebruikt. Een goed voorbeeld is het feedbacksysteem van eBay, een website voor openbare veilingen. Aangezien kopers en verkopers elkaar nog nooit ontmoet hebben, is het belangrijk een reputatiesysteem te hebben zodat gebruikers van eBay weten of iemand betrouwbaar is. Na elke verkoop worden eBay gebruikers aangemoedigd feedback te geven over de afhandeling van de verkoop. De reputatie van gebruikers wordt bepaald op basis van de feedback over hen.

Een reputatiesysteem moet robuust zijn tegen aanvallen. Een aanvaller mag niet in staat zijn de reputatie van peers in een bepaalde richting te veranderen. Dit soort aanvallen is vooral mogelijk in samenwerkende reputatiesystemen en dus minder van toepassing op individuele reputatiebepaling.

Aangezien er geen robuuste gedecentraliseerde digitale geld schema's bestaan, zal men vaak naar reputatiesystemen grijpen, al dan niet in combinatie met een betalingssysteem.

2.1.4 Anonimiteit

Het is mogelijk dat de gebruikers van het peer-to-peer netwerk anoniem willen blijven. Het moet dan onmogelijk zijn te achterhalen welke informatie gebruikers opvragen of welke informatie gebruikers in het netwerk opslaan. De vereiste van *anonimiteit* lijkt te conflicteren met de vereiste

van betrouwbaarheid: hoe kan men iemand vertrouwen die anoniem wil blijven? De oplossing hiervoor is *pseudonimiteit*. Een gebruiker is identificeerbaar door zijn pseudoniem, maar men kan de echte identiteit van de gebruiker niet achterhalen. Pseudoniemen zijn natuurlijk uniek binnen het systeem. Dit maakt het mogelijk om een gebruiker te vertrouwen op basis van zijn pseudoniem, terwijl zijn echte identiteit niet gekend is. Pseudoniemen worden gewoonlijk gerealiseerd door digitale handtekeningen.

Pseudonimiteit is zwakker dan anonimiteit. Bij anonimiteit weet men niet wie een transactie (publiceren of lezen van informatie) doet. Bij pseudonimiteit kunnen alle transacties gekoppeld worden aan een pseudoniem. Als men identiteit en pseudoniem van een gebruiker met elkaar kan verbinden, kunnen alle toekomstige transacties met hetzelfde pseudoniem aan die gebruiker gekoppeld worden.

Anonimiteit in peer-to-peer systemen wordt in de volgende hoofdstukken gedetailleerd behandeld.

2.2 Analyse van bestaande peer-to-peer projecten

De bekendste peer-to-peer bestandsuitwisselingssystemen zijn Napster, Gnutella en KaZaA. Deze systemen zijn bijzonder populair omdat ze snelle en eenvoudige bestandsuitwisseling toelaten. De nadruk ligt vooral op efficiëntie en dit gaat ten koste van beveiliging – vooral het anonimiteitsaspect.

Toch bestaan er systemen waar de nadruk wel op beveiliging ligt. Deze systemen implementeren de beschreven basisprincipes, of toch een deel ervan. Aangezien beveiliging nadelig is voor de efficiëntie, de snelheid en het gebruikersgemak, bepaalt het vooropgestelde doel van het project welke basisprincipes gerealiseerd worden en op welke wijze. De meeste projecten proberen een compromis te vinden tussen efficiëntie en beveiliging.

De beveiligde peer-to-peer systemen zijn meestal het resultaat van academisch onderzoek en de bruikbaarheid laat in de praktijk vaak te wensen over.

Nu volgt een korte beschrijving van een aantal belangrijke beveiligde peer-to-peer opslagsystemen. De nadruk ligt vooral op hetgeen kenmerkend is voor het project. Daarom worden enkel de basisprincipes besproken die van belang zijn voor het project.

Mojo Nation

Mojo Nation [15] beoogt vooral efficiënte en robuuste gedistribueerde opslag: de nadruk ligt op de beschikbaarheid van informatie en niet op anonimiteit.

Bestanden worden in stukjes verdeeld (hoe groter het bestand, hoe meer stukjes). Elk stuk wordt vervolgens opgesplitst via een foutverbeterende code in 8 blokken, terwijl reconstructie mogelijk is met 4 blokken. Deze blokken worden op verschillende peers bewaard, zelfs op peers met beperkte bandbreedte (omdat de blokken klein zijn). Deze *zwermdistributie* van bestanden verdeelt de belasting over alle peers. Toch kunnen een aantal blokken – misschien tijdelijk – onbeschikbaar zijn.

Mojo Nation besteedt ook veel aandacht aan de betrouwbaarheid van gebruikers.

- In het systeem wordt anoniem digitaal geld gebruikt om overgebruik van middelen te beperken. De gebruikte munteenheid heet *mojo* – vandaar ook Mojo Nation.

  Mojo Nation gebruikt een gecentraliseerde bank om alle betalingen te verwerken. Alle andere diensten binnen het systeem worden wel op een gedistribueerde manier gerealiseerd.

- Elke gebruiker is binnen het systeem identificeerbaar met een pseudoniem. Dit laat toe om een reputatie te koppelen aan dit pseudoniem. De reputatie wordt bepaald op basis van alle vroegere transacties met de gebruiker.

Publius

Publius [16] is een web-gebaseerd systeem om uitgevers gemakkelijk en anoniem te laten publiceren. Publius probeert vooral een robuust publicatiesysteem te verwezenlijken dat bestand is tegen censuur.

Een bestand wordt vercijferd met een willekeurige symmetrische sleutel K. Deze sleutel K wordt in n delen gesplitst met het secret sharing schema van Shamir, zodat het mogelijk is de sleutel te reproduceren met slechts k delen. De uitgever plaatst dan het vercijferd bestand op n peers, elke keer vergezeld van een ander deel van de sleutel. Het document moet door meer dan $n-k$ peers verwijderd worden of meer dan $n-k$ peers moeten offline zijn, voordat een document onbeschikbaar wordt.

Publius werkt met een vaste lijst van servers om informatie op te slaan en biedt dus geen gedecentraliseerde ondersteuning om nieuwe servers toe te voegen. Er wordt ook geen aandacht besteed aan de betrouwbaarheid van gebruikers.

Free Haven

Free Haven [17] is een publicatiesysteem met veel nadruk op anonimiteit en beveiliging. Zoals alle publicatiesystemen streeft het persisente en oncensureerbare opslag na.

Het informatieverstrooiingsalgoritme van Rabin wordt gebruikt om een bestand op te splitsen in n delen waarvan k volstaan om het bestand te reproduceren. De uitgever genereert een sleutelpaar en gebruikt de private sleutel om elk deel te handtekenen. Tenslotte worden de delen met wat extra informatie (levensduur, publieke sleutel om de handtekening te verifiëren, nummering van de delen, ...) op de peer van de uitgever bewaard.

De uitgever bewaart dus initieel de delen op zijn eigen peer. Maar peers wisselen periodisch delen met elkaar uit. Zo geraken de delen verspreid over heel het netwerk. De delen worden voortdurend op andere peers bewaard zodat het voor een aanvaller moeilijk is delen te vernietigen.

Aangezien peers niet te vertrouwen zijn, gebruikt Free Haven een reputatiesysteem dat misgedrag afstraft. Elk deel van een bestand heeft zijn *buddy*, een kameraad. Deze kameraad is een ander deel van hetzelfde bestand. De peer die een deel opslaat, zal periodisch controleren of de kameraad van dit deel nog in leven is. Als de kameraad niet langer bestaat, wordt dit aan het reputatiesysteem gemeld, waardoor de reputatie van de peer die de kameraad moest bewaren, verlaagd wordt. Een aanvaller kan een deel enkel succesvol verwijderen als hij toevallig de kameraad ervan op een van zijn andere peers bewaart.

Freenet

Bij Freenet [18] ligt de nadruk vooral op de privacy van de gebruikers en op een hoge beschikbaarheid van de informatie. Het is de bedoeling om een bepaalde graad van anonimiteit aan te bieden, terwijl toch een efficiënte dienstverlening gegarandeerd moet worden.

Bestanden in Freenet worden geïdentificeerd door een semi-unieke *bestandssleutel*, die bekomen wordt door een éénwegsfunctiefunctie (hashfunctie). Er bestaan drie verschillende types

bestandssleutels: *keyword-signed key* (KSK), *signed-subspace key* (SSK) en *content-hash key* (CHK).

Bij KSK wordt de bestandssleutel afgeleid van een omschrijving van het bestand. Om een bestand op te vragen, hoeft men enkel de beschrijving ervan te kennen. Deze bestandssleutels zijn daarom eenvoudig te onthouden en te verspreiden. Daarentegen weerhoudt niets iemand een ander bestand toe te voegen met dezelfde beschrijving.

Het voorgaande probleem wordt aangepakt bij SSK. De volledige sleutelruimte wordt ingedeeld in persoonlijke deelruimtes. De uitgever moet bestanden handtekenen om ze toe te voegen aan zijn persoonlijke deelruimte. Als men een document opvraagt, heeft men naast de beschrijving ervan ook de publieke sleutel van de uitgever nodig – om de handtekening te controleren.

Bij CHK worden bestandssleutels afgeleid van de inhoud van het bestand. CHK laat toe om bestanden te splitsen en later aan te passen.

Bestanden worden symmetrisch vercijferd. Bij KSK en SSK wordt de vercijferingssleutel afgeleid van de omschrijving van het bestand. Lezers kunnen de decryptiesleutel zelf afleiden van de omschrijving. Bij CHK wordt echter een willekeurige vercijferingssleutel gegenereerd. De uitgever moet de sleutel publiceren opdat lezers het bestand kunnen ontcijferen.

Het opvragen en toevoegen van bestanden gebeurt via een routeringsalgoritme. Dit houdt in dat bestanden via een aantal peers verstuurd worden. Deze peers bewaren een kopie van het bestand, wat *caching* genoemd wordt. Caching zorgt ervoor dat populaire bestanden, die frequent opgevraagd worden, op steeds meer peers bewaard worden. Freenet repliceert en verplaatst dus informatie op een dynamische manier als reactie op de vraag naar bepaalde informatie. Als een populair bestand opgevraagd wordt, zal de aanvraag ook sneller beantwoord worden omdat er meer kopies van het bestand in Freenet aanwezig zijn. Onpopulaire informatie verdwijnt echter om plaats te maken voor populaire.

Hoofdstuk 3

Anonimiteit in peer-to-peer systemen

Uit het vorige hoofdstuk is gebleken dat anonimiteit van gebruikers een wenselijk eigenschap kan zijn voor beveiligde peer-to-peer systemen. In dit hoofdstuk wordt in detail op deze materie ingegaan. Eerst dient verduidelijkt te worden wat anonimiteit juist inhoudt bij opslagsystemen. In paragraaf 3.1 worden daarom een aantal vormen van anonimiteit gedefinieerd. Paragraaf 3.2 probeert een aantal basisprincipes te omschrijven om deze vormen van anonimiteit te verwezenlijken. Uiteindelijk worden de basisprincipes geconcretiseerd aan de hand van voorbeelden. Dit gebeurt in paragraaf 3.3 door een analyse van bestaande peer-to-peer systemen. De nadruk ligt bij deze analyse op de gebruikte technieken en de gehaalde anonimiteit.

3.1 Vormen van anonimiteit bij opslagsystemen

In de beschrijving van het Free Haven project [17] worden een aantal vormen van anonimiteit aangebracht om bestaande opslagsystemen te vergelijken.

Informatie wordt opgeslagen in *documenten*. De *auteur* van een document is de persoon die het document initieel gecreëerd heeft. De *uitgever* van een document is de gebruiker die het document in het opslagsysteem plaatst. Documenten kunnen *lezers* hebben, die het document ophalen uit het systeem. De peers die deel uitmaken van het peer-to-peer systeem worden *servers* genoemd.

Het gaat duidelijk om een terminologie specifiek voor publicatiesystemen. Bij gedistribueerde informatie-opslag spreekt men eerder over *bestanden* (i.p.v. documenten), gebruikers zullen bestanden *opvragen* (i.p.v. lezen) en *toevoegen* (i.p.v. uitgeven).

Er worden een aantal vormen van anonimiteit gedefinieerd:

Auteursanonimiteit: Een tegenstander kan de auteur niet verbinden met het document.

Uitgeversanonimiteit: Diegene die een document publiceert, kan niet verbonden worden met het document.

Lezersanonimiteit: Deze vorm dient om de privacy van de gebruikers te beschermen. Het is niet mogelijk te bepalen welk document door wie opgevraagd wordt.

Serveranonimiteit: De tegenstander kan niet te weten komen welke servers een welbepaald document aanbieden.

Documentanonimiteit: De server weet niet welke documenten hij opslaat. Er kan nog een onderscheid gemaakt worden tussen passieve en actieve documentanonimiteit. Als men de term documentanonimiteit gebruikt, wordt altijd passieve documentanonimiteit bedoeld omdat actieve documentanonimiteit in de praktijk niet verwezenlijkt kan worden.

Passieve documentanonimiteit: Als de server enkel naar de data mag kijken die hij bewaart, kan hij de inhoud van een document niet achterhalen.

Actieve documentanonimiteit: De server mag communiceren met andere servers en data vergelijken. Zelfs dan is de server niet in staat het document niet lezen.

Opvragingsanonimiteit: De server weet niet welk document hij aanbiedt wanneer hij een aanvraag tot informatie succesvol beantwoordt.

Het is intuïtief in te zien dat de sterkte van anonimiteit kan verschillen. In het volgende hoofdstuk wordt de graad van anonimiteit formeel gedefinieerd.

3.2 Basisprincipes

Er bestaan een aantal methoden om anonimiteit in peer-to-peer systemen te bekomen. Bepaalde vormen van anonimiteit zijn eenvoudig te realiseren. Andere kunnen enkel bereikt worden door meer complexe mechanismen.

Auteursanonimiteit Als een auteur zijn naam niet vermeldt in zijn document of een schrijverspseudoniem gebruikt, kan niemand de identiteit van de echte auteur achterhalen. Dit is ook het geval in de gewone wereld: een boek kan gewoon anoniem uitgegeven worden, tenzij men naar de uitgever van het boek stapt en deze bedreigt om zijn bron prijs te geven. Auteursanonimiteit kan dus eenvoudig gerealiseerd worden, ook in de elektronische wereld.

Documentanonimiteit Als documenten op een beveiligde manier bewaard worden op peers, wordt – passieve – documentanonimiteit bekomen. De technieken om opslag te beveiligen zijn reeds uitvoerig besproken in paragraaf 2.1.1.

Andere vormen van anonimiteit Server-, lezers- en uitgeversanonimiteit worden op een meer gestructueerde manier bekomen.

In een opslagsysteem kunnen twee scenario's onderscheiden worden.

- De uitgever plaatst een document op een server.
- Lezers halen een document af bij een server.

In het eerste scenario speelt uitgeversanonimiteit een belangrijke rol, terwijl in het tweede lezersanonimiteit en serveranonimiteit belangrijk zijn. In deze twee scenario's worden dezelfde mechanismen gebruikt om informatie uit te wisselen.

3.2.1 Informatie-uitwisselingssystemen

Informatie-uitwisseling kan op twee verschillende manieren gebeuren.

- Men vraagt eerst waar de informatie opgeslagen kan worden of opgeslagen is. Vervolgens wordt de informatie verstuurd naar of afgehaald bij de juiste server. De uitwisseling gebeurt duidelijk in twee fases: eerst een *lokalisatieaanvraag*, dan de eigenlijke informatieoverdracht.

- Als een lezer bepaalde informatie opvraagt, krijgt hij deze informatie *rechtstreeks* als antwoord. De uitgever hoeft niet vooraf te vragen op welke server de informatie opgeslagen kan worden, maar het peer-to-peer systeem regelt zelf waar de informatie bewaard zal worden. De uitwisseling kent maar één fase.

Beide uitwisselingsmechanismen zullen een aantal verschillende zoektechnieken gebruiken. Een uitgebreide beschrijving van alle mogelijk zoekmechanismen wordt gegeven in Decentralized Resource Discovery in Large Peer Based Networks [19].

Uitwisseling via routeringsalgoritme

De lezer en uitgever zullen informatie opvragen en publiceren via een *routeringsalgoritme*.

De lezer zal een aanvraag sturen via een routeringsalgoritme: de aanvraag wordt door een aantal peers doorgestuurd totdat een bepaalde peer over de gevraagde informatie beschikt. Deze peer – de server – zal de gewenste informatie naar de lezer sturen. De informatie kan rechtstreeks naar de lezer gestuurd worden of via het routeringsalgoritme. De informatie volgt dan hetzelfde pad als de aanvraag van de lezer, maar in de omgekeerde richting.

De uitgever kan gewoon een willekeurige server uitkiezen om de informatie op te bewaren, maar meestal stuurt hij de informatie via het routeringsalgoritme. Peers sturen de informatie door totdat ze aankomt bij een peer die het document wil of moet opslaan.

Er zijn een aantal routeringsalgoritmen mogelijk. Gemakshalve wordt enkel het raadplegen van informatie toegelicht, maar deze technieken zijn ook toepasbaar voor de publicatie van informatie.

- Het eenvoudigste routeringsalgoritme is *broadcast*. Een lezer stuurt zijn aanvraag naar alle peers die hij kent. Deze peers sturen op hun beurt de aanvraag door naar alle peers die zij kennen, als ze de informatie zelf niet hebben. Indien de peer echter over de informatie beschikt, wordt de aanvraag niet verder doorgestuurd, maar wordt een antwoord gestuurd naar de gebruiker die de aanvraag doet.

  Deze broadcast-zoekmethode kiest het kortste pad van de lezer tot een peer die de informatie bewaart. Dit gebeurt tegen een zeer hoge kost: het aantal aanvraagberichten stijgt exponentieel. Meestal zal een *teller* gebruikt worden die verlaagd wordt telkens een peer beslist de aanvraag door te sturen. Dit zorgt er voor dat de aanvraag uitsterft wanneer de tellerwaarde nul wordt, zodat de aanvraag niet oneindig lang rondgestuurd zal worden. In het ideale geval wordt de teller geïnitialiseerd op de kortste afstand van de gebruiker tot de peer die de informatie heeft.

- Meer geavanceerde routeringsalgoritmes zullen *selectief* aanvragen doorsturen: een peer stuurt de aanvraag door naar de peer die volgens hem de informatie waarschijnlijk opgeslagen heeft. De beslissing waarnaar de aanvraag door te sturen gebeurt op basis van een bepaalde *heuristiek*.

  - Een naïef routeringsalgoritme stuurt de aanvraag gewoon door naar een willekeurig andere peer. Uiteindelijk zal de aanvraag wel terecht komen bij een peer die over de informatie beschikt en dit hopelijk met minder berichten dan broadcastroutering.
  - De meeste selectieve routeringsalgoritmes maken gebruik van *gedistribueerde hashtabellen*. Bij het gebruik van gedistribueerde hashtabellen worden documenten met een semi-unieke *identificatiesleutel*, meestal een hashwaarde van de inhoud van het document, in het systeem bewaard. De identificatiesleutel wordt gebruikt om de informatie te lokaliseren: de keuze waarnaar een aanvraag gestuurd wordt, gebeurt op basis van de identificatiesleutel. Het netwerk en de peers worden zo ontworpen dat een identificatiesleutel snel gelokaliseerd kan worden, wat de grootte van het netwerk ook is.

    Een voorbeeld van gedistribueerde hashtabellen is DHash [20], dat gebruik maakt van Chord [21].

- Een combinatie van broadcast en selectief doorsturen is ook mogelijk.

Uitwisseling na lokalisatieaanvraag

De lokalisatieaanvraag geeft het adres van de server en dan wordt de eigenlijke informatie-overdracht gedaan. Het opvragen van het adres van de server kan op een aantal manieren gerealiseerd worden. Opnieuw wordt enkel het raadplegen van informatie toegelicht.

- Een *index* wordt *centraal* bewaard. Een gebruiker contacteert deze dienst en krijgt onmiddellijk het adres van een peer die over de gevraagde informatie beschikt. Dit is natuurlijk geen peer-to-peer aanpak aangezien men één enkel punt van faling creëert. Het is echter wel bijzonder efficiënt: er wordt een aanvraag gedaan en men krijgt onmiddellijk een antwoord. Dit is vergelijkbaar met een zoekmachine op internet: als men bepaalde informatie wil hebben, raadpleegt men een zoekmachine en de zoekmachine geeft een lijst met relevante links.

- De index kan *gedecentraliseerd* bewaard worden. Er zal dan een routeringsalgoritme gebruikt worden om in de index te zoeken. Men spreekt soms van *gedistribueerde zoekbomen*. Een voorbeeld van een gedecentraliseerde index is de peer-to-peer zoekmachine Reptile [22].

- Het gebruik van gedistribueerde hashtabellen is ook mogelijk.

- Men kan ook een ander routeringsalgoritme, zoals broadcast en selectief doorsturen, gebruiken om het adres op te vragen van de server die een bepaald document opslaat.

De lokalisatieaanvraag gebeurt in sommige gevallen via een routeringsalgoritme, maar bij de informatie-overdracht wordt geen gebruik gemaakt van routeringsalgoritmes. De informatie wordt gewoon verstuurd via een communicatiekanaal. Dit is het grote verschil met het vorige uitwisselingsmechanisme. Dit communicatiekanaal kan een rechtstreekse verbinding zijn of kan via enkele andere peers gaan.

3.2.2 Anonimiteit bij informatie-uitwisseling

Uitwisseling via routeringsalgoritme

Documenten kunnen zeer *efficiënt* verstuurd worden mits een goede keuze van het routeringsalgoritme. Bij broadcastroutering bijvoorbeeld zal het document het kortste pad bereiken van de uitgever naar de server die het document zal opslaan.

Uitgevers-, lezers- en serveranonimiteit kunnen bereikt worden, maar de graad hiervan is moeilijk te bepalen. De uitgever weet niet op welke server het document terecht zal komen en de lezer weet niet welke server het document aanbiedt, want zij kennen enkel de eerste peer(s) waar zij hun aanvraag naar sturen. Er is dus serveranonimiteit. De server weet ook niet goed wie de lezer is, aangezien de peer die de aanvraag doet, ofwel de lezer is ofwel de aanvraag gewoon doorstuurt. Analoog zal de server ook niet weten wie de uitgever is.

Informatie-overdracht via een routeringsalgoritme is een *ad hoc methode* om anonimiteit te bekomen en helemaal niet theoretisch onderbouwd. De behaalde graad van anonimiteit moet blijken uit de exacte implementatie. Er zal gewoonlijk geen bescherming aangeboden tegen alle mogelijk bedreigingsmodellen. Zodra peers samen beginnen te werken of verkeersanalyse gedaan wordt, zal snel blijken dat de behaalde anonimiteit te wensen overlaat.

Opvragingsanonimiteit is mogelijk wanneer uit de aanvraag niet blijkt om welk document het gaat. Bij het gebruik van gedistribueerde hashtabellen vraagt men het document op met een bepaalde identificatiesleutel. Als de identificatiesleutel de hashwaarde van de bestandsnaam is en als bestanden vercijferd opgeslagen worden, weet de server hierdoor niet welk bestand hij aanbiedt: hij kent de inhoud niet en hij leert de bestandsnaam niet kennen. De server kan immers de hashfunctie niet omkeren.

Uitwisseling na lokalisatieaanvraag

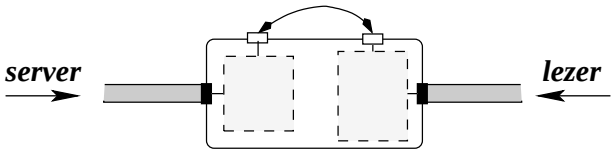
Indien het overdragen van documenten via een communicatiekanaal gebeurt, zal de graad van anonimiteit een stuk duidelijker te bepalen zijn. De anonimiteit van communicatiekanalen is immers reeds uitgebreid onderzocht. Bij een communicatiekanaal stuurt een *zender* een *bericht* naar een *ontvanger*. Nu volgt een korte bespreking van de vormen van anonimiteit die mogelijk zijn bij informatie-overdracht via een communicatiekanaal.

Geen anonimiteit Als men geen extra inspanning doet, zal een communicatiekanaal totaal geen anonimiteit bieden. De zender van een bericht weet naar wie hij het stuurt en de ontvanger weet van wie hij het krijgt. Het is natuurlijk mogelijk om een vals afzenderadres te gebruiken (dit wordt *spoofen* genoemd), maar dit is hier niet aan de orde. Dus indien de overdracht van documenten via een *traditioneel communicatiekanaal* – TCP of UDP connectie – gebeurt, worden geen lezers-, uitgevers- en serveranonimiteit bekomen.

Lezers- en uitgeversanonimiteit Men kan echter ook een *anoniem communicatiekanaal* gebruiken. Bij een anoniem kanaal wordt het bericht van de zender niet rechtstreeks naar de ontvanger gestuurd, maar via een aantal andere peers. De ontvanger van een bericht kan geen onderscheid maken tussen een peer die een bericht zelf stuurt of een peer die gewoon een bericht doorstuurt voor een andere peer. De afzender bepaalt op voorhand wie de bestemmeling is. Er kan toch een antwoord teruggestuurd worden naar de anonieme afzender.

Als een anoniem kanaal dus gebruikt wordt voor het versturen van documenten, zijn lezer en uitgever anoniem. Men krijgt meestal het echte adres van de server als resultaat van een lokalisatieaanvraag. Serveranonimiteit wordt dus niet bereikt.

De volgende hoofdstukken gaan dieper in op de realisatie van anonieme kanalen.



Figuur 3.1: Double blinded kanaal. Server en lezer ontmoeten elkaar op een neutrale ontmoetingsplaats.

Serveranonimiteit Hoewel op het eerst zicht lijkt dat met anonieme kanalen geen serveranonimiteit bereikt wordt, is dit toch mogelijk. Een server legt dan een anoniem kanaal naar een bepaalde peer en vraagt hem een poort te reserveren. De server publiceert dan het adres van die peer als zijn eigen adres. Die peer wordt dus gekozen als een ontmoetingspunt in het midden. Lezers en uitgevers kunnen een anoniem kanaal opzetten naar de peer die als ontmoetingspunt functioneert. Die peer zal alle berichten doorsturen via het anoniem kanaal dat de server met hem heeft opgezet. Dit wordt geïllustreerd in figuur 3.1. In de literatuur wordt dit een *double blinded kanaal* [23] genoemd. Nu is het wel mogelijk om serveranonimiteit te bekomen. De uitgever en lezers denken dat het ontmoetingspunt de eigenlijke server is, wat dus niet het geval is.

Hybride uitwisselingssysteem

Er is een duidelijke afweging tussen efficiëntie – overdracht via routeringsalgoritme – en anonimiteit – overdracht via anoniem kanaal. Daarom is het misschien aangewezen om een hybride systeem te gebruiken. Men gebruikt overdracht via een routeringsalgoritme, maar peers kunnen een anoniem kanaal naar de volgende peer gebruiken – als ze een aanvraag doorsturen – in plaats van een directe verbinding. Dit verhoogt duidelijk de graad van anonimiteit voor de lezer en uitgever. Het heeft alleen zin om een anoniem kanaal te gebruiken tussen de eerste peer – lezer of uitgever – en de tweede peer. Tussen alle andere peers worden directe verbindingen gebruikt. Men spreekt dan van een anonieme *pre-routeringsfase*. De anonimiteit van de gebruiker wordt zo sterk mogelijk gemaakt en het uitwisselingssysteem blijft toch nog efficiënt.

3.3 Analyse van bestaande peer-to-peer projecten

Een korte analyse van de anonimiteit in bestaande peer-to-peer opslagsystemen is hier aan de orde, om de beschreven basisprincipes te verduidelijken.

Hoewel Napster en Gnutella geen anonimiteit bieden, is het misschien toch interessant om te zien hoe er informatie uitgewisseld wordt.

Napster

Napster [24] werkt met een index op een centrale server. Gebruikers registreren in deze index welke bestanden zij aanbieden. Men kan in deze index zoeken: de index geeft een lijst terug waarin staat wie welk bestand aanbiedt. De gebruiker kiest dan het bestand (en bijhorende server)

dat hem interesseert. Bij de overdracht kennen de server en de lezer elkaars internetadres; geen sprake van lezers- of serveranonimiteit dus. De uitgever bewaart alle bestanden lokaal op zijn eigen computer; de uitgever en de server zijn dus dezelfde gebruiker. De uitgever/server weet welke bestanden hij aanbiedt; er is dus geen documentanonimiteit.

Gnutella

Bij Napster weet de centrale server waarop de index staat, van elke gebruiker welke bestanden hij zoekt en aanbiedt. Deze centrale server heeft volledige controle over welke bestanden in de index staan en kan eenvoudig bepaalde documenten censureren.

Gnutella [25] probeert dit te verhinderen door de lokalisatieaanvraag te decentraliseren en anoniem te maken. De lokalisatieaanvraag wordt verstuurd door broadcastroutering. Er wordt een teller (*hops-to-live*) gebruikt zodat aanvragen niet oneindig lang doorgestuurd worden: deze teller wordt verlaagd door elke peer die de aanvraag doorstuurt. Toch blijkt de aanvraag niet anoniem te zijn omdat de teller op een gekende waarde geïnitialiseerd wordt: als een peer een aanvraag 'doorstuurt' met de teller op de initiële waarde, is het duidelijk dat deze peer de aanvraag gegenereerd heeft en niet doorstuurt voor een andere peer.

Mojo Nation

Binnen Mojo Nation worden een aantal diensten aangeboden die een gebruiker contacteert om te ontdekken welke bestanden opgeslaan zijn en waar deze bestanden staan: de *publication tracker* houdt bij welke bestanden allemaal opgeslagen zijn in Mojo Nation, de *content tracker* onthoudt hoe een bestand geconstrueerd kan worden en de *metatracker* houdt bij welke servers allemaal verantwoordelijk zijn om bepaalde delen op te slaan. Al deze diensten worden door de peers zelf aangeboden. Het aanbieden van deze diensten is voor gebruikers een manier om mojo te verdienen.

Mojo Nation realiseert documentanonimiteit aangezien servers slechts een deel van een document opslaan. Lezer- en uitgeversanonimiteit worden niet bekomen omdat de informatie-overdracht rechtstreeks gebeurt, zoals bij Napster en Gnutella.

Publius

Publius bereikt tevens documentanonimiteit omdat de sleutel opgesplitst wordt over n servers: de peer die een document opslaat, beschikt enkel over één deel van de ontcijferingssleutel. De uitgever plaatst een document op de n verschillende servers via een anoniem kanaal. Er is dus uitgeversanonimiteit. Lezers gebruiken echter geen anoniem kanaal om documenten af te halen en dus is er geen lezersanonimiteit.

Free Haven

Aangezien Free Haven bestanden in stukken opsplitst, is er documentanonimiteit. De uitgever plaatst initieel de verschillende delen van een document op zijn eigen peer. Dan begint de peer de delen te *ruilen* met andere peers. Wanneer twee peers bestandsdelen ruilen, kennen ze enkel elkaars pseudoniem. Dit is net iets zwakker dan informatie-uitwisseling via een anoniem kanaal, maar laat het gebruik van een reputatiesysteem toe. De uitgever weet dus niet op welke servers de delen van het document bewaard worden. Er is dus serveranonimiteit en ook uitgeversanonimiteit. Lezers zijn ook anoniem omdat ze een anoniem kanaal gebruiken om bestand af te halen.

Free Haven behaalt dus bijna alle mogelijke vormen van anonimiteit, wat juist het doel van dit project is. De verwezenlijken van dit doel heeft natuurlijk consequenties voor de efficiëntie van het systeem.

Freenet

Er is documentanonimiteit omdat bestanden vercijferd opgeslagen worden.

Freenet implementeert een gedistribueerde hashtabel. Elke peer heeft een eigen identificatiesleutel, afgeleid van zijn internetadres. Peers specialiseren zich in het bewaren van bestanden met een bestandssleutel die in de buurt van hun identificatiesleutel ligt. Informatie-overdracht gebeurt via een routeringsalgoritme dat *sleutelnabijheid* als heuristiek gebruikt.

Als de lezer een bestand met een bepaalde bestandssleutel opvraagt, zoekt zijn peer in zijn lijst gekende peers naar de peer wiens identificatiesleutel het dichtst bij de bestandssleutel ligt. De aanvraag wordt naar deze peer doorgestuurd. Peers blijven de aanvraag doorsturen naar de peer die volgens hen het meeste kans heeft het bestand op te slaan, totdat de aanvraag inderdaad bij een peer komt die het bestand heeft. Het bestand wordt via dezelfde weg teruggestuurd en elke peer onderweg bewaard een kopie van het bestand zodat later aanvragen van ditzelfde bestand sneller beantwoord kunnen worden.

Als een uitgever een bestand toevoegt, wordt dit gewoon een aantal keer doorgestuurd. De keuze naar waar door te sturen, gebeurt weer op basis van de bestandssleutel zodat het document op peers terechtkomt waarvan de identificatiesleutels steeds dichter bij de bestandssleutel ligt. Ook hier gebeurt onderweg caching.

Elke aanvraag heeft een teller zoals bij Gnutella. Opnieuw bestaat het gevaar dat een gekende initiële teller het mogelijk maakt te achterhalen wie de aanvraag genereert – en dus wie de lezer is. Freenet probeert hier een oplossing voor te bieden door de teller niet op een vaste waarde te initialiseren, maar op een willekeurige waarde. Peers kunnen ook kiezen om de teller *niet* te verlagen als ze een aanvraag doorsturen. Er is dan toch lezersanonimiteit, maar deze is niet erg sterk: als de teller een hoge waarde heeft, is er toch nog een hoge kans dat de peer die de aanvraag stuurt, de lezer is. Het is in elk geval een stuk beter dan bij Gnutella. Om de graad van lezersanonimiteit te verhogen, wordt in de Freenet paper voorgesteld om de aanvraag in een *pre-routeringsfase* door een anoniem kanaal te sturen.

Het toevoegen van bestanden gebeurt via dezelfde mechanismen als het opvragen. Er is dus ook zwakke uitgeversanonimiteit.

Peers willen steeds meer andere peers in het netwerk leren kennen om zo efficiënter en sneller bestanden te kunnen opvragen. In Freenet tracht men dit te doen door bij elke aanvraag te vermelden wie de aanvraag stuurt – dus wie de initiator van de aanvraag is. Alle peers onderweg leren zo het internetadres van deze peer kennen. Dit lekt opnieuw informatie die de anonimiteit van gebruikers drastisch verlaagt. Freenet probeert het effect ervan nog wat te beperken door het bronadres in een aanvraag te veranderen: een peer onderweg kan kiezen om het adres te overschrijven door zijn eigen adres. Zo helpt hij bovendien ook aan de verspreiding van zijn eigen bekendheid.

Hoofdstuk 4

Anonieme kanalen

Zoals reeds vermeld in paragraaf 3.2.2, worden anonieme communicatiekanalen gebruikt om anoniem berichten naar iemand te sturen en dit door de berichten via een aantal andere peers om te leiden. Het gebruik ervan maakt het mogelijk om lezer- en uitgeveranonimiteit te realiseren bij opslagsystemen.

In dit hoofdstuk wordt er dieper ingegaan op dit onderwerp. Paragraaf 4.1 beschrijft eerst de topologie en terminologie die voor anonieme kanalen gebruikt worden. Paragraaf 4.2 legt uit wat anonieme kanalen exact trachten te realiseren. In paragrafen 4.3 en 4.4 komen de verschillende soorten tegenstanders en aanvallen aan bod, die de anonimiteit en veiligheid van deze kanalen bedreigen. Vervolgens worden in paragraaf 4.5 de bouwblokken besproken die de communicatie moeten beschermen tegen deze bedreigingen. Paragraaf 4.6 sluit het hoofdstuk af met bijzondere aandacht voor de opbouw van anonieme kanalen.

In het volgende hoofdstuk wordt er dan bekeken hoe deze bouwblokken kunnen gecombineerd worden tot concrete protocols voor de opbouw van anonieme kanalen.

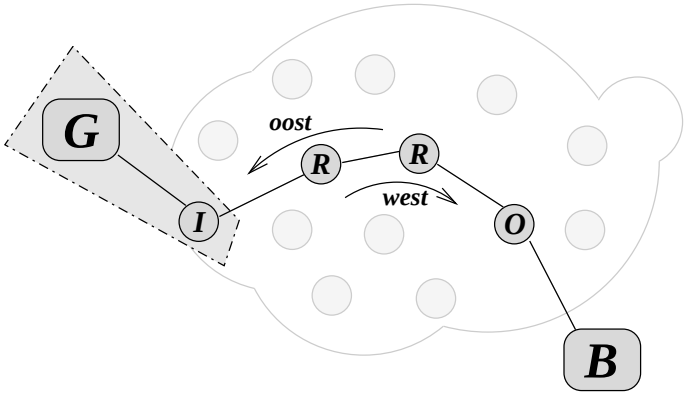
4.1 Topologie

Een anoniem kanaal bestaat in het algemeen uit een ketting van servers die data naar elkaar doorsturen. In principe gaat het hier niet noodzakelijk om peers in de wolk van een peer-to-peer netwerk, maar in de rest van dit hoofdstuk wordt dat wel verondersteld.

De eerste peer van het kanaal wordt de *aanvrager* of *initiator* genoemd, omdat hij het kanaal meestal opzet en controleert. De laatste peer is de *ontvanger*. Ze zijn het begin- en eindpunt van het kanaal en bevinden zich steeds **in** de wolk peers die de kanalen vormt. In figuur 4.1 worden ook *west*- en *oost*-richting gedefinieerd, respectievelijk van de initiator weg en naar de initiator toe.

Naar analogie met de originele betekenis van *relais* – een plaats waar men van paard kon wisselen – worden de peers tussen initiator en ontvanger in deze tekst zo genoemd. Zij geven berichten door aan elkaar. Typisch kennen zij enkel het pad van hun voorganger tot aan hun opvolger – ook wel *kanaalinterval* genoemd in deze tekst. De initiator kent het hele kanaal ofwel enkel het kanaalinterval tot de eerste relais.

De *gebruiker* is diegene die het anonieme kanaal wil gebruiken om zijn identiteit te verbergen voor de buitenwereld en/of voor de *bestemmeling*. Hij vraagt aan de initiator om een anonieme verbinding op te zetten naar de bestemmeling. Gebruiker en bestemmeling kunnen binnen **of** buiten de peer-to-peer wolk liggen. In het eerste geval heeft de gebruiker of de bestemmeling



Figuur 4.1: Topologie van een anoniem kanaal in wolk van peers. I is de initiator, O de ontvanger. G is de gebruiker, B de bestemmeling. R zijn relais. Gebruiker en initiator bevinden zich in dit voorbeeld op dezelfde computer.

dan de nodige peer-to-peer software op zijn eigen computer geïnstalleerd.

Peers kunnen meestal alle bovenbeschreven functies uitvoeren, vaak tegelijkertijd. Een wolk peers kan plaats bieden aan verschillende, gelijktijdige anonieme kanalen.

4.2 Anonimiteit bij communicatiekanalen

Intuïtief voelt iedereen aan wat anonimiteit betekent. Maar het is belangrijk om anonimiteit exact te definiëren. In de literatuur wordt anonimiteit op tal van manieren gedefinieerd. Bovendien zijn de verschillen tussen de definities vaak subtiel.

4.2.1 Definities

In Anonymity, unobservability and pseudonymity – a proposal for terminology [26] probeert men tot een eenduidige terminologie te komen. De terminologie in deze paper spitst zich toe op communicatiekanalen, maar kan ook algemener gebruikt worden.

Anonimiteit betekent niet identificeerbaar zijn binnen een groep van entiteiten, de anonimiteitsgroep. De anonimiteitsgroep is dus de groep van mogelijke verdachten. *Zenderanonimiteit* wil zeggen dat men niet weet wie de zender van een bericht is. *Ontvangeranonimiteit* daarentegen betekent dat men niet weet wie de ontvanger van een bericht is.

Onverbindbaarheid garandeert dat een entiteit meermaals gebruik kan maken van middelen of diensten zonder dat anderen in staat zijn deze gebruiken met elkaar te verbinden. Onverbindbaarheid staat dicht bij *relatieanonimiteit*. Daarbij weet men niet wie met wie communiceert. Zender en ontvanger kunnen dan niet met elkaar verbonden worden. Relatieanonimiteit is een zwakkere eigenschap dan zender- of ontvangeranonimiteit: men weet misschien wel wie welke berichten zendt en wie welke berichten ontvangt, zolang de relatie tussen de zender en ontvanger maar niet achterhaald kan worden.

Niet-waarneembaarheid verzekert dat een entiteit een middel of dienst kan gebruiken zonder dat anderen in staat zijn waar te nemen dat het middel of de dienst gebruikt wordt. Niet-

waarneembaarheid betekent meestal dat men berichten niet kan onderscheiden van 'willekeurige ruis'. Dit is bijvoorbeeld een wenselijke eigenschap bij steganografische systemen.

4.2.2 Graad van anonimiteit

Anonimiteit is sterker naar mate de anonimiteitsgroep groter is. In beschrijving van Crowds [27] worden een aantal niveaus van anonimiteit gedefineerd. Dit is een andere manier om uit te drukken dat de graad van anonimiteit kan verschillen. Deze niveaus kan je eenvoudig statistisch uitdrukken: de kans dat men een actie gedaan heeft, wordt p genoemd.

Bewijsbare identiteit: Het is bewezen wie ik ben, en ik kan daar de juridische gevolgen van dragen ($p = 1$).

Ontmaskerd: Men weet wie ik ben ($p \approx 1$).

Waarschijnlijk onschuldig: De kans dat ik het ben, is maar even groot als dat ik het niet ben ($p = 1/2$).

Buiten verdenking: Er is maar een even grote kans dat ik het ben, dan dat eender wie het is ($p = 1/n$ met n het aantal personen in de anonimiteitsgroep).

Absolute anonimiteit: Er is zelfs geen weet van de actie. De actie is dus niet-waarneembaar ($p = 0$).

4.2.3 Anonieme communicatiekanalen

Zoals reeds gezegd in paragraaf 3.2.2 bieden traditionele communicatiekanalen geen anonimiteit. Anonieme communicatiekanalen daarentegen leveren zenderanonimiteit en onverbindbaarheid – in de betekenis van relatieanonimiteit. Als men bovendien ook nog ontvangeranonimiteit wenst, gebruikt men best een double blinded kanaal.

4.3 Bedreigingsmodellen

De mogelijke tegenstanders worden opgedeeld in verschillende bedreigingsmodellen, en wel volgens verschillende criteria [28]. We beperken ons hier tot twee duidelijk onderscheidbare modellen die van toepassing zijn op peer-to-peer netwerken. Enerzijds kunnen **alle** peers, behalve één, *vals* zijn, anderzijds kunnen **alle verbindingen** geobserveerd worden. Een reële tegenstander is meestal een combinatie van deze twee modellen.

Valse peers In een peer-to-peer netwerk is elke peer onbetrouwbaar. Peers komen en gaan, en ze nemen actief deel aan de communicatie tussen twee willekeurige andere peers. Het is niet ondenkbaar dat er valse peers in het netwerk komen en dat ze actief ingrijpen in de netwerkstroom.

De *lokale tegenstander* die hier gedefinieerd wordt, kan de netwerkstroom op een *beperkt* aantal verbindingen *observeren* en eventueel ook *aanpassen*.

Globale afluisteraar Een alliantie van bedrijven, een grote multinational of zelfs een gouvernementele organisatie heeft in het slechtste geval toegang tot de fysieke verbindingen tussen *alle* peers. Al het netwerkverkeer kan door deze entiteit afgeluisterd worden, zonder dat hij de data kan aanpassen. Hij is met andere woorden *passief*.

Deze entiteit wordt hier gedefinieerd als de *globale afluisteraar*. Hij heeft een meer globaal beeld over het netwerkverkeer dan valse peers. Hij kan verbanden leggen en conclusies trekken uit een aantal observaties:

- pakketinhoud, -grootte en -snelheid
- het globale bandbreedtegebruik
- introductie en verdwijnen van peers
- openen en sluiten van netwerkverbindingen

Deze technieken worden vaak samengebald in de term *verkeersanalyse*. De passieve afluisteraar kan netwerkverkeer afluisteren, maar de informatiestroom niet aanpassen.

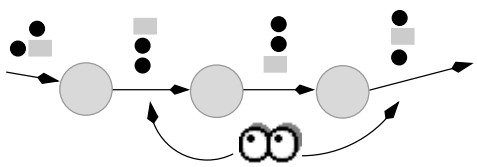
4.4 Aanvallen

In deze paragraaf worden enkele specifieke aanvallen besproken, zoals ze kunnen uitgevoerd worden door een valse peer of een globale afluisteraar. De meeste aanvallen zijn op zich niet voldoende om een anoniem kanaal bloot te leggen of om gebruiker en bestemmeling met elkaar te verbinden, maar combinaties van aanvallen kunnen wel voldoende zijn. Soms kan het herhaaldelijk toepassen van bepaalde aanvallen de anonimiteitsgroep verkleinen en daalt de graad van anonimiteit dus.

4.4.1 Verkeersanalyse

Aanvallen op boodschapsinhoud Indien boodschappen niet van vorm veranderen tijdens hun overdracht, kan hun traject gevolgd worden door een passieve afluisteraar.

Aanvallen op boodschapsgrootte Een globale, passieve afluisteraar kan ontdekken dat twee peers informatie uitwisselen door de pakketten op basis van hun grootte langs de tussenliggende verbindingen te volgen. Figuur 4.2 geeft een illustratie van deze en de vorige aanval.



Figuur 4.2: Verkeersanalyse kan het pad van boodschappen vrijgeven voor een passieve waarnemer, als de boodschappen onderscheidbaar zijn binnen een groep – eventueel onleesbare – boodschappen

Exhaustieve traceermethodes De aanvaller volgt elk mogelijk pad van een boodschap in het netwerk, terwijl hij alle verbindingen in deze paden afluistert. Bijvoorbeeld de volgorde van inkomende en uitgaande boodschappen in een peer kunnen hem helpen de afgelegde weg na te speuren. Met heel wat geluk kan hij zo gebruiker en bestemmeling met elkaar verbinden. Vaak wordt deze methode echter gebruikt om een lijst van mogelijke bestemmelingen vast te leggen.

Tijdstipaanvallen Een tegenstander kan de mogelijke eindpunten van een informatiestroom observeren. Hij zoekt naar tijdscorrelaties tussen gebeurtenissen aan het beginpunt en aan alle mogelijke eindpunten. Om deze aanval succesvol uit te voeren, moet hij in staat zijn deze gebeurtenissen te herkennen, dat kan bijvoorbeeld op twee manieren:

- De aanvaller kan onderscheid maken tussen data- en controlepakketten.
- De aanvaller analyseert het bandbreedtegebruik van het systeem. Dit is niet nuttig wanneer de bandbreedte van het netwerk volledige opgebruikt is.

Intersectieaanval De peers hebben in peer-to-peer netwerken een duidelijk offline/online gedrag. Een aanvaller kan proberen de peers te traceren door dit gedrag over een lange tijdsspanne te observeren. Bij deze aanval kan de aanvaller een bepaalde gebruiker ervan beschuldigen een bepaalde aanvraag uit te voeren. Hij onthoudt de gebruikers die online zijn telkens bepaalde informatie wordt opgevraagd. Als de aanvaller dit vaak genoeg doet en de doorsnede neemt van die gebruikers, kan hij de gebruiker ontdekken.

Men spreekt ook over een *profielaanval*, aangezien de aanvaller profielen opstelt van het gedrag van de gebruikers, en hieruit conclusies trekt. Er werd reeds een uitgebreide evaluatie gemaakt van de haalbaarheid van deze aanval in [29].

4.4.2 Actieve aanvallen

Boodschappen etiketteren Een actieve aanvaller of valse peer kan de boodschappen zodanig manipuleren dat hij ze herkent bij het verlaten van alle mogelijke uitgangspunten van het anonieme kanaal, of van alle uitgangspunten waar de aanvaller in geïnteresseerd is. Enkele voorbeelden:

Schaduwboodschappen: De aanvaller stuurt boodschappen die dezelfde weg volgen als de boodschap die hij wil traceren. Bijvoorbeeld, hij herhaalt (*replay*) boodschappen met wisselende frequentie, zodanig dat hij ze herkent bij het verlaten van het kanaal.

Boodschappen uitstellen: Een valse peer wacht met het doorsturen van ontvangen boodschappen. Als hij zo het bandbreedtegebruik van het netwerk kan beïnvloeden, kan hij informatie vergaren over de route van de pakketten. Het volstaat het bandbreedtegebruik op verschillende punten in het netwerk statistisch te analyseren.

Flooding-aanval De aanvaller kan de anonimiteitsgroep gevoelig verkleinen, door het systeem te verzadigen met zijn eigen boodschappen (men spreekt van *flooding* van het systeem). De weinige andere boodschappen in de netwerkstroom zijn zo gemakkelijker traceerbaar.

Het eenvoudigste voorbeeld van een flooding-aanval, is de $t-1$-*aanval* [28]. Als elke peer wacht op t boodschappen alvorens ze verder te sturen, kan een aanvaller er $t-1$ boodschappen sturen. Het is dan triviaal ingaande en uitgaande boodschappen met elkaar te associëren.

Allianties Een verbond van valse peers staat sterker in het naspeuren van informatiestromen dan enkelingen.

Peers uitschakelen Een gevoelig aantal routes kunnen geschrapt worden, wanneer een actieve aanvaller bepaalde peers ontoegankelijk maakt. Men noemt deze aanval ook wel een *denial-of-service*-aanval. Een adaptieve aanvaller kan de peers uitkiezen terwijl hij boodschappen traceert. Bovendien zullen peers waarvan de routes vernietigd werden zich anders gedragen in het netwerk. Zo kan een aanvaller bijvoorbeeld het kanaalopbouwproces proberen af te luisteren.

4.5 Bouwblokken

In deze paragraaf worden bouwblokken gedefinieerd die een anoniem kanaal implementeren door in te grijpen in de netwerkstroom. Dat kan zowel door *inhoud* als *volgorde* van de netwerkpakketten aan te passen. Sommige bouwblokken bieden expliciet weerstand tegen valse peers, andere zijn beter geschikt tegen verkeersanalyse. Vaak conflicteren beide vereisten, en moet er gezocht worden naar een compromis tussen bepaalde bouwblokken. Heel vaak gebeurt dit op een ad hoc wijze, zonder bewijsbare sterkte.

Voor een uitgebreide bespreking wordt er verwezen naar het APES project [30], waar performantie en sterkte van de verschillende bouwblokken geëvalueerd worden. In de volgende paragrafen worden de bouwblokken aangehaald die toepasbaar zijn op anonieme kanalen in peer-to-peer systemen.

4.5.1 Oplossingen tegen verkeersanalyse

Het is onmogelijk passieve aanvallen te ontdekken. De enige remedie bestaat erin ze onmogelijk te maken.

Ingangs- en uitgangspunten De 'wolk' peers die instaat voor het vormen van het anonieme kanaal kan één of meerdere ingangs- en uitgangspunten hebben. In het laatste geval worden exhaustieve traceertechnieken en tijdstipaanvallen veel moeilijker gemaakt, aangezien de verzameling mogelijke verbindingen enorm stijgt.

Vercijfering Men spreekt van *linkvercijfering* wanneer het netwerkverkeer tussen peers, van relais tot relais, vercijferd wordt. Linkvercijfering maakt de $t-1$-aanval onmogelijk, want de aanvaller kan ingaande en uitgaande boodschappen niet meer met elkaar associëren. Wanneer een extra laag encryptie voorzien is tussen begin- en eindpunt van het anoniem kanaal, spreekt men van *eindpuntvercijfering*. Beide vormen van vercijfering zijn gevoelig voor aanvallen op de boodschapsinhoud, want de berichten hebben bij elke relais dezelfde vorm.

Bij communicatieprotocols wordt deze bouwblok meestal gecombineerd met een bouwblok die authentisering en/of integriteit van de data verzekert, zoals besproken wordt in paragraaf 2.1.2.

Gelaagde vercijfering De aanval op de boodschapsinhoud wordt uitgeschakeld door het proces van vercijfering in verschillende lagen. De gebruikte vercijferingsalgoritmes kunnen zowel symmetrisch als asymmetrisch zijn.

In de *west*-richting – van initiator naar ontvanger – gaat dat als volgt:

1. De boodschap wordt klaargemaakt voor verzending door het anonieme kanaal met relaispeers $t_1, t_2, \ldots, t_l$ op zijn pad. De initiator vercijfert de boodschap eerst met de sleutel van ontvanger t_l, dan met de sleutel van relais t_{l-1}, enzoverder.

2. Bij elke relais t_i wordt een laag vercijfering verwijderd, en wordt de boodschap verder gestuurd naar relais t_{i+1}.

3. De boodschap komt aan bij peer t_l, de ontvanger. Deze verwijdert de laatste laag vercijfering. Ofwel is deze peer de bestemmeling van de boodschap, ofwel wordt de boodschap in klaartekst doorgestuurd naar de bestemmeling.

In de *oost*-richting – van ontvanger naar initiator:

1. De ontvanger en elke relais t_i voegen een laag vercijfering toe en sturen de boodschap door naar peer t_{i-1}.

2. De initiator ontvangt de vercijferde boodschap en verwijdert l lagen vercijfering.

De grootste rekentaak ligt hier bij de initiator. Vooral bij asymmetrische vercijfering mag dit effect bij het ontwerp van real-time peer-to-peer toepassingen niet over het hoofd gezien worden, bijvoorbeeld bij het ontwerp van ‘geïntegreerde’ toepassingen.

Padding Boodschappen worden zo gecodeerd dat hun grootte vast is. Dit gebeurt door opvullen tot vaste grootte of door splitsen in kleinere stukken van vaste grootte. Dit biedt dus bescherming tegen aanvallen op de boodschapsgrootte.

Herordening Boodschappen verschijnen niet noodzakelijk in dezelfde volgorde aan de uitgang van een peer als de volgorde waarin ze binnengekomen zijn. Boodschappen worden tijdelijk bijgehouden in een buffer of *boodschappenpoel*. Deze techniek is vooral effectief tegen exhaustieve traceermethodes en tijdstipaanvallen.

Er bestaan verschillende strategieën om de boodschappen te herordenen. Er kan enige bescherming geboden worden tegen de $t-1$-*aanval*, wanneer de relais steeds enkele boodschappen in de boodschappenpoel houdt.

Uitstel van verzending Peers kunnen de verzending van boodschappen een willekeurig korte tijd uitstellen. De motivatie is simpel: probeer tijdstipaanvallen te bemoeilijken door de tijdsspanne waarover een boodschap zich in het anonieme kanaal bevindt, willekeurig te maken. Bovendien kan er een minimum tijdsspanne opgelegd worden aan de tijd dat een boodschap zich in het kanaal bevindt. Dit belet dat de aanvaller een lange route kan uitsluiten, omdat een pakket nooit binnen die tijd zou kunnen aankomen.

Willekeurige kanaallengte Een tweede methode om tijdstipaanvallen tegen te gaan bestaat erin het aantal peers dat het kanaal vormt willekeurig te maken. Toch zijn deze en de vorige methode weinig nuttig voor real-time peer-to-peer toepassingen, aangezien te grote vertragingen de toepassing onbruikbaar kunnen maken.

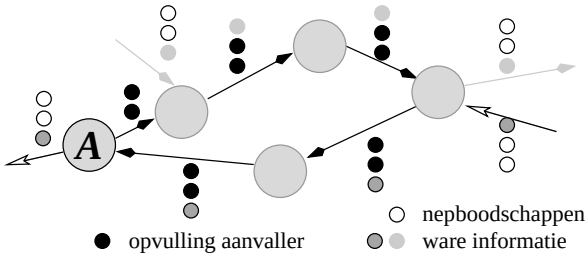
Opvullen *Nepboodschappen* zijn boodschappen zonder informatieve inhoud. Ze worden verstuurd om het aantal boodschappen per tijdsinterval vast te houden. Dat biedt bescherming tegen tijdstipaanvallen.

Verschillende criteria voor het versturen van nepboodschappen kunnen gehanteerd worden.

- Wanneer er binnen een tijdsinterval te weinig boodschappen gereed zijn voor versturing.
- In elk tijdsinterval, ook wanneer er genoeg wachtende boodschappen zijn.

De laatste aanpak biedt beter weerstand tegen de *flooding*-aanval.

Nepboodschappen worden soms niet alleen op directe verbindingen tussen peers verstuurd (*linkopvulling*), maar ook tussen de eindpunten van het kanaal (*eindpuntopvulling* of *kanaalopvulling*). Bij één bepaalde aanval [31] legt de aanvaller een anoniem kanaal in een lus, terug tot bij zichzelf. Vervolgens drijft hij de netwerkstroom op, totdat de vastgelegde bandbreedte tussen de relais-peers verzadigd is. Zoals men kan zien op figuur 4.3 vloeien er dan geen nepboodschappen meer tussen de relais-peers. Een passieve afluisteraar kan nu op alle verbindingen het andere, echte netwerkverkeer onderscheiden en de ware boodschappen traceren.



Figuur 4.3: Aanval op een netwerk waar enkel de bandbreedte tussen de individuele relais-peers vast gehouden wordt. Aanvaller *A* verzadigt de zelf aangelegde lus en kan de ware informatie volgen door rekening te houden met zijn eigen bandbreedtegebruik.

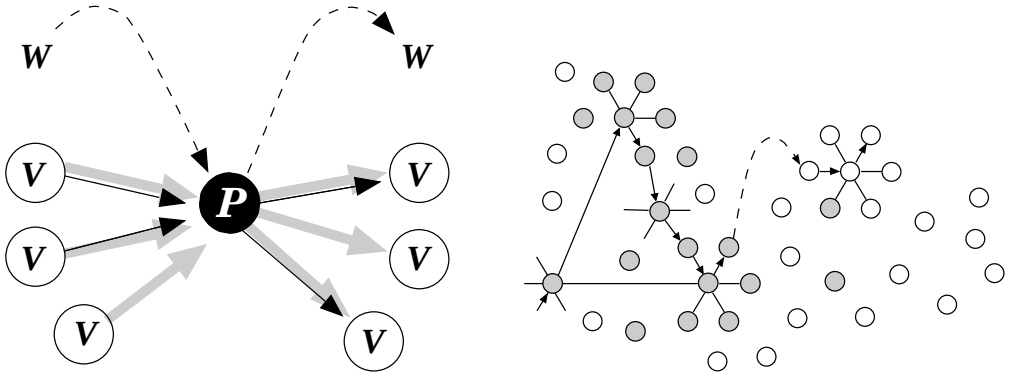
Raymond beschrijft in [28] een methode om de sterkte van deze bouwblok te evalueren en om de juiste parameters te kiezen.

Informatiesubstitutie Alle informatie die een gebruiker kan identificeren, wordt uit de boodschappen gehaald door de initiator. Cookies zijn een goed voorbeeld, maar ook andere informatie die netwerkefficiëntie tot doel heeft kunnen ongewenste details over het kanaal vrijgeven. In paragraaf 3.3 worden hier 2 voorbeelden van gegeven.

- Freenet gebruikt een teller die weergeeft hoe vaak een bericht nog mag doorgestuurd worden (hops-to-live), zodat aanvragen niet oneindig lang doorgestuurd worden.
- Bij elk bericht wordt het adres vermeld van de afzender zodat peers zijn adres leren kennen.

Hierdoor lekt er informatie over de afzender.

Verhoogde connectiviteit Men wil de anonimiteitsgroep voor globale afluisteraars zo groot mogelijk houden. Dat kan op twee manieren.



(a) Verhoogde connectiviteit in peer *P*: Volledige stroom (nep + echt) in grijs, echte informatie tussen nabootsers *V* in zwart en de stroom op 'willekeurige' sprongen (aangeduid door *W*) vloeit langs de stippellijnen.

(b) Willekeurige sprongen vermijden dat allianties van valse peers volledige kanalen kunnen beïnvloeden. Valse peers zijn grijs gekleurd.

Figuur 4.4: Verhoogde connectiviteit

- **Volledige connectiviteit.** In een netwerk met *n* peers worden tussen alle peers verbindingen gelegd en onderling een constante netwerkstroom onderhouden. Dat is uiteraard nefast voor de netwerkefficiëntie.

- **Nabootsers.** Elke peer zorgt ervoor dat hij steeds verbonden is met *k* andere peers uit zijn omgeving – ook wel zijn *nabootsers* of *mimics* [23] genoemd. Ze onderhouden onderling een constante netwerkstroom, eventueel opgevuld met nepboodschappen, zoals geïllustreerd op figuur 4.4(a). Anonieme kanalen worden om efficiëntieredenen bij voorkeur langs deze verbindingen opgezet.

Deze bouwblok verhoogt de weerbaarheid tegen exhaustieve aanvallen, aangezien het aantal mogelijke paden bij elke peer steeds met een factor *k* vermenigvuldigd wordt. Bij kanalen met lengte *l* moet de aanvaller dus steeds k^l mogelijke paden nagaan. De nabootsers verzekeren dat ook bij laag netwerkgebruik het aantal mogelijke paden voor een afluisteraar hoog wordt gehouden.

Een open probleem bij deze bouwblok is de juiste keuze van de omliggende peers. Een actieve aanvaller mag geen invloed hebben op deze keuze, anders zou het mogelijk zijn de netwerkstroom te beïnvloeden. Figuur 4.4(b) illustreert het probleem dat niets een alliantie van peers belet elkaar te kiezen om de verbindingen op te zetten. Een gedeeltelijke oplossing voor dit probleem bestaat erin de kanalen niet enkel langs de vaste verbindingen te leiden. Ook 'willekeurige' sprongen mogen dan door de initiator gekozen worden.

4.5.2 Oplossingen tegen actieve aanvallen

Extra pakketinformatie Het toevoegen van extra pakketinformatie kan enkele aanvallen moeilijker maken.

- Een nonce kan schaduwboodschappen verhinderen.

- Tijdsstempels laten toe dat gedetecteerd wordt dat een valse peer boodschappen vertraagt.

Bandbreedtebeperking Er wordt een bovengrens gesteld aan de absolute bandbreedte die wordt toegekend aan elke inkomende verbinding. Dat kan op een simpele manier toegepast worden door maar een beperkt aantal inkomende verbindingen toe te laten en in elk tijdsinterval maar één boodschap per inkomende verbinding te verwerken.

Reputatiesystemen Een variant op het reputatiesysteem zoals besproken in paragraaf 2.1.3 kan gebruikt worden om de bandbreedtebeperking, zoals hierboven besproken, adaptief te maken. Boodschappen afkomstig van peers met een goede reputatie krijgen in zo'n schema voorrang op andere boodschappen. Men kan er ook voor kiezen deze techniek enkel toe te passen wanneer de netwerkverbindingen verzadigd geraken.

Als in het protocol een afspraak gemaakt wordt over de bandbreedte waarover een inkomende verbinding beschikt, kunnen peers die hiertegen zondigen, afgestraft worden en hun aansluitingen geweigerd worden. Reputatiesystemen worden ook gebruikt om bescherming te bieden tegen de *flooding*-aanval [32].

Elektronische betalingen In sommige systemen [32, 15] worden elektronische betalingen, zoals besproken in paragraaf 2.1.3, gebruikt om denial-of-service-aanvallen tegen te gaan. Bandbreedte wordt in dat geval ingewisseld voor digitaal geld of voor een bewijs van werk.

4.5.3 Actief versus passief trade-off

Het grootste probleem bij het ontwerp van een anoniem en onobserveerbaar systeem bestaat erin het juiste compromis te vinden tussen de bouwblokken die beschermen tegen valse peers en de bouwblokken die beschermen tegen verkeersanalyse door de globale passieve afluisteraar. Vaak worden ad hoc oplossingen gevonden, maar de sterkte van zulke oplossingen is moeilijk theoretisch te evalueren.

De afweging tussen vaste verbindingen tussen nabootsers, als bescherming tegen passieve afluisteraars, en willekeurige verbindingen, als bescherming tegen actieve valse peers, voor de bouwblok van verhoogde connectiviteit is een mooi voorbeeld van zo'n ad hoc compromis.

4.6 Kanaalopbouw

4.6.1 Keuze van het pad

Het pad van anonieme kanalen kan vast zijn, verschillend voor elke verbinding, of verschillend voor elk bericht. Ook al lijkt het dat variabele paden de anonimiteitsgroep vergroten, toch kunnen ze nadelen hebben [33].

Voor de keuze van een variabel pad zijn er twee mogelijkheden:

1. De initiator legt het pad op voorhand vast.

2. De relais-peers beslissen onafhankelijk de volgende relais in het pad. De initiator heeft geen controle over het gekozen pad, noch kent hij de tussenliggende relais. Afspraken maken i.v.m. geheime sleutels worden in dat opzicht moeilijk, wat de beveiliging van het pad ook moeilijker maakt. Gelaagde vercijfering is in dat geval problematisch.

Het systeem waarbij de initiator alle relais bepaalt, is kwetsbaar voor de *setup-aanval* [29]. In deze aanval wordt een onschuldige peer in het netwerk als initiator afgeschilderd, door de ware initiator of door een groep van initiators.

De criteria bij de selectie van de volgende peer in het pad kunnen verschillen:

- Willekeurige keuze uit lijst van alle mogelijke peers.
- Wanneer de bouwblok van verhoogde connectiviteit uit paragraaf 4.5.1 wordt toegepast, geeft men voorkeur aan één van de vaste verbindingen naar een nabootser.
- Een andere heuristiek, bijvoorbeeld een verbinding waar bandbreedte op overschot is. Dat maakt het netwerk wel gevoeliger voor aanvallen op de bandbreedte, omdat valse peers die invloed kunnen uitoefenen op het bandbreedtegebruik van het netwerk dan ook de keuze kunnen beïnvloeden.

4.6.2 Afspraken

Alvorens informatie kan uitgewisseld worden langs een anoniem kanaal, moeten er afspraken gelden tussen de relais. Dat is nodig om de oorspronkelijke inhoud en volgorde van de netwerkpakketten te herstellen.

De relais-peers in het kanaal moeten overeenkomen welke bouwblokken gebruikt worden en welke parameters hierbij gebruikt worden. Vaak gaat het hier om de sleutel in een vercijferingsalgoritme, maar ook andere parameters in het protocol zijn denkbaar. Enkele voorbeelden:

- Relais-peers moeten informatie verkrijgen over de route van pakketten in hun interval van het kanaal. Dit noemt men *vorige- en volgende-relais-informatie*.
- Identificatienummers om pakketten aan welbepaalde stromen te verbinden.

De afspraken kunnen op twee manieren tot stand komen.

- Peers beslissen onderling de parameters van de netwerkstroom voor het kanaalinterval tussen hen beide.
- De initiator legt alle afspraken vast en stuurt ze door naar elke relais en naar de ontvanger. Dit gebeurt recursief: telkens er een stukje van het anonieme kanaal opgebouwd is, wordt het gebruikt om de afspraken naar de volgende relais door te sturen.

Een combinatie van beide is ook mogelijk.

4.6.3 Beveiligde uitwisseling van afspraken

Uiteraard moeten de afspraken beveiligd verstuurd worden. Het komt erop neer dat de peers onderling geheimen moeten delen, of dat de initiator een geheim deelt met elke relais en met de ontvanger. Zoals in paragraaf 2.1.2 beschreven is, kan men kiezen om al dan niet een publieke-sleutel-infrastructuur te gebruiken. Het gebruik van een PKI maakt tussenpersoonaanvallen onmogelijk, maar systemen om deze sleutel-infrastructuur gedecentraliseerd te realiseren werden nog maar recent onderzocht en staan nog in hun kinderschoenen.

4.6.4 Persistente verbindingen

Soms is het te verkiezen bestaande verbindingen tussen peers te herbruiken, zodat het opzetten van een kanaal niet lijkt op een bliksemschicht in het donker. Wanneer een peer toetreedt tot het netwerk, maakt hij een vast aantal verbindingen met andere peers. De *nabootsers* die in paragraaf 4.5.1 aan bod kwamen, implementeren dit principe. Uiteraard mag het toetreden van een peer ook geen informatie lekken.

4.6.5 Robuustheid

Bij het falen van een verbinding tussen twee peers kan het kanaal best stuk voor stuk heropgebouwd worden. Als het hele kanaal zou worden afgebroken, wordt het gemakkelijker voor tegenstanders om de verbindingen tussen peers te traceren. Zij zouden op basis van tijdscorrelaties tussen de hoeveelheid netwerkverkeer en het falen van de verbindingen stukken kanaal kunnen blootleggen.

Hoofdstuk 5

Evaluatie van protocols voor anonieme kanalen

In dit hoofdstuk worden een aantal protocols besproken voor anonieme kanalen.

Eerst wordt uitgelegd op welke manier de protocols geïmplementeerd kunnen worden. Dan wordt een opsomming van een aantal protocols gegeven. Hierbij ligt de nadruk op de gebruikte bouwblokken en de sterkte van de anonimiteit. Telkens worden ook de bekendste implementaties vermeld. Het zal blijken dat de meeste implementaties niet volledig gedecentraliseerd zijn en vaak centraal beheerd worden.

Het doel van dit hoofdstuk bestaat erin een geschikt protocol te vinden voor anonieme communicatie. Er wordt ook gekeken of het kan gebruikt worden bij het ontwerp van een anoniem peer-to-peer netwerk. Op het einde wordt, als voorbeeld, eens kort besproken welke protocols in aanmerking komen om de anonimiteit in Freenet te verbeteren.

5.1 Lagenmodel

Anonieme kanalen kunnen op verschillende lagen uit het OSI model gerealiseerd worden.

- Op de *netwerklaag* werkt men volledig transparant voor de uiteindelijke toepassing. Men kan alle internetverkeer anoniem maken, zowel TCP als UDP. Als op de toepassingslaag echter het internetadres van de zender expliciet nog eens vermeld wordt, is de zenderanonimiteit om zeep. Dit is het nadeel van de transparantie en toepassingsonafhankelijkheid.

- Werkt men op de *transportlaag*, dan kan men enkel UDP of TCP anonimizeren, wat misschien een probleem is voor sommige toepassingen. De toepassing zal licht aangepast moeten worden. De transparantie is dus iets meer beperkt.

- Als het anoniem kanaal op de *toepassingslaag* geïmplementeerd wordt, is de implementatie totaal geïntegreerd in de toepassing. De graad van anonimiteit kan dan sterker gemaakt worden dan bij een implementatie op de netwerklaag of transportlaag, omdat men de gebreken van de toepassing kent.

De protocols die in de rest van dit hoofdstuk besproken worden, kunnen in principe al deze lagen geïmplementeerd worden. In de praktijk bestaan veel systemen die op de toepassingslaag werken: er is vooral vraag naar systemen om anoniem te surfen en anoniem e-mail te versturen.

5.2 Proxies

Simpele proxies zijn vaste servers die boodschappen in naam van andere gebruikers doorsturen. Ze bieden een heel beperkte vorm van (zender)anonimiteit doordat ze informatiesubstitutie toepassen. Afzenderadressen, cookies en andere persoonlijke informatie worden verwijderd.

Soms wordt linkvercijfering toegepast tussen gebruiker en proxy, ter bescherming tegen een lokale afluisteraar, maar proxies zijn niet ontworpen met verkeersanalyse in het achterhoofd. *Gecascadeerde proxies* zijn enkele proxies die een *vaste* ketting vormen, met ertussen linkvercijfering. Dit systeem is niet bestand tegen allianties van valse peers. Proxies kunnen in de praktijk moeilijk een nuttige functie vervullen in een peer-to-peer systeem, aangezien ze de gedecentraliseerde natuur ervan compleet negeren.

5.3 Probabilistische routering

Er zijn enkele systemen uitgedacht die vertrekken vanuit de gedecentraliseerde natuur van de peer-to-peer architectuur. Ze hebben zowel zenderanonimiteit als onverbindbaarheid van gebruiker en bestemmeling tot doel. Deze systemen hechten groot belang aan de bescherming tegen exhaustieve aanvallen en allianties van valse peers. Toch zijn ze niet ontworpen om weerstand te bieden tegen een krachtige tegenstander die alle verbindingen afluistert.

De keuze van het pad in deze systemen ligt bij de relais-peers. Zij kiezen met waarschijnlijkheid $1 - p$ willekeurig de volgende peer in het pad. Met waarschijnlijkheid p wordt beslist om het kanaal te beëindigen en de boodschap naar de uiteindelijke bestemmeling te sturen.

Linkvercijfering wordt voorzien ter bescherming tegen lokale afluisteraars. Valse peers zien de boodschap nog steeds in klaartekst, informatiesubstitutie is noodzakelijk om de anonimiteit van de gebruiker te vrijwaren. De aanvrager kent het pad van de boodschappen niet, dus gelaagde vercijfering kan niet toegepast worden. Bijgevolg blijven deze systemen kwetsbaar voor aanvallen op de boodschapsinhoud.

Het anonieme kanaal is te vergelijken met een gecascadeerde proxy-ketting, maar het pad verschilt voor elke verbinding. In dat opzicht heeft een alliantie van valse peers het moeilijker om gebruiker met bestemmeling te verbinden.

5.3.1 Crowds

Crowds [27] is een implementatie van het voorgaande principe om anoniem te surfen. De peers in het netwerk worden *jondos* genoemd. Verder is er de *blender* die netwerkkennis verspreidt en sleutels verdeelt. Omdat de blender zich centraal bevindt en cruciaal is voor de werking van het netwerk, kan het als kwetsbaar knelpunt beschouwd worden.

Crowds verwerkt WWW-aanvragen en -antwoorden parallel, zodanig dat tijdspatronen die typisch zijn voor webverkeer niet in tijdstipaanvallen kunnen gebruikt worden.

5.3.2 Invisible IRC Project

Het Invisible IRC Project (*IIP*) [34] heeft anonieme IRC tot doel. In principe is het een verarmde versie van Crowds, met enkel een vaste IRC server als uitgangspunt uit de anonimiteitswolk. Het voordeel ten opzichte van Crowds ligt dan weer in het bredere toepassingsgebied aangezien IIP op de transportlaag werkt.

Er zijn wel enkele beperkingen aan IIP.

- Het netwerk is statisch. Het netwerk bestaat uit een groep toegewijde peers. Indien er nieuwe peers toetreden, moeten alle peers hun lijst met peers aanpassen. Het eindpunt van het kanaal (de IRC server) gebruikt zijn eigen 'lijst'. Er is hier maar één element, namelijk een referentie naar de lokale IRC server-poort.

- De gebruikers nemen niet actief deel aan het systeem.

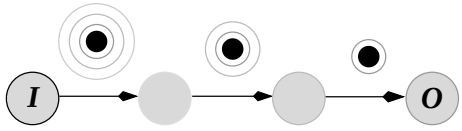
- Terwijl elke peer wel als toegangspunt tot de IIP-wolk kan dienen, is er maar één uitgangspunt uit de wolk van peers. De peer die als IRC server dient, is een duidelijk 'aanzuigpunt' van netwerkverkeer en als dusdanig kwetsbaar voor *denial-of-service*-aanvallen. Het aanzuigpunt maakt de anonimiteitsgroep voor een globale observator kwetsbaar klein.

5.4 Mixnets

In principe is een mix een peer die *herordening* en *gelaagde vercijfering* toepast. Traditioneel wordt publieke sleutel vercijfering gebruikt.

Chaum [35] heeft aangetoond dat voor een mixnet waar k peers het anoniem kanaal vormen, de gebruiker zelfs bij verkeersanalyse door een krachtige globale afluisteraar onverbindbaar blijft met de bestemmeling. Hier mogen alle $k+1$ verbindingen afgeluisterd worden of kunnen $k-1$ peers vals spelen, zonder dat de anonimiteit van het kanaal in het gedrang komt.

De gelaagde vercijfering maakt dat de boodschappen er als een 'ui' uitzien, met geneste lagen van vercijfering (figuur 5.1). De uien worden door de initiator samengesteld en vervolgens doorgestuurd naar de eerste relais. Deze ontdekt onder de eerste laag vercijfering de benodigde informatie over de volgende relais in de ketting. Op zijn beurt stuurt hij de (nu kleinere) ui door. Dit proces gaat zo voort tot de ui helemaal gepeld 'als bij wonder' bij de bestemmeling aankomt. Het pad dat de ui aflegt werd niet op voorhand afgesproken, de ui zelf bevat alle benodigde informatie.



Figuur 5.1: Ui-structuur van boodschappen in Chaum's model. Elke boodschap wordt ingepakt in verschillende lagen vercijfering.

Hoewel mixnets succesvol toegepast worden voor anonieme e-mail, hebben ze enkele belangrijke nadelen waardoor ze in basisvorm niet praktisch bruikbaar zijn bij het ontwerp van peer-to-peer toepassingen. Er moet immers rekening gehouden worden met het grote online/offline gedrag van de gebruikers.

1. Het pad van het anonieme kanaal is vast. Bovendien is alle moeite verloren als er een relais niet bereikbaar is. Er is geen directe manier voor de initiator om dit falen te ontdekken, of om te weten te komen waar en waarom dit gebeurde.

2. Publieke sleutel vercijfering vereist een publieke-sleutel-infrastructuur en is te traag voor real-time toepassingen.

3. Enkel éénwegsverkeer is mogelijk. *Reply-blocks* [36] worden hier als oplossing gebruikt voor e-mail, maar zijn niet praktisch in real-time communicatie.

5.4.1 Onion routing

Onion routing [37] verschilt op twee vlakken van standaard mixnets.

1. De communicatie gebeurt in twee richtingen en kan real-time.

2. De anonieme kanalen zijn toepassingsonafhankelijk, door gebruik van specifieke *proxies* die de toepassingsprotocols vertalen in het onion routing protocol.

Tweerichtingsverkeer wordt mogelijk gemaakt door het gebruik van langdurige verbindingen tussen de relais, of ook nog onion routers genoemd. Voor elk anoniem kanaal worden in een eerste fase de onion routers op het pad op de hoogte gebracht van het kanaal dat opgezet wordt, eveneens door het gebruik van een ui die zich door het netwerk beweegt. Bijgevolg is de initiator in deze fase nog steeds blind voor het mogelijke falen van de opzet van het kanaal.

Onion routers hebben als nevenvoorwaarde dat ze boodschappen maar een beperkte tijdsspanne kunnen bijhouden alvorens ze verder te sturen. De sterkte van tegen verkeersanalyse in vergelijking met Chaum's *mix servers* wordt zo beknot ten voordele van real-time netwerkverkeer.

De onion routers vormen een netwerk op zich, waar de gebruiker buiten staat. In het geval dat gebruiker of bestemmeling zich niet achter een betrouwbare onion router bevinden, zijn ze gevoelig voor verkeersanalyse aan de eindpunten van het anonieme kanaal.

5.4.2 WebMIX

WebMIX [32] is een systeem dat werd uitgedacht om anoniem te surfen. De auteurs stellen dat het netwerk net zo bestand is tegen verkeersanalyse als Chaum's mixnets, maar met een hogere netwerkefficiëntie en dus ook beter toepasbaar voor real-time toepassingen zoals surfen. In dat opzicht is dit systeem beter geschikt als vertrekpunt voor het ontwerp van peer-to-peer netwerken.

Om deze doelstelling te verwezenlijken werden enkele bouwblokken toegevoegd:

- Efficiënte padding en bandbreedtebeperking, gecombineerd in het adaptief *chop-and-slice* algoritme. Lange boodschappen worden versneden in kleinere pakketten en per tijdsinterval verzonden, met een gelijkmatige netwerkstroom tot doel.

- Opvullen van de netwerkstroom met nepboodschappen, ook tussen de eindgebruikers. De motivatie hiervoor is terug te vinden in paragaaf 4.5.1 en bijhorende figuur 4.3.

- Een variant op een reputatiesysteem als bescherming tegen denial-of-service- en flooding-aanvallen, op basis van *geauthentiseerde tickets*. Er wordt gebruik gemaakt van blinde digitale handtekeningen.

- Een nonce wordt toegevoegd aan de pakketten.

Alle gebruikers installeren een lokale 'proxy' (*JAP*) die de beschermende bouwblokken toepast. Ze plaatsen zich op die manier *in* het beveiligde netwerk.

Er is één controlerende *info-server* aanwezig die informatie voor de controle en werking van het netwerk verstrekt. De info-server plaatst blinde handtekeningen op tickets. Dat is inderdaad nodig, want het mixnet principe verplicht de initiator nog steeds op een externe manier de status van het netwerk te kennen, wil hij een goede kans hebben dat de datapakketten bij de bestemmeling zullen aankomen.

Deze laatste opmerking maakt dat ook deze architectuur moeilijk toepasbaar is bij het ontwerp van een anoniem peer-to-peer systeem, waar de peers een sterk online/offline gedrag vertonen. Bovendien doet de centrale info-server – als enkel punt van faling – afbreuk aan het gedecentraliseerde karakter van peer-to-peer netwerken.

5.5 Pipenet

Wei Dai beschrijft in [38] een protocol voor efficiënte anonieme connecties. Net zoals mixnets wordt herordening en gelaagde vercijfering toegepast. De vercijfering is echter symmetrisch. Het aantal berichten per tijdsinterval tussen gebruiker en bestemmeling wordt in beide richtingen constant gehouden, om verkeersanalyse tegen te gaan. Meer bepaald de tijdstipaanval kan nog moeilijk uitgevoerd worden, aangezien er voor een globale afluisteraar geen specifieke gebeurtenissen waarneembaar zijn. Er wordt enkel een vaste, onherkenbare stroom data waargenomen.

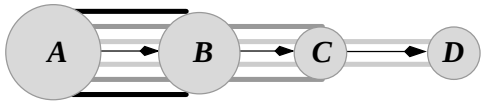
Het bedreigingsmodel is strikter dan bij mixnets. Actieve aanvallers kunnen boodschappen uitstellen zoals aangehaald in paragraaf 4.4.2. Pipenet lost dat als volgt op. In een tijdsinterval waarin een relais voor eender welk anoniem kanaal te weinig berichten ontvangt, wordt in dat tijdsinterval door die relais van geen enkel kanaal berichten doorgestuurd. Zo wordt ervoor gezorgd dat uitstel van berichten niet kan gebruikt worden om een berichten te traceren.

Het grote verschil met mixnets is het gebruik van persistente, robuuste en – eventueel – dynamische kanalen. Het anonieme kanaal wordt in een voorbereidingsfase helemaal opgebouwd. Pas daarna wordt de echte informatie uitgewisseld. Dit principieel verschil met mixnets is subtiel, maar zorgt ervoor dat de Pipenet-achtige protocols beter geschikt zijn bij het ontwerp van een anoniem peer-to-peer netwerk.

Pipenet steunt op het principe van vercijfering in *telescopische verbindingen*. Figuur 5.2 toont dat de anonieme kanalen eerder een telescopische vorm vertonen, zoals een 'prei'. De preistengels worden *recursief* in enkele stappen opgebouwd:

1. *A* en *B* komen een sleutel overeen
2. *B* wordt gevraagd een link te maken met *C*
3. *A* doet een sleutelovereenkomst met *C* via *B*
4. *A* vraagt *B*
    - alle pakketten van *A* te ontcijferen en naar *C* door te sturen
    - alle pakketten van *C* te vercijferen en naar *A* door te sturen

Het Pipenet protocol werkt met vaste pakketgrootte (padding) en past opvulling van de netwerkstroom toe. De neppakketten worden tussen de eindpunten van het kanaal verstuurd.



Figuur 5.2: De anonieme kanalen in *Pipenet* vertonen de vorm van een ‘prei’. De langdurige verbindingen worden met meerdere lagen symmetrische vercijfering beschermd.

Net zoals bij mixnets is één betrouwbare peer in het kanaal voldoende opdat gebruiker en bestemmeling onverbindbaar zijn. Ten opzichte van elkaar is de anonimiteit asymmetrisch. De gebruiker is anoniem, de bestemmeling minder.

Er dient opgemerkt te worden dat het probleem van de sleutelovereenkomst hier niet opgelost wordt. Er is nog steeds nood aan een publieke-sleutel-infrastructuur. Ofwel moet er een (anonieme) Diffie-Hellman sleuteluitwisseling gebeuren, die gevoelig is voor de tussenpersoon-aanval.

5.5.1 Freedom

Het Freedom protocol is de commerciële implementatie van een protocol dat lijkt op het Pipenet protocol. Het netwerk is opgebouwd uit een aantal vaste servers, die beheerd worden door ZeroKnowledge, het bedrijf achter Freedom.

Weliswaar ontbreken er twee essentiële bouwblokken die in Wei Dai’s specificatie aan bod komen:

- Vast bandbreedtegebruik door middel van neppakketten tussen de eindpunten van het kanaal.
- Extra pakketinformatie zoals MAC’s, meer bepaald voor **elke** relais, als bescherming tegen actieve aanvallen door valse peers.

Wei Dai beschrijft twee mogelijke aanvallen op het Freedom protocol in [31].

5.5.2 Tarzan

Tarzan [23] is een onderzoeksproject aan het Massachusetts Institute of Technology. Men wil een anoniem peer-to-peer protocol in de netwerklaag construeren. Het protocol kan beschouwd worden als een aanhanger van het Pipenet protocol, met enkele toevoegingen en aanpassingen ten voordele van de efficiëntie, decentralisatie en transparantie.

Tarzan biedt zenderanonimiteit en onverbindbaarheid van zender en ontvanger. De idee van een *double blinded* kanaal, zoals reeds in paragraaf 3.2.2 werd besproken, kwam in dit project trouwens voor het eerst aan de orde als strategie om ontvangeranonimiteit te bereiken.

Mimics De *mimics* zijn de nabootsers in de bouwblok ‘verhoogde connectiviteit’ (zie paragraaf 4.5.1). Dit concept beschermt alle peers, niet enkel de servers in de kern, zoals bij Onion Routing of Freedom het geval is.

De mimics worden toegekend door middel van een protocol waarop actieve aanvallers geen invloed kunnen hebben. Dat biedt enige weerstand tegen de *setup-aanval* die in paragraaf 4.6 beschreven wordt, aangezien de initiator niet meer helemaal vrij is in de keuze van het pad.

De vermenging van willekeurige sprongen met het mimic-concept zoekt naar een compromis in bescherming tegen valse peers versus bescherming tegen krachtige globale verkeersanalyse. Dit werd reeds besproken in paragraaf 4.5.1. Men zoekt op die manier een tussenweg tussen Crowds en Pipenet.

Netwerkkennis en kanaalopbouw WebMIX, Freedom en Onion Routing steunen op een centrale authoriteit of server om de peers zich elkaar te laten authentiseren. Tarzan gebruikt Chord [21] – een gedistribueerd lokalisatieprotocol. Er werd een extra stap toegevoegd, die de authenticiteit van de opgezochte sleutels verifieert.

Tarzan werd uitgedacht met toepassing in een peer-to-peer netwerk in het achterhoofd. Het werd reeds voorgesteld als implementatie van anonieme kanalen in het Free Haven project [17].

5.6 Anonimiteit in Freenet verhogen

Zoals besproken in paragraaf 3.3 is de lezer- en uitgeveranonimiteit in Freenet tamelijk zwak. Freenet biedt ook geen afdoende bescherming tegen verkeersanalyse. In de Freenet paper wordt voorgesteld om aanvragen eerst langs een anoniem kanaal te sturen. Daarom is het interessant een opsomming te geven van de extra anonimiteitssterkte die de protocols kunnen geven.

- De anonimiteit van proxies is zwak. Dus het gebruik ervan in Freenet verhoogt de behaalde anonimiteit onvoldoende.
- In Freenet kunnen peers met een bepaalde kans *p* beslissen om de hops-to-live teller *niet* te verlagen. Als een peer een aanvraag stuurt met de hops-to-live teller op een bepaalde waarde, kan men niet langer met zekerheid zeggen dat deze peer de aanvraag genereert. Als in een *pre-routeringsfase* probabilistische routering gebruikt wordt, zullen aanvragen door een aantal peers doorgestuurd worden, zonder dat de hops-to-live waarde verlaagd wordt. Men creëert zo gewoon extra onzekerheid over de waarde van de teller, maar de graad van anonimiteit wordt hierdoor niet zwaar opgedreven.

  Probabilistische routering biedt ook weinig extra weerstand tegen verkeersanalyse.
- Mixnet en pipenet verhogen de anonimiteit aanzienlijk. Ze bieden immers bescherming tegen verkeersanalyse, wat ontbreekt in Freenet. Een aanvraag kan nu anoniem gestuurd worden naar een bepaalde peer, die dan de aanvraag in Freenet doet. Voor de Freenet peers lijkt het alsof deze peer de aanvraag zelf doet. De echte lezer of uitgever blijft echter anoniem.

Om de anonimiteit in Freenet te verbeteren heeft men de keuze tussen mixnet en pipenet als protocol voor een anonieme pre-routeringsfase. Beide kunnen dezelfde graad van anonimiteit en bescherming tegen verkeersanalyse realiseren.

Hoofdstuk 6

Ontwerp van een anoniem peer-to-peer kader

Er is altijd al vraag geweest naar een algemeen kader voor beveiligde en anonieme informatie-uitwisseling. Enerzijds wil men elke netwerktoepassing of -dienst die men zich kan inbeelden, beschermen en er anonimiteit aan verschaffen. Anderzijds kan men niet voorbij gaan aan de steeds stijgende rekenkracht en connectiviteit bij de eindgebruiker. Het ligt dan ook voor de hand dat het ontwerp van een algemeen kader met deze evolutie rekening houdt, en dat het dus bij voorkeur een peer-to-peer architectuur krijgt.

Het verwezenlijken van peer-to-peer systemen en toepassingen kan op verschillende manieren gebeuren. In de literatuur [39] worden twee mogelijke aanpakken tegen elkaar afgewogen. In een eerste geval tracht men een volledig systeem te bedenken dat aan alle vereisten voor een veilig en anoniem netwerk voldoet. Men spreekt hier van een *über-netwerk*. In een andere denkschool worden de beste eigenschappen van bestaande peer-to-peer systemen met elkaar gecombineerd in een soort *inter-connectie* van systemen.

Onze oplossing zoekt een compromis tussen beide denkscholen. Het systeem dat uitgedacht wordt, maakt anonieme kanalen tussen de peers. De juiste combinatie van opkomende peer-to-peer technieken, systemen en netwerken zal zo'n systeem verwezenlijken. Het ontwerp baseert zich op peer-to-peer systemen die zich zowel op het vlak van *zoektechnieken* en *informatie-opslag*, als op het vlak van *anonieme communicatie* situeren.

Deze aanpak heeft twee voordelen. Enerzijds houdt het ontwerp rekening met de opkomende revolutie in het peer-to-peer domein. Anderzijds tracht het ontwerp een algemene oplossing te bieden, opdat eender welke bestaande netwerkdienst of -toepassing zijn communicatie anoniem en veilig kan maken. Hopelijk kan de oplossing dan ook nog toegepast worden op bestaande peer-to-peer systemen die al een beperkte vorm van beveiliging of anonimiteit aanbieden. Zo kan de toegevoegde waarde geëvalueerd en vergeleken worden.

In dit hoofdstuk worden eerst de vereisten van een beveiligd en anoniem peer-to-peer kader besproken. Dan wordt de algemene strategie besproken, waarbij men uitgaat van een juiste combinatie of inter-connectie van gekende bouwblokken, protocols en subsystemen. Vervolgens worden de gekozen algoritmes en datastructuren ervan beschreven. De implementatie, vertaling naar programmacode en evaluatie van de bekomen resultaten komen aan bod in het volgende hoofdstuk.

6.1 Vereisten

Beveiliging en anonimiteit Een anonieme peer-to-peer structuur moet weerstand bieden tegen de twee onderscheidbare bedreigingsmodellen uit paragraaf 4.3. Er moet dus een protocol ontworpen worden om anonieme kanalen te realiseren. Men kan best vertrekken van één van de beproefde protocols uit hoofdstuk 5.

Toepassingsonafhankelijkheid Er is een grote vraag naar een universele en onderliggende 'anonimizeringslaag' voor gebruik in allerhande systemen. Zo'n systeem kan op een transparante manier een hele reeks toepassingen een verhoogde graad van anonimiteit bieden. Bovendien moet het systeem transparant zijn voor de toepassing, zodat de toepassing die ervan gebruikt maakt, niet moet aangepast worden.

Inherent aan het peer-to-peer concept De pijlers van het peer-to-peer principe komen hier weer aan bod:

Decentralisatie Het systeem kan geen centraal beheerde publieke-sleutel-infrastructuur of peer-lokalisatiesysteem veronderstellen. Toch is het voor het slagen van elk anoniem systeem van cruciaal belang dat peers elkaars adres en eventueel elkaars publieke sleutels leren kennen. Het eenvoudigste is om hiervoor een bestaand systeem te gebruiken.

Als het anonimizeringssysteem in een peer-to-peer toepassing gebruikt wordt, bestaat natuurlijk het nadeel dat je dubbel werk doet: de peer-to-peer toepassing zal ook een eigen peer-lokalisatiesysteem hebben.

De rand van het netwerk plooit naar binnen Het zijn de peers zelf die ervoor zorgen dat het peer-to-peer netwerk anoniem wordt. Zij moeten er dus zelf voor instaan dat de communicatie tussen twee willekeurige peers anoniem gemaakt wordt.

Dynamiek De informatie over het netwerk moet dynamisch bijgehouden worden. Deze eis wordt bovendien nog problematischer ten gevolge van het gedecentraliseerde karakter. Plotse verdwijningen van peers uit het netwerk moeten opgevangen worden, om de anonieme connecties in stand te houden.

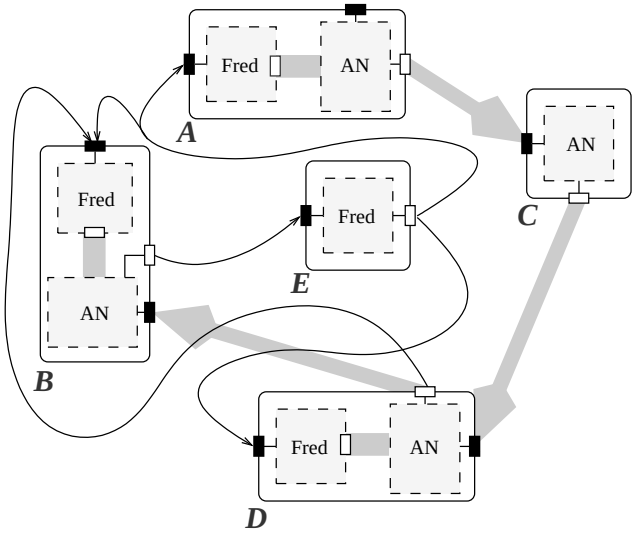
6.2 Systeembeschrijving

In deze paragraaf wordt de structuur en de werking van het ontworpen systeem beschreven, en worden bepaalde keuzes kort toegelicht vanuit de vereisten en besproken bouwblokken. De terminologie uit paragraaf 4.1 wordt herbruikt.

6.2.1 Overzicht

Het anonimizeringssyteem is een combinatie – een inter-connectie – van afzonderlijke autonome componenten. Dit laat toe om reeds bestaande systemen te gebruiken. Het is mogelijk om eventueel later een bepaalde component te vervangen.

Het opzetten van *anonieme kanalen* wordt gerealiseerd door een component die *AnonNet* genoemd wordt, naar analogie met een éénmansproject in ontwikkeling, gestart door William



Figuur 6.1: Een bestaande peer-to-peer toepassing met onderliggend peer-to-peer netwerk. Fred's zijn Freenet software modules, AN zijn AnonNet software modules. Witte connectoren zijn uitgaande poorten, zwarte connectoren inkomende. Op hybride peers zijn Fred en AN verbonden door het onderscheppingsmechanisme: de witte Fred connector is hier verbonden met AN. Gewone Freenet netwerkstroom vloeit langs de zwarte lijnen, dikke grijze verbindingen zijn anonieme kanalen. Niet alle mogelijke verbindingen worden afgebeeld.

Ahern [40]. AnonNet bevindt zich op de transportlaag en kan gebruikt worden door toepassingen die anonieme verbindingen willen. De interactie met de uiteindelijke toepassing zal gebeuren via transparante *onderschepping van de verbindingsaanvragen*: de verbindingsaanvraag wordt dan aan AnonNet doorgegeven. Dit wordt in detail beschreven in paragraaf 6.2.3.

De *decentralisatie* van het systeem wordt verzekerd door de tweede component. Deze component houdt dynamisch bij welke peers actief zijn in het netwerk. Hiervoor wordt het bestaande systeem *Chord* gebruikt. Chord zorgt voor de efficiënte en snelle lokalisatie van peers. De AnonNet component zal Chord contacteren om het adres van een bepaalde peer op te vragen.

De *opslag* van de publieke sleutels van peers is de taak van een derde component, *DHash*. De AnonNet component kan DHash raadplegen om de publieke sleutel van een bepaalde peer te weten te komen. DHash en Chord zijn tamelijk sterk met elkaar geïntegreerd, want DHash gebruikt ook Chord. Chord en DHash, en de interactie tussen beide, worden in paragraaf 6.2.7 besproken.

Het anonimizeringssysteem bestaat dus uit *hybride peers*, aangezien elke peer deze drie verschillende componenten zal draaien.

Figuur 6.1 toont hoe het hele systeem – de combinatie van de drie componenten – interageert met een bestaande toepassing, hier Freenet.

6.2.2 AnonNet taken

De AnonNet component voert de volgende taken uit:

1. Beheer – opbouw, afbraak en herstel – van anonieme kanalen langs de door hem gekozen relais-peers. Dit zijn de taken van een initiator, telkens wanneer de software een aanvraag tot anonieme verbinding ontvangt.

2. Doorsturen van pakketten voor kanalen die andere peers hebben opgezet: de taak van een relais.

3. Doorsturen van pakketten naar de bestemmeling: de taak van de ontvanger.

6.2.3 Onderscheppen van verbindingsaanvragen

De software van de bestaande toepassing wordt niet aangepast. De uitgaande netwerkverbindingen van de toepassing worden onderschept en langs de AnonNet software geleid. Op dat moment wordt een anoniem kanaal van willekeurige lengte opgebouwd, gaande van nul (geen anoniem kanaal) tot maximaal vijf, waarvan vier relais-peers en de ontvanger. Het komt er dus op neer dat de rechtstreekse verbindingen die een bestaande toepassing probeert aan te leggen, af en toe vervangen wordt door een onrechtsreekse verbinding via een anoniem kanaal.

Figuur 6.1 illustreert een voorbeeld. Stel dat de toepassing op hybride peer *A* een verbinding wil maken met peer *B*. De aanvraag wordt onderschept op peer *A*, de AnonNet software maakt een kanaal via peers *C* en *D*. Aan peer *D* wordt gevraagd het kanaal te beëindigen en alle berichten door te sturen naar peer *B*, naar de poort waarop de toepassingssoftware luistert.

6.2.4 Netwerkmodel

Men onderscheidt twee netwerklagen in het ontwerp.

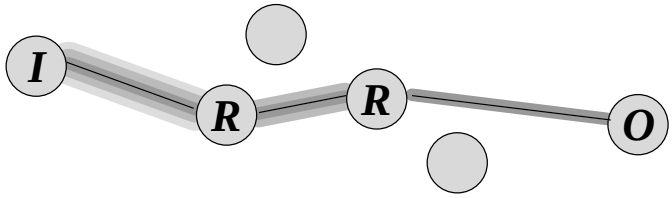
Linklaag Vooreerst zijn er de directe verbindingen tussen twee AnonNet peers. Ze worden *links* genoemd, en bevinden zich in figuur 6.2 tussen elk relais. De relais-peers krijgen opdracht van de initiator om links tussen elkaar te openen, maar ze kunnen herbruikt worden voor verschillende kanalen. Het gevaar dat, bij zijn opbouw, een kanaal als een bliksem oplicht en zo de initiator blootgeeft, wordt zo ook verkleind.

Er wordt een maximum aantal pakketten per seconde en per link vastgelegd. Er is geen vast opgelegde connectiviteit tussen de AnonNet peers, zoals bij het mimic concept [23]. Toch kan de anonimiteitsgroep vergroot worden door de idee van *virtuele circuits*, zoals verder besproken zal worden. Tussen twee opeenvolgende relais-peers in een anoniem kanaal worden soms controle-informatie of neppakketten uitgewisseld. Een stroomcijfer beschermt deze informatie voor afluisteraars.

Kanaallaag In deze laag liggen de anonieme kanalen, zoals ze beschreven worden door het Pipenet model. In dit ontwerp werd voor het Pipenet model gekozen omwille van de grote dynamiek van een peer-to-peer netwerk. In een Pipenet heeft de initiator goede controle en overzicht over de door hem geconstrueerde anonieme kanalen. Bij mixnets is het moeilijker om de staat van de anonieme verbindingen te achterhalen, of om zich er zeker van te stellen dat uitgezonden netwerkberichten goed aankomen. De initiator kan ingrijpen waar nodig, cruciaal om een reëel netwerk werkende te houden.

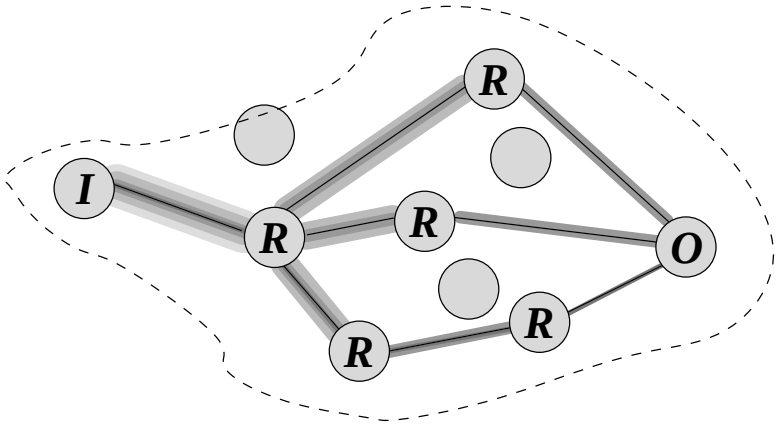
De initiator en ontvanger zijn ervoor verantwoordelijk dat langs de kanalen een vast aantal berichten per tijdsinterval vloeit. Bij gebrek aan echte informatie worden nepboodschappen verstuurd.

Enkelvoudige anonieme kanalen In een eerste, eenvoudige aanpak wordt een *enkelvoudige* anonieme connectie gelegd tussen initiator en ontvanger, waarover netwerkpakketten vloeien van oost naar west en vice versa. De 'prei' is een enkelvoudige ketting van relais-peers, zoals duidelijk gemaakt wordt in figuur 6.2.



Figuur 6.2: Een enkelvoudig anoniem kanaal. De *links* zijn in zwart getekend.

Virtuele circuits Verschillende enkelvoudige anonieme kanalen kunnen gebundeld worden in een *virtueel circuit*. Dit is een virtuele route tussen initiator en ontvanger, waarlangs een stroom bijeenhorende netwerkpakketten vloeit. Bijeenhorend wil hier zeggen dat ze allemaal behoren tot eenzelfde, onderschepte netwerkverbinding. De initiator staat in voor het toevoegen en weghalen van enkelvoudige kanalen uit de virtuele circuits, zoals figuur 6.3 illustreert. Initiator en ontvanger kennen een sessienummer toe aan elk virtueel circuit, waarmee berichten uit bijeenhorende kanalen bestempeld worden.



Figuur 6.3: Virtueel circuit met 3 anonieme kanalen.

Vooreerst biedt dit concept een grotere robuustheid in een dynamisch netwerk van verdwijnende peers, maar bovendien vervult het ook de wens naar *hoge connectiviteit* tussen peers. Het effect op de grootte van de anonimiteitsgroep is positief, terwijl de redundante verbindingen toch nog nuttig gebruikt worden.

6.2.5 Pakketoverdracht

De pakketten hebben een vaste lengte, en zijn ingepakt in verschillende lagen vercijfering. Figuur 6.4 toont zo'n pakket.

Elk pakket is 129 bytes lang. De eerste byte bevat een controle-informatie voor de linklaag, de overige 128 bytes zijn bestemd voor de kanaallaag. De eigenlijke informatie bevindt zich in

de laatste 48 bytes van het pakket. Er zijn 5 keer 16 bytes voorzien als informatievelden voor de relais-peers.

Hierin vindt elke relais informatie over het type pakket:

Nepboodschap: Deze worden genegeerd.

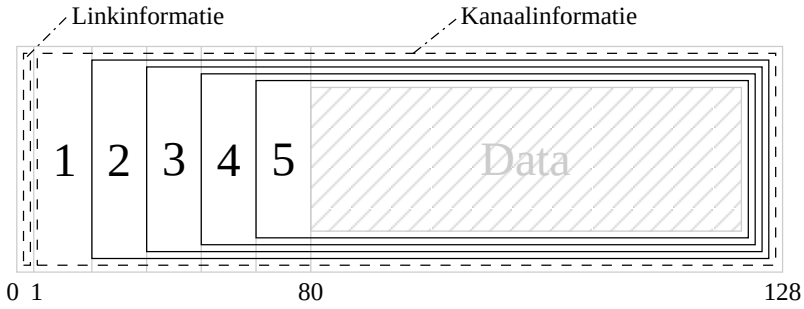
Forward: Pakket moet doorgestuurd worden.

Kanaalopbouwaanvraag: Deze boodschap wordt niet doorgestuurd. De aanvraag wordt behandeld door de relais in kwestie en het kanaalopbouwprotocol wordt opgestart. In de 'prei'-voorstelling betekent dit dat er een laag toegevoegd wordt van initiator tot aan de relais in kwestie.

Verbindopdracht: De initiator vraagt een verbinding te maken met een andere node. Hier zijn twee types te onderscheiden: ofwel wil de initiator het anoniem kanaal verlengen, ofwel vraagt hij het anoniem kanaal af te sluiten en een verbinding te leggen met de bestemmeling. In het laatste geval neemt de relais in kwestie de ontvanger-functie over.

Afsluitopdracht: Het kanaal moet aan deze relais afgebroken worden. Het kan hier gaan om een gedeeltelijke heropbouw van het kanaal.

Verder is er een MAC aanwezig, berekend op het pakket zoals de relais het ontvangt (dus niet noodzakelijk op de klaartekst) en een volgnummer. Pakketten met foute MAC of uit volgorde worden niet behandeld.



Figuur 6.4: Pakketten met vaste lengte (129 bytes), waarvan 1 byte linkinformatie, 5 informatievelden van 16 bytes lang, bestemd voor de relais-peers, en 48 bytes eigenlijke informatie.

De pakketten worden geconstrueerd door de initiator. Hij voegt herhaaldelijk een laag vercijfering toe voor elke gekozen relais, nadat hij de juiste informatie in het overeenkomende informatieveld geplaatst heeft.

De relais weet welk informatieveld voor hem bestemd is, omdat het hem bij het openen van de link door de initiator werd meegedeeld. De initiator kan ervoor kiezen dat het volgnummer van de informatievelden niet overeenkomt met het werkelijke volgnummer op het pad, opdat minder route-informatie gelekt zou worden. Dat is enkel mogelijk indien de kanaallengte kleiner is dan vijf. Er is dan een speling mogelijk in het plaatsen van de informatievelden, weliswaar met de ware volgorde. Bijvoorbeeld, voor een kanaal van lengte 3 kan zowel plaats 1, 2 als 3 van figuur 6.4 aan de eerste relais toegekend worden.

De relais-peers verwerken inkomende pakketten, zoals beschreven in paragraaf 4.5.1. Eerst wordt een laag vercijfering verwijderd en alle informatie uit het juiste informatieveld gehaald. Vervolgens worden integriteit, authenticiteit en het type van het pakket geverifieerd. Indien het

een forward pakket is, overschrijft de relais zijn informatieveld en de vorige informatievelden met willekeurige data en stuurt het door naar de volgende relais. Het is essentieel dat ook de vorige informatievelden overschreven worden, omdat anders valse relais-peers de informatievelden kunnen gebruiken om informatie aan elkaar door te geven, bv. het adres van wie ze het pakket gekregen hebben.

In de omgekeerde richting worden de ontcijferingsoperaties vervangen door vercijfering.

6.2.6 Kanaalopbouw

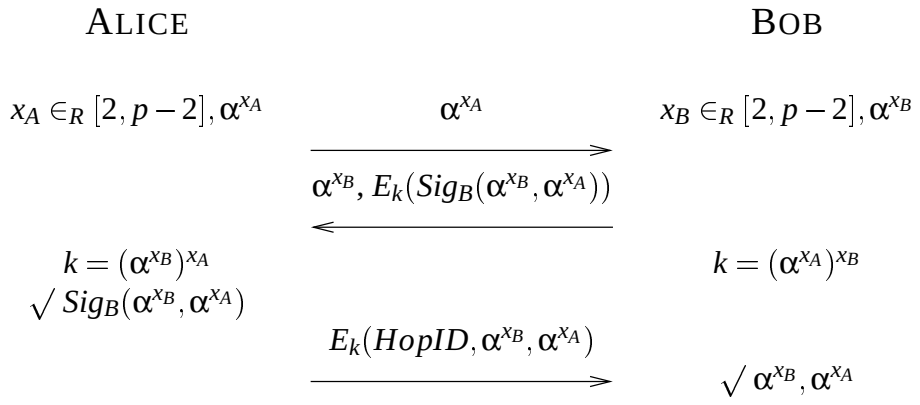
De initiator kiest een aantal relais-peers in het netwerk en start een iteratief algoritme om het kanaal op te bouwen. De initiator stuurt telkens via het kanaal tot bij relais t_i een aanvraag om een link te openen naar t_{i+1} en deze te associëren met de link tussen t_{i-1} en t_i. Algoritme 1 beschrijft de kanaalopbouw.

Algoritme 1 Kanaalopbouw

$l :=$ willekeurige lengte van het kanaal $\in [0,5]$
$t_0 :=$ zichzelf
$t_1 := kies\_willekeurige\_peer()$
maak *link* en $kanaal_1$ met t_1
voor $i = 1$ tot $l-1$ **doe**
  als maximum *aantal_kansen* overschreden **dan**
    stop kanaalopbouw
  einde als
  $t_{i+1} := kies\_willekeurige\_peer()$
  antwoord := stuur $maak\_link\_tussen(t_i, t_{i+1})$ naar t_i door $kanaal_i$
  als *antwoord* $= ok$ binnen tijdslimiet **dan**
    $kanaal_{i+1}$:= stuur $onderhandel\_kanaal(t_0, t_{i+1})$ naar t_{i+1} door $kanaal_i$
    zolang $onderhandel\_kanaal()$ mislukt of te lang duurt **doe**
      als maximum *aantal_kansen* overschreden **dan**
        $i := i-1$
      einde als
      stuur $verbreek\_link\_met(t_{i+1})$ naar t_i
      $t_{i+1} := kies\_willekeurige\_peer()$
      antwoord := stuur $maak\_link\_tussen(t_i, t_{i+1})$ naar t_i door $kanaal_i$
      $kanaal_{i+1}$:= stuur $onderhandel\_kanaal(t_0, t_{i+1})$ naar t_{i+1} door $kanaal_i$
    einde zolang
    aantal_kansen $:= 0$
  anders
    $i := i-1$
  einde als
  aantal_kansen $:= 0$
einde voor

Tijdens dit proces worden tussen initiator en alle relais-peers de cryptografische algoritmes, hashfuncties en symmetrische sleutels afgesproken.

Zoals in paragraaf 4.6.3 besproken is, maakt het gebruik van een publieke-sleutel-infrastructuur tussenpersoonaanvallen onmogelijk. Het ontwerp van de gedecentraliseerde PKI komt aan bod in de volgende paragraaf. Als protocol voor sleuteluitwisseling wordt een uit-



Figuur 6.5: Uitgebreide Diffie-Hellman sleutelovereenkomst voor kanaal- en linkinitiatie. $Sig_B(M)$ betekent dat boodschap M gehandtekend werd met het private sleutel van Bob. $\surd\ Sig_B(M)$ duidt op de verificatie van de handtekening van Bob. Bij de linkinitiatie wordt de derde fase niet uitgevoerd.

breiding van anonieme Diffie-Hellman sleutelovereenkomst gebruikt. Deze uitbreiding is hard geïnspireerd door het Station to Station protocol [3].

Het ontworpen protocol, geïllustreerd op figuur 6.5, heeft de volgende eigenschappen:

Wederzijdse sleutelbevestiging: Bob weet dat een andere, niet geïdentificeerde partij – mogelijk Alice – een bepaalde sleutel bezit.

Eenzijdige expliciete sleutelauthentisering: Alice heeft een bewijs dat een andere geïdentificeerde partij – Bob – een bepaalde sleutel bezit.

Eenzijdige authentisering van entiteiten: Alice wordt overtuigd van de identiteit van Bob en van het feit dat Bob actief aanwezig is tijdens het uitvoeren van het protocol.

Dit protocol wordt gebruikt om zowel een sessiesleutel af te spreken voor de linkvercijfering (op de linklaag) als voor de eindpuntvercijfering (op de kanaallaag). Bij de kanaalinitiatie is het belangrijk dat de initiator – Alice – zich *niet* identificeert aan de relais – Bob – zodat de initiator anoniem blijft. Bij de linkinitiatie wordt tijdens de derde fase de HopID, die aanduidt waar de relais zich in het kanaal bevindt, niet gestuurd; dit wordt enkel tijdens de kanaalinitiatie gedaan.

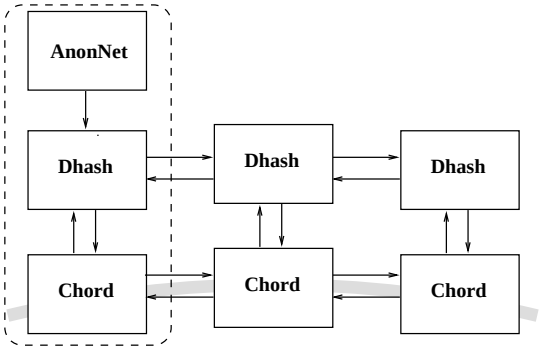
Een aantal beveiligingsparameters worden bij de kanaalinitiatie afgeleid van het overeengekomen geheim k: de sleutels voor de vercijfering en de boodschap-authentiseringscode (MAC), de initiële toestand IV van het blokcijfer, alsook de initiële volgnummers. Bij de linkinitiatie wordt de sessiesleutel voor het stroomcijfer hiervan afgeleid.

Een andere alternatief protocol lijkt het gebruik van sleuteltransport: Alice vercijfert de sessiesleutels met de publieke sleutel van Bob en stuurt dit bericht naar Bob. Er is een subtiele aanval mogelijk. De PKI is zo geïmplementeerd dat elke peer zijn eigen publieke sleutel bewaart. Als Bob aan iedere peer in het systeem een andere publieke sleutel vertelt, kan Bob achterhalen wie de initiator van een kanaal – Alice – is. Wanneer Bob de boodschap van Alice krijgt met de sessiesleutels in, probeert hij alle private sleutels die hij heeft, totdat hij de juist sleutel vindt. Hij kijkt dan aan wie hij de bijhorende publieke sleutel gegeven heeft, en kent de echte identiteit van Alice. Om dit probleem op te lossen zal men naar een ad hoc oplossing moeten grijpen [23]. De uitgebreide Diffie-Hellman sleuteluitwisseling heeft meer fases dan sleuteltransport, maar is niet kwetsbaar voor deze aanval.

6.2.7 Lokalisatie van peers

Een dynamisch netwerk Vaak gebruiken de reeds bestaande peer-to-peer toepassingen de activiteiten van hun gebruikers om tegelijkertijd andere peers in het netwerk te leren kennen. Meestal wordt de kennis over **actieve** peers in deze toepassingen echter heel traag vernieuwd en weerspiegelt ze slecht de huidige toestand van het netwerk, omdat dat in veel *file sharing* toepassingen van ondergeschikt belang is. Voor een anonimizeringslaag is een goede kennis over de actieve peers natuurlijk van cruciaal belang. Bovendien, voor een transparante anonimizeringslaag kan er niet gesteund worden op deze lokalisatiesystemen. Daarom werd er besloten om een autonoom systeem te gebruiken, dat op zich een peer-to-peer architectuur heeft.

Gedistribueerde zoekmethode Het ontwerp gebruikt het Chord zoekalgoritme [21] voor de ontdekking van actieve peers. Iedere hybride peer in AnonNet neemt deel aan één enkele Chord ring, parallel aan AnonNet. Bovenop de Chord ring wordt een DHash gedistribueerde hashtabel gebruikt om de internetadressen en publieke sleutels van de peers op te slaan. Hashtabellen werden reeds beschreven in paragraaf 3.2.1. Het ontwerp van deze gedistribueerde zoekmethode voor peerlokalisatie wordt geïllustreerd in figuur 6.6.



Figuur 6.6: Structuur van de autonome Chord ring met bovenliggende DHash gedistribueerde hashtabel. De verticale pijlen zijn lokale interfaces, de horizontale pijlen zijn netwerkinterfaces. De AnonNet software modules maken gebruik van DHash.

Chord associeert een uniek identificatienummer aan elke peer, door de hash te nemen van zijn internetadres. In de gedistribueerde hashtabel worden internetadres en publieke sleutel van de peer opgeslagen, met zijn uniek identificatienummer als indexwaarde. Chord definieert de *opvolger* van een bepaalde indexwaarde als het kleinste identificatienummer van een **actieve** peer, dat volgt op die indexwaarde.

De initiator *I* gebruikt Chord om een willekeurige relais te vinden in 3 stappen:

1. *I* genereert een willekeurige indexwaarde *r*
2. *I* vraagt via de Chord ring de *opvolger k* van deze indexwaarde op
3. Het internetadres en publieke sleutel van de willekeurige peer wordt uit de DHash tabel opgevraagd met *k* als indexwaarde

Deze dynamische opzoekmethode verzekert dat enkel de internetadressen van actieve peers gebruikt worden door de initiator. Bovendien hebben peers via de gedistribueerde hashtabel direct kennis over de topologie van het **gehele** netwerk, van zodra ze toetreden tot de Chord ring. Het volstaat één andere peer in het AnonNet netwerk te kennen.

6.3 Robuustheid

6.3.1 Impact van netwerkvertragingen

Het is duidelijk dat een zuiver Pipenet met enkelvoudige anonieme kanalen niet praktisch te realiseren is. De maatregel die in paragraaf 5.5 beschreven is, beschermt Pipenet tegen actieve aanvallen waarbij berichten in de tijd uitgesteld worden. In dat geval zal een Pipenet relais volledig weigeren verder te werken. Hij verstuurt dan op geen enkele uitgaande verbinding nog pakketten, totdat voor elke inkomende verbinding een pakket beschikbaar is. Dat maakt het netwerk enorm kwetsbaar voor denial-of-service-aanvallen. Dit soort aanvallen zal de identiteit van de gebruikers in een Pipenet niet prijsgeven, maar het kan het netwerk wel gemakkelijk op de knieën krijgen.

In dit ontwerp werd er dus voor gekozen ook een minimale stroom aan te houden langs de links, tussen elk paar relais-peers. Wanneer een relais langer dan een maximale tijdsduur stil heeft gelegen, wordt er op de verbindingen waar geen uitgaand pakket voor beschikbaar is, een neppakket verstuurd. Dit zijn boodschappen in de linklaag. Het opvullen van de ongebruikte, maar opgelegde bandbreedte heeft een nefaste weerslag op de netwerkefficiëntie. Het herbruiken van links voor verschillende anonieme kanalen kan die weerslag verkleinen.

6.3.2 Heropbouw van de kanalen

Gefaalde links worden door de relais-peers gesignaleerd aan de initiator. Hij beslist dan hoe dit falen kan worden opgevangen. Zowel omwille van netwerkefficiëntie als omwille van het behouden van de anonimiteit, wordt bij het falen of verdwijnen van een peer best niet heel het kanaal terug opgebouwd van aan de initiator. Wanneer een actieve aanvaller een link uitschakelt, wil men vermijden dat het falen van het hele kanaal, het kanaal als een bliksem doet oplichten in het netwerk. *Gedeeltelijke* heropbouw is hier een betere aanpak.

Hoofdstuk 7

Implementatie van een peer-to-peer architectuur voor anonieme kanalen

In hoofdstuk 6 werd de keuze voor vier autonome componenten toegelicht, die respectievelijk de taken van onderschepping, kanaalbeheer, opslag van publieke sleutels en peerlokalisatie voor hun rekening nemen. Ze hebben de vorm gekregen van drie aparte programma's; opslag van publieke sleutels en peerlokalisatie worden gecombineerd in één programma. Ze worden op elke peer opgestart en communiceren met elkaar via lokale interfaces.

Alle programmacode en instructies voor het opzetten van een anoniem peer-to-peer netwerk zijn beschikbaar op `http://atnet.sourceforge.net` [41] als vrije software.

7.1 Onderscheppen van verbindingsaanvragen

SOCKS is een protocol dat gebruikt wordt om TCP netwerkverkeer langs SOCKS proxy servers om te leiden. Het geïmplementeerde systeem maakt gebruik van *tsocks* [42]. Dit is een programma voor Unix platforms, dat uitgaande netwerkverbindingen in de transportlaag onderschept en het SOCKS protocol implementeert.

Tsocks wordt automatisch in de procesruimte van het gevraagde programma geladen. Hier onderschept het elke `connect()` functie die dat programma uitvoert wanneer het een netwerkverbinding aanvraagt. De originele tsocks implementatie laat niet toe dat lokale SOCKS proxies gebruikt worden. De C broncode werd aangepast en opnieuw gecompileerd, zodat de lokale AnonNet component kan gebruikt worden.

In dit geval implementeert de AnonNet component, die instaat voor het kanaalbeheer, een SOCKS interface, zodat beide componenten kunnen communiceren en de verbindingen wel degelijk 'omgeleid' worden. Op Windows platformen kan *SocksCap* [43] deze taak vervullen.

7.2 AnonNet component

Voor de implementatie van de component die de taken uit paragraaf 6.2.2 vervult, werd vertrokken van een bestaande implementatie in de C programmeertaal [40]. Deze implementatie stond echter helemaal nog niet op poten. In principe was enkel de taak van een relais geïmplementeerd en konden 'links' tussen peers onderhandeld worden. Anonieme kanalen konden echter nog niet opgezet worden. Verschillende onderdelen werden toegevoegd, de gebruikte types boodschappen kwamen in paragraaf 6.2.5 aan bod:

Implementatie van de SOCKS interface. De software luistert op een lokale netwerkpoort op inkomende SOCKS verbindingsaanvragen, afkomstig van tsocks. Er werd gekozen voor de simpliciteit van SOCKS versie 4. SOCKS versie 5 ondersteunt authentisatie en UDP, maar dat heeft geen nut voor dit systeem.

De taak van de initiator. Bij een inkomende SOCKS verbindingsaanvraag wordt een kanaal opgebouwd volgens algoritme 1 uit paragraaf 6.2.6. De initiator verwerkt inkomende pakketten en zet ze op het anonieme kanaal. Hij stuurt neppakketten indien nodig.

De taak van de ontvanger. De ontvanger vormt de poort naar het publieke internet. Zodra hij van de initiator een verbindopdracht krijgt, maakt hij een verbinding met de bestemmeling in kwestie. Vervolgens stuurt hij alle pakketten uit dit kanaal naar de bestemmeling.

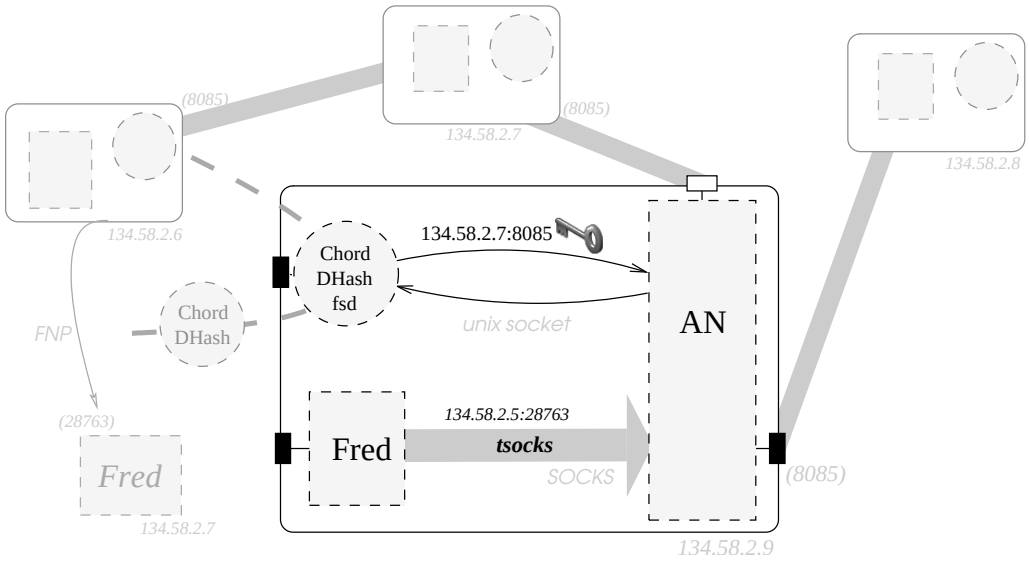
Implementatie van de Chord/DHash interface. Er werd een simpel protocol geïmplementeerd voor de uitwisseling van internetadressen en publieke sleutels met de Chord/DHash interface. Telkens de AnonNet component leest uit het speciale bestand `/tmp/anonnet-sock` – een *UNIX socket* – krijgt hij het internetadres en de publieke sleutel van een willekeurige andere actieve AnonNet peer, in geserializeerde vorm. De AnonNet component kan ook schrijven naar het bestand. Zo vraagt hij aan de Chord/DHash laag om zijn adres en publieke sleutel in de gedistribueerde hashtabel te registreren.

Sleutelovereenkomst met éénzijdige authentisering. Het Diffie-Hellman protocol voor sleutelovereenkomst werd uitgebreid. Men maakt gebruik van de kennis van de publieke sleutels die opgeslagen zijn in de DHash gedistribueerde hashtabel. Deze uitbreiding werd beschreven in paragraaf 6.2.6.

De interfaces met de andere componenten worden geïllustreerd in figuur 7.1. Het programma heeft individuele threads die de verschillende taken onafhankelijk van elkaar uitvoeren, zodat verschillende aanzoeken gelijktijdig en asynchroon kunnen behandeld worden.

Server threads Er zijn twee threads die wachten op inkomende netwerkverbindingen. Ze luisteren standaard op poorten 8085 en 6060. Bij een inkomende verbinding starten ze respectievelijk een nieuwe relais/ontvanger thread of SOCKS thread op.

Mixmaster thread De mixmaster thread is verantwoordelijk voor de *herordening* van de berichten en het opleggen van een *vaste bandbreedte*. Hij staat ook in voor het beheer van de datastructuren voor de netwerkpakketten – inkomende en uitgaande wachtrijen bij elke netwerklink. Deze thread overloopt de uitgaande wachtrijen van alle links. Bij elke tijdsstap wordt voor elke link – in willekeurige volgorde – welgeteld één 129 byte pakket verstuurd. Dit normale gedrag wordt tijdelijk gestopt wanneer er in de uitgaande wachtrij van minstens één link geen pakket beschikbaar is. Als er binnen een bepaalde tijdsduur in die wachtrij nog steeds geen pakket beschikbaar wordt, zullen tijdelijk neppakketten in de linklaag verstuurd worden. Dit komt overeen met de aanpassing op het Pipenet protocol, die gestaafd werd in paragraaf 6.3. Deze thread leest ook de interfaces van alle links uit. Bij elke tijdsstap wordt één 129 byte pakket in de inkomende wachtrij van de overkomende link geplaatst, ter behandeling door de relais/ontvanger thread.



Figuur 7.1: Interactie tussen de verschillende componenten. Freenet peer Fred is hier de toepassing. Fred's aanvraag tot verbinding met Fred op adres 134.58.2.5 en poort 28763 wordt onderschept en geanonimizeerd door het systeem. De DHash tabel geeft peers in het AnonNet netwerk, zodat een anoniem kanaal tot bij de gevraagde Fred kan gemaakt worden.

SOCKS thread Deze thread implementeert de juiste afhandeling van een inkomende netwerkverbinding volgens het SOCKS protocol. Uit de SOCKS-aanvraag wordt het adres van de bestemmeling gehaald. Vervolgens wordt een nieuwe initiator thread opgestart en dit adres doorgegeven. Indien de initiator thread binnen een bepaald tijdsbestek een succesvolle kanaalopbouw signaleert, wordt dit volgens het SOCKS protocol aan de gebruiker gemeld. Anders wordt een faling gemeld. De taak van deze thread is dan afgelopen en hij wordt vernietigd.

Initiator thread Zoals de naam al zegt, staat deze thread in voor de het vervullen van de taken van de initiator. Hij rapporteert het resultaat van de kanaalopbouw terug aan de SOCKS thread. Vervolgens worden alle boodschappen die via deze interface binnenkomen opgesplitst in stukken met van 48 bytes. Elk stuk wordt ingepakt in een pakket, zoals in figuur 6.4. Dat gebeurt met een recursieve aanroep van de functie die telkens één laag vercijfering legt. Dan wordt de versneden en ingepakte boodschap in de uitgaande wachtrij van de verbinding naar de juiste relais geplaatst.

De initiator stuurt neppakketten zodat het bandbreedtegebruik tussen gebruiker en ontvanger vast blijft. Wanneer de verbinding gesloten wordt door de gebruiker, stuurt de initiator een afsluitopdracht naar de ontvanger en wordt het kanaal gesloten. Het kanaal wordt ook afgebroken wanneer een afsluitopdracht van de initiator ontvangen wordt. De 'links' tussen peers blijven open als ze nog in gebruik zijn door andere kanalen, of ze blijven nog een beperkte tijd open voor het geval dat een volgend kanaal erlangs zou geopend worden. In het laatste geval wordt de netwerkstroom langs de link tijdelijk opgevuld, zodat de anonimiteitsgroep niet afgebroken wordt.

Relais/ontvanger thread Deze thread zorgt voor het versassen van de pakketten tussen de links van één welbepaald kanaal of virtueel circuit. Alle inkomende wachtrijen van de links worden geïnspecteerd. Van elk wachtend pakket wordt de linkvercijfering verwijderd, en de juiste actie uitgevoerd volgens het pakkettype. Indien het om een type *forward* pakket gaat, wordt

linkvercijfering toegepast op het pakket en wordt het in de uitgaande wachtrij geplaatst van de juiste link. Wanneer de verbinding gesloten wordt door de bestemmeling, stuurt de ontvanger een afsluitopdracht naar de initiator en wordt het kanaal afgebroken. Dat gebeurt analoog als bij de initiator.

De juiste synchronisatie van de initiator, relais/ontvanger en mixmaster threads is cruciaal voor de implementatie van kanaalopvulling. Als de caller thread voor- of achterloopt op de mixmaster thread, worden de verbindingen onnodig volgestouwd met neppakketten of zullen de relais-peers steeds meer neppakketten beginnen opstapelen.

7.3 fsd: de Chord/DHash server-client

De bestaande Chord/DHash server *(lsd)* is de implementatie van de Chord en DHash protocols. Het programma registreert de peer in de Chord ring, luistert op een netwerkpoort naar andere Chord/DHash servers, en maakt het speciale bestand `/tmp/dhash-sock` – de UNIX socket – aan. Via deze interfaces kunnen via RPC (*Remote Procedure Call*) blokken data opgeslagen worden in de DHash tabel.

Het programma werd herschreven zodat het de internetadressen en publieke sleutels kan leveren waar de AnonNet component om vraagt. Een extra UNIX socket wordt aangemaakt onder het bestand `/tmp/anonnet-sock`, als interface naar de AnonNet component. De implementatie maakt gebruik van de C++ API (*Application Program Interface*) die toegang bieden tot de Chord en DHash functionaliteit.

Registratie in het netwerk Bij het opstarten van het programma, contacteert de peer een gekende Chord server en registreert zich in de Chord ring via een RPC. Hierbij wordt hem een semi-unieke identificatiesleutel *ChordID* toegekend. Vervolgens slaat het programma een informatieblok op in de DHash tabel, onder identificatiesleutel $ChordID + 1$. Dit informatieblok bestaat uit het internetadres en de poort van de AnonNet component, en de publieke sleutel van de AnonNet component. Dit gebeurt met een RPC op de lokale `/tmp/dhash-sock` interface.

Ontdekking van peers Op regelmatige tijdstippen wordt een willekeurige identificatiesleutel r gegenereerd. Met behulp van de Chord `find_successor(`r`)` oproep komt men de eerste actieve opvolger *succID* van r in de Chord ring te weten. Dit maakt het mogelijk het informatieblok onder identificatiesleutel $succID + 1$ uit de DHash tabel te halen. Deze informatieblokken worden opgeslagen in een circulaire buffer, zodat de lijst toch recent blijft.

Telkens de AnonNet component uit het bestand `/tmp/anonnet-sock` leest, wordt een willekeurig informatieblok met internetadres, poort en publieke sleutel van een AnonNet peer uit de circulaire buffer naar dit bestand geschreven, in geserializeerde vorm.

Beveiliging De informatieblokken worden op *regelmatige tijdstippen* opgezocht in de DHash tabel, niet wanneer de AnonNet component om informatie vraagt. Dat komt omdat de opzoeking anders informatie zou lekken over het anonieme kanaal dat op dat moment zal opgezet worden.

De DHash API liet nog niet toe dat er informatieblokken onder specifieke identificatiesleutels opgeslagen werden. We hebben die functionaliteit toegevoegd. Bovendien wordt ervoor gezorgd

dat deze blokken niet kunnen overschreven worden, anders zou een tegenstander de publieke sleutel kunnen aanpassen.

Het nieuwe programma werd *fsd* gedoopt, als verwijzing naar *find_successor daemon*.

7.4 Cryptografische primitieven

Hier volgt een kort overzicht van de gebruikte cryptografische algoritmen en basisblokken. Er kan wel opgemerkt worden dat er in het kanaalverbindingsprotocol plaats is voor onderlinge afspraak van andere basisblokken. De verbindingsaanvraagboodschappen hebben hiervoor plaats voorzien.

Linklaag De pakketten op de links tussen twee AnonNet peers worden vercijferd met het Arcfour stroomcijfer.

Kanaallaag De gelaagde encryptie van de AnonNet pakketten wordt uitgevoerd door het Rijndael blokcijfer. Dat gebeurt in CBC mode, ter bescherming tegen actieve aanvallen zoals herhaling.

Authenticiteit en authentisering van de informatie Elk pakket bevat een HMAC op basis van SHA-1 of Tiger/192.

Sleutelafleiding Alle gebruikte sleutels worden afgeleid van het ene geheim dat overeengekomen werd bij de uitgebreide Diffie-Hellman uitwisseling. Zoals bij TLS 1.0 wordt een sleutelstroom gegenereerd met een *pseudo random function*. Hieruit worden de Rijndael en Arcfour sleutels gehaald, alsook de initialisatievector, de gebruikte HMAC sleutel en de initiële pakketvolgnummers.

7.5 Voorlopige resultaten

De evaluatie van de implementatie kan op basis van drie criteria gebeuren: anonimiteit, efficiëntie en robuustheid. Tijdens het ontwerp lag de nadruk hoofdzakelijk op het realiseren van anonimiteit om het gebruik van de beschreven bouwblokken te verduidelijken.

Beveiliging en anonimiteit Hoewel anonimiteit niet meetbaar is, lijkt het gerealiseerde systeem goed te beschermen tegen verkeersanalyse. Het Pipenet model werd immers bijna volledig gevolgd, enkel het gebruik van linkopvulling is toegevoegd om het systeem robuuster te maken.

Momenteel biedt het ontworpen systeem weinig weerstand tegen denial-of-service-aanvallen. Dit kan dus nog verbeterd worden.

Efficiëntie en gebruiksgemak Het opzetten van een anoniem kanaal duurt enkel seconden omdat meerder publieke sleuteloperaties moeten gebeuren, en verschillende relais-peers moeten gecontacteerd worden. Dit kan voor de gebruiker als hinderlijk ondervonden worden. Nadien zal de vertraging gering zijn omdat het gebruik van symmetrische vercijfering geen probleem vormt op hedendaagse computers. Een gebruiker dient er ook rekening mee te houden dat een anoniem kanaal beperkt is in bandbreedte.

Robuustheid en bruikbaarheid Momenteel is het mogelijk om simpele toepassingen zoals *e-mail*, *telnet* en *SSH* over een anoniem kanaal om te leiden. Ook *HTTP*-verkeer is mogelijk, maar de implementatie is nog niet stabiel genoeg om complexe webpagina's in te laden met browsers die deze pagina's met parallelle HTTP-stromen binnenhalen. Het inladen van `http://www.google.com` gaat bijvoorbeeld al wel. Een peer-to-peer toepassing zoals Freenet werd nog niet getest.

Als een peer uitvalt in een kanaal, wordt geprobeerd het kanaal gedeeltelijk terug op te bouwen. Er wordt geen rekening mee gehouden dat de gefaalde peer een aantal nuttige pakketten in zijn buffer had staan. Deze pakketten gaan verloren. Als anonieme kanalen op de netwerklaag (IP) geïmplementeerd worden, zal de bovenliggende transportlaag (TCP) voor het betrouwbaar transport van pakketten zorgen: de ontvangst van een pakket wordt bevestigd en er gebeurt heruitzending van verdwenen pakketten. Misschien moet een vergelijkbaar systeem toegevoegd worden aan de huidige implementatie zodat bij het uitvallen van een peer in het kanaal de verloren pakketten opnieuw gestuurd worden. Het zal uit de praktijk moeten blijken of dit de moeite loont.

7.6 Suggesties voor uitbreidingen

Virtuele circuits In de huidige implementatie wordt een enkelvoudige anoniem kanaal gelegd tussen de initiator en de ontvanger. Zoals in paragraaf 6.2.4 beschreven is, kan men ook meerdere anonieme kanalen leggen en deze bundelen in een *virtueel circuit*. Dit concept heeft een aantal voordelen:

- Het biedt een grotere robuustheid. Als één van de kanalen onderbroken wordt door het uitvallen van een relais-peer, kan het virtuele circuit toch gewoon blijven verder werken. Het onderbroken kanaal wordt tijdelijk niet gebruikt, totdat het opnieuw opgebouwd is.

- Het vervult tevens de wens naar hoge connectiviteit tussen peers, aangezien het aantal mogelijke paden van een inkomend pakket vergroot wordt. Bovendien voorzien de peers elkaar zo van meer inkomend verkeer. Het is duidelijk dat de anonimiteitsgroep aanzienlijk vergroot.

Misschien wegen de conceptuele voordelen echter niet op tegen de implementatieproblemen. Dit moet verder onderzocht worden. Aan elk pakket dat op een virtueel circuit verstuurd wordt, moet immers een volgnummer toegekend worden. Binnen een enkelvoudig kanalen komen pakketten in volgorde aan bij de ontvanger, maar bij een virtueel circuit moet de ontvanger op basis van de volgnummers de pakketten terug in de juiste volgorde plaatsen. Men kan zich ook inbeelden dat de pakketbuffers bij de ontvanger heel groot zullen moeten zijn, om tijdelijk gefaalde enkelvoudige kanalen op te vangen.

Herbruik van links tussen peers In de huidige implementatie worden links tussen peers nog niet herbruikt voor verschillende kanalen. Toch biedt dit een aantal voordelen:

- Alvorens een link gebruikt kan worden, moet een sessiesleutel voor het stroomcijfer afgesproken worden tussen de twee peers. Dit vraagt een bepaalde tijd omdat in dit protocol publieke sleutel operaties gebruikt worden. Door links te herbruiken kan men gewoon de reeds afgesproken sessiesleutel gebruiken.
- Als links herbruikt worden, moeten er ook minder nepboodschappen op een link gestuurd worden.

Een pseudoniemensysteem implementeren in DHash is dus nog een open probleem.

Overgebruik verhinderen De huidige implementatie biedt geen beveiliging tegen denial-of-service-aanvallen. Zoals in paragraaf 4.5.2 besproken is, kan men dit verhinderen door het gebruik van een betalingsysteem of een reputatiesysteem.

- Als men een betalingsysteem gebruikt, betaalt de gebruiker/initiator aan elke relais een bepaald bedrag om de peer in een anoniem kanaal te gebruiken. Het bedrag is evenredig met de tijdsduur dat de initiator het kanaal wil gebruiken. Een gebruiker zal zelf geld verdienen door zelf relais te spelen voor andere kanalen. Het is natuurlijk de vraag of het betalingsysteem gedecentraliseerd kan geïmplementeerd worden.
- Het gebruik van een bewijs van werk kan ook denial-of-service-aanvallen verhinderen – vertragen. Dit vereist geen bank en kan dus eenvoudiger op een gedecentraliseerd manier geïmplementeerd worden.
- Als een reputatiesysteem gebruikt wordt, zal de initiator zich identificeren aan elke relais met zijn pseudoniem. Zoals in paragraaf 2.1.4 beschreven staat, is pseudonimiteit een zwakkere vorm van anonimiteit.

  Het dient opgemerkt te worden dat een ander sleutelpaar gebruikt moet worden als pseudoniem. Momenteel wordt in DHash de publieke sleutel en het internetadres van een peer opgeslagen met een bepaalde ChordID. Zo is het mogelijk om de publieke sleutel van een peer aan het internetadres ervan te koppelen. Het internetadres identificeert de initiator, waardoor die publieke sleutel niet gebruikt kan worden al pseudoniem. Peers zullen dus een tweede publieke sleutel in DHash opslaan met een andere ChordID. De initiator identificeert zich aan de relais door het ChordID van die tweede publieke sleutel te zeggen en een digitale handtekening te plaatsen met de private sleutel die bij deze publieke sleutel hoort.

Er zijn nog een aantal onopgeloste problemen bij het implementeren van een pseudoniemensysteem:

- Er moet verhinderd worden dat een peer meerdere pseudoniemen gebruikt en zo het reputatiesysteem omzeilt.
- Het is niet zeker dat DHash geschikt is om pseudoniemen op te slaan. Een aanvaller zal trachten te achterhalen wie welk pseudoniem in DHash plaatst om zo de ware identiteit – het internetadres – van een pseudoniem te achterhalen.

Momenteel slaat elke peer zijn publieke sleutel zelf op. De peer mag zijn eigen pseudoniem echter niet zelf opslaan, anders zou de opzoeking in de DHash tabel zijn identiteit prijsgeven. Het pseudoniem moet dus op een andere peer bewaard worden. De opslag moet beveiligd worden met de technieken die in paragraaf 2.1.1 besproken zijn, zodat dat opgeslagen pseudoniemen niet verwijderd kunnen worden.

- Als een gebruiker zijn pseudoniem op een andere peer plaatst, kan deze peer de identiteit van de gebruiker met het pseudoniem verbinden. Men zou het plaatsen van een pseudoniem anoniem kunnen maken, maar dan bestaat de mogelijkheid dat gebruikers meerdere pseudoniemen publiceren in het systeem, wat juist verhinderd moest worden.

Een pseudoniemensysteem implementeren met DHash blijft voorlopig dus een open probleem.

Hoofdstuk 8

Algemeen Besluit

Heel wat peer-to-peer netwerken zagen nog recent het daglicht, als een logische reactie op een herschapen internetlandschap. De eindgebruikers zagen hun aandeel in rekenkracht en connectiviteit gevoelig stijgen. Ze worden steeds belangrijkere spelers in het netwerk, maar ze vertonen een veel groter online/offline gedrag dan servers. Deze recente evolutie maakt dat de peer-to-peer architectuur deel uitmaakt van een jong en voortdurend vernieuwend onderzoeksdomein. Er zijn reeds heel wat technieken ontwikkeld die de eigenschappen van dit nieuwe landschap proberen te benutten. Toch is er nog niet veel werk verricht dat een overzicht tracht te schetsen van de nieuwe uitdagingen en technieken. Het boek *Peer-to-Peer – Harnessing the Benefits of a Disruptive Technology* [44] is voorlopig het enige document dat deze leegte heeft geprobeerd te vullen. Uit het boek blijkt dat er in dit onderzoeksdomein twee belangrijke onderzoeksaspecten te onderscheiden zijn.

Enerzijds is er het *netwerkaspect* van peer-to-peer systemen. Het dynamische karakter van een peer-to-peer systeem zorgt ervoor dat niet vast staat welke peers deelnemen aan het systeem. Het is een interessante uitdaging om toch dienstverlening mogelijk te maken. Er wordt onderzoek gedaan naar efficiënte en snelle zoekalgoritmen en systemen om te ontdekken welke peers allemaal deelnemen aan het systeem. Daarenboven probeert men systemen voor informatie-opslag aan te bieden door redundante opslag te voorzien.

Anderzijds heb je het *beveiligingsaspect*. Tot voor kort waren client en server duidelijk gescheiden. Het was dan de taak van de beheerder van een server om de nodige beveiligingsmaatregelingen te nemen zodat gebruikers niets verkeerd konden doen. Bij een peer-to-peer systeem verdwijnt het onderscheid tussen gebruiker en server. Alle peers zijn gelijke spelers en bieden zelf de dienstverlening aan, maar ze zijn a priori onbetrouwbaar. Dit zorgt voor interessante uitdagingen omdat valse gebruikers de dienstverlening nu *van binnen uit* kunnen ondermijnen.

We hebben de aandacht eerst gevestigd op het *beveiligingsaspect*. Soms wordt er verwezen naar het netwerkaspect, waar nodig. Er zijn een aantal peer-to-peer toepassingen mogelijk, maar file sharing is de populairste en meest bestudeerde toepassing. Dit heeft men te danken aan de hype rond Napster. Daarom hebben we ervoor gekozen om extra aandacht te geven aan systemen die informatie-opslag tot doel hebben. De behoefte om communicatie en opslag te beveiligen is niet echt nieuw. De technieken om dit te realiseren bestaan al jaren. De grote uitdaging ligt in systemen om te achterhalen welke peers betrouwbaar zijn. We onderscheiden hierin systemen die een betalingssysteem gebruiken enerzijds, en systemen die gebruik maken van een reputatiesysteem anderzijds. Echt robuuste betalingssystemen bestaan er momenteel nog niet. In de praktijk zal daarom vaak een reputatiesyteem gebruikt worden of een combinatie van beide.

Vervolgens kwam de vraag naar *anonimiteit* in peer-to-peer netwerken aan bod. Zowel graad als haalbaarheid werden in vraag gesteld. Als conclusie kwamen er twee technieken uit de bus om de anonimiteit van gebruikers te behouden: informatie-uitwisseling via een routeringsalgoritme en informatie-overdracht via een communicatiekanaal naar keuze. Dit onderscheid werd in de literatuur nog nooit expliciet gemaakt. De beschrijving van de mogelijke routeringsalgoritmes is een netwerkaspect, maar wel essentieel om te begrijpen hoe anonimiteit nu juist gerealiseerd wordt. Als bestanden uitgewisseld worden door een routeringsalgoritme is het moeilijk te bepalen wat de behaalde graad van anonimiteit juist is. Dit blijkt duidelijk uit de uitgebreide analyse van Freenet. Het gebruik van anonieme kanalen werd wel al beter bestudeerd, waardoor de graad van anonimiteit veel duidelijker bepaald kan worden.

De voorkeur bij informatie-overdracht via een communicatiekanaal gaat naar *anonieme communicatiekanalen*. Men onderscheidt twee bedreigingsmodellen: passieve verkeersanalyse door een krachtige globale waarnemer versus actief samenwerkende valse peers. Ter illustratie van deze bedreigingen werden een aantal specifieke aanvallen uit de literatuur geïllustreerd. Er werden reeds verschillende bouwblokken opgesomd in de literatuur [30]. We hebben getracht ze te classificeren bij verdedigingsmethoden tegen verkeersanalyse of bij technieken die beschermen tegen actieve valse peers. Het concept kanaalopbouw is een belangrijk begrip dat verklaard moet worden, alvorens men naar concrete implementaties van anonieme kanalen kan kijken.

De besproken bouwblokken worden meestal op een ad hoc methode gebruikt. Het gebruik werd in kaart gebracht voor bestaande protocols, en de aanpassingen geanalyseerd. De meeste implementaties van anonieme kanalen zullen probabilistische routering, mixnet of Pipenet gebruiken. Mixnet en Pipenet lijken sterk op elkaar aangezien ze beide gelaagde vercijfering realiseren. In de praktijk wordt een mixnet vaak gebruikt als men een centraal beheerd anonimizeringssysteem wil maken, terwijl het Pipenet model interessanter is als men een dynamisch peer-to-peer anonimizeringssysteem wil realiseren.

We ontwierpen een *peer-to-peer* systeem dat *anonieme kanalen* verwezenlijkt. Op die manier trachten we de steeds stijgende vraag naar een algemeen kader voor beveiligde en anonieme informatie-uitwisseling te beantwoorden, terwijl we nuttig gebruik maken van de verschuivingen in het internetlandschap. Daarom ging de voorkeur naar een peer-to-peer architectuur. Het ontwerp gaat een stap verder dan de reeds bestaande gecentraliseerde anonimizeringssystemen. Het maakt nuttig gebruik van de bestudeerde peer-to-peer technieken en protocols ter implementatie van het systeem.

Er werd gekozen voor het Pipenet [38] model voor anonieme kanalen, omdat het geschikt is bij het ontwerp van een *gedecentraliseerd* systeem. Het ontwerp vroeg toch naar een aantal uitbreidingen op het Pipenet model. Het gebruik van virtuele circuits zorgt voor verhoogde connectiviteit tussen peers. Virtuele circuits kunnen bijgevolg de anonimiteitsgroep groot houden, vergelijkbaar met de nabootsers bij Tarzan. Om efficiëntie en robuustheid te verhogen willen we zo veel mogelijk verhinderen dat kanalen helemaal afgebroken worden: de kanalen worden iteratief opgebouwd, links worden herbruikt en indien nodig worden tijdelijk neppakketten verstuurd op de linklaag. Dit beschermt ons tevens tegen denial-of-service-aanvallen, maar verlaagt de weerbaarheid tegen verkeersanalyse. Tussenpersoonaanvallen worden vermeden door de Diffie-Hellman sleutelovereenkomst eenzijdig te authentiseren.

De juiste combinatie van opkomende peer-to-peer technieken, systemen en netwerken verwezenlijkt het systeem. Het ontwerp baseert zich op peer-to-peer systemen die zich zowel op het vlak van *zoektechnieken* en *informatie-opslag*, als op het vlak van *anonieme communicatie* situeren. Deze eisen en wensen hebben geresulteerd in een symbiose van drie systemen. Chord [21]

staat in voor de *zoek*tocht naar actieve peers en DHash [20] is verantwoordelijk voor de *opslag* van de lokatie en publieke sleutels van de peers. AnonNet baseert zich tenslotte op het Pipenet[38] model voor de opbouw, het beheer en de uitvoering van de anonieme kanalen.

In het ontwerp staat ook *toepassingsonafhankelijkheid* centraal. We hebben gekozen om op de transportlaag te werken. Op de netwerklaag werken, zoals Tarzan [23], biedt een hogere graad van transparantie, maar vereist aanpassingen op het systeemniveau. Toepassingsonafhankelijkheid werd wel gedeeltelijk gerealiseerd door gebruik te maken van een programmeerbibliotheek die pakketten onderschept alvorens ze aan het besturingssysteem doorgegeven worden.

Voor de implementatie van het ontwerp hebben we samengewerkt met William Ahern, die aan de wieg stond van AnonNet [40], maar spijtig genoeg de tijd niet had zijn ideeën uit te werken. De bestaande C code werd grondig uitgebreid ter implementatie van verschillende nieuwe en cruciale functionaliteiten, volgens het besproken ontwerp. Vooreerst werd de taak van initiator en ontvanger geïmplementeerd, waarbij de kanaalopbouw en vast bandbreedtegebruik tussen de eindpunten van het kanaal centraal stonden. DHash voegt een gedecentraliseerde publieke-sleutel-infrastructuur toe. Dat heeft toegelaten het sleutelovereenkomst protocol te herdenken. We stellen een protocol voor met éénzijdige authentisering. De autonome DHash server-client (*fsd*) werkt in een parallelle Chord ring. Aan de Chord en DHash API's werden nieuwe functies toegevoegd om een veilige werking van *fsd* te verzekeren. Er was nood aan interfaces, enerzijds tussen de toepassingen en AnonNet, en anderzijds tussen AnonNet en zijn hulpmodules. Dit heeft geleid tot de implementatie van een SOCKS interface en van een interface met *fsd*.

Het is moeilijk een uitgebreide evaluatie te maken van de behaalde graad van anonimiteit. Het Pipenet model wordt in dit domein beschouwd als een sterk anoniem protocol en werd relatief strikt gevolgd. Anonimiteit is moeilijk meetbaar, maar we vermoeden dat het systeem een sterke weerstand tegen verkeersanalyse biedt. Voor het eerst kan dit op een volledig gedecentraliseerde manier, met een peer-to-peer architectuur als ruggengraat.

Uiteraard biedt de implementatie slechts een conceptueel model. Er kan nog heel wat gewerkt worden aan de robuustheid en bruikbaarheid van de software. Bovendien zijn er nog talloze onderwerpen voor verder onderzoek mogelijk. De introductie van een betalingssysteem of reputatiesysteem kan een oplossing bieden tegen denial-of-service-aanvallen. Dit vergt echter het gebruik van pseudoniemen. De gedecentraliseerde implementatie van zo'n pseudoniemensysteem blijkt een moeilijk en open probleem. Tenslotte suggereren we nog onderzoek naar virtuele circuits en het herbruik van links. Het is nog niet duidelijk of de extra anonimiteit en robuustheid opwegen tegen de implementatieproblemen.

Leuven, 17 mei 2002

Bibliografie

[1] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, november 1979.

[2] Michael O. Rabin. Efficient disperal of information for security, load balancing and fault tolerance. *Journal of the ACM*, 36(2):335–348, april 1989.

[3] Alfred J. Menezes, Paul C. van Oorschot, en Scott A. Vanstone. *Handbook of Applied Cryptography*, hoofdstuk 12. CRC Press, oktober 1996.

[4] Ronald L. Rivest. Chaffing and winnowing: confidentiality without encryption. *CryptoBytes*, 4(1):14–18, maart 1998.

[5] Roger Dingledine, Michael J. Freedman, en David Molnar. Accountability. In *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*. O'Reilly, maart 2001.

[6] Richard Lethin. Reputation. In *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*. O'Reilly, maart 2001.

[7] Stephen Adler. The slashdot effect: An analysis of three internet publications, maart 1999.

[8] Dan J. Bernstein. Syn cookies. `http://cr.yp.to/syncookies.html`.

[9] Cynthia Dwork en Moni Naor. Pricing via processing or combating junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology: Proceedings of Crypto'92, the 12th Annual International Cryptology Conference*, pagina 139–147, Santa Barbara, California, USA, augustus 1992. Springer-Verlag.

[10] Adam Back. Hash cash: Partial hash collision based postage scheme. `http://www.cypherspace.org/~adam/hashcash/`, 1997.

[11] Ari Juels en John Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In *Proceedings of Networks and Distributed Security Systems '99*, pagina 151–165, 1999.

[12] Ronald L. Rivest en Adi Shamir. Payword and micromint – two simple micropayment schemes. *CryptoBytes*, 2(1), 1996.

[13] David Chaum. Blind signatures for untraceable payments. In Ron L. Rivest, A. Sherman, en David Chaum, editors, *Advances in Cryptology, Proceedings of Crypto 82*, pagina 199–203, Santa Barbara, California, 1983. Plenum Press.

[14] Stefan Brands. Untraceable off-line cash in wallets with observers. In D. R. Stinson, editor, *Advances in Cryptology: Proceedings of Crypto '93, the 12th Annual International Cryptology Conference*, pagina 302–318. Springer-Verlag, 1993.

[15] Autonomous Zone Industries. Mojo nation: An internet distributed content system with market based resource allocation. `http://mojonation.sourceforge.net`.

[16] Marc Waldman, Aviel D. Rubin, en Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pagina 59–72, augustus 2000.

[17] Roger Dingledine, Michael J. Freedman, en David Molnar. The free haven project: Distributed anonymous storage service. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pagina 67–95, Berkeley, California, juli 2000.

[18] Ian Clarke, Oskar Sandberg, Brandon Wiley, en Theodore T. Wong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pagina 46–66, Berkeley, California, juli 2000.

[19] coderman@mindspring.com. Decentralized resource discovery in large peer based networks. `http://cubicmetercrystal.com/alpine/discovery.html`.

[20] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, en Ion Stoica. Wide-area cooperative storage with cfs. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, oktober 2001.

[21] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, en Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM 2001 Conference*, San Diego, California, augustus 2001.

[22] Reptile. `http://reptile.openprivacy.org`.

[23] Michael J. Freedman, Emil Sit, Josh Cates, en Robert Morris. Introducing tarzan, a peer-to-peer anonymizing network layer. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, Cambridge, MA, maart 2002.

[24] Napster. `http://www.napster.com`.

[25] Gnutella. `http://gnutella.wego.com`.

[26] Andreas Pfitzmann en Marit Köhntopp. Anonymity, unobservability and pseudonymity – a proposal for terminology. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pagina 1–9, Berkeley, California, juli 2000.

[27] Michael K. Reiter en Aviel D. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, november 1998.

[28] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pagina 10–29, Berkeley, California, juli 2000.

[29] Matthew Wright, Micah Adler, Brian N. Levine, en Clay Shields. An analysis of the degradation of anonymous protocols. In *Network and Distributed System Security Symposium*, San Diego, CA, February 2002.

[30] Bart de Win, Vincent Naessens, Claudia Diaz, Stefaan Seys, Carolien Goemans, Joris Claessens, prof. Bart De Decker, prof. Jos Dumortier, en prof. Bart Preneel. Anonymity and privacy in electronic services – technologies overview, november 2001.

[31] Wei Dai. Two attacks against freedom, a commercially implemented pipenet-like protocol. `http://www.eskimo.com/~weidai/freedom-attacks.txt`, mei 1999.

[32] Oliver Berthold, Hannes Federrath, en Stefan Köpsell. Web mixes: A system for anonymous and unobservable internet access. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pagina 115–129, Berkeley, California, juli 2000.

[33] Oliver Berthold, Ronny Standtke, en Andreas Pfitzmann. The disadvantages of free mix routes and how to overcome them. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pagina 27–42, Berkeley, California, juli 2000.

[34] Invisible irc project. `http://invisiblenet.net/`.

[35] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, februari 1981.

[36] Lance Cottrell. Mixmaster & remailer attacks. `http://www.obscura.com/~loki/remailer/remailer-essay.html`.

[37] Michael G. Reed, Paul F. Syverson, en David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Special Areas in Communications*, 16(4):482–494, 1998.

[38] Wei Dai. Pipenet 1.1. `http://www.eskimo.com/~weidai/pipenet.txt`, november 1998.

[39] Brandon Wiley. Interoperability through gateways. In *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*. O'Reilly, maart 2001.

[40] William Ahern. Anonnet: Anonymous network project. `http://www.authnet.org/anonnet/`.

[41] Atnet, anonymous tunneling network. `http://atnet.sourceforge.net`.

[42] Tsocks, a transparent socks proxying library. `http://tsocks.sourceforge.net`.

[43] Sockscap. `http://www.socks.nec.com/reference/sockscap.html`.

[44] Andy Oram, editor. *Peer-to-Peer. Harnessing the Power of Disruptive Technologies*. O'Reilly, maart 2001.