Dankwoord

De eerste persoon die ik wil vermelden in dit dankwoord is Prof. Haegemans. Toen ik vorig jaar onder de thesisonderwerpen van het departement Computerwetenschappen geen enkele thesis vond die me echt fascineerde, was zij degene die me aanraadde met Frank Piessens te gaan praten over een alternatief onderwerp. Zonder haar had deze thesis gewoon niet bestaan.

Ik zou ook Prof. De Decker willen bedanken voor zijn medewerking aan de paper voor het World Computer Congress 2000, vooral voor zijn interessant idee mobiele agenten te gebruiken om de netwerkbelasting van de Secret Query Database draagbaar te maken.

Erik Van Hoeymissen mag hier zeker niet ontbreken. Bedankt voor de hulp bij de papers, het "blokken" van de eerste versies van deze tekst, maar bovenal bedankt voor de steun en de kritiek die niet gewoon opbouwend maar echt motiverend hebben gewerkt.

En dan is er natuurlijk Frank Piessens, die de gecumuleerde rollen van begeleider en promotor van deze thesis glansrijk heeft vervuld. Toen ik vorig jaar bij hem kwam aankloppen had hij na één gesprek volledig begrepen wat ik in een thesis zocht, en een week later heeft hij dit splinternieuwe thesisonderwerp uit de kast getoverd. Ik zou hem vooral willen bedanken voor zijn niet aflatende enthousiasme, zijn onvoorwaardelijke steun en zijn prachtige begeleiding door me op de juiste momenten een nieuwe uitdaging voor te schotelen. Hij was meer dan promotor en begeleider alleen, hij was een coach. Hartelijk bedankt ook voor de grote hulp bij het schrijven en het indienen van de paper en de nuttige opmerkingen op deze tekst.

Tot slot zou ik graag het thuisfront bedanken, de mensen zonder wie ik nooit tot hier was geraakt. Bedankt Mams en Taps voor de steun doorheen de laatste vijf jaren, vooral op de momenten dat het even moeilijker ging. Tsjoep, bedankt van mij te laten winnen bij de ontspannende spelletjes Tetrinet, en Marilyn, bedankt voor geduldig mijn wekelijks gezanik te aanhoren tijdens de autoritten tussen Lanaken en Leuven.

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Historisch overzicht

De cryptografie heeft als wetenschap een opmerkelijke evolutie achter de rug. Reeds in de klassieke oudheid was er nood aan beveiliging van gevoelige documenten. Julius Caesar gebruikte de naar hem genoemde *Caesar-cijfer* die simpelweg elke letter verving door de letter drie plaatsen verder in het alfabet, dus de 'A' wordt een 'D', de 'B' wordt een 'E', ... Sindsdien is de cryptografie door de eeuwen heen gestaag gevorderd, net als andere wetenschappen, met als enige echte toepassing het veilig uitwisselen van strategische boodschappen in oorlogstijd.

Tot in het begin van de twintigste eeuw werd er heel weinig onderzoek verricht naar cryptografie. De *Playfair-cijfer*, die door de Britten tijdens de Eerste Wereldoorlog gebruikt werd, is met de huidige kennis van het domein helemaal niet moeilijk te kraken. Na de Eerste Wereldoorlog kwam er eindelijk schot in de zaak. Cryptografie werd toen enkel bestudeerd voor militaire doeleinden. Militaire organisaties van alle grote naties, waaronder ook de Verenigde Staten, wierpen zich gretig op de ontwikkeling van sterke encryptie-algoritmes en het kraken van de algoritmes van andere staten. Het was van cruciaal belang dat onderzoeksresultaten niet in handen van de vijand vielen, en dus werd alle kennis over cryptografie angstvallig geheim gehouden. Tot en met de Tweede Wereldoorlog was de publieke cryptografische literatuur op sterven na dood. Nochtans had de wetenschap tegen die tijd al fundamentele vorderingen gemaakt, maar die werden geklasseerd als staatsgeheimen. De enige uitzondering hierop was een paper van Shannon, "The Communication Theory of Secrecy Systems," die in oorlogstijd ontwikkeld was maar na de oorlog, waarschijnlijk per vergissing, aan het publiek bekend is gemaakt.

Deze uitzondering niet te na gesproken bleef het publieke landschap van de cryptografie in de periode vlak na de oorlog uiterst schaars bezaaid. Pas aan het einde van de jaren zeventig is hier drastische verandering in

gekomen, mede dankzij het ontwerp van DES (Data Encryption Standard) en de uitvinding van publieke-sleutel cryptografie. Enkele pogingen van de NSA (National Security Agency, de Amerikaanse staatsveiligheid) om de plotselinge opmars van degelijke publieke cryptografie te stuiten mochten niet baten.

Sindsdien is de cryptografie met reuzensprongen vooruit gegaan. Er zijn sterke algoritmes ontwikkeld voor klassieke encryptie en decryptie van gegevens, maar daarnaast zijn er ook nieuwe principes en toepassingen bedacht onder de vorm van cryptografische protocols. Die protocols lossen vaak problemen op waarvan men intuïtief zou denken dat ze niet oplosbaar zijn, zoals bijvoorbeeld eerlijk kop-of-munt spelen over de telefoon. De problemen die de cryptografie oplost worden meer en meer geavanceerd. Eén van die geavanceerde problemen is het onderwerp van deze thesis: Secure Distributed Computing.

1.2 Omschrijving van het probleem

Alice en Bob[1] zijn twee miljonairs die tijdens hun dagelijkse lunch op de golfclub in een discussie gewikkeld geraken over wie van hen getweeën het rijkst is. Zoals het goede Belgische miljonairs betaamt heeft de fiscus het raden naar de juiste omvang van hun fortuin en wensen ze dat ook zo te houden. Daarom is geen van beide partijen bereid het juiste bedrag in het oor van de ander te fluisteren. Dit probleem is bekend als *Yao's miljonairsprobleem*, omdat Andrew Yao het als eerste geformuleerd en opgelost heeft.

Blijkbaar geraken Alice en Bob uit hun probleem (misschien hebben ze de hulp ingeroepen van de ober die lang geleden cryptografie gestudeerd heeft maar nooit aan de bak gekomen is), want uiteindelijk zijn ze het erover eens dat Bob het rijkst is. Met elk glaasje champagne dat ze bij de kaviaar nuttigen begint Alice er in Bobs ogen steeds knapper uit te zien. Ook Alice is niet ongevoelig voor de gemoedelijke stemming aan tafel, het gezellige kaarslicht en het idee haar fortuin meer dan te verdubbelen. Ze hebben echter allebei een zekere reputatie hoog te houden op de golfclub, dus noch Alice noch Bob is van plan van wal te steken met een passionele liefdesverklaring. Geen van hen wil door de ander afgewezen worden en daarbij gezichtsverlies lijden voor de volledige high-society van de streek. Als hun gevoelens wederzijds zijn zullen ze de kans niet laten liggen om samen nog lang en gelukkig door het leven te gaan. Ze zoeken een manier om uit te vinden of er voor hen een toekomst is weggelegd zodanig dat als Bob Alice wel ziet zitten maar Alice toch meer verlangt van een man dan een goed inkomen, Alice nooit zal weten dat Bob een oogje had op haar, en omgekeerd. Op die manier behoudt iemand die afgewezen wordt zijn

[1] Alice en Bob zullen we in deze tekst nog vaak tegenkomen. Zij maken het verhaal net iets levendiger dan de twee abstracte partijen A en B.

waardigheid, als hij tenminste met een stalen gezicht zijn ontgoocheling kan verbergen. Dit probleem wordt het *match-making probleem* genoemd.

Het algemene probleem van Secure Distributed Computing is het volgende: stel dat n verschillende partijen P_i, $i = 1 \ldots n$, een geheime invoer x_i hebben. Ze willen allemaal graag de functie $f(x_1, \ldots, x_n)$ berekenen, maar niemand wil dat er meer over zijn geheime invoer bekend gemaakt wordt dan wat door het functieresultaat geïmpliceerd wordt. Het is bijvoorbeeld zinloos met twee partijen Secure Distributed Computing te gebruiken om de som van twee geheime getallen te berekenen, want elke partij kan de invoer van de ander berekenen als het verschil tussen het functieresultaat en zijn eigen invoer. Ook mag één van de partijen, na uitvoering van een protocol om $f(x_1, \ldots, x_n)$ te berekenen, niet in staat zijn zonder enige hulp van de andere partijen een andere functie $g(x_1, \ldots, x_n)$ te berekenen, want ook dan kan hij meer te weten komen dan wat geïmpliceerd wordt door het resultaat van f alleen.

Het Secure Distributed Computing probleem is natuurlijk triviaal op te lossen als er een bijkomende partij is die door iedereen vertrouwd wordt: iedereen stuurt zijn geheime invoer naar de vertrouwde partij zodat hij $f(x_1, \ldots, x_n)$ kan berekenen. Hij maakt dan het juiste resultaat aan alle partijen bekend. In deze tekst gaan we echter op zoek naar oplossingen indien er zo geen vertrouwde partij voorhanden is.

De bovenstaande problemen waarmee Alice en Bob te kampen kregen zijn speciale gevallen van het Secure Distributed Computing probleem. Het miljonairsprobleem is equivalent met Secure Distributed Computing voor twee partijen, Alice en Bob, waarbij de geheime invoeren x_A en x_B de juiste bedragen van hun fortuin voorstellen en de te evalueren functie gelijk is aan $f(x_A, x_B) = (x_A > x_B)$. Na het protocol weet Bob niets over x_A behalve het feit dat x_A groter of kleiner is dan x_B, en mutatis mutandis geldt hetzelfde voor Alice. Het match-making probleem past ook in dit plaatje als we het als volgt beschouwen: de geheime invoer van Alice bestaat uit één enkele bit a, die 1 is als ze Bob wel ziet zitten en die 0 is als dat niet het geval is. Op dezelfde manier heeft Bob een geheime invoerbit b. De te evalueren functie is de logische AND tussen de twee bits: $f(a, b) = a \wedge b$. Merk op dat als Bob een 1 als invoer geeft en het resultaat toch een 0 blijkt te zijn, hij zeker weet dat Alice een 0 heeft ingegeven. Dit is niet erg, want dit wordt louter door het functieresultaat geïmpliceerd. De bedoeling is dat Alice in dit geval niet kan weten dat Bob een 1 als invoer had. Nog een ander voorbeeld is een geheime stemming: de functie f is een optelling van alle stemmen en elke partij heeft een geheime invoer die ja ($x_i = 1$) of nee ($x_i = 0$) kan zijn. De uitvoer van het protocol is het totaal aantal ja-stemmen. Mits enkele aanpassingen aan de invoer en de functie behoort ook een gewogen stemming tot de mogelijkheden.

Elke functie met een invoer van eindige grootte is implementeerbaar als een Booleaans circuit. Een vaak gevolgde tactiek om een protocol te ontwer-

pen dat elke functie volgens de regels van Secure Distributed Computing kan evalueren is dan ook regels op te stellen die toelaten een willekeurig Booleaans circuit te evalueren. Daar elk logisch circuit kan gerealiseerd worden met enkel NOT- en AND-poorten, zullen de meeste protocols zich beperken tot regels voor deze twee poorten. Om een willekeurig circuit volgens de regels van Secure Distributed Computing te evalueren, is het echter niet voldoende regels vast te leggen om een AND- en een NOT-poort op zich veilig te evalueren. Er moet ook een manier worden voorzien om die poorten "aan mekaar te koppelen". Het is geen goede oplossing elke poort op zich veilig te evalueren en zo doorheen het circuit naar de uitvoer toe te werken, want op die manier komen de partijen niet alleen de uitvoer te weten maar ook alle tussenresultaten van het circuit. Hieruit kunnen ze informatie afleiden over elkanders invoer die niet geïmpliceerd wordt door het functieresultaat alleen.

Stel dat we in een situatie zitten waarbij één van de partijen een functie g wil geheim houden, terwijl het protocol dat we willen gebruiken enkel gegevens geheim kan houden. We kunnen dit probleem omzeilen door een soort *universeel* circuit als functie f te nemen, dat een codering voor g kan interpreteren en uitvoeren. Langs de andere kant, als één van de partijen gegevens geheim willen houden terwijl het protocol enkel geheimhouding van functies toelaat, kan hij de gegevens "hardwired" in de geheime functie aanbrengen. Dankzij deze equivalentie tussen gegevens en functies is het dus zinloos een onderscheid tussen deze twee te maken, ze zijn perfect uitwisselbaar.

Een nog algemenere maar weinig gebruikte versie van Secure Distributed Computing laat toe dat elke partij P_i een eigen uitvoer $f_i(x_1, \ldots, x_n)$ krijgt, maar niets kan te weten komen over de geheime uitvoer van een andere partij. Hoewel er protocols zijn waar private uitvoer efficiënter kan gebeuren, kunnen we deze uitbreiding gemakkelijk realiseren door aan elke uitvoerdraad van het circuit nog een bijkomende XOR-poort te schakelen waarvan de andere ingang verbonden is met een geheime invoerbit van P_i.

1.3 Doelstellingen en overzicht

Het doel van deze thesis was vooreerst een inwerking in het domein van de cryptografie, om vervolgens na te gaan wat precies de huidige stand van de wetenschap is voor Secure Distributed Computing in het bijzonder. Deze thesis bestond dan ook grotendeels uit literatuurstudie: het is niet vanzelfsprekend in de hooimijt aan cryptografische publicaties een naald over Secure Distributed Computing te vinden. Het doorworstelen van deze publicaties om het voorgestelde protocol van begin tot einde te begrijpen is nog veel minder vanzelfsprekend. Tenslotte was het ook de bedoeling een stap verder te zetten door bestaande protocols te verbeteren of het praktisch

nut ervan in te schatten.

Deze tekst is niet enkel bedoeld als een verslag van mijn thesiswerkzaamheden doorheen het afgelopen jaar. Deze tekst tracht meerwaarde te bieden ten opzichte van de originele papers door de protocols op een meer intuïtieve manier aan te brengen. De oorspronkelijke publicaties geven exacte beschrijvingen en rigoureuze bewijzen van netjes afgeronde protocols, zoals het een degelijke wetenschappelijke paper ook betaamt, maar laten het verwerven van inzicht in het protocol over aan de lezer. Deze tekst tracht, naast de beschrijving van het protocol, de achterliggende manier van denken mee te geven, het waarom van de verschillende elementen in het protocol duidelijk te maken en de aandacht te trekken op de invloed die ogenschijnlijk kleine details hebben op de veiligheid.

In het volgende hoofdstuk verduidelijken we wat we precies verstaan onder "veiligheid van een protocol". We zullen zien dat er verschillende modellen zijn voor het (wan)gedrag en de capaciteiten van eventuele valsspelers. Dit laat ons toe de veiligheid van de verschillende protocols beter in te schatten. In hoofdstuk 3 zullen we enkele begrippen uit de discrete wiskunde aanbrengen die we verderop nodig hebben. Hoofdstuk 4 zal de lezer wapenen met de nodige cryptografische bagage om de besproken protocols te begrijpen. Hoofdstuk 5 behandelt een opmerkelijk protocol dat Secure Distributed Computing oplost met behulp van speelkaarten in plaats van met computers. In hoofdstuk 6 bespreken we een aantal protocols voor Secure Distributed Computing met twee partijen, terwijl hoofdstuk 7 enkele protocols behandelt voor een willekeurig aantal partijen. We maken dit onderscheid tussen protocols voor twee partijen en voor meerdere partijen omdat het ontwerp van deze laatste nog net iets meer problemen met zich meebrengt. In hoofdstuk 8 bespreken we enkele toepassingen die voorlopig misschien nog tot de verbeelding spreken maar met efficiënte protocols voor Secure Distributed Computing praktisch gerealiseerd kunnen worden. Tot slot geven we enkele nabeschouwingen in hoofdstuk 9.

Hoofdstuk 2

Wat is veiligheid?

2.1 Het gedrag van de spelers

Als we ervan uit mogen gaan dat de mens door en door eerlijk is, is er geen cryptografie nodig. Cryptografie is eigenlijk de meest misantrope van alle wetenschappen, want ze is volledig gebaseerd op wantrouwen. Iemand die cryptografie gebruikt vertrouwt het liefst helemaal niemand, en als hij dan toch iemand moet vertrouwen wil hij precies weten hoe diep dat vertrouwen moet gaan.

We zullen vanaf nu een speler die cryptografie gebruikt om zijn geheime gegevens te beschermen en geen pogingen onderneemt om beschermde gegevens van anderen te achterhalen *eerlijk* noemen. Iemand die daarentegen wel uit is op gegevens die niet voor zijn ogen bestemd zijn, noemen we *oneerlijk* of een *tegenstander*. Of deze benamingen terecht zijn is een andere zaak: de drugsdealer die zijn volgende deal via geëncrypteerde e-mail afspreekt noemen we eerlijk, de gerechtelijke politie die zijn boodschappen tracht te kraken noemen we oneerlijk. Het zwaard van de cryptografie zal altijd langs twee kanten snijden.

Een eerlijke speler volgt perfect de regels van het protocol, gaat geen extra communicatie of berekeningen aan buiten wat er vastgelegd is door het protocol en houdt zijn private gegevens ook echt geheim. Een eerlijke speler die het recept van Coca-Cola geëncrypteerd met de Caesar-cijfer toegestuurd krijgt, zal niet eens de moeite nemen dit belachelijk eenvoudige algoritme te kraken. Elke speler die niet aan deze voorwaarden voldoet is oneerlijk. We maken echter een fundamenteel onderscheid tussen twee verschillende soorten tegenstanders. Een *semi-eerlijke* tegenstander (Engels: *semi-honest adversary*) is een tegenstander die de regels zoals voorgeschreven door het protocol perfect volgt, maar alles in het werk zal stellen om uit de gegevens die hij vergaart in de loop van het protocol zo veel mogelijk informatie te halen. Een *kwaadaardige* tegenstander (Engels: *malicious adversary*) kan bovendien afwijken van de regels van het protocol. Hij

kan boodschappen doorsturen die helemaal niet geconstrueerd zijn zoals het protocol voorschrijft, maar op een manier die hem toelaat informatie van anderen te achterhalen of het protocol te verstoren. Semi-eerlijke en kwaadaardige tegenstanders worden in de literatuur soms ook *passieve* en *actieve* tegenstanders genoemd.

In een protocol voor meerdere partijen kunnen er ook meerdere tegenstanders zijn. Om het slechtst mogelijke geval te beschouwen moeten we veronderstellen dat al deze tegenstanders gestuurd worden door één enkel "brein van de bende". In het geval van semi-eerlijke tegenstanders heeft dit centrale brein toegang tot de gegevens van alle partijen die hij in zijn macht heeft. In het geval van kwaadaardige tegenstanders kan hij de partijen die hij beheerst opdragen op een gecoördineerde manier de correcte en veilige werking van het protocol te verstoren.

We mogen ons echter niet blind staren op deze twee categorieën van tegenstanders, we kunnen de veiligheid van protocols sterker nuanceren. Op het eerste gezicht zou men zeggen dat een protocol dat enkel bestand is tegen semi-eerlijke tegenstanders, maar niet tegen kwaadaardige tegenstanders, weinig praktische relevantie heeft omdat we moeilijk kunnen veronderstellen dat alle tegenstanders in de echte wereld netjes de regels van het protocol volgen. In praktische situaties maakt het echter een groot verschil of, indien kwaadaardig valsspelen mogelijk is, het valsspelen door de eerlijke spelers gedetecteerd wordt of niet. Een bedrijf dat haar naam wil hooghouden kan het zich niet permitteren kwaadaardig vals te spelen in een protocol waar ze dit niet ongemerkt kunnen doen, ook al is het protocol strikt gezien niet bestand tegen kwaadaardige tegenstanders.

Voor Secure Distributed Computing in het bijzonder maken we nog twee opmerkingen. Een eerste is dat het door de formulering van de probleemstelling onmogelijk is te controleren of een partij liegt over zijn invoer. Alice kan bijvoorbeeld in het miljonairsprobleem ongemerkt een veel grotere waarde opgeven dan zij werkelijk bezit. Het protocol garandeert de geheimhouding van haar invoer, dus niemand zal haar daar ooit voor op de vingers tikken. Dit probleem is nu eenmaal inherent aan de probleemstelling. Daarom definiëren we de correcte uitvoer van de functie als de waarde van de functie voor de invoer die de partijen opgegeven hebben, niet voor de invoer die ze hadden moeten opgeven.

Ten tweede moeten we constateren dat er in gedistribueerde systemen geen gelijktijdigheid bestaat. Het is dus ook onmogelijk dat alle partijen op exact hetzelfde ogenblik de uitvoer van het circuit te weten komen. Dit impliceert dat een partij die de uitvoer eerder krijgt dan een andere, kan verhinderen dat de andere partij de uitvoer ooit te weten komt door het protocol vroegtijdig af te breken. Sommige protocols voorzien de mogelijkheid om de uitvoer bit per bit te openen, zodat een kwaadaardige tegenstander maximaal één bit voordeel kan hebben ten opzichte van de andere spelers door het protocol stop te zetten. Misschien is het mogelijk de stapgrootte verder

te verfijnen door de uitvoer te encrypteren en de sleutel ervan bit per bit te onthullen. Op die manier bereikt een kwaadaardige tegenstander met het vroegtijdig afbreken slechts een voordeel van een factor twee aan rekentijd nodig om de uitvoer te kraken ten opzichte van de andere spelers.

2.2 De rekenkracht van de spelers

We verwachten van het protocol dat het garandeert dat andere partijen bepaalde informatie "niet te weten kunnen komen". Iets niet te weten kunnen komen is echter relatief: met een industriële staalbrander krijg je veel sterkere kluizen open dan met een koevoet. In de cryptografie is het niet anders. Het FBI heeft heel wat meer brute rekenkracht ter beschikking om de factorisatie van een groot getal te berekenen dan het bescheiden PC'tje op mijn bureau kan opbrengen.

De enige werkelijk tot op de graat zuivere protocols zijn de protocols die ontworpen zijn in de veronderstelling dat alle deelnemers over een onbegrensde rekenkracht beschikken. We noemen zo een protocol *informatietheoretisch veilig*. Informatietheoretische veiligheid impliceert dat de doorgestuurde informatie totaal geen informatie over de te beschermen geheimen bevat. Stel dat Alice bij de uitvoering van een protocol zekere gegevens x geheim wenst te houden. Informatietheoretische veiligheid van haar gegevens betekent dat het transcript[1] van de communicatie net zo goed het transcript had kunnen zijn van een communicatie waarin het geheim van Alice eender welke $y \neq x$ is, maar ze een andere keuze voor willekeurige waarden heeft gemaakt. Als dit niet mogelijk is, zit er heel diep toch informatie over x verborgen in het transcript. Hoe diep dit ook is, iemand die beschikt over onbegrensde rekenkracht kan die informatie er in een vingerknip uithalen.

Het mag duidelijk zijn dat informatietheoretische veiligheid een loodzware eis is. Het is zelfs zo dat veel cryptografische bouwblokken die we verderop zullen bespreken vanuit informatietheoretisch standpunt gewoon onmogelijk zijn. In het bijzonder toont Cramer in [13] aan dat er voor het evalueren van een AND-poort met bescherming van de invoer van beide partijen (het match-making probleem) informatietheoretisch gezien geen protocol *kan* bestaan[2]. Stel dat Alice en Bob, twee partijen met onbegrensde rekenkracht, respectievelijk invoerbits a en b hebben. We gaan aantonen dat als $a = 0$ Alice altijd in staat is te beslissen of $b = 0$ of $b = 1$, ongeacht het gebruikte protocol, hetgeen rechtstreeks indruist tegen de definitie van

[1] Met de term *transcript* bedoelen we de sequentie van boodschappen die tijdens een protocol heen en weer worden gezonden.

[2] Althans als de communicatiekanalen perfect zijn. Als we bijvoorbeeld een communicatiekanaal hebben dat bits slechts met een kans van één op twee op hun bestemming brengt, is het verzenden van een bit equivalent met een oblivious transfer van Rabin (zie sectie 4.5). Rabin-OT is equivalent met $\binom{2}{1}$-OT, en Cramer geeft in [13] een protocol dat Secure Distributed Computing mogelijk maakt enkel op basis van $\binom{2}{1}$-OT.

Secure Distributed Computing.

Zij $\mathcal{T}$ het transcript van de communicatie voor het volledige protocol en zij $x_A \in \{0,1\}^*$ en $x_B \in \{0,1\}^*$ respectievelijk de willekeurige bits die Alice en Bob hebben gebruikt in hun berekeningen. Als $b = 0$, dan vereist informatietheoretische veiligheid dat er een andere keuze van willekeurige bits $x'_A \in \{0,1\}^*$ bestaat zodat $\mathcal{T}$ het transcript is van een uitvoering van het protocol waarbij $a = 1$. Enkel als zo'n keuze x'_A bestaat, bevat $\mathcal{T}$ geen Shannon-informatie over a. Omdat Alice onbegrensde rekenkracht heeft, mogen we er gerust van uitgaan dat zij een dergelijke x'_A in een wip gevonden heeft.

Als $b = 1$, kan $\mathcal{T}$ onmogelijk consistent zijn met $a = 1$ en een andere keuze van willekeurige bits x''_A, want als Alice haar input verandert in $a = 1$ verandert de uitvoer van het protocol! Het is onmogelijk dat $\mathcal{T}$ net zo goed het transcript kan zijn van een communicatie waarbij de uitvoer 0 is als van een communicatie waarbij de uitvoer 1 is. De enige manier waarop $\mathcal{T}$ consistent zou kunnen zijn met twee verschillende uitvoeren is als $\mathcal{T}$ geen Shannon-informatie bevat over de uitvoer, hetgeen zou betekenen dat het protocol helemaal geen AND-poort evalueert. Alice besluit dat $b = 0$ als er een x'_A bestaat zodat $\mathcal{T}$ consistent is met $a = 1$ en de willekeurige bits x'_A, en besluit dat $b = 1$ als er zo geen x'_A bestaat.

Als informatietheoretische veiligheid de enige vorm van veiligheid zou zijn, zouden we deze thesis beter hier stopzetten. We hebben immers net aangetoond dat er voor een elementair basisprobleem geen oplossing kan bestaan. De veronderstelling dat iedereen beschikt over onbegrensde rekenkracht is inderdaad zeer veilig, maar overdreven voorzichtig. Meestal vragen we niet dat het geheim *nooit* kan achterhaald worden. Als het met alle computers van de wereld tesamen gemiddeld nog duizend jaar duurt eer het geheim kan ontfutseld worden, zijn we praktisch zeker dat niemand zich die moeite zal getroosten. Maar waar trekken we dan de grens wat betreft de beschikbare rekenkracht en -tijd? Vanaf wanneer kunnen we zeggen dat een bepaald algoritme veilig is? Misschien wil de Pakistaanse regering haar kernprogramma wel strikt geheim houden gedurende de komende honderd jaar, maar het geheime verjaardagscadeau voor Jan mag hij met zijn PC op enkele weken tijd kunnen achterhalen, dan is zijn verjaardag toch al voorbij. Hier verschijnt complexiteitstheorie op het toneel. Complexiteitstheorie voorziet een methodologie om de "moeilijkheid" van algoritmes te analyseren. Informatietheorie vertelt ons dat nagenoeg alle cryptografische algoritmes kunnen gebroken worden, complexiteitstheorie vertelt of ze kunnen gebroken worden voor het heelal instort. Een uitvoerige beschrijving van de complexiteitstheorie ligt buiten het bestek van deze tekst. We zullen enkel kort aanraken wat cryptografische veiligheid precies inhoudt.

Elk cryptografisch probleem heeft een *veiligheidsparameter* k. Deze parameter is meestal evenredig met de omvang van de invoer, bijvoorbeeld het aantal bits van een priemgetal. Als er een algoritme bestaat dat de op-

lossing voor het probleem berekent en dat een tijds- en ruimtecomplexiteit heeft die kan uitgedrukt worden als een veelterm in k, beschouwt de cryptografie dit probleem als een gemakkelijk probleem. We zeggen dan dat er een polynomiaal algoritme bestaat om het probleem op te lossen. Merk op dat er geen beperkingen worden opgelegd aan de graad of de coëfficiënten van de veelterm. Als het beste algoritme om het probleem op te lossen echter van exponentiële complexiteit is in k, wordt het probleem als moeilijk beschouwd. Intuïtief (maar wiskundig niet helemaal juist) kunnen we stellen dat een algoritme van exponentiële complexiteit is als een verhoging van de veiligheidsparameter met één eenheid het probleem dubbel zo moeilijk maakt.

We spreken hier over het beste algoritme om een probleem op te lossen. Maar hoe weten we wat het beste algoritme is? Het feit dat de huidige wetenschap geen beter algoritme kent wil niet zeggen dat er geen beter algoritme bestaat. Er zijn een aantal problemen waaraan gedurende lange tijd grondige aandacht is besteed in onderzoek, maar waarvoor nog steeds geen polynomiaal algoritme gevonden is. Van die problemen wordt aangenomen dat het moeilijke problemen zijn, maar het blijft natuurlijk een veronderstelling. Het bekendste voorbeeld is het factoriseren van een groot getal dat een product is van twee grote priemgetallen. Verder in de tekst zullen we nog enkele problemen tegenkomen waarvan vermoed wordt dat er geen polynomiaal algoritme voor bestaat.

Een algoritme of protocol noemen we *cryptografisch veilig* als het ontfutselen van de geheime informatie een moeilijk probleem is, dus als er geen polynomiaal algoritme bestaat dat die informatie achterhaalt. In de praktijk is dit een veel relevanter criterium dan informatietheoretische veiligheid. Een moeilijk probleem legt een groot complexiteitsvoordeel bij de eerlijke gebruiker: als hij de veiligheidsparameter maar een klein beetje verhoogt wordt het kraken van het systeem toch een heel pak moeilijker.

Maar laten we informatietheoretische veiligheid niet volledig overboord gooien als een speelgoedje voor theoretici. Er is onderzoek lopende naar zogenaamde quantumcomputers. Dit zijn computers die gebaseerd zijn op quantummechanische principes. Dergelijke computers kunnen in meerdere toestanden tegelijk verkeren, in tegenstelling tot klassieke computers die altijd in één bepaalde toestand verkeren. In 1994 heeft Peter Shor een algoritme voor een quantumcomputer gepubliceerd dat in staat is in polynomiale tijd grote getallen te factoriseren en discrete logaritmen te berekenen. Dit vormt voorlopig geen reden tot paniek in de cryptografische wereld, want momenteel zijn enkel quantumcomputers van enkele bits groot gerealiseerd. Alhoewel het onwaarschijnlijk is dat quantumcomputers in de nabije toekomst praktisch bruikbaar worden, kunnen we de informatietheoretische veiligheid toch als troef achter de hand blijven houden.

Hoofdstuk 3

Een beetje getaltheorie

Heel wat cryptografische algoritmes en protocols bouwen voort op resultaten uit de getaltheorie. De lezer is waarschijnlijk reeds vertrouwd met de beginselen van groepentheorie en discrete wiskunde. We zullen hier toch nog enkele basisbegrippen opfrissen om daarna onmiddellijk van wal te steken met meer geavanceerde wiskunde. Het is zeker niet de bedoeling een algemeen overzicht te geven van de huidige stand van zaken in de getaltheorie. Integendeel, we willen de lezer enkel de wiskundige bagage meegeven die later haar nut zal bewijzen in de bespreking van cryptografische protocols.

Deze sectie is grotendeels gebaseerd op [6]. Voor meer details kan de lezer meestal daar terecht, of natuurlijk in de gespecialiseerde literatuur.

3.1 Basisbegrippen

Een *groep* is een verzameling G met een binaire bewerking $*$ op de elementen ervan. We noteren de groep als $\mathbb{G} = \langle G, * \rangle$. Een groep voldoet aan de volgende eigenschappen:

1. De bewerking is intern: $a, b \in G \Rightarrow a * b \in G$
2. De bewerking is associatief: $(a * b) * c = a * (b * c)$
3. Er is een neutraal element 1 zodat $1 * a = a * 1 = a$ voor alle $a \in G$.
4. Elk element $a \in G$ heeft een inverse $a^{-1} \in G$ zodat $a * a^{-1} = a^{-1} * a = 1$

Het aantal elementen in een groep noemen we de *orde* van de groep. Voor elke groep $\mathbb{G}$ geldt[1]

$$\forall\, a \in G: \quad a^{|G|} = 1$$

[1] De machtsverheffing heeft de gebruikelijke betekenis van het herhaaldelijk samenstellen van een element met zichzelf

Een groep is *cyclisch* als er een element $g \in G$ bestaat zodat voor elke $a \in G$ er een geheel getal i is waarvoor geldt $g^i = a$. We noemen g een *generator* van de groep $\mathbb{G}$.

Een eenvoudig voorbeeld van een groep is $\mathbb{Z}_n = \langle Z_n, + \rangle$, de gehele getallen van 0 tot $n-1$ met de optelling modulo n. Als we Z_n^* definiëren als de verzameling elementen kleiner dan n en copriem met n,

$$Z_n^* = \{a \in \mathbb{Z} : 1 \leq a < n \text{ en } \text{ggd}(a, n) = 1\}$$

dan kunnen we aantonen dat Z_n^* een groep $\mathbb{Z}_n^* = \langle Z_n^*, \cdot \rangle$ vormt met de vermenigvuldiging modulo n. De orde van deze groep wordt gegeven door de *Euler Totiënt Functie* $\phi(n)$, die met behulp van de volgende regels kan berekend worden:

- Voor een priemgetal p : $\phi(p) = p - 1$
- Voor een macht α van een priemgetal p : $\phi(p^\alpha) = p^{\alpha-1}(p-1)$
- Voor het produkt van twee gehele getallen m en n : $\phi(mn) = \phi(m)\phi(n)$

In de cryptografie werkt men vaak in de cyclische groep $\mathbb{Z}_p^*$ met p een priemgetal. De orde van deze groep is $\phi(p) = p - 1$, dus voor elke $a \in Z_p^*$ geldt dat

$$a^{p-1} \equiv 1 \mod p$$

Een element p keer met zichzelf vermenigvuldigen geeft terug het element zelf. Dit impliceert dat we machten modulo $(p-1)$ mogen beschouwen.

Als g een generator is van $\mathbb{Z}_p^*$, dan bestaat er voor elk element $a \in \mathbb{Z}_p^*$ een unieke i $(1 \leq i \leq p-1)$ zodat $a \equiv g^i \mod p$. Bovendien geldt er dat

$$g^i \equiv g^j g^k \mod p \iff i \equiv j + k \mod p - 1$$

Er bestaat dus een isomorfisme tussen de groepen $\mathbb{Z}_p^* = \langle Z_p^*, \cdot \rangle$ en $\mathbb{Z}_{p-1} = \langle Z_{p-1}, + \rangle$. We mogen zo veel heen en weer springen tussen deze twee groepen als we willen en we mogen altijd de voorstelling gebruiken die ons het best uitkomt.

3.2 Discrete Logaritme Veronderstelling

Uitbreidingen van het algoritme van Euclides zijn in staat de inverse van een element in $\mathbb{Z}_p^*$ efficiënt te berekenen. Ook de machtsverheffing kunnen we in polynomiale tijd in $|p|$, het aantal bits van p, uitvoeren op de volgende manier. Stel dat we $g^i \mod p$ willen berekenen. We stellen de rij g, g^2, g^4, ..., $g^{|i|}$ op en we vermenigvuldigen die elementen van de rij met elkaar waarvoor er in de binaire voorstelling van i een 1 staat.

Er bestaan dus efficiënte algoritmes om gegeven g en i de waarde g^i mod p te berekenen. De inverse functie van de machtsverheffing blijkt helemaal niet zo eenvoudig te zijn. Er zijn voorlopig nog geen algoritmes gevonden die gegeven g en $a \in Z_p^*$ de waarde i kunnen berekenen zodat $a \equiv g^i$ mod p. Zo'n i noemen we het *discreet logaritme* van a ten opzichte van g modulo p. Algemeen wordt er aangenomen dat het onmogelijk is in $\mathbb{Z}_p^*$ het discreet logaritme in polynomiale tijd te berekenen. Deze veronderstelling staat bekend als de *Discrete Logaritme Veronderstelling (DLV)*. Onder de DLV is de machtsverheffing modulo een priemgetal een éénrichtingsfunctie, d.w.z. dat de functie zelf efficiënt berekenbaar is maar dat er voor de inverse functie geen polynomiaal algoritme bestaat.

3.3 Kwadratische residu's

3.3.1 Definitie

Definitie 3.1 (Kwadratisch Residu). *Een element a van de groep $\mathbb{Z}_n^*$ is een* kwadratisch residu *of kortweg* kwadraat *als het een wortel heeft in $\mathbb{Z}_n^*$, dit wil zeggen dat er een $x \in Z_n^*$ bestaat zodat $a \equiv x^2 \mod n$. Indien a zo geen wortel heeft noemen we a een* kwadratisch nonresidu *of kortweg* niet-kwadraat.

We zullen eerst ingaan op kwadratische residu's in $\mathbb{Z}_p^*$ met p een priemgetal. Gebruik makende van het homomorfisme tussen $\mathbb{Z}_p^*$ en $\mathbb{Z}_{p-1}$ kunnen we volgende stelling bewijzen.

Stelling 3.1. *$a \in Z_p^*$ is een kwadratisch residu modulo p met p een oneven priemgetal als en slechts als $a \equiv g^i \mod p$ waarbij $1 \leq i \leq p-1$ en i even is.*

Bewijs. Laat g een generator zijn van $\mathbb{Z}_p^*$.
$\Leftarrow$ Stel dat $a = g^{2j}$ voor een bepaalde j. Dan is $a = x^2$ waarbij $x = g^j$.
$\Rightarrow$ Beschouw het kwadraat van een element $x = g^j$. Er geldt dan dat $x^2 = g^{2j} \equiv g^i \mod p$ met i een even getal, want het homomorfisme leert ons dat $2j$ gereduceerd wordt modulo $p-1$, hetgeen een even resultaat geeft omdat $p-1$ zelf ook even is.
Daarom zijn enkel elementen die kunnen geschreven worden als g^i met i een even getal kwadratische residu's. □

Een gevolg van deze stelling is dat exact de helft van de elementen in $\mathbb{Z}_p^*$ kwadratische residu's zijn. Inderdaad, er zijn juist $\frac{p-1}{2}$ even getallen tussen 1 en $p-1$. Deze stelling zal goed van pas komen in het bewijs van volgende stellingen, maar ze levert geen praktisch bruikbaar algoritme om te beslissen of een element een kwadratisch residu is of niet. De stelling zegt dat een getal een kwadratisch residu is als en slechts als haar discreet

logaritme even is, maar herinner uit de vorige sectie dat het vinden van het discreet logaritme in $\mathbb{Z}_p^*$ een moeilijk probleem is.

3.3.2 Het Legendre-symbool

Definitie 3.2 (Legendre-symbool). *Het* Legendre-symbool $\mathbf{J_p}(a)$ *bepaalt of* a *een kwadraat is in* $\mathbb{Z}_p^*$ *met* p *priem:*

$$\mathbf{J_p}(a) = \begin{cases} 1 & \textit{als } a \textit{ een kwadraat is in } \mathbb{Z}_p^* \\ -1 & \textit{als } a \textit{ geen kwadraat is in } \mathbb{Z}_p^* \end{cases}$$

Het is duidelijk dat als we een efficiënte manier hebben om $\mathbf{J_p}(a)$ te berekenen, we ook een goed algoritme hebben om te beslissen of a een kwadraat is of niet. Nu blijkt zo'n manier inderdaad te bestaan dankzij de volgende stelling.

Stelling 3.2 (Criterium van Euler). *Het Legendre-symbool* $\mathbf{J_p}(a)$ *kan berekend worden als*

$$\mathbf{J_p}(a) \equiv a^{\frac{p-1}{2}} \mod p$$

Bewijs. Zij g een generator van $\mathbb{Z}_p^*$.
Beschouw als eerste geval dat $a = g^i$ met i even. Volgens stelling 3.1 is a dan een kwadraat modulo p. Als we $i = 2j$ stellen, geldt er dat

$$a^{\frac{p-1}{2}} = g^{j(p-1)} \equiv 1 \mod p$$

want exponenten mogen gereduceerd worden modulo $p-1$.
Het tweede geval is $a = g^i$ met i oneven. Volgens stelling 3.1 is a dan een kwadratisch nonresidu modulo p. Als we $i = 2j+1$ stellen, geldt er dat

$$a^{\frac{p-1}{2}} = g^{j(p-1)} g^{\frac{p-1}{2}} \equiv 1 \cdot (-1) = -1 \mod p$$

want inderdaad

$$\begin{aligned} (-1) \cdot (-1) &\equiv 1 \mod p \\ \Leftrightarrow g^x g^x = g^{2x} &\equiv g^{p-1} \mod p \\ \Leftrightarrow x &\equiv \frac{p-1}{2} \mod p-1 \end{aligned}$$

en dus $g^{\frac{p-1}{2}} \equiv -1 \mod p$. □

Stelling 3.3. *Zij* $a, b \in Z_p^*$ *met* p *een oneven priemgetal. Dan geldt er dat* $\mathbf{J_p}(ab) = \mathbf{J_p}(a)\mathbf{J_p}(b)$.

Bewijs. Zij g een generator van $\mathbb{Z}_p^*$. We kunnen a en b schrijven als $a = g^i \mod p$ en $b = g^j \mod p$. Het produkt van a en b is gelijk aan $ab = g^{i+j} \equiv g^k \mod p$ met $k \equiv i+j \mod (p-1)$. We kunnen nu drie gevallen onderscheiden:

1. i en j even:
   Dan is ook hun som $i + j$ even. Gereduceerd modulo $p - 1$, een even getal, geeft dit opnieuw een even getal k. Volgens stelling 3.1 zijn a, b en ab kwadratische residu's, en dus is $\mathbf{J_p}(ab) = 1 = \mathbf{J_p}(a)\mathbf{J_p}(b)$.

2. i even en j oneven (of omgekeerd):
   $i + j$ is nu oneven, dus is k ook oneven. Weer volgens stelling 3.1 is a een kwadratisch residu terwijl b en ab kwadratische nonresidu's zijn, zodat $\mathbf{J_p}(ab) = -1 = \mathbf{J_p}(a)\mathbf{J_p}(b)$.

3. i en j oneven:
   Dan is $i + j$ even zodat ook k even is. Zowel a als b zijn kwadratische nonresidu's, maar ab is wel een kwadratisch residu en dus is $\mathbf{J_p}(ab) = 1 = \mathbf{J_p}(a)\mathbf{J_p}(b)$.

□

Voor de machtsverheffing modulo p hebben we al een snel algoritme, dus voor het bepalen of een getal een kwadraat is modulo een priemgetal draaien we onze hand ook niet meer om. Maar kunnen we ook de wortel van zo'n kwadraat berekenen? Als p een oneven priemgetal is, is $p \equiv 1 \mod 4$ of $p \equiv 3 \mod 4$. In de volgende stelling bespreken we enkel het laatste geval, omdat dit het eenvoudigste is en we het andere geval verderop niet meer nodig hebben. In [6] wordt eveneens een algoritme gegeven om wortels te trekken als $p \equiv 1 \mod 4$.

Stelling 3.4. *Laat p een oneven priemgetal zijn met $p \equiv 3 \mod 4$ en laat a een kwadratisch residu zijn in $\mathbb{Z}_p^*$. Er bestaat een polynomiaal algoritme dat x berekent zodat $x^2 \equiv a \mod p$.*

Bewijs. $p \equiv 3 \mod 4$, en dus is $p = 4m+3$ voor een geheel getal m. Omdat a een kwadraat is in $\mathbb{Z}_p^*$ hebben we

$$\begin{aligned} 1 = \mathbf{J_p}(a) \equiv a^{\frac{p-1}{2}} \mod p \\ \Leftrightarrow a^{2m+1} \equiv 1 \mod p \\ \Leftrightarrow a^{2m+2} = (a^{m+1})^2 \equiv a \mod p \end{aligned}$$

en dus is a^{m+1} een wortel van a modulo p. □

Het is eenvoudig in te zien dat als x een wortel is van a, $-x$ ook een wortel is van a. Dit zijn de enige twee wortels van a want als $y^2 \equiv a \equiv x^2 \mod p$, dan is $x^2 - y^2 \equiv 0 \mod p$ oftewel $p|[(x-y)(x+y)]$. Dan moet ofwel gelden dat $p|(x-y)$ ofwel dat $p|(x+y)$, zodat $y \equiv \pm x \mod p$. In het algemeen kunnen we stellen dat de vergelijking $a \equiv x^2 \mod p$ ofwel geen ofwel twee oplossingen heeft.

Als $p \equiv 3 \mod 4$ is het bovendien zo dat één van de wortels x en $-x$ op zijn beurt een kwadraat is en de andere een kwadratisch nonresidu is. Inderdaad, stel $x \equiv g^i \mod p$, dan is $-x \equiv g^{\frac{p-1}{2}} g^i = g^{\frac{p-1}{2}+i} \mod p$. Er bestaat een m zodat $p = 4m+3$, dus is $\frac{p-1}{2} = 2m+1$ een oneven getal. Als i even is, is $\frac{p-1}{2} + i$ oneven en omgekeerd. De wortel van a die zelf ook een kwadratisch residu is, noemen we de *principale wortel* van a.

3.3.3 De Chinese Reststelling

Op het eerste gezicht lijkt het dat kwadraten in $\mathbb{Z}_p^*$ niet interessant zijn voor cryptografische toepassingen omdat voor ongeveer alles wat we ermee kunnen doen een polynomiaal algoritme bestaat. Om van cryptografisch nut te zijn moeten er bepaalde dingen mogelijk zijn maar andere dingen weer niet. We zullen aantonen dat de groep $\mathbb{Z}_{pq}^* = \langle Z_{pq}^*, \cdot \rangle$ met p en q twee priemgetallen congruent met 3 mod 4, op dit vlak veel meer mogelijkheden biedt. Maar om dat in te zien hebben we de *Chinese Reststelling*[2] en het constructieve bewijs ervan nodig.

Stelling 3.5 (Chinese Reststelling). *Zij $m_1, m_2, \ldots, m_k$ onderling coprieme gehele getallen, d.w.z. dat $ggd(m_i, m_j) = 1$ voor $1 \leq i < j \leq k$. Zij $a_i \in Z_{m_i}$ voor $1 \leq i \leq k$ en stel $m = m_1 m_2 \cdots m_k$. Dan bestaat er een unieke $y \in Z_m$ zodat $y \equiv a_i \mod m_i$ voor $i = 1 \ldots k$.*

Bewijs. Zij $n_i = \frac{m}{m_i} = m_1 \cdots m_{i-1} m_{i+1} \cdots m_k \in \mathbb{Z}$. Omdat $m_1, \ldots, m_k$ copriem zijn is ook $\text{ggd}(m_i, n_i) = 1$ en dus bestaat er een $b_i \in Z_{m_i}$ zodat $n_i b_i \equiv 1 \mod m_i$. Als we $c_i = n_i b_i$ stellen, geldt er dat

$$c_i = \begin{cases} 1 & \mod m_i \\ 0 & \mod m_j \text{ voor } j \neq i \end{cases}$$

want $c_i \equiv 1 \mod m_i$ en $m_j | c_i$ voor $j \neq i$.
Stel nu $y = \sum_{i=1}^{k} c_i a_i \mod m$. Dan is $y \equiv a_i \mod m_i$ voor elke i, zoals gevraagd was.
We moeten nog bewijzen dat deze y enig is. Stel dat $y' \equiv a_i \mod m_i$ voor elke i dan is $y' \equiv y \mod m_i$ en daar de m_i's copriem zijn volgt dat $y' \equiv y \mod m$ en dus dat y uniek is. □

We hebben daarnet aangetoond dat de vergelijking $a \equiv x^2 \mod p$ twee verschillende oplossingen heeft, x en $-x$, als a een kwadraat is modulo p. Laten we nu eens $n = pq$ nemen met $p \equiv q \equiv 3 \mod 4$. Hoe zit het met de oplossing van de vergelijking $a \equiv z^2 \mod n$? Wel, deze vergelijking is enkel oplosbaar als zowel $a \equiv x^2 \mod p$ als $a \equiv y^2 \mod q$ oplossingen hebben. Stel dat de oplossingen van de eerste vergelijking x en $-x$ zijn en die van

[2] Deze stelling heeft haar naam te danken aan het feit dat de oude Chinezen haar reeds kenden.

de tweede y en $-y$. We kunnen dan de Chinese Reststelling gebruiken om deze oplossingen modulo p en q te combineren tot een oplossing modulo n. Zij c_1 en $c_2 \in \mathbb{Z}_n$ de coëfficiënten zoals ze gebruikt werden in het bewijs van de Chinese Reststelling, dan geldt dus

$$c_1 \equiv \begin{cases} 1 & \mod p \\ 0 & \mod q \end{cases}$$

en

$$c_2 \equiv \begin{cases} 0 & \mod p \\ 1 & \mod q \end{cases}$$

We kunnen inzien dat de vergelijking $a \equiv z^2 \mod n$ vier verschillende oplossingen heeft die we kunnen schrijven als

$$\begin{aligned} z_1 &= c_1 x + c_2 y \\ z_2 &= c_1 x - c_2 y \\ z_3 &= -c_1 x + c_2 y \\ z_4 &= -c_1 x - c_2 y \end{aligned}$$

3.3.4 Kwadratische Residuositeit Veronderstelling

We hebben eigenlijk net een algoritme gegeven dat in staat is in polynomiale tijd de vier oplossingen van $a \equiv z^2 \mod n$ te berekenen. Alweer niks voor onze cryptografische toepassingen? Toch wel. Om dit algoritme te kunnen gebruiken moeten we de factoren p en q kennen. Als we echter enkel n en niet de ontbinding ervan kennen, staan we voor een veel zwaardere klus. Het is vrij eenvoudig in te zien dat het worteltrekken modulo n minstens zo moeilijk is als het factoriseren van n. We kunnen immers bewijzen dat de kennis van twee verschillende wortels de factoren van n verraadt. Als we twee wortels x en y hebben met $x \not\equiv \pm y \mod n$, geldt er dat $(x+y)(x-y) \equiv 0 \mod n$ en dus dat $n|(x+y)(x-y)$. Maar $n \not| (x+y)$ omdat $x \not\equiv y \mod n$ en $n \not| (x-y)$ omdat $x \not\equiv -y \mod n$. Daarom zijn $\text{ggd}(x-y, n)$ en $\text{ggd}(x+y, n)$ niet-triviale delers van n oftewel de factoren van n.

Als we wortels kunnen trekken modulo n kunnen we natuurlijk ook bepalen of een getal een kwadraat is modulo n. Het is echter niet ondenkbaar dat er een manier bestaat om te bepalen of een getal een kwadraat is modulo n zonder daarvoor de wortels te moeten berekenen. Er is echter geen efficiënt algoritme bekend dat zonder de ontbinding van n te kennen kan beslissen of a een kwadratisch residu is of niet. Algemeen wordt aangenomen dat er geen polynomiaal algoritme bestaat dat kwadratische residuositeit modulo een samengesteld getal kan beslissen. Deze veronderstelling staat bekend als de *Kwadratische Residuositeit Veronderstelling (KRV).*

3.3.5 Het Jacobi-symbool

Even terug hebben we het Legendre-symbool besproken. Dit had enkel betekenis in groepen modulo een priemgetal. We zullen nu het Legendre-symbool veralgemenen tot het Jacobi-symbool dat ook voor samengestelde moduli werkt.

Definitie 3.3 (Jacobi-symbool). *Het* Jacobi-symbool $\mathbf{J_n}(a)$ *wordt gedefinieerd als volgt:*

1. *Als n priem is, dan is het Jacobi-symbool $\mathbf{J_n}(a) = 1$ als a een kwadratisch residu is modulo n, en $\mathbf{J_n}(a) = -1$ als a een kwadratisch nonresidu is modulo n.*

2. *Als n een samengesteld getal is, dan is het Jacobi-symbool $\mathbf{J_n}(a) = \mathbf{J_{p_1}}(a) \cdot \mathbf{J_{p_2}}(a) \cdot \ldots \cdot \mathbf{J_{p_k}}(a)$ met $p_1 \ldots p_k$ de priemfactoren van n.*

Net als voor het Legendre-symbool bestaan er ook voor het Jacobi-symbool manieren om het in polynomiale tijd te berekenen, zelfs als de priemontbinding van n niet gekend is. Maar merk op dat het Jacobi-symbool niet kan gebruikt worden om te bepalen of a een kwadratisch residu is modulo n of niet, tenzij n priem is natuurlijk want dan herleidt het Jacobi-symbool zich tot het Legendre-symbool.

Laten we opnieuw de groep $\mathbb{Z}_n^*$ beschouwen met $n = pq$ en $p \equiv q \equiv 3 \mod 4$. Als een element $a \in Z_n^*$ een kwadratisch residu is modulo n, heeft het vier wortels van de vorm $z = \pm c_1 x \pm c_2 y$, waarbij $\pm x$ en $\pm y$ de wortels zijn modulo p en q, respectievelijk, en c_1 en c_2 de coëfficiënten van de Chinese Reststelling. Zonder aan algemeenheid te verliezen kunnen we stellen dat x en y de principale wortels zijn. Dit betekent dat x en y kwadratische residu's zijn modulo p en q, terwijl $-x$ en $-y$ kwadratische nonresidu's zijn. Een getal is enkel een kwadraat modulo n als het tegelijk een kwadraat is modulo p en modulo q. We zien dat van de vier wortels van a enkel de wortel $z_1 = c_1 x + c_2 y$ een kwadratisch residu is modulo n. Dit is de principale wortel van a modulo n. Uitgaande van het feit dat $\mathbf{J_p}(x) = \mathbf{J_q}(y) = 1$ en $\mathbf{J_p}(-x) = \mathbf{J_q}(-y) = -1$ kunnen we ook opmerken dat twee wortels, namelijk $z_1 = c_1 x + c_2 y$ en $z_4 = -c_1 x - c_2 y$, Jacobi-symbool 1 hebben en de andere twee wortels, $z_2 = c_1 x - c_2 y$ en $z_3 = -c_1 x + c_2 y$, Jacobi-symbool -1 hebben. Hier zien we trouwens duidelijk dat het Jacobi-symbool voor samengestelde getallen geen band meer heeft met het al dan niet kwadratisch zijn: $\mathbf{J_n}(z_4) = 1$ terwijl z_4 toch een kwadratisch nonresidu is modulo n.

3.3.6 De groep $\mathbb{Z}_n^*[+1]$

Laten we de verzameling Z_n^* eens beperken tot enkel die elementen die Jacobi-symbool 1 hebben. We noteren deze verzameling als $Z_n^*[+1]$. We kunnen nagaan dat $Z_n^*[+1]$ samen met de vermenigvuldiging modulo n aan

de voorwaarden voldoet om een groep $\mathbb{Z}_n^*[+1] = \langle Z_n^*[+1], \cdot \rangle$ te vormen. In deze groep zijn opnieuw exact de helft van de elementen, namelijk de a's met $\mathbf{J_p}(a) = \mathbf{J_q}(a) = 1$, kwadratische residu's en zijn de andere elementen, waarvoor $\mathbf{J_p}(a) = \mathbf{J_q}(a) = -1$, kwadratische nonresidu's. Bovendien kunnen we volgende stelling bewijzen.

Stelling 3.6. *Zij $n = pq$ met p en q priemgetallen congruent met 3 mod 4. Zij $a, b, c \in Z_n^*[+1]$ zodat $c \equiv a \cdot b \mod n$. Als a en b allebei kwadratische residu's of allebei kwadratische nonresidu's zijn is c een kwadratisch residu modulo n. Als a een kwadratisch residu is en b niet (of omgekeerd) dan is c een kwadratisch nonresidu modulo n.*

Bewijs. We onderscheiden drie gevallen:

1. a en b allebei kwadratische residu's modulo n:
   Dan zijn a en b ook kwadratische residu's modulo p en q, zodat $\mathbf{J_p}(a) = \mathbf{J_p}(b) = \mathbf{J_q}(a) = \mathbf{J_q}(b) = 1$. Volgens stelling 3.3 zijn dan ook $\mathbf{J_p}(ab) = \mathbf{J_q}(ab) = 1$ zodat ab een kwadratisch residu is modulo p en q, en dus ook modulo n.

2. a kwadratisch residu, b kwadratisch nonresidu modulo n (of omgekeerd):
   Er geldt dus dat $\mathbf{J_p}(a) = \mathbf{J_q}(a) = 1$, terwijl $\mathbf{J_p}(b) = \mathbf{J_q}(b) = -1$. Het is niet mogelijk dat $\mathbf{J_p}(b) = 1$ en $\mathbf{J_q}(b) = -1$ of omgekeerd, want dan is $\mathbf{J_n}(b) = -1$ en dus $b \notin Z_n^*[+1]$. Volgens stelling 3.3 geldt dan dat $\mathbf{J_p}(ab) = \mathbf{J_q}(ab) = -1$, zodat ab een kwadratisch nonresidu is modulo p en q, en dus ook modulo n.

3. a en b allebei kwadratische nonresidu's modulo n:
   Nu geldt er dat $\mathbf{J_p}(a) = \mathbf{J_p}(b) = \mathbf{J_q}(a) = \mathbf{J_q}(b) = -1$. Andere mogelijkheden zijn er niet om dezelfde reden als in de vorige stap. We passen weer stelling 3.3 toe en zien dat $\mathbf{J_p}(ab) = \mathbf{J_q}(ab) = 1$, zodat ab een kwadratisch residu is modulo p en q, en dus ook modulo n.

□

Zoals reeds gezegd zijn er snelle algoritmes om van een getal het Jacobisymbool te bepalen. Het is dus niet moeilijk een willekeurig element r van $Z_n^*[+1]$ te genereren, ook als de factorisatie van n niet gekend is. Als we dit getal met zichzelf vermenigvuldigen, zijn we zeker dat het resultaat r^2 een kwadratisch residu is. Omdat $-1 \in Z_n^*[+1]$ en -1 altijd een kwadratisch nonresidu is modulo n, kunnen we r^2 vermenigvuldigen met -1 om een niet-kwadraat te genereren. Samengevat kunnen we naar believen kwadraten en niet-kwadraten genereren in $\mathbb{Z}_n^*[+1]$ zonder daarvoor de factorisatie van n te moeten kennen. Wat we niet kunnen is van een getal beslissen of het al dan niet een kwadratisch residu is; met de factoren p en q op zak herleidt dit probleem zich plots tot een eenvoudige klus.

Hoofdstuk 4

Cryptografische bouwblokken

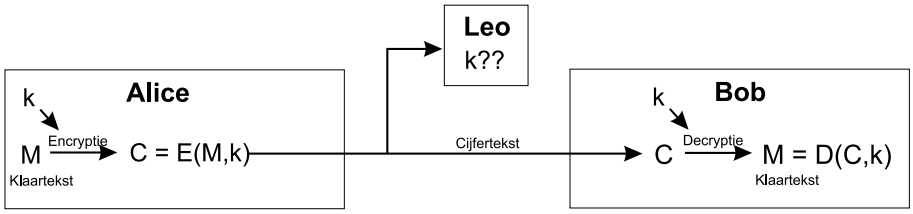
De aandacht van massa's onderzoekers sinds de jaren zeventig heeft de cryptografie geen windeieren gelegd. In de laatste vijfentwintig jaar zijn er ongelooflijk veel nieuwe cryptografische principes, algoritmes en protocols gepubliceerd. Van een aantal van deze principes zullen we verderop in deze tekst dankbaar gebruik maken om het Secure Distributed Computing probleem op te lossen. Meestal zullen we op dat moment die cryptografische bouwblokken als een zwarte doos beschouwen om de aandacht niet van het protocol zelf af te leiden. In dit hoofdstuk bespreken we de verschillende bouwblokken die we tot onze beschikking hebben en om de lezer niet met een kluitje in het riet te sturen, gluren we ook even naar de inhoud van de zwarte dozen.

Nagenoeg de volledige literatuur over cryptografie is in het Engels geschreven. We zullen cryptografische begrippen enkel vertalen naar het Nederlands als er een geschikte vertaling voor de hand ligt. Begrippen die door vertaling naar het Nederlands zodanig aangetast worden dat de lezer ze in de Engelse literatuur met moeite als hetzelfde begrip zou herkennen, zullen we daarentegen niet vertalen.

Dit hoofdstuk is grotendeels gebaseerd op [35], waarin Bruce Schneier een mooi overzicht geeft van de volledige cryptografie. Voor meer informatie verwijzen we de lezer dan ook naar dat standaardwerk.

4.1 Encryptie en decryptie

Stel dat Alice een boodschap naar Bob wil sturen. Het is mogelijk dat Leo, de luistervink, het kanaal waarover ze die boodschap wil sturen afluistert. Alice wil niet dat Leo de boodschap die voor Bob bestemd is kan lezen. Ze spreekt daarom op voorhand met Bob een geheime sleutel k af. Als ze een boodschap naar Bob wil zenden, gebruikt ze een *symmetrisch*



Figuur 4.1: Schematische voorstelling van symmetrische encryptie

encryptie-algoritme om de boodschap met de geheime sleutel k te vercijferen. De leesbare boodschap noemen we de *klaartekst* (Eng: plaintext), de geëncrypteerde boodschap noemen we de *cijfertekst* (Eng: ciphertext). Voor Leo, die de sleutel k niet kent, lijkt de cijfertekst maar een hoop willekeurige data zonder verdere betekenis. Bob kan echter met behulp van de sleutel k de cijfertekst terug decrypteren tot de oorspronkelijke klaartekst. Dit principe wordt verduidelijkt in figuur 4.1.

Er zijn verscheidene algoritmes voor symmetrische encryptie ontwikkeld. Het meest gekende algoritme is ongetwijfeld DES, dat jarenlang de standaard voor data-encryptie geweest is. DES behandelt gegevens in blokken van 64 bits met een sleutel van 56 bits. Door deze relatief kleine sleutellengte wordt DES tegenwoordig niet meer als volledig veilig beschouwd. Triple-DES, het drievoudig toepassen van DES met drie onafhankelijke sleutels, is een goed alternatief. Ook andere algoritmes zoals IDEA en AES (Advanced Encryption Standard, die deze zomer uit een aantal kandidaat-algoritmes zal gekozen worden), zijn voldoende sterk om de rekenkracht van vandaag en morgen te trotseren. Een sleutellengte van 64 bits kan voldoende zijn voor minder gevoelige data, voor echt sterke veiligheid is een sleutellengte van 128 bits aangewezen.

Bijna alle symmetrische encryptie-algoritmes gaan ervan uit dat Leo over misschien wel grote maar toch beperkte rekenkracht beschikt. Dit betekent dat deze algoritmes wel cryptografisch veilig zijn, maar informatietheoretisch gezien niet. Er is één encryptiesysteem dat wel informatietheoretisch veilig is. Het heet een *one-time pad*, en is eigenlijk niets meer dan een oneindig lange reeks van willekeurige bits die Alice en Bob op voorhand hebben afgesproken, maar die Leo niet kent. Alice encrypteert haar boodschap door elke bit van de klaartekst met de volgende bit van het one-time pad te XOR'en. Ze gebruikt elke bit van het one-time pad exact één keer, voor slechts één boodschap. Bob decrypteert de boodschap door de bits opnieuw te XOR'en met het one-time pad. Het is belangrijk dat het one-time pad volledig willekeurige bits bevat en dat elke bit van het one-time pad slechts één keer gebruikt wordt. Dat dit encryptiesysteem informatietheoretisch veilig is, kunnen we als volgt inzien. Stel dat Alice en Bob op voorhand het one-time pad 100101 hebben afgesproken en dat Alice de boodschap $M = 000111$

naar Bob wil sturen. Ze berekent de XOR met het one-time pad en stuurt $C = 100101 \oplus 000111 = 100010$ naar Bob. Leo kan uit C geen enkele informatie halen over M, want hij kent het one-time pad niet. Voor hem kan de sleutelsequentie net even goed 001010 zijn, wat zou betekenen dat de boodschap $C \oplus 001010 = 101000$ is. Voor elke mogelijke boodschap is er een one-time pad dat exact dezelfde cijfertekst C had opgeleverd.

Het probleem van symmetrische algoritmes is dat Alice en Bob over een veilig kanaal moeten beschikken om de sleutel uit te wisselen. In 1976 hebben Diffie en Hellman dat paradigma voorgoed verbroken. Zij stelden voor een sleutelpaar te gebruiken in plaats van één enkele sleutel, waarvan één sleutel publiek is en de andere privaat. Een boodschap die geëncrypteerd is met de publieke sleutel kan enkel met de private sleutel gedecrypteerd worden. Er bestaat natuurlijk een wiskundig verband tussen beide sleutels, maar het afleiden van de private sleutel uit de publieke sleutel is een moeilijk probleem. Het is nog best te vergelijken met een gesloten brievenbus: iedereen kan brieven in de brievenbus steken, maar enkel de persoon met de sleutel kan de post uit de brievenbus halen.

Als Alice een boodschap naar Bob wil sturen vraagt ze Bob om zijn publieke sleutel. Bob stuurt zijn publieke sleutel naar Alice, Alice encrypteert haar boodschap met Bobs publieke sleutel en ze stuurt het resultaat naar Bob. Ook al heeft Leo de publieke sleutel van Bob afgeluisterd, toch kan hij de boodschap niet decrypteren want daarvoor heeft hij Bobs private sleutel nodig. Bob is de enige die die private sleutel kent en is dus ook de enige die de boodschap kan decrypteren.

Het meest bekende publieke-sleutel algoritme is RSA, gebaseerd op de moeilijkheid van het factoriseren van grote getallen. Een goede eigenschap van dit algoritme is dat de gebruiker zelf de grootte van de sleutel — en dus de kracht van het algoritme — kan bepalen. Iemand die bereid is honderd miljoen frank neer te tellen kan een machine kopen die in staat is een 512-bit getal binnen enkele maanden te factoriseren. Schneier raadt dan ook sleutels van 1024 bits aan voor gewoon gebruik, tot zelfs 2048 bits voor uiterst gevoelige gegevens.

4.2 Probabilistische encryptie

4.2.1 Algemeen

Het idee van publieke-sleutel cryptografie is elegant, maar kan ons in sommige situaties volledig in de steek laten. Stel dat Leo weet dat de boodschap M die Alice naar Bob stuurt ofwel M_1 ofwel M_2 is, bijvoorbeeld een bevestiging of een ontkenning. Leo kan inderdaad de cijfertekst C die hij heeft afgeluisterd niet decrypteren, want hij heeft Bobs private sleutel niet. Leo heeft wel Bobs publieke sleutel waarmee hij M_1 en M_2 kan encrypteren tot C_1 en C_2. Hij kan C vergelijken met C_1 en C_2 en zo beslissen of $M = M_1$

of $M = M_2$.

In het algemeen kunnen we stellen dat publieke-sleutel cryptografie niet mag gebruikt worden als het aantal verschillende mogelijkheden voor de boodschap vrij beperkt is. Het idee van publieke-sleutel cryptografie impliceert immers dat Leo zelf ook kan encrypteren en dus de doorgezonden cijfertekst kan vergelijken met de encrypties van de verschillende mogelijkheden voor de boodschap.

Het antwoord op deze problemen is *probabilistische encryptie*, oorspronkelijk bedacht door Goldwasser en Micali ([24]). Zij maken het encryptie-algoritme probabilistisch in plaats van deterministisch, wat wil zeggen dat bij het encrypteren het toeval ook een rol speelt. Dit is enkel mogelijk indien een groot aantal cijferteksten kunnen overeenkomen met dezelfde klaartekst, en de precieze cijfertekst bij encryptie willekeurig wordt gekozen. Voor één en dezelfde boodschap M hebben we dus[1]

$$C_1 = E_k(M), C_2 = E_k(M), C_3 = E_k(M), \ldots, C_n = E_k(M)$$
$$M = D_k(C_1) = D_k(C_2) = D_k(C_3) = \ldots = D_k(C_n)$$

Stel dat Leo de cijfertekst $C_i = E_k(M)$ heeft afgeluisterd. Zelfs als hij M juist raadt, kan hij dit niet controleren: wanneer hij zelf M encrypteert bekomt hij een totaal andere cijfertekst C_j. Hij kan C_i en C_j niet vergelijken en kan dus niet zeker weten of hij de boodschap juist geraden heeft. Om het met de woorden van Bruce Schneier te zeggen: "This is amazingly cool stuff". Zelfs als Leo de publieke sleutel, de boodschap en de cijfertekst heeft, kan hij nog steeds niet bewijzen dat de cijfertekst de encryptie is van de klaartekst.

De cijfertekst van een probabilistische encryptie is natuurlijk altijd groter dan de klaartekst. Hier kunnen we niet omheen, het is een logisch gevolg van het feit dat veel cijferteksten overeenkomen met één en dezelfde klaartekst.

Aan welke vereisten moet een probabilistisch encryptiesysteem voldoen om veilig te zijn? Volgens Goldwasser en Micali moet een degelijk probabilistisch encryptiesysteem voldoen aan de eigenschap dat al wat efficiënt berekenbaar is over de klaartekst gegeven de cijfertekst, ook efficiënt berekenbaar is zonder de cijfertekst. Ze hebben deze eigenschap *semantische veiligheid* gedoopt. Formeler is een probabilistische encryptiefunctie E semantisch veilig als er geen polynomiale-tijd algoritme bestaat dat, gegeven de publieke sleutel k, twee klaarteksten M_1 en M_2 en een cijfertekst C, in staat is te beslissen of $C = E_k(M_1)$ of $C = E_k(M_2)$ met een kans op juistheid niet-verwaarloosbaar groter dan willekeurig raden.

[1] We zullen doorheen deze tekst de encryptie van een boodschap M met een sleutel k consequent noteren als $E_k(M)$. De decryptie van een cijfertekst C met sleutel k noteren we als $D_k(C)$.

4.2.2 Probabilistische encryptie met behulp van kwadratische residu's

De concrete implementatie die Goldwasser en Micali in [24] geven voor een probabilistisch encryptiesysteem wordt door Schneier afgedaan als onpraktisch omdat de cijfertekst ervan wel erg veel groter is dan de klaartekst. Ze heeft echter nog enkele bijkomende voordelen die het bijzonder bruikbaar maken voor Secure Distributed Computing, vandaar dat we haar hier toch bespreken.

Het systeem berust op de Kwadratische Residuositeit Veronderstelling (KRV) zoals aangebracht in sectie 3.3. Herinner daaruit dat het beslissen of er voor een gegeven $a \in Z_n^*[+1]$ een x bestaat zodat $a \equiv x^2 \mod n$, met $n = pq$ met p en q twee grote priemgetallen congruent met 3 modulo 4, als een moeilijk probleem wordt beschouwd indien de factorisatie van n niet gekend is. We encrypteren de klaartekst bit per bit, dus er zijn eigenlijk maar twee mogelijke klaarteksten: een 0 en een 1. We definiëren de probabilistische encryptie van een bit b als[2]

$$E_n(b) = (-1)^b \cdot r^2 \mod n \quad \text{met } r \in_R Z_n^*[+1]$$

Omdat -1 altijd een niet-kwadraat is modulo n, is de decryptiefunctie

$$D_n(x) = \begin{cases} 0 & \text{als } x \text{ een kwadraat is modulo } n \\ 1 & \text{als } x \text{ een niet-kwadraat is modulo } n \end{cases}$$

De publieke sleutel van dit encryptiesysteem is n, de private sleutel bestaat uit de factoren p en q. Iemand die n kent kan een willekeurig getal $r \in_R Z_n^*[+1]$ kiezen (herinner dat het Jacobi-symbool ook efficiënt te berekenen is als de factorisatie van n niet gekend is) en een encryptie van een bit b berekenen. Het feit dat er bij de encryptie een getal willekeurig getal moet gekozen worden zorgt voor het probabilistische aspect van dit encryptiesysteem. Vanwege de KRV is het decrypteren echter een moeilijk probleem, tenzij de factoren p en q gekend zijn. In $Z_n^*[+1]$ zijn bovendien exact de helft van de elementen kwadratische residu's en de andere helft kwadratische nonresidu's, zodat Leo ook met willekeurig gokken niet verder komt dan een kans op juistheid van één op twee.

4.2.3 Het ElGamal algoritme

Sinds de uitvinding van het concept van publieke-sleutel cryptografie, zijn er verschillende publieke-sleutel algoritmes voorgesteld. Veel ervan zijn achteraf onveilig gebleken, andere zijn onpraktisch omdat de sleutel te groot is of omdat de cijfertekst veel groter is dan de klaartekst. Het meest bekende algoritme dat tot nog toe alle beproevingen heeft overleefd is RSA,

[2] Met de notatie "$\in_R$" bedoelen we "een willekeurig element van".

gebaseerd op de moeilijkheid van het factoriseren van grote getallen. Het is echter niet het enige: in 1985 heeft Taher ElGamal — die overigens tot voor kort hoofd van RSA Data Security Engineering was — een algoritme voorgesteld gebaseerd op de Discrete Logaritme Veronderstelling.

Om de veiligheid van ElGamal-encryptie beter te begrijpen zullen we eerst een protocol van Diffie en Hellman aanbrengen dat een veilige uitwisseling van sleutels toelaat, eveneens op basis van de DLV.

Protocol 1 (Diffie-Hellman sleuteluitwisseling). *Een protocol dat twee partijen A en B toelaat een geheime sleutel uit te wisselen over een onveilig kanaal.*

1. A en B spreken eerst een priemgetal p en een generator g van $\mathbb{Z}_p^*$ af. Verder kiest A een willekeurig getal k_A en kiest B een willekeurig getal k_B.
2. $A \longrightarrow B$: $g^{k_A} \mod p$
3. $B \longrightarrow A$: $g^{k_B} \mod p$
4. De geheime sleutel wordt gegeven door $g^{k_A k_B} \mod p$. A kan dit berekenen als $(g^{k_B})^{k_A} = g^{k_A k_B}$ en B kan dit berekenen als $(g^{k_A})^{k_B} = g^{k_A k_B}$.

Iemand die het kanaal aftapt ziet enkel g^{k_A} en g^{k_B} voorbijkomen. Het bepalen van $g^{k_A k_B}$ als g^{k_A} en g^{k_B} gegeven zijn staat bekend als het *Diffie-Hellman probleem.* Als de luistervink discrete logaritmen zou kunnen berekenen, kan hij k_A en k_B berekenen en kan hij de geheime sleutel $g^{k_A k_B}$ reconstrueren. Een algoritme voor het discrete logaritme probleem zou dus in één klap ook het Diffie-Hellman probleem oplossen. Volgens de DLV is deze manier van werken onmogelijk, maar het is niet ondenkbaar dat er een manier bestaat om het Diffie-Hellman probleem op te lossen zonder onderweg discrete logaritmen te moeten berekenen. Tot nog toe is er echter geen polynomiaal algoritme gekend dat hiertoe in staat is. Men veronderstelt dat het Diffie-Hellman probleem, net als discrete logaritmen, een moeilijk probleem is waarvoor geen polynomiaal algoritme bestaat.

Het ElGamal publieke-sleutel algoritme wordt meestal als volgt gedefinieerd. We kiezen een priemgetal p en een getal $g \in Z_p^*$, dat niet noodzakelijk een generator is maar wel van "grote" orde moet zijn. We kiezen ook nog een private sleutel $x \in_R Z_{p-1}$ en we berekenen $y = g^x \mod p$. Het triplet $[p, g, y]$ vormt de publieke sleutel en x is de private sleutel. De encryptie van een boodschap M ziet eruit als

$$E(M) = [g^r \mod p,\ M \cdot (g^r)^x \mod p]$$

waarbij r een willekeurig getal is tussen 1 en $p-1$. De decryptie kan enkel gebeuren met behulp van de private sleutel x:

$$D([\alpha, \beta]) = \beta \cdot \alpha^{-x} \mod p$$

De veiligheid van ElGamal encryptie lijkt gebaseerd op discrete logaritmen, maar komt eigenlijk beter overeen met het Diffie-Hellman probleem. Inderdaad, als we in staat zijn uit g^r en g^x het getal g^{rx} te berekenen kunnen we ElGamal kraken. Het Diffie-Hellman probleem is dus minstens zo moeilijk als ElGamal kraken, maar of het omgekeerde ook geldt is tot nog toe niet geweten.

Als we enkele extra voorwaarden opleggen aan p en g bekomen we een versie van ElGamal waarover meer bekend is wat de veiligheid ervan betreft. Als p een priemgetal is en bovendien voldoet aan $p = aq + 1$ waarbij q eveneens een priemgetal is en a een klein even[3] getal, dan bestaat er een deelgroep $\mathbb{G}_q$ van $\mathbb{Z}_p^*$ van orde q. We kiezen een generator g van die deelgroep $\mathbb{G}_q$. Encryptie en decryptie werken op dezelfde manier als in de eerste versie, behalve dat r moet gekozen worden tussen 1 en $q-1$ en dat de boodschap M een element moet zijn van G_q. In [36] wordt bewezen dat de semantische veiligheid van dit encryptiesysteem equivalent is met het Diffie-Hellman beslissingsprobleem. Het Diffie-Hellman beslissingsprobleem is niet helemaal hetzelfde als het oorspronkelijke Diffie-Hellman probleem. Stel dat we een triplet elementen $[\alpha = g^r, \beta = g^s, \gamma]$ kennen (maar niet de discrete logaritmen r en s zelf), dan bestaat het Diffie-Hellman beslissingsprobleem erin te bepalen of $\gamma = g^{rs}$ of niet. Er wordt aangenomen dat er geen polynomiaal algoritme bestaat dat deze beslissing met meer kans op correctheid kan nemen dan willekeurig gokken. Dankzij zijn bewijsbare semantische veiligheid is deze versie van ElGamal geschikt voor probabilistische encryptie.

4.3 Verdelen van een geheim

Ergens in een Amerikaanse woestijn staan nucleaire raketten op Moskou gericht. Alleen de president kent de geheime code die het lanceermechanisme in gang kan zetten. Daarmee vormt de president een "single point of failure" voor het nucleaire defensieprogramma van de Verenigde Staten. De president beseft dit en besluit de top van het Amerikaanse leger, die uit vijf officieren bestaat, op de hoogte te brengen van de geheime code. Hij wil echter niet dat één enkele officier in een destructieve bui in staat is Moskou van de kaart te vegen. Hij wil dat er minstens drie officieren nodig zijn om het lanceermechanisme in gang te zetten.

De president wil zijn geheime code op een zodanige manier verdelen over vijf officieren dat er minstens drie officieren nodig zijn om de geheime code te reconstrueren. We veralgemenen dit probleem tot het verdelen van een geheim s over n partijen met een drempel t, zodat elke groep van minstens t partijen s kan reconstrueren, maar elke groep van $t-1$ partijen of minder geen informatie over s kan verkrijgen. Adi Shamir heeft een elegante oplossing voor dit probleem gevonden die we nu kort bespreken.

[3] a kan niet oneven zijn want dan zou p even zijn

We hebben een licht gewijzigde versie van veelterminterpolatie volgens Lagrange nodig. Zij K een eindig veld. Stel dat t punten (x_i, y_i) van het vlak K^2 gegeven zijn. Dan bestaat er juist één veelterm $f(X)$ van graad $t-1$ die door die t punten gaat. Inderdaad, als we stellen dat

$$f_i(X) = \prod_{1 \leq j \leq t, j \neq i} \frac{(X - x_j)}{(x_i - x_j)}$$

voor $i = 1 \ldots t$, dan is $f_i(X)$ van graad $t-1$ en voldoet f_i aan $f_i(x_i) = 1$ terwijl $f_i(x_j) = 0$ als $j \neq i$. Er volgt dan onmiddellijk dat

$$f(X) = \sum_{i=1 \ldots t} y_i \cdot f_i(X)$$

de veelterm is die we zoeken.

Het geheimverdelingsschema van Shamir werkt als volgt. De verdeler heeft een geheim $s \in K$ dat hij wil verdelen over n partijen zodat er minstens t partijen nodig zijn om het geheim te reconstrueren. Hij kiest daartoe een willekeurige veelterm $f(X)$ van graad $t-1$ zodat $f(0) = s$. Concreet betekent dit dat hij s als constante term neemt en de overige $t-1$ coëfficiënten willekeurig kiest uit K. Hij zendt $s_i = f(i)$ naar elke partij $i = 1 \ldots n$.[4] We noemen s_i een *schaduw* van s.

Als een groep van minstens t deelnemers het geheim willen reconstrueren, maken ze hun schaduwen s_i bekend. Ze gebruiken Lagrange interpolatie om de veelterm $f(X)$ op te stellen zodat $f(i) = s_i$ en reconstrueren het geheim door $s = f(0)$ te berekenen.

Als een groep van $t-1$ deelnemers gaat samenzitten om het geheim te reconstrueren, komen ze bedrogen uit. Ze hebben één punt te weinig om de veelterm f eenduidig te kunnen bepalen. Het is zelfs zo dat voor elke $s' \in K$ er een veelterm $f'(X)$ bestaat zodat $f'(0) = s'$ en $f'(X)$ door al hun punten gaat. Geen enkel element van K heeft meer kans het geheim te zijn dan een ander. Het samenleggen van $t-1$ of minder schaduwen levert geen enkele informatie over het geheim op, zodat zelfs onbeperkte rekenkracht hen geen stap vooruit helpt. Shamir's geheimverdelingsschema is daarom zelfs informatietheoretisch veilig.

4.4 Bit commitment

Alice heeft de bijzondere gave de winnaar van een paardenren onfeilbaar te kunnen voorspellen. Zelf heeft ze principiële bezwaren tegen gokken, maar ze is wel bereid haar gave tegen een kleine vergoeding ten dienste te stellen van Bob, die minder scrupules heeft. Hij twijfelt echter aan de oprechtheid

[4] Maak niet de fout de partijen te nummeren van 0 tot $n-1$, want dan krijgt partij 0 het geheim zelf.

van Alice, en vraagt haar haar kunsten te bewijzen door de winnaar van de eerstvolgende paardenren te voorspellen. Alice weigert natuurlijk: "Niks van, jij wil gewoon op dat paard inzetten en een klein fortuin verdienen zonder mij een cent te betalen. Vorige week had ik voorspeld dat Gouden Bliksem ging winnen, en inderdaad, Gouden Bliksem heeft gewonnen". "Hoe kan ik weten dat jij echt Gouden Bliksem voorspeld had? Je kan net zo goed de dag erna even de krant hebben opgengeslagen. Ik zal heus niet inzetten op het paard dat je nu voorspelt, vertrouw me." "Nee, echt, ik had Gouden Bliksem voorspeld, geloof me."

Zo gaan ze er natuurlijk nooit uit geraken. Gelukkig heeft Alice naast haar bijzondere gave ook nog praktisch doorzicht. Ze schrijft haar voorspelling voor de volgende ren op een briefje en steekt het briefje in een verzegelde enveloppe. Alice en Bob gaan samen naar Sterrebeek. Na afloop van de wedstrijd openen ze samen de enveloppe en overtuigt ze Bob van haar gave.

Is dit ook mogelijk vanop afstand? Bestaat er zoiets als een elektronische verzegelde enveloppe? Alice wil zich binden aan een voorspelling, een bit of een reeks bits, maar wil dat Bob pas later haar voorspelling te weten komt. Van zijn kant wil Bob dat Alice intussen niet meer van gedacht kan veranderen. De cryptografie heeft hiervoor inderdaad een oplossing gevonden: bit commitments.

We zullen eerst een eenvoudig bit commitment schema aanbrengen gebaseerd op symmetrische cryptografie. Bob genereert een willekeurige bitstring R en zendt die naar Alice. Alice creëert een boodschap bestaande uit R en de bit b waaraan ze zich wil binden, kiest een willekeurige sleutel k en stuurt de encryptie $C = E_k(R, b)$ naar Bob. Bob kan deze boodschap niet decrypteren want hij kent de sleutel k niet. Op het moment dat Alice haar bit wil bekend maken, zendt ze de sleutel k naar Bob. Bob decrypteert de boodschap om de bit te onthullen. Hij controleert dat het eerste deel van de boodschap inderdaad R is.

Als Bob R niet zou controleren kan Alice vals spelen. Als ze enkele keren probeert C te decrypteren met andere sleutels, vindt ze al snel een sleutel k' zodat $D_{k'}(C) = (R', \overline{b})$. Door Bob de sleutel k' te overhandigen in plaats van k kan ze dan liegen over de inhoud van haar bit commitment. Bobs string R verhindert haar echter op deze manier vals te spelen, want nu moet ze een sleutel vinden die niet alleen een andere waarde voor haar bit oplevert maar bovendien ook R als eerste deel van de boodschap geeft. Als ze een degelijk encryptie-algoritme gebruiken, is de kans van Alice om zo'n sleutel te vinden miniem.

De probleemstelling van bit commitments doet een beetje denken aan probabilistische encryptie. En terecht: Alice kan een probabilistische encryptie van haar bit b als bit commitment gebruiken en naar Bob zenden. Om het bit commitment te openen verklapt ze haar private sleutel aan

Bob zodat die het bit commitment kan decrypteren. Een private sleutel verklappen ... dit moet koude rillingen veroorzaken over de rug van elke cryptograaf. Hoewel dit geen onoverkomelijk probleem vormt — Alice kan voor elk bit commitment een nieuw sleutelpaar genereren — hadden we het toch graag anders gezien.

Voor het probabilistisch encryptiesysteem gebaseerd op kwadratische residu's dat we in sectie 4.2 besproken hebben, bestaat er een andere manier om Bob van de inhoud van het bit commitment te overtuigen. Alice construeert een bit commitment voor een bit b als de probabilistische encryptie

$$E_n(b) = (-1)^b \cdot r^2 \mod n$$

waarbij $r \in_R Z_n^*[+1]$ en $n = pq$ met p en q priemgetallen congruent met 3 modulo 4. Een bit commitment voor een 0 is een willekeurig kwadratisch residu modulo n, een bit commitment voor een 1 is een willekeurig kwadratisch nonresidu modulo n. De private sleutel is de factorisatie van n. Alice kan Bob overtuigen van de inhoud van het bit commitment zonder de factorisatie van n te moeten verklappen. Als a een bit commitment is voor een 0, is de eenvoudigste manier om Bob hiervan te overtuigen hem een wortel x van a te zenden zodat $a \equiv x^2 \mod n$. Als a een bit commitment is voor een 1, dan is a een kwadratisch nonresidu en dit kan Alice weer bewijzen door Bob een wortel x van $-a$ te overhandigen. Hoewel Bob uit twee wortels die niet op het teken na gelijk zijn de factorisatie van n gemakkelijk kan berekenen, helpt de kennis van één enkele wortel hem daarbij geen stap verder.

Dit bit commitment schema heeft enkele opmerkelijke eigenschappen die we later zullen benutten. Ten eerste is dit schema *vergelijkbaar*. Dit wil zeggen dat Alice van een commitment a voor een bit b en een commitment a' voor een bit b', aan Bob kan bewijzen dat a en a' commitments zijn voor dezelfde of net verschillende bits, zonder daarbij a of a' te openen. Formeler kan Alice Bob de waarde $b \oplus b'$ aantonen zonder enige verdere informatie over b en b' te lekken. Inderdaad, Alice moet daartoe gewoon een wortel van $(-1)^{b \oplus b'} \cdot a \cdot a' \mod n$ bekendmaken.

Ten tweede is dit schema *blindeerbaar*. De voorwaarden voor deze eigenschap zijn iets ingewikkelder. Stel dat Alice een bit commitment a voor een bit b maakt, dan moet Bob in staat zijn hieruit een bit commitment a' voor $b \oplus b'$ te construeren zodat Alice geen verband meer kan zien tussen a en a'. Alice moet echter in staat zijn a' te openen alsof het één van haar eigen commitments was. Bovendien moet Bob aan de hand van a en a' de waarde van b' kunnen bewijzen. Sterker nog, Bob moet aan de hand van a' en een tweede geblindeerd commitment a'' voor een bit $b \oplus b''$, Alice kunnen overtuigen van de waarde van $(b \oplus b') \oplus (b \oplus b'') = b' \oplus b''$.

Ons schema voldoet aan al deze vereisten. Bob blindeert een bit com-

mitment a met een bit b' door

$$a' = (-1)^{b'} \cdot s^2 \cdot a \mod n$$

te berekenen, met s een willekeurig getal van $Z_n^*[+1]$. We noemen s de *blindeerfactor*. Als $a = (-1)^b \cdot r^2 \mod n$, dan kan Bob de waarde van b' aantonen door een wortel op te geven van $(-1)^{b'} \cdot a' \cdot a^{-1}$. Bob kent hier inderdaad een wortel van, want

$$(-1)^{b'} \cdot a' \cdot a^{-1} = (-1)^{b'} \cdot \left((-1)^{b \oplus b'} \cdot (rs)^2\right) \cdot \left((-1)^b \cdot r^{-2}\right) = s^2 \mod n$$

en bijgevolg is s een wortel. Op dezelfde manier kan Bob vanuit twee blinderingen a' en a'' Alice overtuigen van de waarde van $b' \oplus b''$ door een wortel modulo n op te geven van

$$(-1)^{b' \oplus b''} \cdot a' \cdot (a'')^{-1} \mod n$$

Alle geblindeerde bit commitments blijven net als vroeger kwadraten als de bit die erin verborgen zit gelijk is aan 0 en niet-kwadraten als de bit gelijk is aan 1. Alice kan dus nog steeds haar kennis van de factorisatie van n gebruiken om elk bit commitment te openen alsof het het hare was.

Stel dat Alice en Bob samen op een eerlijke manier kop-of-munt willen spelen van op afstand. Dit is hetzelfde als samen een willekeurige bit kiezen. Alice wil niet dat Bob de bit op zijn eentje kiest en omgekeerd. Het is geen oplossing Alice een willekeurige bit naar Bob te laten sturen, Bob een willekeurige bit naar Alice te laten sturen en de XOR van beide bits als willekeurige bit te laten doorgaan, want Bob kan zijn keuze laten afhangen van de bit die hij van Alice krijgt en zo nog steeds de waarde van de resulterende bit zelf bepalen. Dit probleem is in de literatuur gekend als een *fair coin flip*. Bit commitments bieden een oplossing. Alice kiest een willekeurige bit b en stuurt een commitment voor b naar Bob. Bob kiest op zijn beurt een willekeurige bit b' en stuurt hier eveneens een commitment voor naar Alice. Beide openen hun commitments en nemen $b \oplus b'$ als willekeurige bit. Dit protocol is eenvoudig uit te breiden tot meerdere partijen.

Een bit commitment kan op twee verschillende manieren gerealiseerd worden. De eerste is dat Alice data doorgeeft aan Bob waar heel diep wel informatie over b in zit, maar Bob kan die informatie er niet in polynomiale tijd uithalen. Het bit commitment schema dat gebruikt maakt van kwadratische residu's is hier een voorbeeld van. De tweede manier is dat Alice data doorgeeft aan Bob die net zo goed een commitment zou kunnen zijn voor $\overline{b}$, maar dat Alice de gegevens die ze bij opening van het bit commitment aan Bob moet overhandigen om te liegen over de inhoud van haar commitment niet in polynomiale tijd kan berekenen. Het bit commitment schema gebaseerd op symmetrische encryptie is hier een voorbeeld van. In

het eerste geval gaan we ervan uit dat Bobs rekenkracht beperkt is, in het tweede geval moet die van Alice beperkt zijn. Het is dus mogelijk een bit commitment uit te voeren tussen een partij met beperkte rekenkracht en één met onbeperkte rekenkracht. Een bit commitment voor twee partijen met onbeperkte rekenkracht is echter onmogelijk.

4.5 Oblivious transfer

Dit cryptografisch principe heeft niet enkel een onvertaalbare naam ("vergeetachtige overdracht" houdt echt geen steek), maar is tevens ongetwijfeld het meest surrealistische principe dat we in dit hoofdstuk aanbrengen. Er bestaan verschillende versies van oblivious transfer (afgekort OT). Oorspronkelijk is het door Michael Rabin voorgesteld als volgt. In het begin van het protocol heeft Alice een bit b, en aan het einde van het protocol heeft Bob met een kans van één op twee b ontvangen en met een kans van één op twee helemaal niets ontvangen. Alice heeft echter geen idee of Bob b ontvangen heeft of niet. Deze versie wordt ook *Rabin-OT* genoemd.

Even, Goldreich en Lempel hebben later de oblivious transfer op een andere manier gedefinieerd ([15]). In hun definitie heeft Alice twee bits b_0 en b_1 en heeft Bob een selectiebit s. Aan het einde van het protocol heeft Bob b_s ontvangen, maar hij heeft geen idee van $b_{\overline{s}}$. Alice van haar kant heeft geen idee welke bit Bob nu precies gekozen heeft. Deze versie is bekend onder de naam *chosen one-out-of-two oblivious transfer*, dikwijls kort genoteerd als $\binom{2}{1}$-OT. Crépeau heeft aangetoond dat deze definitie en die van Rabin equivalent zijn, d.w.z. dat een $\binom{2}{1}$-OT kan geïmplementeerd worden met als enig hulpmiddel de Rabin-OT en vice versa.

Nog een andere versie is *oblivious bitstring transfer*, kortweg $\binom{2}{1}$-OT^k. Hier heeft Alice in plaats van twee bits b_0 en b_1 twee bitstrings M_0 en M_1 van lengte k. Bob heeft opnieuw een selectiebit s en ontvangt M_s, maar kan niets te weten komen over $M_{\overline{s}}$. Ook in deze versie heeft Alice het raden naar s.

We geven een concrete implementatie van een oblivious bitstring transfer, gebaseerd op discrete logaritmen ([26]). Alice kiest een groot priemgetal p, een generator g van $\mathbb{Z}_p^*$ en een $c \in_R Z_p^*$. Bob kiest een willekeurige $x \in Z_{p-1}$ en berekent

$$\beta_s = g^x \mod p \qquad \text{en} \qquad \beta_{\overline{s}} = c \cdot \beta_s^{-1} \mod p$$

Bobs publieke sleutel is β_0, β_1 en zijn private sleutel is x. Bob kent enkel het discrete logaritme van β_s, niet dat van $\beta_{\overline{s}}$ want hij kent het discrete logaritme van c niet. Alice kan controleren dat Bob zijn publieke sleutel eerlijk heeft opgesteld door na te gaan of $\beta_0 \cdot \beta_1 \equiv c \mod p$. Ze kan hieruit echter niet afleiden van welk van de twee Bob het discrete logaritme kent.

Alice kiest willekeurige getallen y_0 en y_1 tussen 0 en $p-2$, berekent voor $i = 0, 1$

$$\begin{aligned} a_i &= g^{y_i} \mod p \\ \gamma_i &= \beta_i^{y_i} \mod p \\ r_i &= M_i \oplus \gamma_i \end{aligned}$$

en zendt a_0, a_1, r_0 en r_1 naar Bob. Gebruik makende van zijn private sleutel x berekent Bob $a_s'^{x} = g^{x \cdot y_s} = \beta_s^{y_s} = \gamma_s$. Hij kan de bitstring van zijn keuze berekenen als $M_s = \gamma_s \oplus r_s$, maar omdat hij het discrete logaritme van $\beta_{\overline{s}}$ niet kent, kan hij $\gamma_{\overline{s}}$ niet berekenen en kan hij dus ook $M_{\overline{s}}$ niet achterhalen.

4.6 Zero-knowledge proofs

Peggy[5] kent een bepaald geheim maar Victor gelooft niet dat zij dat geheim kent. De eenvoudigste manier om Victor ervan te overtuigen dat zij het geheim wel degelijk kent is hem even terzijde te nemen en hem het geheim in het oor te fluisteren. Jammer genoeg houdt het geheim daar op met een geheim te zijn, want nu kent Victor het geheim ook en kan hij het gaan doorvertellen aan al wie het horen wil. Wat Peggy nodig heeft is een zogenaamd *zero-knowledge proof*. Dit is een protocol dat Peggy toelaat aan Victor te bewijzen dat ze bepaalde kennis bezit zonder hem enige informatie over die kennis door te geven en zonder dat Victor na het bewijs zich bij een derde persoon kan voordoen alsof hij die kennis zelf ook bezit.

Zero-knowledge proofs zijn meestal interactieve protocols, alhoewel er ook niet-interactieve zero-knowledge proofs bestaan. Victor stelt Peggy een aantal vragen die ze allemaal kan beantwoorden als ze het geheim inderdaad kent. Kent ze het niet, heeft ze een bepaalde kans kleiner dan één om toch juist te antwoorden. Na een aantal vragen is Victor ervan overtuigd dat Peggy het geheim kent. De vragen zijn echter zodanig dat de antwoorden erop hem geen informatie over het geheim geven, enkel over het feit dat Alice het geheim kent.

We zullen dit even illustreren met een voorbeeld. Stel dat Peggy het isomorfisme tussen twee grote grafen G_1 en G_2 kent, maar Victor dit niet zomaar wil geloven. Het vinden van het isomorfisme tussen twee grote grafen is een moeilijk probleem, dus Bob kan het isomorfisme niet op eigen houtje berekenen. Peggy past een willekeurige permutatie toe op G_1 zodat ze een nieuwe isomorfe grafe H bekomt. Ze stuurt H naar Victor. Victor vraagt Peggy ofwel hem het isomorfisme tussen G_1 en H te geven, ofwel hem het isomorfisme tussen G_2 en H te geven. Peggy kent het isomorfisme tussen G_1 en H (want dat heeft ze zelf gekozen). Als ze bovendien het isomorfisme

[5] Voor een keer spelen Alice en Bob niet de hoofdrollen: Peggy is de "prover", Victor is de "verifier"

tussen G_1 en G_2 kent, kan ze gemakkelijk ook het isomorfisme tussen G_2 en H berekenen. Ze geeft Victor het isomorfisme dat hij vroeg, maar houdt het andere isomorfisme geheim. Als Victor zowel het isomorfisme tussen G_1 en H als dat tussen G_2 en H in handen krijgt, kan hij immers gemakkelijk zelf het isomorfisme tussen G_1 en G_2 berekenen.

Als Peggy het isomorfisme tussen G_1 en G_2 eigenlijk niet kent, kan ze H isomorf construeren met ofwel G_1 ofwel G_2, maar niet met beide tegelijk. Ze heeft dan één kans op twee dat Victor net dat isomorfisme vraagt dat zij kent. Met zo'n grote kans om bij de neus genomen te worden zal Victor zeker geen genoegen nemen. Daarom herhalen ze dit protocol enkele malen met telkens een andere grafe H. Na n herhalingen heeft Peggy nog maar een kans van $\frac{1}{2^n}$ om ongemerkt vals te spelen, zodat Victor uiteindelijk overtuigd wordt.

Dit protocol gebruikt het zogenaamde *cut-and-choose* principe. Bijna alle zero-knowledge proofs in de cryptografie zijn op dit principe gebaseerd. Algemeen kunnen we een cut-and-choose protocol als volgt beschrijven. Peggy kent de oplossing van een moeilijk probleem en wil aan Victor bewijzen dat ze die oplossing kent. Ze transformeert het oorspronkelijk probleem naar een willekeurig isomorf moeilijk probleem. Uit de oplossing van het eerste probleem en het isomorfisme kan ze ook het isomorfe probleem oplossen. Ze zendt het isomorfe probleem naar Victor. Victor daagt haar uit

- ofwel hem te bewijzen dat het nieuwe en het oude probleem isomorf zijn
- ofwel hem de oplossing van het nieuwe probleem te geven.

Peggy antwoordt met de gevraagde informatie. Deze stappen herhalen ze n keer totdat Victor volledig overtuigd is.

Zero-knowledge proofs kunnen zeer handig van pas komen in protocols voor Secure Distributed Computing. Veel van die protocols zijn enkel veilig als de tegenstanders semi-eerlijk zijn. Van zodra de tegenstanders beginnen af te wijken van de regels van het protocol en valse informatie doorsturen komt de veiligheid in het gedrang. Een protocol ontwerpen dat ook die kwaadaardige tegenstanders de baas kan is een veel zwaardere opgave. Het is veel eenvoudiger een protocol te gebruiken dat enkel semi-eerlijke tegenstanders aankan en na elke stap de partijen te laten bewijzen dat ze de regels van het protocol correct gevolgd hebben. Een klassiek bewijs zou hierbij de privacy van hun geheime gegevens schenden. Als ze hun eerlijkheid echter in zero-knowledge bewijzen, blijven hun geheimen bewaard terwijl de andere partijen toch overtuigd zijn van hun eerlijkheid.

Hoofdstuk 5

Een fysische oplossing

Niemi en Renvall stellen in [30] een protocol voor om een willekeurige booleaanse functie te evalueren volgens de regels van Secure Distributed Computing, zij het niet op een computer maar met behulp van speelkaarten. De argumenten die zij aanhalen voor deze aanpak (mensen die meer vertrouwen hebben in kaarten dan in computers, elektriciteit die uitvalt of computers die crashen op kritieke momenten) snijden misschien niet veel hout, maar het protocol is wel zeer origineel.

Het uitgangspunt van het protocol is het *vijfkaartentruukje* (oorspronkelijk bedacht door Bert den Boer) dat twee partijen toelaat om een logische AND van hun inputs te berekenen met behulp van vijf kaarten. Twee van die kaarten zijn rood en drie ervan zijn groen. Kaarten van eenzelfde kleur zijn niet van mekaar te onderscheiden en de rug van elke kaart, of die nu rood is of groen, is identiek. Merk op dat het hier dus niet gaat om het alledaagse boek kaarten dat in elk Vlaams café achter de toog ligt.

Het vijfkaartentruukje gaat als volgt: initieel hebben Alice en Bob elk één rode en één groene kaart. De overblijvende groene kaart ligt omgekeerd op tafel. Alice legt haar twee kaarten hier bovenop in een volgorde bepaald door haar geheime bit a: als $a = 0$ legt ze de groene kaart bovenaan, als $a = 1$ legt ze de rode bovenaan. Nu legt Bob zijn twee kaarten onder dit stapeltje volgens een complementaire regel: de groene kaart onderaan als $b = 0$ en de rode als $b = 1$. Vervolgens mogen Alice en Bob beurtelings het stapeltje *afpakken* — zoals dat heet in kaart-middens — oftewel cyclisch permuteren over een aantal kaarten dat ze willekeurig kiezen maar geheim houden voor de ander. Het resultaat is dat op het stapeltje kaarten een cyclische permutatie is toegepast die ze geen van beide kennen. Tenslotte worden de kaarten omgedraaid. Als de twee rode kaarten naast mekaar liggen (cyclisch gezien) is de uitvoer een 1, anders een 0. Men kan gemakkelijk inzien dat de rode kaarten enkel naast elkaar kunnen liggen als $a = b = 1$ en dat de overige drie mogelijkheden niet van mekaar te onderscheiden zijn.

Het uitbreiden van dit systeem tot willekeurige booleaanse functies is

niet triviaal. Om het redeneren te vereenvoudigen zullen we enkele notaties invoeren. Een rode kaart noteren we als r, een groene als g. We voeren ook de notatie $\overline{g} = r$ en $\overline{r} = g$ in. Een stapel kaarten is dan een woord $\boldsymbol{w}$ over $\{r, g\}$ en we noteren dit als $\boldsymbol{w} = w_1 w_2 \dots w_n$ met $w_i \in \{r, g\}$. Een cyclische permutatie over i kaarten noteren we als $f_i(\boldsymbol{w}) = w_{i+1} \dots w_n w_1 \dots w_i$. Verder definiëren we de functie F die een willekeurige cyclische permutatie van een woord teruggeeft: $F(\boldsymbol{w}) = f_i(\boldsymbol{w})$, $i \in_R \{0, \dots, |\boldsymbol{w}| - 1\}$. We stellen een 0-bit voor door een stapeltje van een groene op een rode kaart ($\boldsymbol{d_0} = gr$) en een 1-bit door een rode op een groene ($\boldsymbol{d_1} = rg$). De decoderingsfunctie δ wordt natuurlijk gegeven door $\delta(\boldsymbol{d_b}) = b$. Een stapeltje van een rode en een groene kaart in de juiste volgorde kan als bit commitment dienst doen door het omgekeerd op tafel te leggen. Men kan het commitment openen door de kaarten om te draaien.

Ook een fysisch protocol is gebaseerd op bepaalde "cryptografische" veronderstellingen. In dit protocol veronderstellen we dat

- twee kaarten van dezelfde kleur niet van mekaar te onderscheiden zijn
- de ruggen van alle kaarten identiek zijn
- een kaart slechts kan omgedraaid worden als alle partijen hieraan deelnemen
- elke speler kan *afpakken* zonder de willekeurige index i te verklappen, zodat F op een veilige manier kan berekend worden

Om een circuit van booleaanse poorten op een veilige manier te kunnen evalueren hebben we voor elke logische poort een protocol nodig dat uit commitments voor de input bit(s) een commitment voor de output bit kan berekenen zonder deze commitments te openen of enige informatie te lekken over de waarde van de input of output bits. We kunnen dan vertrekken vanuit commitments voor de (geheime) invoer van elke partij en poort voor poort doorheen het circuit propageren. Op die manier zullen we uiteindelijk commitments voor de uitvoer construeren en deze openen.

Het berekenen van een commitment voor de logische NOT van een gegeven commitment $\boldsymbol{d_b} = x\overline{x}$ is triviaal: we berekenen gewoon $\boldsymbol{d_{\overline{b}}} = f_1(\boldsymbol{d_b}) = \overline{x}x$. De veilige evaluatie van een AND-poort is echter minder evident. Het vijfkaartentruukje voldoet niet meer omdat de uitvoer ervan geen commitment van de vorm $x\overline{x}$ is. Bij elke AND-poort een commitment aanmaken voor de output bit is natuurlijk geen goede oplossing omdat zo de tussenresultaten uitlekken. Het volgende protocol is gebaseerd op het vijfkaartentruukje maar is gewijzigd om aan de bijkomende eisen te voldoen. Het veronderstelt echter een protocol dat toelaat een commitment $\boldsymbol{\alpha} = x\overline{x}$ te kopiëren zonder het te openen; we komen hierop terug in protocol 3.

Protocol 2. *Een protocol om vertrekkende vanuit twee commitments* $\boldsymbol{\alpha} = x\overline{x}$ *en* $\boldsymbol{\beta} = y\overline{y}$ *een commitment te berekenen voor* $\delta(\boldsymbol{\alpha}) \wedge \delta(\boldsymbol{\beta})$.

1. De partijen creëren $\boldsymbol{w} = \boldsymbol{\alpha}^2 = x\overline{x}x\overline{x}$ en $\boldsymbol{w'} = \boldsymbol{\beta}^2 = y\overline{y}y\overline{y}$ met behulp van protocol 3 en combineren deze stapels tot

$$\boldsymbol{u} = w_1 w_2 g w_2' w_1' w_3 w_4 g w_4' w_3' = (x\overline{x}g\overline{y}y)^2 = \boldsymbol{v}^2$$

2. Alle partijen pakken om beurten de stapel af zodat $\boldsymbol{u}$ omgevormd wordt tot $\boldsymbol{u'} = F(\boldsymbol{u})$. De bovenste kaart u_1' wordt omgedraaid. Als deze kaart groen is wordt ze teruggelegd op de stapel en hernemen we deze stap. Pas als de bovenste kaart rood is wordt er verder gegaan met de volgende stap.

3. We nemen de twee volgende kaarten $u_2'u_3'$ van de stapel en draaien ze pas om nadat ze nog eens door iedereen gepermuteerd zijn.

4. Als deze kaarten beide groen zijn nemen we $u_{10}'u_9'$ als commitment voor de uitvoer. Als het één rode en één groene kaart blijken te zijn nemen we $u_7'u_8'$.

We zullen eerst de correctheid van het protocol aantonen. De stapel $\boldsymbol{u}$ bestaat eigenlijk uit twee identieke stapels $\boldsymbol{v} = x\overline{x}g\overline{y}y$. Zo een stapel $\boldsymbol{v}$ is van exact dezelfde structuur als de stapel in het vijfkaartentruukje. Een cyclische permutatie $\boldsymbol{u'}$ van $\boldsymbol{u}$ bestaat dan op haar beurt uit twee identieke stapels $\boldsymbol{v'}$ die een cyclische permutatie zijn van $\boldsymbol{v}$. Dus gelden dezelfde regels als in het vijfkaartentruukje: de rode kaarten (cyclisch) na mekaar betekent een 1, alle andere gevallen betekenen een 0 als uitvoer. Na stap 2 zijn we zeker dat de bovenste kaart rood is ($v_1' = r$). We hebben dan nog maar vier mogelijke gevallen: $\boldsymbol{v'} = rrggg$, $\boldsymbol{v'} = rgrgg$, $\boldsymbol{v'} = rggrg$ of $\boldsymbol{v'} = rgggr$. Hiervan stellen het eerste en het laatste geval een uitkomst 1 voor, het tweede en het derde geval staan voor een 0. In stap 3 worden de tweede en derde kaart ook omgekeerd. Als deze allebei groen zijn zitten we in het derde of het vierde geval. De uitvoer is dan $u_{10}'u_9'$, hetgeen gelijk is aan $v_5'v_4'$. En inderdaad, deze kaarten vormen een commitment voor de juiste bit: gr voor een 0 in het derde geval, rg voor een 1 in het vierde. Als de tweede en derde kaart uit één groene en één rode blijken te bestaan[1] zitten we in ofwel het eerste ofwel het tweede geval. De uitvoer van het protocol is dan $u_7'u_8'$, hetgeen gelijk is aan $v_2'v_3'$. En opnieuw zien we dat deze kaarten het juiste commitment vormen: rg voor een 1 in het eerste geval, gr voor een 0 in het tweede.

We zullen nu nagaan op welke manier het protocol de geheimhouding van de input en output bits verzekert. De veiligheid van de input bits kan men gemakkelijk inzien door de analogie met het vijfkaartentruukje: er worden hier zelfs *minder* kaarten omgedraaid (slechts de eerste drie in plaats van alle vijf) zodat er onmogelijk meer informatie kan lekken over de invoer.

[1] Ze kunnen niet allebei rood zijn omdat v_1' al rood was en er slechts twee rode kaarten zijn in elk groepje van vijf.

De veiligheid van de output bit ligt iets delicater. Na stap 2 weten we dat de eerste kaart rood is en dat de stapel gelijk is aan één van de vier hoger vermelde gevallen, maar we hebben geen idee aan welk van die vier gevallen precies. Als in stap 3 de twee omgedraaide kaarten allebei groen blijken worden de mogelijkheden beperkt tot het derde en het vierde geval. Daar echter in het derde geval de output bit een 0 is en in het vierde geval een 1 kunnen we hier nog steeds geen enkele informatie over de waarde van de output bit uit halen: beide waarden blijven even waarschijnlijk. De kaarten die de output vormen zijn $u'_{10}u'_9 = u'_5u'_4$ waarover geen informatie gekend is omdat ze niet omgedraaid zijn sinds de laatste permutatie. Als de in stap 3 omgedraaide kaarten één rode en één groene kaart zijn, kunnen we in het eerste of het tweede geval zitten. Het eerste geval zorgt voor een output bit 1 en het tweede geval voor een 0, dus hier kunnen we opnieuw niets uit leren. Het commitment voor de output bit wordt gevormd door de kaarten $u'_7u'_8$, maar zoals eerder aangetoond zijn deze dezelfde als $u'_2u'_3$ en die kaarten hebben we net onthuld! Een lek in het protocol? Nee, alvorens die twee kaarten om te keren hebben we een willekeurige permutatie toegepast zodat we de oorspronkelijke volgorde niet kennen. Zowel het commitment voor een 0 als dat voor een 1 bestaan uit één rode en één groene kaart, maar het is net de volgorde van de kaarten die de waarde van de output bit bepaalt. Ook in dit geval lekt er dus geen informatie over de waarde van de output bit. We concluderen dat het protocol aan alle veiligheidseisen voldoet.

Voor de volledigheid geven we nog een protocol om meerdere exemplaren van een commitment te creëren zonder de waarde ervan vrij te geven. De veiligheid en correctheid ervan zijn vrij gemakkelijk in te zien, voor bewijzen en berekeningen van het aantal kaarten dat nodig is voor elk protocol verwijzen we de lezer naar [30].

Protocol 3. *Een protocol om k copies van een commitment $\boldsymbol{\alpha} = x\overline{x}$ te creëren zonder $\boldsymbol{\alpha}$ te openen.*

1. De partijen construeren gezamenlijk een stapel van $2(k+1)$ afwisselend groene en rode kaarten: $\boldsymbol{u} = (gr)^{k+1}$.

2. Iedereen pakt de stapel af: $\boldsymbol{u'} = F(\boldsymbol{u}) = (y\overline{y})^{k+1}$. We nemen de bovenste twee kaarten van de stapel en leggen die onder het originele commitment $\boldsymbol{\alpha}$. We hebben dan twee stapels op tafel liggen: $\boldsymbol{v} = \boldsymbol{\alpha}u'_1u'_2 = x\overline{x}y\overline{y}$ en $\boldsymbol{u}\text{"} = u'_3u'_4 \ldots u'_{2(k+1)}$.

3. Iedereen pakt het stapeltje van vier kaarten af ($\boldsymbol{v'} = F(\boldsymbol{v})$) en het wordt open gelegd.

4. Als de twee rode kaarten niet langs mekaar liggen (opnieuw cyclisch gezien) geven we $\boldsymbol{u}$" als uitvoer, anders permuteren we de stapel over één plaats zodat $f_1(\boldsymbol{u}\text{"})$ als uitvoer wordt teruggegeven.

Hoofdstuk 6

Protocols voor twee partijen

6.1 Een opwarmertje

Om de smaak te pakken te krijgen zullen we eerst een eenvoudig maar praktisch vrij beperkt protocol beschrijven. Het biedt enkel een oplossing voor het bijzondere geval van Yao's miljonairsprobleem, niet voor Secure Distributed Computing van een willekeurige functie.

Stel dat Alice en Bob de twee miljonairs zijn. Het fortuin van Alice bedraagt i miljoen frank, dat van Bob bedraagt j miljoen frank. Ze willen allebei weten of $i \leq j$ of $i > j$, maar Alice wil het juiste bedrag van haar vermogen niet aan Bob bekend maken en langs de andere kant wil Bob niet dat Alice te weten komt hoeveel er precies op zijn geheime rekening bij de KB-Lux staat. Laten we veronderstellen dat i en j tussen 1 en 100 liggen.

Voor het protocol hebben we een publieke-sleutel algoritme nodig. Het maakt absoluut geen verschil welk algoritme we hiervoor precies kiezen. We zullen een encryptie met de publieke sleutel van Bob daarom abstract noteren als $E_B(\cdot)$ en een decryptie met Bobs private sleutel als $D_B(\cdot)$. Het moet wel een degelijk encryptie-algoritme zijn. Meer bepaald moet het veranderen van één enkele bit in de klaartekst een cijfertekst opleveren die totaal geen verband houdt met de eerste cijfertekst.

Alice kiest eerst een groot willekeurig getal x en encrypteert dat met Bobs publieke sleutel tot $c = E_B(x)$. Ze trekt i van deze cijfertekst af en stuurt het resultaat naar Bob:

$$\text{Alice} \longrightarrow \text{Bob: } c - i$$

Bob berekent achtereenvolgens de getallen $y_u = D_B(c - i + u)$ voor u gaande van 1 tot 100. Er geldt natuurlijk dat $y_i = x$, maar omdat x een volledig willekeurig getal is kan hij y_i niet onderscheiden van de andere y_u's. Vervolgens kiest hij een willekeurige priemgetal p. Het aantal bits van p moet kleiner zijn dan het aantal bits van x. Bob berekent $z_u = (y_u \mod p)$ voor $1 \leq u \leq 100$. Hij stuurt de j eerste z_u's ongewijzigd en in volgorde naar

Alice. Bij de overblijvende z_u's telt hij 1 op en hij stuurt het resultaat, ook weer in volgorde, naar Alice. Hij laat Alice ook weten welke p hij precies gekozen heeft.

$$\text{Bob} \longrightarrow \text{Alice: } z_1, z_2, \ldots, z_j, z_{j+1}+1, z_{j+2}+1, \ldots, z_{100}+1, p$$

Alice bekijkt het i^{e} element in de rij en vergelijkt het met $x \mod p$. Als het congruent is met x modulo p, is $i \leq j$. Als het congruent is met $x+1$ modulo p, is $i > j$. Ze stuurt het resultaat naar Bob. Uit de andere elementen kan Alice geen informatie over j halen. Als ze een degelijk encryptiesysteem gebruiken zijn de y_u's en dus ook de z_u's volledig willekeurige getallen, want ze kent de private sleutel van Bob niet om alles na te rekenen. Aan volledig willekeurige getallen kan ze niet zien of er 1 bij opgeteld is of niet.

Het nut van het priemgetal p is te verzekeren dat Alice ook niet in de andere richting kan beginnen narekenen. Als Bob niet de z_u's naar Alice zendt maar de y_u's zelf, kan Alice achtereenvolgens het eerste, het tweede, ... element van de rij terug encrypteren met Bobs publieke sleutel. De cijferteksten die daaruit komen zullen achtereenvolgens $c-i+1$, $c-i+2$, ... zijn, tot op een bepaald punt vanaf waar er schijnbaar willekeurige getallen uitkomen. Alice weet dan dat dat punt het $(j+1)$e element in de lijst moet zijn. Door Alice de rest van y_u bij deling door p te laten weten, heeft ze de exacte decryptie niet en kan ze ook niet alles opnieuw encrypteren.

Er kan zich een probleem voordoen. In een niet-probabilistisch encryptiesysteem levert de decryptie van twee verschillende cijferteksten altijd twee verschillende klaarteksten op. Het getal $c-i+u$ is altijd verschillend voor twee verschillende waarden van u, dus zijn ook alle y_u's steeds verschillend. Het is echter wel mogelijk dat er door een stom toeval twee waarden u_1 en u_2 zijn $(u_1 < u_2)$ zodat $y_{u1} \equiv y_{u_2} + 1 \mod p$. Als daarenboven $u_1 \leq j < u_2$ ziet Alice op posities u_1 en u_2 toch twee dezelfde waarden in de lijst die ze van Bob kreeg. Omdat het voorgaande scenario de enige mogelijkheid is om twee dezelfde waarden te bekomen, krijgt ze extra informatie over j, namelijk dat $u_1 \leq j < u_2$. Bob kan heel eenvoudig op voorhand detecteren of dit probleem zich voordoet: hij controleert of voor alle u_1 en u_2, $u_1 < u_2$, geldt dat $|z_{u1} - z_{u2}| > 1$ en of er niet tegelijk geldt dat $z_{u1} = 0$ en $z_{u2} = p-1$. Als deze voorwaarden niet voldaan zijn kiest hij een andere p en probeert hij opnieuw.

6.2 Goldreich, Micali en Wigderson

Het basisidee van dit protocol is aangereikt door Oded Goldreich, Silvio Micali en Avi Wigderson in [23]. In deze paper wordt aan het eigenlijke protocol slechts iets meer dan één bladzijde vuil gemaakt, zodat de beschrijving ervan zeer vaag en summier blijft. Franklin werkt in [17] hun idee verder uit tot een overzichtelijk geheel maar maakt daarbij de notatie hier en daar

onnodig zwaar en laat het verwerven van inzicht in het protocol grotendeels over aan de lezer. Bovendien kunnen we hem betrappen op een foutje (de encryptiefunctie E moet helemaal niet probabilistisch zijn, een gewone symmetrische encryptiefunctie volstaat) en kan de nodige hoeveelheid data voor een AND-poort gehalveerd worden. In de volgende paragrafen zullen we enkele verbeteringen aanbrengen en de achterliggende ideeën van het protocol beter in de verf zetten. We maken hierbij gebruik van een notatie gebaseerd op die van Franklin, maar we zullen er op gepaste tijdstippen van afwijken om het geheel te verlichten.

Vanuit vogelperspectief bekeken bestaat het protocol uit twee fasen. In de eerste fase neemt Alice het logische circuit onder handen en zet het om tot een "geëncrypteerde" versie van het circuit. Alice overhandigt dit geëncrypteerde circuit en haar geëncrypteerde invoer aan Bob. In de tweede fase evalueert Bob het circuit voor de specifieke invoer van Alice en hemzelf. Tenslotte maakt hij het functieresultaat bekend aan Alice.

Stel dat het te evalueren circuit uit m draden bestaat die op mekaar zijn aangesloten door middel van logische AND- en NOT-poorten. De verzameling van draden die overeen komen met een invoerbit van Alice noemen we I_A, de verzameling van draden die overeenkomen met een invoerbit van Bob I_B. Verder noteren we de verzameling draden waarop een uitvoerbit komt te staan als U. Met de notatie b_i bedoelen we de bit die op draad i zou staan als de invoer van Alice en Bob aan het circuit aangelegd werd. De invoerbits van Alice zijn dan gelijk aan b_i, $i \in I_A$, analoog kunnen we de invoerbits van Bob noteren als b_i, $i \in I_B$, en wordt de uitvoer gevormd door b_i, $i \in U$.

Zoals reeds vermeld mag Alice het protocol in gang zetten door het circuit te encrypteren. Hiertoe kiest ze voor elke draad i twee willekeurige bitstrings r_i^0 en r_i^1. Het centrale idee is dat Bob straks in de tweede fase van het protocol enkel $r_i^{b_i}$ te weten zal komen, maar totaal geen informatie krijgt over $r_i^{\bar{b_i}}$. We kunnen r_i^b dus zien als een codering voor de bit b op draad i, waarbij de decoderingsfunctie gegeven wordt door $\phi_i : r_i^b \mapsto b$.

Om de logische poorten voor te stellen hebben we een datastructuur nodig die Bob later zal toelaten vanuit de bitstrings die de inputs van de poort voorstellen de juiste bitstring voor de uitvoerdraad te berekenen. Dit wordt gerealiseerd door de codering van de uitvoerbit van elke poort te encrypteren met de overeenkomstige invoerbit(s) als sleutel. We maken hiervoor gebruik van een klassiek symmetrisch encryptie-algoritme, zoals DES of IDEA. Welk algoritme juist gebruikt wordt is irrelevant — zolang het maar veilig is. Bob moet een correcte decryptie kunnen detecteren, hetgeen niet vanzelfsprekend is daar de bitstrings volledig willekeurig zijn. Alice kiest daartoe nog een willekeurige bitstring R die ze telkens aan de gegevens plakt alvorens ze te encrypteren. Bob kan hetgeen hij gedecrypteerd heeft dan vergelijken met R.

Voor een NOT-poort met invoerdraad *in* en uitvoerdraad *uit* construeert Alice een willekeurige permutatie van het tuple

$$\langle\ E_{r^0_{in}}(R \cdot r^1_{uit}), E_{r^1_{in}}(R \cdot r^0_{uit})\ \rangle$$

waarbij $\cdot$ staat voor de concatenatie van bitstrings. De willekeurige permutatie is nodig opdat Bob geen informatie kan halen uit het feit dat hij het eerste of het tweede element van het tuple kan decrypteren. Voor een AND-poort is de situatie net iets moeilijker omdat we nu met twee invoerdraden *links* en *rechts* en een uitvoerdraad *uit* te doen hebben. In dit geval construeert Alice een willekeurige permutatie van het tuple

$$\langle\ E_{r^0_{links} \oplus r^0_{rechts}}(R \cdot r^0_{uit}), E_{r^0_{links} \oplus r^1_{rechts}}(R \cdot r^0_{uit}),$$

$$E_{r^1_{links} \oplus r^0_{rechts}}(R \cdot r^0_{uit}), E_{r^1_{links} \oplus r^1_{rechts}}(R \cdot r^1_{uit})\ \rangle$$

waarin $\oplus$ voor de bit-gewijze XOR staat.

Alice zendt al deze tuples naar Bob, samen met R, de codering voor haar eigen invoerbits r_i ($i \in I_A$) en de decoderingsfunctie van de uitvoerbits ϕ_i ($i \in U$). Maar hoe kan Alice de juiste codering voor Bob's input bits tot bij hem krijgen? Ze wil niet zowel r^0_i als r^1_i ($i \in I_B$) aan hem verklappen, want dan kan Bob zonder medeweten van Alice het circuit meermaals evalueren, telkens met andere invoer, en zo trachten meer te weten te komen over de invoer van Alice. Langs de andere kant wil Bob niet verklappen welke bitstrings hij precies nodig heeft, want dat staat gelijk met het prijsgeven van zijn geheime invoer. Deze impasse wordt netjes opgelost door een one-out-of-two oblivious bitstring transfer (vaak genoteerd als $\binom{2}{1}$-OT^k; zie sectie 4.5). Voor elke $i \in I_B$ biedt Alice r^0_i en r^1_i aan met behulp van een oblivious transfer en Bob kiest volgens de waarde van zijn input bit b_i. De oblivious transfer zorgt ervoor dat Alice b_i niet te weten komt terwijl Bob geen informatie krijgt over $r^{\bar{b}_i}_i$.

Nu is het aan Bob om de tweede fase van het protocol in te zetten. Vertrekkende van de codering van alle invoerbits kan hij het volledige circuit evalueren als volgt. Als er een NOT-poort is met invoerdraad *in* waarvan hij reeds een codering r^*_{in} kent[1], probeert hij de twee elementen van het tuple dat bij die NOT-poort hoort te decrypteren met r^*_{in} als sleutel. Een geslaagde decryptie kan hij herkennen aan het feit dat de eerste gedecrypteerde bits gelijk zijn aan R. Slechts één van die twee elementen zal hij succesvol kunnen decrypteren en levert hem de codering r^*_{uit} van de bit op de uitvoerdraad. Als er een AND-poort is waarvan hij voor beide invoerdraden *links* en *rechts* reeds een codering kent maar nog niet voor de uitvoerdraad *uit*, probeert hij de vier elementen van het bijhorende tuple te decrypteren met $r^*_{links} \oplus r^*_{rechts}$

[1] We gebruiken de notitie r^*_i omdat Bob geen idee heeft of het om r^0_{in} of r^1_{in} gaat (tenzij de waarde enkel afhankelijk is van zijn eigen inputs).

als sleutel. Opnieuw zal dit slechts bij één van de elementen lukken en zal hij daaruit de codering r^*_{uit} van de uitvoerdraad halen.

Op deze manier propageert Bob doorheen het volledige circuit tot hij een codering heeft voor alle draden in het circuit. Hij kent dan ook een codering r^*_i voor de uitvoerdraden $i \in U$ die hij gemakkelijk kan decoderen tot de echte uitvoer van het circuit door ϕ_i erop toe te passen. Tenslotte zendt Bob de uitvoer van het circuit naar Alice.

De correctheid van het protocol is eenvoudig na te gaan als we de datastructuren van de poorten nader bekijken. Stel dat Bob de codering r^*_{in} kent op de invoerdraad van een NOT-poort. Als Alice het circuit correct geëncrypteerd heeft zijn de twee elementen van het tuple zodanig geconstrueerd dat de bitstring r^*_{uit} die Bob uit het element haalt dat hij succesvol kan decrypteren steeds voldoet aan

$$\phi_{uit}(r^*_{uit}) = \overline{\phi_{in}(r^*_{in})}$$

Op een analoge wijze kan men inzien dat de verkregen bitstring r^*_{uit} die Bob verkrijgt uit het tuple van een AND-poort met invoerdraden *links* en *rechts* steeds voldoet aan

$$\phi_{uit}(r^*_{uit}) = \phi_{links}(r^*_{links}) \wedge \phi_{rechts}(r^*_{rechts})$$

Hieruit volgt dat op elke draad i geldt dat $\phi_i(r^*_i) = b_i$. Dit geldt a fortiori ook voor de uitvoerdraden, zodat de correctheid van de uitvoer verzekerd is.

De veiligheid van Alices invoer wordt gegarandeerd door het feit dat Bob de codering van haar inputs krijgt maar hier geen informatie uit kan halen over de waarde ervan. Verder komt Bob voor geen enkele draad i zowel r^0_i als r^1_i te weten, zodat hij het circuit slechts voor één combinatie van inputs kan evalueren. De veiligheid van Bobs invoer wordt dan weer gegarandeerd door de oblivious transfer.

Het protocol zoals voorgesteld door de drie auteurs en besproken door Franklin is enkel bestand tegen semi-eerlijke tegenstanders. Het is voor een kwaadaardige tegenstander bijna belachelijk eenvoudig om de andere partij met een kluitje in het riet te sturen. In de laatste stap moet Bob de uitvoer van het circuit naar Alice zenden. Alice heeft geen enkele controle of de uitvoer die Bob haar toestuurt ook effectief de uitvoer is die hij net berekend heeft, het kan net zo goed om een hoop willekeurige bits gaan. Hoe kunnen we Alice meer zekerheid geven? Als Alice de decoderingsfuncties ϕ_i, $i \in U$, niet op voorhand verklapt kent Bob na de evaluatie van het circuit voor elke uitvoerdraad slechts één codering, nl. die van de juiste uitvoer. Alice kan Bob vragen haar deze coderingen toe te sturen zodat zij ze kan decoderen en de gedecodeerde uitvoer naar Bob kan terugsturen. Bob heeft geen mogelijkheid meer om vals te spelen, want om Alice met een foutieve

uitvoer op te zadelen moet hij haar de andere codering van de uitvoerdraad toesturen, en die kent hij niet. Maar nu bevindt Bob zich weer in een heel kwetsbare positie, want Alice kan eender wat doorsturen als decodering van zijn uitvoer. Het lijkt erop dat we het probleem gewoon verschoven van Alice naar Bob.

We moeten inderdaad toegeven dat de laatste versie geen haar beter is dan de oorspronkelijke, maar we willen wel de aandacht richten op een opmerkelijke eigenschap ervan: er bestaat een zware asymmetrie tussen de veiligheid van Alice en die van Bob. Als Alice een kwaadaardige tegenstander is kan ze Bob op verschillende manieren bedriegen: ze kan het circuit foutief encrypteren, ze kan Bobs inputs verwisselen of ze kan hem eender welke uitvoer aansmeren als de decodering van wat hij net berekend heeft. Bob staat echter met zijn rug tegen de muur. Zijn enige hoop is het kraken van het encryptie-algortime E — niet meteen een haalbare kaart — of valsspelen tijdens de oblivious transfer. Als het gebruikte protocol voor de oblivious transfer kwaadaardige tegenstanders aankan, smelt zijn laatste hoop als sneeuw voor de zon. Het enige dat hij kan doen is het protocol vroegtijdig afbreken zodat Alice de uitvoer niet te weten komt, hetgeen hem geen kennisvoordeel ten opzichte van Alice oplevert want hij kan de uitvoer niet op eigen houtje decoderen.

Deze versie van het protocol kan een kwaadaardige tegenstander aan[2], maar alleen als die de rol van Bob speelt. Blijkbaar is de rol van Alice, de encryptor, een machtigere rol dan die van Bob, de evaluator. Iedereen die aan een dergelijk protocol deelneemt zal dus liefst de rol van encryptor spelen, al is het maar om zeker te zijn dat de ander niet zal valsspelen. We kunnen overwegen eerst een fair coin flip uit te voeren om te bepalen welke partij welke rol mag spelen, maar dat geeft een tegenstander een kans van één op twee om ongemerkt vals te spelen, hetgeen absoluut onaanvaardbaar is. Een betere oplossing is het protocol dubbel uit te voeren: Alice en Bob spelen eerst beide de rol van encryptor: ze nemen allebei een exemplaar van het circuit onder handen en sturen de gegevens die nodig zijn voor de evaluatie ervan naar mekaar door. Ze evalueren beide het circuit dat ze van de ander gekregen hebben tot ze de codering van de uitvoerbits hebben. Tenslotte sturen ze afwisselend de codering van één uitvoerbit naar mekaar door, zodat het vroegtijdig afbreken van het protocol maximaal één bit aan informatievoordeel ten opzichte van de andere partij kan betekenen. In protocol 4 geven we een overzicht van het gewijzigde protocol. Om het onderscheid te kunnen maken tussen de twee encrypties van het circuit noteren we alle bitstrings in de encryptie van Alice met de letter r en de bitstrings in het circuit dat Bob encrypteert met de letter s.

[2]Het heeft weinig zin het geval te beschouwen waarin beide partijen kwaadaardige tegenstanders zijn. De bedoeling van een cryptografisch protocol is altijd eerlijke partijen te beschermen voor valsspelers. Als er geen eerlijke partijen zijn heeft het geen zin een protocol uit te voeren.

Protocol 4. *Een protocol gebaseerd op dat van Goldreich, Micali en Wigderson, maar gewapend tegen kwaadaardige tegenstanders.*

1. Alice encrypteert het circuit op dezelfde manier als hiervoor beschreven met behulp van de willekeurige bitstrings R, r_i^0, r_i^1.

   $$\text{Alice} \longrightarrow \text{Bob: het geëncrypteerde circuit, } R,\ r_i^{b_i}\ \forall i \in I_A$$
   $$\text{Alice} \longrightarrow \text{Bob : } r_i^{b_i}\ \forall i \in I_B \text{ via oblivious transfer}$$

2. Bob encrypteert het circuit op dezelfde manier als hiervoor beschreven met behulp van de willekeurige bitstrings S, s_i^0, s_i^1.

   $$\text{Bob} \longrightarrow \text{Alice: het geëncrypteerde circuit, } S,\ s_i^{b_i}\ \forall i \in I_B$$
   $$\text{Bob} \longrightarrow \text{Alice: } s_i^{b_i}\ \forall i \in I_A \text{ via oblivious transfer}$$

3. Alice en Bob evalueren het circuit dat ze net van mekaar hebben gekregen. Alice berekent op deze manier s_i^*, Bob berekent r_i^*.

4. Herhaal voor elke $i \in U$:

   $$\text{Alice} \longrightarrow \text{Bob: } s_i^*$$
   $$\text{Bob} \longrightarrow \text{Alice: } r_i^*$$

Het is belangrijk dat Bob en Alice afwisselend een uitvoerbit aan mekaar bekend maken. Als Alice eerst alle uitvoerbits in één keer aan Bob bekend maakt, kan hij door het protocol stop te zetten de volledige uitvoer kennen terwijl Alice er het raden naar heeft. Door om beurten een uitvoerbit bekend te maken kan Bob hoogstens één bit meer weten dan Alice als hij het protocol vroegtijdig afbreekt. In hoofdstuk 2 hebben we reeds de vraag gesteld of het mogelijk is de stap nog verder te verfijnen zodat Bob bijvoorbeeld hoogstens het voordeel kan hebben in half zo lange tijd als Alice op eigen houtje een extra uitvoerbit kan achterhalen.

6.3 Abadi en Feigenbaum

Alice heeft een geheim circuit voor een functie f en Bob heeft geheime gegevens x. Bob zou graag $f(x)$ berekenen, maar hij heeft geen algoritme voor f of hij kent f niet exact. Alice is bereid haar circuit door Bob te laten gebruiken, maar wil het circuit niet aan Bob verklappen. Langs de andere kant wil Bob niet dat Alice x te weten komt. Dit probleem lijkt op het eerste gezicht verschillend van de algemene probleemstelling van Secure Distributed Computing, maar komt toch op hetzelfde neer vanwege de equivalentie tussen data en functies. Het volgende protocol, voorgesteld door Martín Abadi en Joan Feigenbaum in [1], helpt Alice en Bob uit hun impasse.

Dit protocol is volledig gebaseerd op kwadratische residuositeit, meer bepaald op kwadratische residu's in de groep $\mathbb{Z}_n^*[+1]$, de groep van de getallen copriem met n en met Jacobi-symbool 1 zoals beschreven in sectie 3.3.6. De modulus n is van de vorm $n = pq$ met p en q priemgetallen en $p \equiv q \equiv 3 \mod 4$. Volgens de Kwadratische Residuositeit Veronderstelling (KRV) bestaat er geen polynomiale-tijd algoritme om te beslissen of een getal een kwadraat is modulo een samengesteld getal als de factoren van de modulus niet gekend zijn. We hebben reeds aangetoond dat het daarentegen heel eenvoudig is een kwadraat of niet-kwadraat te construeren in $\mathbb{Z}_n^*[+1]$, ook als de factoren van n niet gekend zijn: neem gewoon een willekeurig element r en kwadrateer het om een kwadraat modulo n te bekomen, of vermenigvuldig dit nog eens met -1 om een niet-kwadraat te verkrijgen. We gebruiken het probabilistisch encryptiesysteem dat we in sectie 4.2 besproken hebben. We encrypteren een bit b als

$$E_n(b) = (-1)^b \cdot r^2 \quad \text{met } r \in_R Z_n^*[+1]$$

terwijl de decryptiefunctie gegeven wordt door

$$D_n(x) = \begin{cases} 0 & \text{als } x \text{ een kwadraat is modulo } n \\ 1 & \text{als } x \text{ een niet-kwadraat is modulo } n \end{cases}$$

Dit encryptiesysteem heeft twee opmerkelijke eigenschappen waar dit protocol dankbaar gebruik van zal maken.

Eigenschap. *Als $E_n(b)$ een probabilistische encryptie is van een bit b, is $(-1) \cdot E_n(b) \mod n$ een probabilistische encryptie van $\overline{b}$:*

$$E_n(\overline{b}) = (-1) \cdot E_n(b) \mod n$$

Het is dus mogelijk de waarde verborgen door een encryptie te inverteren zonder iets over de waarde zelf te weten te komen.

Eigenschap. *Als $E_n(b_1)$ en $E_n(b_2)$ probabilistische encrypties zijn van de bits b_1 en b_2, dan volgt uit stelling 3.6 dat $E_n(b_1) \cdot E_n(b_2) \mod n$ een probabilistische encryptie van $b_1 \oplus b_2$ is*[3] *:*

$$E_n(b_1 \oplus b_2) = E_n(b_1) \cdot E_n(b_2) \mod n$$

Het is dus mogelijk een encryptie te berekenen voor de XOR van twee geëncrypteerde bits zonder die bits te decrypteren. Er bestaat blijkbaar een homomorfisme tussen de XOR in $\mathbb{Z}_2$ en de vermenigvuldiging in $\mathbb{Z}_n^*[+1]$. In het algemeen noemen we een encryptiesysteem E *homomorf* als het voldoet aan een eigenschap van de vorm $E(x \ \mathbf{op} \ y) = E(x) \ \mathbf{op'} \ E(y)$ voor bepaalde operaties **op** en **op'**. Een homomorf encryptiesysteem laat toe operaties

[3] Vergelijk deze eigenschap met de blindeerbaarheid van bit commitments uit sectie 4.4

uit te voeren op geëncrypteerde data en kan goede diensten bewijzen voor Secure Distributed Computing.

Bob zet het protocol in door twee priemgetallen p en q te kiezen congruent met 3 mod 4. Hij berekent $n = pq$ en encrypteert elke bit x_i van zijn geheime gegevens x. Bob zendt n en zijn geëncrypteerde gegevens $E_n(x_i)$ naar Alice. De factorisatie van n houdt hij geheim. Zonder deze factorisatie kan Alice de geheime data van Bob niet decrypteren. Vervolgens begint Alice het circuit te evalueren. Als ze een NOT-poort met input $E_n(b)$ tegenkomt, berekent ze de output van de poort als $-E_n(b)$. Een XOR-poort met inputs $E_n(b_1)$ en $E_n(b_2)$ kan ze ook zonder probleem evalueren: de uitvoer ervan is $E_n(b_1) \cdot E_n(b_2) \mod n$.

Voor de evaluatie van een AND-poort met inputs $E_n(b_1)$ en $E_n(b_2)$ heeft ze echter Bob's hulp nodig. Ze kiest twee willekeurige bits c_1 en c_2, construeert encrypties $E_n(c_1)$ en $E_n(c_2)$ voor deze bits, berekent $E_n(b_i \oplus c_i) = E_n(b_i) \cdot E_n(c_i) \mod n$ en zendt $E_n(b_1 \oplus c_1)$ en $E_n(b_2 \oplus c_2)$ naar Bob. Bob kent de factorisatie van n en kan deze bits dus decrypteren. Hij ziet echter alleen $d_1 = b_1 \oplus c_1$ en $d_2 = b_2 \oplus c_2$, hij kan hieruit geen informatie afleiden over b_i of c_i zelf. Bob stuurt de volgende gegevens naar Alice:

$$z_{00} = E_n(d_1 \wedge d_2),\ z_{01} = E_n(d_1 \wedge \overline{d_2}),\ z_{10} = E_n(\overline{d_1} \wedge d_2),\ z_{11} = E_n(\overline{d_1} \wedge \overline{d_2})$$

naar Alice. Alice neemt $z_{c_1 c_2}$ als uitvoer van de AND-poort.

Op deze manier kan Alice doorheen het volledige circuit propageren totdat ze een encryptie heeft van de uitvoerbits. Deze geëncrypteerde bits stuurt ze naar Bob. Als Alice het resultaat zelf ook wil weten zendt Bob haar de gedecrypteerde uitvoerbits. Bob mag ook nu in geen geval de factoren p en q aan Alice verklappen, want als zij ergens een transcript van de communicatie bewaard heeft kan ze alsnog x achterhalen.

Voor de evaluatie van een NOT- of XOR-poort is er geen interactie nodig, voor de evaluatie van een AND-poort moet Alice twee geëncrypteerde bits naar Bob sturen en moet Bob een tuple van vier geëncrypteerde bits terugsturen, een totaal van zes bit-encrypties die over het netwerk moeten. Dit aantal kan herleid worden tot drie door een booleaans truukje toe te passen, geïnspireerd op de evaluatie van een AND-poort in het protocol van Franklin en Haber uit sectie 7.3. Net als in het origineel protocol kiest Alice twee willekeurige bits c_1 en c_2 en stuurt ze $E_n(b_1 \oplus c_1)$ en $E_n(b_2 \oplus c_2)$ naar Bob. Bob decrypteert deze bits als d_1 en d_2, maar stuurt enkel $E_n(d_1 \wedge d_2)$ terug naar Alice. Als we gebruik maken van het feit dat

$$(a \oplus b) \wedge c = (a \wedge c) \oplus (b \wedge c)$$

kunnen we aantonen dat

$$\begin{aligned} & d_1 \wedge d_2 && = (b_1 \oplus c_1) \wedge (b_2 \oplus c_2) \\ & && = (b_1 \wedge b_2) \oplus (b_1 \wedge c_2) \oplus (b_2 \wedge c_1) \oplus (c_1 \wedge c_2) \\ \Leftrightarrow\ & b_1 \wedge b_2 && = (d_1 \wedge d_2) \oplus (b_1 \wedge c_2) \oplus (b_2 \wedge c_1) \oplus (c_1 \wedge c_2) \end{aligned}$$

en als we het homomorfisme tussen XOR van bits en vermenigvuldiging van encrypties toepassen resulteert dit in

$$E_n(b_1 \wedge b_2) = E_k(d_1 \wedge d_2) \cdot E_n(b_1 \wedge c_2) \cdot E_n(b_2 \wedge c_1) \cdot E_n(c_1 \wedge c_2) \mod n$$

Het linkerlid van deze vergelijking is wat Alice nodig heeft om verder te kunnen met de evaluatie van het circuit. De eerste factor in het rechterlid kent ze, want die heeft ze net gekregen van Bob. De bits c_1 en c_2 heeft ze zelf gekozen, dus kan ze gemakkelijk ook de laatste factor berekenen. Een encryptie voor $b_1 \wedge c_2$ berekenen is iets moeilijker. Als ze $c_2 = 0$ gekozen had, is $b_1 \wedge c_2 = 0$, dus kan ze een willekeurig kwadraat nemen als encryptie voor $b_1 \wedge c_2$. Als ze $c_2 = 1$ gekozen had, is $b_1 \wedge c_2 = b_1$. Ze kent reeds een geldige encryptie $E_n(b_1)$, want dat was één van de inputs van de AND-poort. De factor $E_n(b_2 \wedge c_1)$ berekent Alice op volledig analoge manier. Het vermenigvuldigen modulo n van deze vier factoren levert een encryptie voor de uitvoer van de AND-poort op.

Tijdens het protocol krijgt Bob geen informatie over de geheime functie f van Alice behalve het aantal AND-poorten in het circuit. Voor elke praktische toepassing van dit protocol moet nagekeken worden of het aantal AND-poorten voor dat concrete geval belangrijke informatie voor Bob kan bevatten. Eventueel kan Alice een aantal valse AND-poorten inlassen door enkele keren twee willekeurige elementen van $Z_n^*[+1]$ naar Bob te sturen en het antwoord van Bob weg te gooien. Op die manier kent Bob enkel nog een bovengrens voor het aantal AND-poorten. Protocols voor Secure Distributed Computing zijn echter al vrij zwaar op zich, dus is het eigenlijk af te raden "nutteloze" communicatie-overhead te creëren.

De gegevens x van Bob worden geëncrypteerd en blijven voor Alice verborgen op voorwaarde dat de KRV stand houdt. Theoretisch gezien is dit een belangrijk onderscheid met de veiligheid die Alice geniet voor haar functie f. Alice stuurt, behalve het aantal AND-poorten, totaal geen informatie over f naar Bob, dus zelfs als Bob beschikt over ongelimiteerde rekenkracht kan hij f niet ontfutselen. Als Alice daarentegen over ongelimiteerde rekenkracht beschikt, kan ze n factoriseren en dus de geëncrypteerde data die ze van Bob gekregen heeft decrypteren. In de praktijk is deze discrepantie veel minder relevant, zolang we n maar groot genoeg kiezen opdat Alice met alle rekenkracht die zij tot haar beschikking heeft niet in staat is binnen afzienbare tijd de factorisatie van n te berekenen. De prijs die we hiervoor betalen is netwerkbelasting: alle doorgestuurde gegevens zijn getallen modulo n. Hoe groter we n kiezen, hoe beter Bob's gegevens beschermd zijn maar hoe groter ook de getallen modulo n.

Ook dit protocol is enkel bestand tegen semi-eerlijke tegenstanders. Hoe kan een kwaadaardige tegenstander roet in het eten gooien? Als Bob kwaadaardig is kan hij bij de evaluatie van een AND-poort foutieve waarden terugsturen naar Alice. Hij kan in de slotfase van het protocol ook liegen over

de waarde van de uitvoerbits zonder Alice's argwaan te wekken. Als Alice kwaadaardig is kan ze natuurlijk van de regels om het circuit te evalueren afwijken, maar een belangrijker punt is dat ze in de slotfase van het protocol niet de encrypties van de uitvoerbits maar bijvoorbeeld Bobs geëncrypteerde data of belangrijke tussenresultaten door Bob laat openen. Ze kan zelfs Bobs geëncrypteerde data vermenigvuldigen met encrypties van willekeurige bits, zodat Bob een hoop nonsens als uitvoer ziet terwijl eigenlijk zijn eigen invoer erin verborgen zit. Meestal is het aantal uitvoerbits wel veel kleiner dan het aantal invoerbits zodat Alice niet Bob's volledige invoer kan achterhalen. Maar een paar bits kunnen al te veel zijn natuurlijk ...

Het is verleidelijk de veiligheid van dit protocol te verbeteren op een analoge manier als we gedaan hebben in het protocol van Goldreich, Micali en Wigderson. In de slotfase van het protocol krijgt Bob de geëncrypteerde uitvoer toegestuurd om die te decrypteren en het resultaat naar Alice terug te sturen. Het is al te gemakkelijk om een valse uitvoer naar Alice terug te sturen. We kunnen dit misschien verhinderen door Bob te laten bewijzen dat de bit die hij stuurt ook werkelijk de decryptie is van wat Alice daarnet naar hem gestuurd heeft. Hij kan dit doen door Alice een wortel van $E_n(b)$ te geven als $b = 0$ en een wortel van $-E_n(b)$ als $b = 1$. Bob kan inderdaad niet meer liegen over de uitvoer, maar nu schuilt er voor hemzelf geen gewoon addertje maar een heuse boa constrictor onder het gras. Stel dat Alice een willekeurig getal x kiest en $a = x^2 \mod n$ naar Bob stuurt in plaats van de uitvoer van het circuit. We weten dat a vier wortels heeft modulo n, namelijk x, $-x$, y en $-y$. Bob zal één van deze vier wortels naar Alice sturen. Met een kans van één op twee stuurt hij y of $-y$ naar Alice. In dat geval kent zij twee echt verschillende wortels en kan ze, zoals aangetoond in sectie 3.3, de factoren p en q berekenen en zo Bob's volledige invoer decrypteren! Dit voorbeeld toont mooi aan hoe delicaat de veiligheid van een cryptografisch protocol kan zijn; ze kan vallen of staan met een ogenschijnlijk piepklein detail.

6.4 Sander en Tschudin

De voorgaande protocols hebben allemaal een vervelend trekje: ze gebruiken meer communicatierondes dan strikt noodzakelijk. In principe moet het mogelijk zijn een protocol te ontwerpen waarbij Alice haar geëncrypteerde data naar Bob stuurt, Bob berekeningen uitvoert op deze geëncrypteerde data en het resultaat terug naar Alice stuurt. Zo'n protocol wordt een *niet-interactief* of *autonoom* protocol genoemd, wat een beetje een verwarrende naam is want er vindt wel degelijk — zij het een minimale hoeveelheid — interactie plaats. Merk op dat een niet-interactief protocol niet noodzakelijk een lichtere netwerkbelasting heeft, het kan in één enkele communicatieronde meer data versturen dan een ander protocol in totaal doet gespreid over

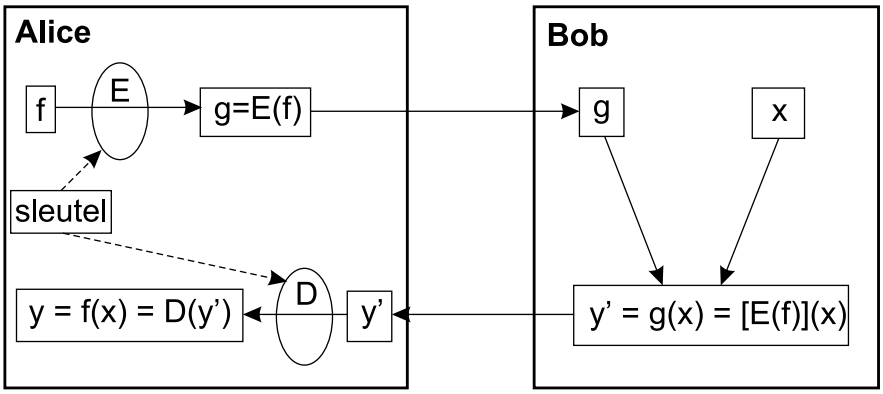
meerdere communicatierondes. Het grote voordeel van een niet-interactief protocol is dat Alice haar data naar Bob kan sturen en Bob op zijn eentje alles kan berekenen. Alice kan intussen off-line gaan want haar tussenkomst is niet meer nodig. Bob kan haar het resultaat terugsturen van zodra ze terug on-line is. Een niet-interactief protocol is perfect geschikt voor het beveiligen van mobiele code. Als mobiele code tijdens zijn uitvoering op een andere machine om de haverklap de tussenkomst van zijn uitzender nodig heeft, gaan de meeste voordelen van het gebruik van mobiele code in rook op. Een niet-interactief protocol laat ons toe de veiligheid van de code te garanderen zonder de voordelen van mobiele code prijs te geven. We gaan hier dieper op in in sectie 8.2.

Het protocol van Goldreich, Micali en Wigderson uit sectie 6.2 benadert het ideaal geval van een niet-interactief protocol nog het best. Alice stuurt het geëncrypteerde circuit en de codering voor haar invoer naar Bob, voert een oblivious transfer uit om Bob van de codering van zijn eigen invoer te voorzien, Bob evalueert het circuit en stuurt het resultaat terug naar Alice. Jammer dat die oblivious transfer nodig is, want die introduceert twee communicatierondes extra als we de implementatie van sectie 4.5 gebruiken. Het protocol van Abadi en Feigenbaum dat we daarnet hebben besproken is een hel wat communicatierondes betreft: voor elke AND-poort in het circuit is er interactie nodig, zodat het aantal communicatierondes evenredig is met de diepte van het circuit.

In [33] en [34] zetten Sander en Tschudin een stap in de richting van privacy voor mobiele code. Ze beschouwen het probleem van niet-interactieve evaluatie van geëncrypteerde functies, dat door de equivalentie tussen data en code nauw verwant is met Secure Distributed Computing. Stel dat Alice een algoritme heeft voor een functie f. Bob heeft een invoer x en is bereid $f(x)$ voor haar te berekenen, maar Alice wil niet dat Bob enige "substantiële" kennis over f krijgt. Bovendien moet Bob in staat zijn zonder interactie met Alice $f(x)$ te berekenen.

Een schematisch verloop van een dergelijk protocol wordt gegeven in figuur 6.1. Alice encrypteert haar functie f tot een functie g waaruit Bob geen informatie kan halen over f. Bob berekent zonder interactie met Alice $g(x)$ en stuurt het resultaat terug naar Alice. Zij decrypteert dit tot $D(g(x)) = f(x)$.

We zouden natuurlijk graag eender welke functie f op die manier kunnen evalueren, maar het protocol van Sander en Tschudin beperkt zich tot veeltermen over de ring $\mathbb{Z}_n$. Het ideale gereedschap om geëncrypteerde veeltermen niet-interactief te berekenen is een encryptiesysteem dat zowel additief als multiplicatief homomorf is, d.w.z. waarin zowel $E(x + y)$ als $E(x \cdot y)$ gemakkelijk te berekenen zijn uit $E(x)$ en $E(y)$. De vraag naar zo'n encryptiesysteem is in 1991 gesteld door Feigenbaum en Merritt, maar tot op heden is er geen antwoord op gevonden. Voor het rekenen met geëncrypteerde veeltermen is het echter niet noodzakelijk dat het encryptiesysteem zowel



Figuur 6.1: Schematische voorstelling van niet-interactieve evaluatie van geëncrypteerde functies

additief als multiplicatief homomorf is. Het volstaat dat de encryptiefunctie additief homomorf en *gemengd* multiplicatief homomorf is. Een encryptiefunctie E noemen we gemengd multiplicatief homomorf als $E(xy)$ gemakkelijk te berekenen is uit $E(x)$ en y (en dus niet $E(y)$) zonder x te onthullen. Dit is helemaal geen strenge voorwaarde, want we kunnen aantonen dat elke additief homomorfe encryptiefunctie ook gemengd multiplicatief homomorf is[4].

Stelling 6.1. *Als E een additief homomorfe encryptiefunctie is, dan is E tevens gemengd multiplicatief homomorf.*

Bewijs. De opgave bestaat erin uit $E(x)$ en y een encryptie $E(xy)$ te berekenen. Zij $y = \sum_{i=0}^{\log n} y_i 2^i$. Bereken eerst door herhaalde toepassing van het additief homomorfisme de lijst $E(2^i x)$, $1 \leq i \leq \log n$. Gebruik opnieuw het additief homomorfisme om de $E(2^i x)$ op te tellen waarvoor $y_i = 1$. □

Stel dat $p = \sum a_{i_1 \ldots i_s} X_1^{i_1} \ldots X_s^{i_s} \in \mathbb{Z}_n[X_1 \ldots X_s]$ de veelterm is die Alice wil evalueren voor Bobs gegevens $x_1 \ldots x_s$. Alice en Bob gaan dan als volgt te werk.

1. Alice construeert een programma g dat Bobs data $x_1 \ldots x_s$ als invoer neemt en een lijst $L = [\ldots, (x_1^{i_1} \ldots x_s^{i_s}), \ldots]$ construeert met de eentermen van p geëvalueerd op $x_1 \ldots x_n$. Vervolgens gebruikt g het gemengd multiplicatief homomorfisme om vanuit $E(a_{i_1 \ldots i_s})$ en L een lijst $M = [\ldots, E(a_{i_1 \ldots i_s} x_1^{i_1} \ldots x_s^{i_s}), \ldots]$ te construeren. Tenslotte telt g alle elementen van M samen door het additief homomorfisme toe te passen. Deze functie g stuurt Alice naar Bob.

[4]De tactiek is analoog aan het efficiënte algoritme voor machtsverheffing in eindige velden uit sectie 3.2

2. Bob voert g uit op zijn geheime invoer $x_1 \ldots x_s$ en verkrijgt $g(x)$. Hij stuurt dit resultaat terug naar Alice.

3. Alice decrypteert het resultaat $D(g(x)) = p(x)$.

Voor dit protocol zijn we er wel van uitgegaan dat we een additief homomorf encryptiesysteem over $\mathbb{Z}_n$ ter beschikking hebben. Voor $\mathbb{Z}_2$ kunnen we gebruik maken van het encryptiesysteem gebaseerd op kwadratische residu's dat ook in het protocol van Abadi en Feigenbaum uit sectie 6.3 reeds goede diensten heeft bewezen. De auteurs verwijzen daarnaast naar een nog niet verschenen paper van Lipton en Sander waarin een additief homomorf encryptiesysteem over $\mathbb{Z}_n$ wordt voorgesteld als n een "zachte" integer is (d.w.z. dat n enkel kleine priemfactoren heeft).

In het protocol blijft de veelterm p zeker niet volledig geheim. Er lekt een hoeveelheid informatie die, afhankelijk van de concrete toepassing, onaanvaardbaar kan zijn. Meer bepaald kan Bob de verzameling eentermen met coëfficiënt verschillend van nul uit het programma g halen. We noemen deze verzameling het *skelet* van p. Als E een semantisch veilige encryptie is voor getallen uit $\mathbb{Z}_n$, is het skelet de enige informatie die lekt, de coëfficiënten $a_{i_1 \ldots i_s}$ blijven veilig verborgen. Stel dat de geheime veelterm van Alice gelijk is aan $p(X_1, X_2) = 13X_1^{15}X_2^{11} + 6X_1^8X_2^{12} + 8X_1^3X_2$, dan zal Bob het skelet van p, namelijk $\{X_1^{15}X_2^{11}, X_1^8X_2^{12}, X_1^3X_2\}$ te weten komen, maar niet de waarde van de coëfficiënten 13, 6 en 8.

Het protocol laat enkel toe geëncrypteerde veeltermen over een ring $\mathbb{Z}_n$ te evalueren. De auteurs geven toe dat dit een zware hypotheek legt op het nut van het protocol, maar zij zien het als een eerste stap naar de uitbreiding tot algebraïsche en uiteindelijk Booleaanse circuits. Als iemand een protocol vindt voor niet-interactieve evaluatie van geëncrypteerde Booleaanse circuits, ligt de weg open voor de beveiliging van eender welke mobiele code.

Een ander zwak punt is de veiligheid van de veelterm. Bob heeft geen idee van de coëfficiënten, maar krijgt wel het volledige skelet van de veelterm in handen. Dit is toch nog een hemelsbreed verschil met volledige veiligheid van de veelterm. Een voorbeeld van een concrete toepassing waarbij de coëfficiënten van een veelterm geheim moeten blijven maar het lekken van het skelet geen probleem vormt, had een veel degelijkere fundering kunnen geven aan het protocol. Een dergelijk voorbeeld wordt echter niet gegeven.

Hoofdstuk 7

Protocols voor meerdere partijen

7.1 Cramer

We hebben in sectie 4.3 aangebracht hoe een geheim kan verdeeld worden onder n partijen zodat elke groep van t partijen het geheim kan reconstrueren. Dit principe past perfect in Secure Distributed Computing voor n partijen waarvan maximaal $t-1$ semi-eerlijke (roddelende) tegenstanders zijn. Als elke partij zijn geheime invoer verdeelt over de n partijen, is het misschien mogelijk uit schaduwen van de verschillende inputs ook schaduwen van de outputs te berekenen, die de deelnemers aan het eind van het protocol bekendmaken om het resultaat te onthullen. Dit idee is het onderwerp geweest van een heel aantal papers, waaruit Ronald Cramer in [13] een elegant en begrijpbaar protocol destilleert.

Even ter herinnering: als iemand een geheim $s \in K$, met K een eindig veld, wil verdelen over n partijen zodat een groep van t partijen het geheim kan reconstrueren, kiest hij een willekeurige veelterm $f(X) \in K[X]$ van graad $t-1$ met s als constante term. Er geldt dan dat $f(0) = s$. Hij stuurt naar elke partij $i = 1 \ldots n$ de schaduw $s_i = f(i)$. Elke groep van t spelers kan het geheim achterhalen door hun schaduwen s_i aan mekaar bekend te maken, met behulp van Lagrange interpolatie de veelterm $f(X)$ te reconstrueren en het geheim $s = f(0)$ te berekenen.

We nemen aan dat de functie die de partijen willen evalueren gegeven is als een algebraïsch circuit over het eindige veld K. Algebraïsche circuits zijn vergelijkbaar met booleaanse circuits, behalve dat ze werken over K in plaats van over $\mathbb{Z}_2$.[1] We hebben twee soorten poorten nodig: twee-input-poorten voor optelling en vermenigvuldiging, en één-input-poorten voor optelling en vermenigvuldiging met een constante. Bovendien nemen we aan dat het

[1] Dit is geen beperking: als we een vaste K kiezen, is elke efficiënt berekenbare functie ook berekenbaar door een algebraïsch circuit van polynomiale omvang over K.

aantal kwaadaardige spelers $t-1 < \frac{n}{2}$. De reden hiervoor zal even verder duidelijk worden.

Zoals reeds vermeld voorziet elke deelnemer alle partijen van een schaduw van zijn geheime invoer. Uiteindelijk heeft elke deelnemer een schaduw van elke invoer. Het tweede stadium van het protocol verloopt weer op de poort-per-poort manier. We onderscheiden de vier verschillende types van poorten. De realisatie van de eerste drie is gemakkelijk in te zien als we de interpolatie van veeltermen in het achterhoofd houden.

- Optellen van geheim s en een constante c. Stel dat s verdeeld is door een veelterm $f(X)$ zodat $f(0) = s$. Als alle spelers c optellen bij hun schaduw s_i, hebben ze allemaal een schaduw van $s + c$ want de interpolerende veelterm tussen de nieuwe punten is $f(X) + c$.

- Vermenigvuldigen van geheim s met constante c. Alle spelers vermenigvuldigen hun schaduw s_i met c. Zo bekomen ze een schaduw van $s \cdot c$.

- Optellen van twee geheimen s en s'. Alle spelers berekenen $s_i + s'_i$, hetgeen een correcte schaduw is van $s + s'$.

- Vermenigvuldigen van twee geheimen s en s'. Het kon niet blijven duren, er moest toch één poort roet in het eten gooien. Stel dat s en s' verdeeld zijn door de veeltermen $f(X)$ en $g(X)$, beide van graad $t-1$. Als alle spelers hun schaduwen vermenigvuldigen tot $s_i \cdot s'_i = f(i) \cdot g(i)$, hebben ze allemaal een punt van de veelterm $(f \cdot g)(X)$. Deze veelterm voldoet wel aan $(f \cdot g)(0) = s \cdot s'$, maar is niet meer van de juiste graad: hij is van graad $2t-2$. Om deze veelterm eenduidig vast te leggen, en dus om het geheim $s \cdot s'$ te onthullen, zijn er $2t-1$ punten nodig in plaats van t. Indien $2t-1 > n$ hebben alle deelnemers samen niet eens voldoende gegevens om $(f \cdot g)(X)$ te reconstrueren. Vandaar dat aan de voorwaarde $t-1 < \frac{n}{2}$ moet voldaan zijn.

  Nu komt het moeilijkste punt. Er bestaat een vaste lineaire combinatie, waarvan iedereen de coëfficiënten $r_1, \ldots, r_n \in K$ kan berekenen, van de "product-schaduwen" $s_i \cdot s'_i$ die $s \cdot s'$ oplevert. De interpolatieformule van Lagrange zegt ons immers dat

$$(f \cdot g)(0) = \sum_{1 \leq i \leq n} \left(\frac{\prod_{1 \leq j \leq n, j \neq i} (-i)}{\prod_{1 \leq j \leq n, j \neq i} (i-j)} \right) \cdot s_i s'_i$$

  en dus is

$$r_i = \frac{\prod_{1 \leq j \leq n, j \neq i} (-i)}{\prod_{1 \leq j \leq n, j \neq i} (i-j)}$$

  Iedereen kan inderdaad alle r_i's berekenen, want ze hangen enkel af van i.

Het is belangrijk dat minstens t van deze waarden verschillend van nul zijn, anders is een groep van $t-1$ of minder deelnemers in staat $s \cdot s'$ te berekenen. Dit vormt geen probleem, want stel zonder verlies aan algemeenheid dat $r_t = \ldots = r_n = 0$. Er geldt dan dat bijvoorbeeld $(f \cdot 1)(0) = s = \sum_{1 \leq i \leq t-1} r_i(s_i \cdot 1)$. Dit zou betekenen dat spelers $1, \ldots, (t-1)$ het geheimverdelingsschema van Shamir kunnen breken, hetgeen een contradictie is want de drempel om het geheim s te kunnen onthullen ligt op t deelnemers.

De spelers gaan nu samenwerken om de graad van de veelterm terug tot $t-1$ te herleiden. Eerst herverdelen ze hun product-schaduwen: elke speler i verdeelt zijn $s_i s'_i$ onder de andere partijen door een willekeurige veelterm $h_i(X)$ van graad $t-1$ te kiezen zodat $h_i(0) = s_i s'_i$ en partij j te voorzien van een schaduw $u_{ij} = h_i(j)$. Beschouw nu de veelterm

$$h(X) = \sum_{1 \leq i \leq n} r_i \cdot h_i(X)$$

Deze veelterm is van graad $< t$ en bovendien is $h(0) = \sum_{1 \leq i \leq n} r_i \cdot s_i s'_i = s \cdot s'$. Als elke speler i nu

$$v_i = \sum_{1 \leq j \leq n} r_j u_{ji}$$

berekent, heeft iedereen een schaduw v_i uit de veelterm $h(X)$ van graad kleiner dan t zodat $h(0) = s \cdot s'$. En dat is net wat we wilden bereiken.

De deelnemers propageren op deze manier doorheen het volledige circuit totdat ze schaduwen van de uitvoer hebben. Ze maken hun schaduwen bekend aan de andere partijen, berekenen de interpolerende veelterm $f(X)$ en halen hier de uitvoer $s = f(0)$ uit.

We mogen ervan uitgaan dat het geheimverdelingsschema van Shamir veilig is. Dan is de beginfase van het protocol veilig en is ook de berekening voor de eerste drie poorten veilig, want de deelnemers voeren enkel berekeningen uit op gegevens die ze reeds hadden. Voor een twee-input vermenigvuldigingspoort worden daarentegen wel extra gegevens uitgewisseld. We moeten erop toezien dat op geen enkel ogenblik een groep van minder dan t spelers een tussenresultaat kan onthullen. Dit is inderdaad voldaan: stel dat $t-1$ spelers samenzweren om $h(X)$ te reconstrueren. De veelterm h bevat echter minstens één h_i die gekozen is door iemand buiten de samenzwering, want we hebben reeds aangetoond dat minstens t van de $r_1, \ldots, r_n$ verschillend van nul zijn.

Merk op dat we in dit hele protocol geen encryptie, priemgetal of kwadratisch residu gezien hebben. Het enige bouwblok dat we nodig hebben gehad is het geheimverdelingsschema van Shamir. We weten dat het geheimverdelingsschema van Shamir informatietheoretisch veilig is als er minder dan t

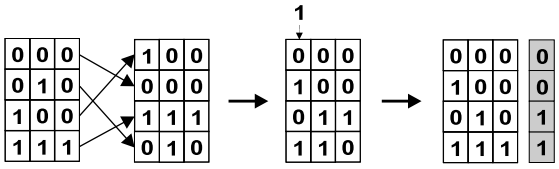
semi-eerlijke tegenstanders zijn. Daaruit kunnen we besluiten dat het volledige protocol informatietheoretisch veilig is als maximaal $t - 1 < \frac{n}{2}$ van de spelers semi-eerlijke tegenstanders zijn. Bovendien kunnen we aantonen dat deze grens strikt is, d.w.z. dat er geen protocol bestaat dat informatietheoretische veiligheid biedt voor een groter aantal semi-eerlijke tegenstanders. Stel immers dat er een protocol voor n partijen bestaat dat informatietheoretisch veilig blijft als meer dan een strikte minderheid van de spelers semi-eerlijk samenzweren, dus dat het aantal tegenstanders minstens $\frac{n}{2}$ bedraagt. Dit zou impliceren dat er een protocol bestaat om een AND-poort informatietheoretisch veilig te evalueren (laat elk van de twee spelers gewoon een andere helft van de n deelnemers simuleren). In sectie 2.2 hebben we echter bewezen dat dit onmogelijk is, en vandaar is de grens $t - 1 < \frac{n}{2}$ sermi-eerlijke tegenstanders optimaal. In [13] wordt met behulp van veel moeilijkere protocols aangetoond dat de optimale grens voor kwaadaardige tegenstanders op $k < \frac{n}{3}$ ligt.

Vanuit theoretisch oogpunt is dit een belangrijk resultaat. Mocht er morgen een ruimteschip landen van een planeet bij Alpha Centauri met superintelligente wezens die over oneindig veel meer rekenkracht beschikken dan wij eenvoudige aardbewoners (bijvoorbeeld met behulp van quantumcomputers), dan is het toch mogelijk met hen een veilig protocol voor Secure Distributed Computing aan te gaan, zolang we erop kunnen vertrouwen dat hoogstens een strikte minderheid van de spelers vals speelt. We stellen het nu een beetje absurd voor, maar er wordt over de hele wereld onderzoek gedaan naar quantumcomputers. Het is niet ondenkbaar dat binnen enkele jaren de resultaten van dit onderzoek bruikbaar zijn voor het kraken van cryptografische algoritmes zodat informatietheoretische veiligheid sterk aan belangrijkheid zal winnen.

7.2 Chaum, Damgård en van de Graaf

Dit protocol is in 1987 gepubliceerd door David Chaum, Ivan Damgård en Jeroen van de Graaf ([11]). In grote lijnen werkt het als volgt: de partijen vertrekken van de volledige, niet-geëncrypteerde waarheidstabel van de te evalueren functie. Om beurten transformeren de verschillende partijen de waarheidstabel op een zodanige manier dat de waarde van hun eigen input bits op een bepaalde rij verborgen blijft voor de anderen. Met behulp van bit commitments en een zero knowledge proof overtuigen ze de anderen ervan dat de transformatie correct is uitgevoerd. Als de laatste partij dit gedaan heeft openen ze de uitvoer van precies die rij van de waarheidstabel die met hun geheime invoer overeen komt.

We gaan de werking van dit protocol bewust op een ietwat vreemde maar meer inzichtelijke manier aanbrengen. We volgen stap voor stap een gedachtegang die Chaum, Damgård en van de Graaf zelf misschien hebben



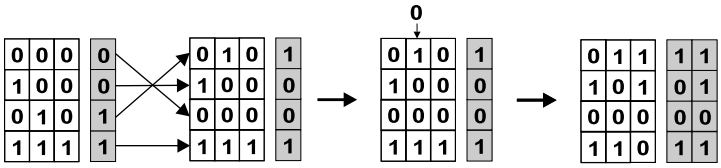
Figuur 7.1: Transformatie door Alice

doorlopen om tot dit protocol te komen. Deze aanpak is wiskundig gezien minder rigoureus maar geeft een grondiger inzicht in het werkingsprincipe en de veiligheid van het protocol. Voor de formele wiskunde erachter verwijzen we naar de oorspronkelijke paper [11], voor een implementatie-gerichte voorstelling kan de lezer best [17] raadplegen.

Laten we vertrekken van het eenvoudige voorbeeld waarin Alice en Bob hun liefde voor mekaar willen betuigen (of net niet) zonder risico op gezichtsverlies, oftewel de logische AND willen berekenen van hun respectievelijke input bits a en b. Alice vertrekt van de waarheidstabel van een AND-poort. Ze spreken af dat de eerste kolom overeenkomt met Alices geheime bit a en de tweede kolom met Bobs geheime bit b. Als Alice de volgorde van de rijen van de waarheidstabel grondig door mekaar schudt en vervolgens de kolom van haar invoer XOR't met een willekeurige bit c_1 weet Bob niet meer welke rij overeen komt met welke invoer van Alice, want in een waarheidstabel komen in één kolom steeds even veel nullen als enen voor. Alhoewel ... de uitvoerkolom is nog onaangeroerd: Bob kan zien welke rij een 1 als uitvoer levert, weet dat de invoer van Alice op die rij een 1 moet zijn en kan gemakkelijk c_1 bepalen. Als Alice ook de uitvoerkolom XOR't met een willekeurige bit komen we geen stap verder: de assymmetrie tussen het aantal nullen en enen in de uitvoerkolom verraadt de encryptiebit. Een betere oplossing is dat Alice elke bit in de uitvoerkolom met een aparte sleutelbit r_l ($l = 1 \ldots 4$) XOR't en een commitment voor r_l aan rij l van de waarheidstabel hangt (zodat Alice niet kan liegen over de waarde van r_l bij het openen van de uitvoer). Alice zendt de getransformeerde waarheidstabel met de bit commitments naar Bob. Dit alles wordt verduidelijkt in Figuur 7.1. In deze en alle volgende figuren duiden we een bit commitment aan door de bit op een grijze achtergrond te plaatsen.

Nu is het Bobs beurt om de waarheidstabel die hij net van Alice gekregen heeft op een analoge wijze te transformeren. Hij begint met een permutatie van de rijen van de tabel, waarbij de commitments mee permuteren. Bob kiest een willekeurige bit c_2 en XOR't de kolom die overeenkomt met zijn invoer b hiermee. Hij kiest ook vier willekeurige bits r_l, $l = 1 \ldots 4$, XOR't de bit op rij l van de uitvoerkolom met r_l, hangt een commitment voor r_l aan rij l en zendt dit geheel terug naar Alice (zie Figuur 7.2).

We zullen het geduld van Alice en Bob niet langer op de proef stellen. Om eindelijk te weten of ze voor mekaar bestemd zijn gaan Alice en Bob als



Figuur 7.2: Transformatie door Bob

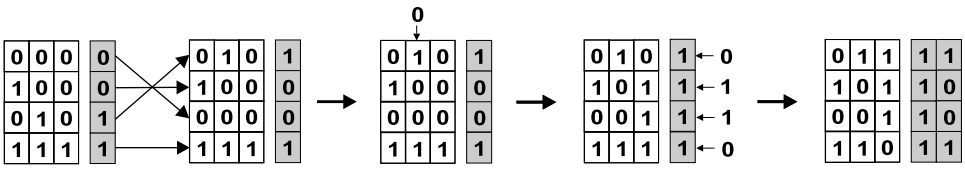
volgt te werk: ze berekenen allebei de geëncrypteerde versie van hun input bit en zenden die naar mekaar. Alice berekent dus $a' = c_1 \oplus a$, Bob berekent $b' = c_2 \oplus b$. Hiermee geven ze mekaar geen informatie over a en b zelf, want c_1 en c_2 zijn willekeurig gekozen. Maar a' en b' bepalen wel eenduidig de juiste rij van de waarheidstabel. Alice en Bob moeten nu enkel nog de commitments die aan de geselecteerde rij hangen openen om de uitvoer te berekenen.

In de figuren heeft Alice $c_1 = 1$ gekozen en heeft Bob $c_2 = 0$ gekozen. Stel dat Bob smoorverliefd is op Alice ($b = 1$) maar Alice Bob helemaal niet ziet zitten ($a = 0$). Alice stuurt dan $c_1 \oplus a = 1 \oplus 0 = 1$ naar Bob en Bob stuurt $c_2 \oplus b = 0 \oplus 1 = 1$ naar Alice. Op de laatste waarheidstabel in Figuur 7.2 zien we dat ze op deze manier de laatste rij selecteren[2]. Alice en Bob openen hun bit commitments van de laatste rij en zien dat de uitvoer van het circuit $0 \oplus 1 \oplus 1 = 0$ is.

Geen happy end voor Bob, hij blijft in de kou staan met zijn mooie gevoelens en kan beginnen denken over de tekst voor een contactadvertentie in de krant. Maar tot overmaat van ramp gaat Alice aan al haar vriendinnen rondbazuinen dat Bob een oogje op haar had. Hoe kan ze dit weten? De bedoeling van het protocol is net dat Alice dit niet te weten kan komen. Naast een harde levensles krijgt Bob van Alice ook nog een lesje in cryptografie. Kijk maar eens goed naar de commitments die Alice aan elke rij heeft gekoppeld: ze verbergen een 0 op de rijen waar de input van Bob een 0 is en een 1 waar de input van Bob een 1 is. De commitments worden mee gepermuteerd met de rijen, dus bij de selectie van één bepaalde rij weet Alice dat de waarde van b gelijk is aan de waarde verborgen door haar eigen bit commitment bij die rij.

Het eigenlijke probleem is dat er informatie lekt over de geheime permutatie van Bob. We kunnen dit verhinderen door een bit commitment schema te gebruiken dat *blindeerbaar* is (zie sectie 4.4). Kort samengevat wil dit zeggen dat Bob een commitment van Alice voor een bit x kan transformeren tot een commitment voor $x \oplus x'$, met x' een door Bob gekozen bit, zonder enige informatie te verkrijgen over x. Door elk bit commitment van Alice te blinderen kan Bob ervoor zorgen dat Alice niet weet wat het oorspronkelijk

[2]Als we de permutaties vanuit de niet geëncrypteerde tabel tot het einde volgen zien we dat dit inderdaad de juiste rij is.



Figuur 7.3: Transformatie door Bob

bit commitment was. Het blijven natuurlijk wel bit commitments die enkel door haar kunnen geopend worden en niet door Bob.

Alice onderneemt net dezelfde stappen als daarnet (zie Figuur 7.1). Bob kiest echter na de permutatie van de rijen en encryptie van zijn invoerkolom voor elke rij l een willekeurige bit s_l om het commitment van Alice mee te blinderen (zie Figuur 7.3). Om de consistentie van de waarheidstabel te bewaren moet hij dan ook de bit in de uitvoerkolom XOR'en met die gekozen bit, anders zal bij de opening in de laatste stap de uitvoer niet correct zijn. Tenslotte kiest Bob opnieuw een willekeurige bit r_l per rij, past daarmee een XOR toe op de uitvoerkolom en hangt een commitment voor r_l aan rij l. De evaluatie van het circuit kan nu op net dezelfde manier gebeuren als daarnet, met dit verschil dat Alice geen achterpoortje meer heeft om b te weten te komen. Door dit protocol te gebruiken had Bob zich heel wat leed en geroddel kunnen besparen.

We moeten wel nog even opmerken dat de permutaties van Alice en Bob zeker niet volledig geheim blijven. Van de tabellen in Figuur 7.1 krijgt Bob enkel de eerste en de laatste onder ogen. Alice heeft de kolom van Bobs input bits tijdens haar transformatie ongemoeid gelaten. Bob weet dus bijvoorbeeld zeker dat de eerste rij (met $a = 0$ en $b = 0$) na de permutatie ofwel op de eerste ofwel op de tweede rij is terecht gekomen en zeker niet op de derde of de vierde. Hij heeft echter geen enkele aanwijzing op welke van die twee rijen precies, voor hem lijken ze allebei even waarschijnlijk. Eén van de twee rijen staat voor de invoer $a = 0$ en $b = 0$, de andere rij staat voor $a = 1$ en $b = 0$. Als hij de exacte rij wel kon aanduiden zou hij na selectie van de gepaste rij in de laatste stap van het protocol de invoer van Alice kunnen achterhalen. Bobs absolute onzekerheid over welke van de twee rijen de juiste is, is een weerspiegeling van zijn absolute onzekerheid over de waarde van a. Er lekt dus inderdaad informatie over de permutaties, maar er lekt slechts zo veel als er mag lekken.

Door van een andere waarheidstabel te vertrekken kunnen we eender welke binaire poort op bovenstaande manier evalueren. Het is vrijwel triviaal in te zien dat we dit protocol ook algemener voor de evaluatie van een t-aire poort kunnen gebruiken. De waarheidstabel heeft dan een omvang van 2^t rijen op $t + 1$ kolommen. Alice en Bob voeren net dezelfde bewerkingen uit; ze hebben nu wel meerdere input bits en dus meerdere kolommen die ze

moeten encrypteren met een willekeurige bit.

Ook het toelaten van meerdere partijen is niet moeilijk. Partij i neemt de getransformeerde tabel van partij $(i-1)$ als vertrekpunt en voert net dezelfde bewerkingen uit als Bob: de rijen permuteren, de eigen invoerkolommen encrypteren, de bestaande bit commitments blinderen, de eigen bit commitments toevoegen en dit resultaat doorgeven aan partij $(i+1)$. Partij i moet wel *alle* bit commitments blinderen, niet enkel die van partij $(i-1)$, anders lekt er opnieuw te veel informatie over de permutatie.

In theorie kunnen we nu al elke booleaanse functie evalueren, maar het is duidelijk dat de kost van de doorgestuurde data al snel veel te groot wordt: voor een circuit met t inputs moet er telkens een waarheidstabel doorgestuurd worden waarvan de omvang van de grootte-orde $O(t \cdot 2^t)$ is. Het zou veel efficiënter zijn voor elke poort in het circuit een afzonderlijke waarheidstabel op te stellen en die tabellen op één of andere manier met mekaar te "verbinden". We reduceren dan de kost van het protocol tot iets dat lineair is in het aantal poorten van het circuit.

Een voor de hand liggende manier om dit te realizeren is elke waarheidstabel te behandelen zoals we daarnet deden en als laatste stap de gepaste rij te selecteren in elke tabel, de uitvoerbit te openen en die dan weer te gebruiken als invoerbit in de tabel van de volgende poort. Het is echter meteen duidelijk dat deze oplossing niet veilig is: alle tussenresultaten van de functie-evaluatie worden te grabbel gegooid.

We zoeken eigenlijk een methode om de tussenresultaten geheim te houden maar ze toch te kunnen gebruiken als invoer voor de volgende poort. We verwezenlijken dit door de corresponderende uitvoer- en invoerkolom te XOR'en met dezelfde willekeurige bit. Van al deze willekeurige bits worden enkel de bits die de uitvoer van het volledige circuit beschermen in de laatste stap van het protocol geopend, de andere bits worden niet vrijgegeven zodat de tussenresultaten beschermd blijven.

Om het overzicht niet te verliezen zullen we nu een stap-voor-stap overzicht geven van het volledige protocol dat partij i moet volgen om de waarheidstabellen die hij van partij $(i-1)$ krijgt te transformeren. Eén aspect van stap 3 hebben we nog niet toegelicht: elke partij moet voor de willekeurige bits die hij gebruikt om kolommen te encrypteren een commitment aan die kolom hangen. Deze commitments zullen we nodig hebben in een zero knowledge proof, maar daarop komen we later terug. Met deze commitments erbij bestaat een getransformeerde waarheidstabel uit een tabel met 2^t rijen en $t+1$ kolommen, met aan elke rij en elke kolom een — eventueel lege — lijst bit commitments.

Partij i voert de volgende stappen uit op elke waarheidstabel T die partij $(i-1)$ hem gegeven heeft:

1. Kies een willekeurige permutatie σ en pas die toe op de rijen van de tabel T. De lijsten van commitments die aan de rijen hangen moeten

mee gepermuteerd worden. Noem de tabel na permutatie T'.

2. Aan iedere rij l van T' hangt een lijst van $(i-1)$ bit commitments. Kies willekeurige bits $s_l^{(j)}$, met $j = 1 \ldots i-1$. Pas een XOR met al deze bits toe op de bit in de uitvoerkolom van rij l en blindeer het j e bit commitment in de lijst met $s_l^{(j)}$.

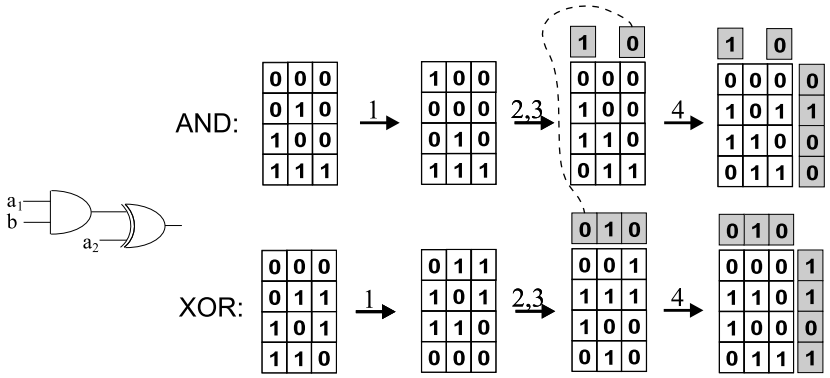
3. Ga voor elke kolom k van T' de volgende stappen na. Indien kolom k overeenkomt met een invoerbit van een andere partij, doe niets. Indien kolom k in het circuit verbonden is met een kolom waarvoor reeds een encryptiebit b gekozen is (voor het "verbinden" van poorten), stel dan $c_k = b$. In alle andere gevallen, kies een willekeurige bit c_k. Pas een XOR met c_k toe op de volledige kolom en voeg een bit commitment voor c_k toe aan de kolomlijst.

4. Kies voor elke rij l van T' een willekeurige bit r_l, XOR de bit in de uitvoerkolom van rij l met r_l en voeg een bit commitment voor r_l toe aan de rijlijst.

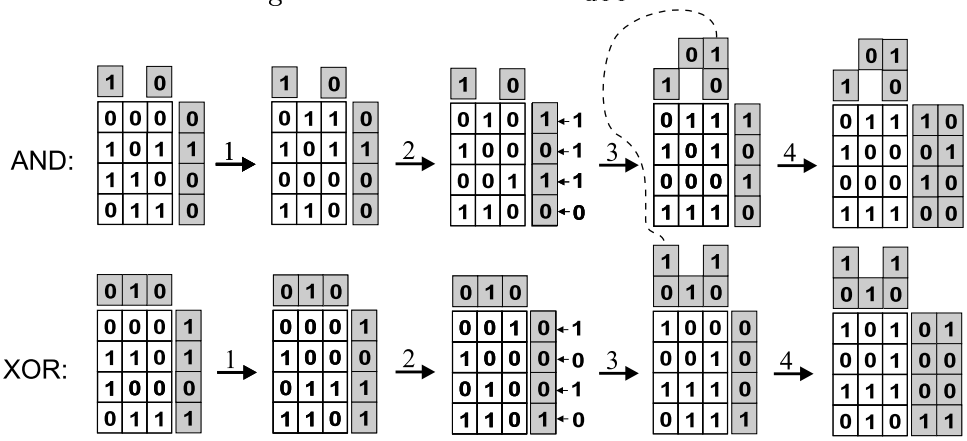
In Figuur 7.4 en Figuur 7.5 illustreren we het volledige protocol. Alice heeft geheime invoerbits a_1 en a_2 en Bob heeft een geheime invoerbit b. Samen willen ze op een veilige manier de functie $(a_1 \wedge b) \oplus a2$ berekenen. De bits die gelijk moeten zijn omdat ze de verbinding tussen twee poorten verzorgen zijn met een stippellijn verbonden.

Stel dat $a_1 = 1$, $b = 0$ en $a_2 = 1$. De evaluatie wordt uitgevoerd op de twee rechtertabellen van Figuur 7.5. We evalueren eerst de AND-poort. Alice maakt haar geëncrypteerde input $a_1 \oplus 1 = 0$ bekend, Bob maakt $b \oplus 0 = 0$ bekend. De derde rij komt met deze inputs overeen. Alice en Bob onthullen hun bit commitments die aan de derde rij hangen en berekenen de XOR met de uitvoerbit: $0 \oplus 1 \oplus 0 = 1$. Deze bit doet in de tabel van de XOR-poort dienst als invoerbit, samen met Alice haar tweede geëncrypteerde input $a_2 \oplus 1 = 0$. Daar selecteren we dus de eerste rij, en omdat de uitvoerbit van deze poort tevens de uitvoer van het volledige circuit is openen Alice en Bob naast hun commitments van de eerste rij ook hun commitments van de kolom. Ze berekenen dan de uitvoer als $1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$, hetgeen inderdaad de juiste uitvoer is. Merk op dat noch Alice noch Bob het tussenresultaat kunnen achterhalen omdat ze het allebei beschermen met een willekeurige bit.

Het protocol zoals tot dusver beschreven werkt enkel in het semi-eerlijke model, d.w.z. als de spelers zich netjes aan de regels van het protocol houden. Als we met kwaadaardige tegenstanders te kampen hebben faalt het. In Figuur 7.5 kan Bob bijvoorbeeld de uitvoerkolom van de XOR-poort encrypteren met een 1 en toch een commitment voor een 0 aan de kolomlijst hangen. De uitvoer van het protocol zal dan het inverse zijn van de echte



Figuur 7.4: Transformatie door Alice



Figuur 7.5: Transformatie door Bob

uitvoer van het circuit, maar enkel Bob weet dit. Hij kan Alice dus verhinderen het juiste resultaat te weten te komen en tegelijkertijd zelf wel de exacte uitkomst kennen.

We volgen een tactiek die vaak gehanteerd wordt om een protocol dat bestand is tegen semi-eerlijke tegenstanders ook te wapenen tegen kwaadaardige tegenstanders. Het principe is dat het protocol ongewijzigd blijft, maar de deelnemende partijen moeten met behulp van een zero knowledge proof (zie sectie 4.6) bewijzen dat ze de regels van het protocol gevolgd hebben. Chaum, Damgård en van de Graaf stellen een zero knowledge proof voor dat erop steunt dat het gebruikte bit commitment schema *vergelijkbaar* is (zie sectie 4.4). Een bit commitment schema dat aan deze eigenschap voldoet laat toe dat een speler, die twee bit commitments heeft gemaakt voor bits x en y, de waarde van $x \oplus y$ te bewijzen aan een andere speler zonder verdere informatie weg te geven over x en y.

Stel dat speler i elke waarheidstabel S in het circuit heeft omgezet naar een waarheidstabel T. Hij construeert dan T', een nieuwe transformatie

van S, met onafhankelijke keuzes voor de permutatie en alle willekeurige bits, en maakt T' publiek. De $n-1$ andere spelers komen gezamelijk een willekeurige bit b overeen, bijvoorbeeld met behulp van een protocol zoals gegeven in sectie 4.4.
Als $b = 0$ moet speler i voor elke T'

- de permutatie σ' die hij gebruikt heeft in de constructie van T' bekend maken
- al de bit commitments die hij aan T' heeft gehangen openen en alle blindeerfactoren onthullen.

Iedereen kan nu nagaan of T' correct geconstrueerd was uit S. Als $b = 1$ moet speler i een bijzonder verband aantonen tussen elk paar (T, T') in het circuit:[3]

- Speler i maakt eerst de permutatie $\pi = \sigma' \circ \sigma^{-1}$ bekend. Als l een bepaalde rij is van tabel T, dan is $\pi(l)$ de overeenkomende rij in tabel T'.
- Als speler i een eigen bit commitment aan kolom k gehangen heeft moet hij, met behulp van de vergelijkbaarheidseigenschap, de waarde van $c_k \oplus c'_k$ aantonen.
- Voor elke rij l van de waarheidstabel moet hij op dezelfde manier de waarde van $r_l \oplus r'_{\pi(l)}$ aantonen.
- En tenslotte moet speler i voor elke rij l en elke voorgaande speler $j < i$ met behulp van de blindeerbaarheid de waarde $s_l^{(j)} \oplus s'^{(j)}_{\pi(l)}$ aantonen.

We definiëren $K(k, T)$ als de XOR van de bits in de lijst bit commitments die aan kolom k van T hangt. Op dezelfde manier noteren we de XOR van de bits in de lijst bit commitments die aan rij l van T hangt als $R(l, T)$. Dan kunnen alle spelers aan de hand van de waarden die speler i net bekend heeft gemaakt voor alle kolommen behalve de laatste controleren dat

$$T_{lk} = T'_{\pi(l)k} \oplus K(k, T) \oplus K(k, T')$$

en voor de uitvoerkolom van de tabel nagaan dat

$$T_{lk} = T'_{\pi(l)k} \oplus K(k, T) \oplus K(k, T') \oplus R(l, T) \oplus R(l, T')$$

Het zero knowledge proof is een typisch *cut-and-choose* protocol. Als $b = 0$ wordt speler i uitgedaagd te bewijzen dat T' correct geconstrueerd

[3] In de volgende stappen bedoelen we met een symbool zonder accent de gekozen waarde in de constructie van T, een symbool met accent staat voor de waarde gekozen in de constructie van T'.

is. Als $b = 1$ wordt hij uitgedaagd te bewijzen dat T correct geconstrueerd is als en slechts als T' correct geconstrueerd is. Stel dat speler i probeert vals te spelen en T foutief geconstrueerd heeft. Als $b = 0$ kan hij zijn geloofwaardigheid staande houden door een T' op tafel te gooien die wel juist geconstrueerd is, als $b = 1$ moet hij daartoe een T' bekend maken die net dezelfde fout bevat als T. Het is echter onmogelijk een T' te construeren die aan beide eisen voldoet, en daar hij de waarde van b niet op voorhand kent heeft hij één kans op twee dat de fraude niet opgemerkt wordt. Als we dit controleprotocol m keer herhalen slinken zijn kansen drastisch tot een exponentieel kleine kans van één op 2^m, zodat het praktisch onmogelijk is onopgemerkt vals te spelen.

Tot slot richten we nog even de aandacht op een opmerkelijke eigenschap van dit protocol. Het is namelijk zo dat dit protocol opgewassen is tegen *elke* mogelijke samenzwering. Zelfs als $n-1$ van de n partijen samenzweren kunnen ze de geheime invoer van de n^{e} partij niet achterhalen en kunnen ze hem evenmin ongemerkt met een verkeerd functieresultaat opzadelen. Het is zelfs zo dat één deelnemer zijn geheimen *onvoorwaardelijk* kan verbergen, d.w.z. dat een deelnemer die beperkt is in rekenkracht op een veilige manier aan Secure Distributed Computing kan doen met een deelnemer die over onbeperkte rekenkracht beschikt. De deelnemer met beperkte rekenkracht moet dan wel een bit commitment gebruiken dat op zich totaal geen informatie geeft over de verborgen bit, maar niet in polynomiale tijd te ontkennen is. Het bit commitment schema gebaseerd op kwadratische residu's is hiertoe dus niet geschikt.

Deze eigenschap gaat niet in tegen de bewering uit de vorige sectie dat een protocol voor meerdere partijen maximaal $\frac{n}{3}$ kwaadaardige partijen kan tolereren, want daar hadden we het over informatietheoretische veiligheid, waarbij *alle* partijen verondersteld worden over onbeperkte rekenkracht te beschikken. Dit protocol is niet informatietheoretisch veilig want een bit commitment tussen twee partijen die allebei over onbegrensde rekenkracht beschikken is onmogelijk. Het is echter wel cryptografisch veilig.

7.3 Franklin en Haber

In [18] presenteren Matt Franklin en Stuart Haber een Secure Distributed Computing protocol voor meerdere partijen gebaseerd op groep-georiënteerde publieke-sleutel cryptografie. In een groep-georiënteerd cryptosysteem kan eender welke groep spelers deelnemen aan een encryptie. De publieke sleutels van de deelnemers zijn nodig om een boodschap te encrypteren, en de medewerking van *alle* deelnemers is vereist om te kunnen decrypteren.

Een groep-georiënteerd cryptosysteem voor n partijen $1 \ldots n$ is bepaald door een publieke sleutel K_{pub}, n private sleutels $K_1, \ldots, K_n$, een verzameling encryptiefuncties $\{E_S : S \subseteq \{1 \ldots n\}\}$ en een verzameling decryptie-

functies $\{D_i : i \in \{1 \ldots n\}\}$. Het verband tussen deze functies voor een boodschap M is gegeven door

$$D_i(E_S(M)) = E_{S \setminus \{i\}}(M)$$

Het moet gemakkelijk zijn E_S te berekenen als de publieke sleutel K_{pub} gekend is, maar het moet moeilijk zijn D_i te berekenen zonder de private sleutel K_i. Tenslotte moet het eenvoudig zijn M te berekenen uit $E_\emptyset(M)$. Het meest voorkomende geval is natuurlijk $E_\emptyset(M) = M$.

Het idee van een groep-georiënteerd cryptosysteem is aanschouwelijk voor te stellen als een kist waar de boodschap M in gestoken wordt. Ieder lid van de groep heeft een sleutel en een heleboel sloten waar die sleutel op past. De geopende sloten delen ze aan mekaar uit, hun sleutel houden ze altijd zelf bij. We kunnen een boodschap encrypteren voor bepaalde deelnemers van de groep door de boodschap in de kist te steken en de kist af te sluiten met een slot van elke deelnemer. Een deelnemer kan op eender welk moment zijn slot van de kist openen. De sloten moeten niet in een bepaalde volgorde geopend worden, maar de kist kan pas worden geopend als de eigenaar van het laatste slot zijn slot opent.

De aanpak van dit protocol is vergelijkbaar met die van Abadi en Feigenbaum (zie sectie 6.3) in die zin dat het protocol ook een probabilistisch homomorf encryptiesysteem gebruikt om de invoer te encrypteren en poort per poort doorheen het circuit te propageren. Ook hier gebeurt de evaluatie van een NOT- of XOR-poort door het homomorfisme toe te passen en is voor de evaluatie van een AND-poort een zekere interactie met de andere partijen nodig. Het grote verschil tussen de twee protocols is dat dat van Abadi en Feigenbaum voor twee partijen is ontworpen, terwijl het volgende protocol ook voor meerdere partijen werkt.

Doorheen hun volledige paper hanteren Franklin en Haber een vrij ingewikkeld encryptiesysteem dat min of meer een kruising is tussen kwadratische residu's en ElGamal encryptie. Een belangrijk nadeel van dit systeem is dat alle bewerkingen gebeuren modulo een samengestelde modulus N waarvan *niemand* de ontbinding mag kennen. Er is een centrale server nodig om een geschikte modulus te kiezen en om elke partij van een private sleutel te voorzien. In een kleine opmerking vermelden ze ook nog een ander encryptiesysteem dat kan gesubstitueerd worden in het protocol. Om wat meerwaarde te geven ten opzichte van de oorspronkelijke paper zullen we het protocol bespreken met behulp van dit alternatieve encryptiesysteem. Dit encrypteert een bit b als een tuple

$$E_S(b) = \left[g^r \mod N,\ (-1)^b \left(\prod_{j \in S} g^{x_j} \right)^r \mod N \right]$$

met $r \in_R Z_N$. De publieke sleutel is $[N, g, g^{x_1} \mod N, \ldots, g^{x_n} \mod N]$, de private sleutel van partij i is x_i. De decryptie van partij i wordt gegeven

door

$$D_i([\alpha, \beta]) = [\alpha,\ \beta\alpha^{-x_i} \mod N]$$

De auteurs zwijgen echter in alle talen over de voorwaarden waaraan N, g, x en r moeten voldoen opdat dit systeem veilig is. Het is verleidelijk omwille van de analogie met ElGamal encryptie te oordelen dat het systeem veilig is als N priem is en g een generator is van $\mathbb{Z}_N^*$. Deze overhaaste conclusie mogen we zeker niet trekken, want als N een priemgetal is congruent met 3 modulo 4 (en algemener als $\mathbf{J_N}(-1) = -1$) is het systeem zelfs bewijsbaar onveilig. Inderdaad, als $[\alpha,\ \beta]$ een geëncrypteerde bit is, leert $\mathbf{J_N}(\alpha)$ ons volgens definitie 3.2 en stelling 3.1 of r even of oneven is. Uit de publieke sleutels kunnen we op dezelfde manier afleiden of de x_i's even of oneven zijn, en dus ook of $x = \sum_{i=1}^{n} x_i$ even of oneven is. Als $N \equiv 3 \mod 4$ geldt altijd dat $\mathbf{J_N}(-1) = -1$. Het is eenvoudig na te gaan dat als x of r even is $\mathbf{J_N}(\beta)$ de waarde van de geëncrypteerde bit b verraadt: als $\mathbf{J_N}(\beta) = 1$ is $b = 0$, als $\mathbf{J_N}(\beta) = -1$ is $b = 1$. Als x en r beide oneven zijn geldt net het omgekeerde.

De oorzaak van het lek is dat deze versie van ElGamal niet semantisch veilig is. In sectie 4.2.3 hebben we een lichtjes andere versie besproken die wel bewijsbaar semantisch veilig is. Deze versie vereist dat $N = aq + 1$ waarbij N en q priemgetallen zijn en a een klein even getal is, dat g een generator is van een deelgroep $\mathbb{G}_q$ van orde q en dat de te encrypteren boodschap een element is van G_q. Semantische veiligheid is perfect wat we nodig hebben, maar ons encryptiesysteem kan nooit aan die voorwaarden voldoen. Stel dat h een generator is van $\mathbb{Z}_N^*$ (herinner dat g een generator is van de deelgroep $\mathbb{G}_q$, niet van $\mathbb{Z}_N^*$ zelf). De deelgroep $\mathbb{G}_q$ bestaat uit de elementen $\{h^a, h^{2a}, \ldots, h^{q(a-1)}, h^{qa} = 1\}$. Een element h^y van Z_N^* is tevens een element van G_q als en slechts als y een veelvoud is van a. De te encrypteren boodschap M is 1 als $b = 0$ en is -1 als $b = 1$. Het getal 1 is het neutraal element en zit bijgevolg in elke deelgroep, ook in G_q. Voor -1 ligt dat anders. In het bewijs van stelling 3.2 hebben we aangetoond dat $-1 \equiv h^{\frac{N-1}{2}} = h^{\frac{aq}{2}} \mod N$. De voorwaarde dat $-1 \in G_q$ is voldaan als en slechts als $\frac{aq}{2}$ een veelvoud is van a. Daar q een oneven priemgetal is zou dit betekenen dat $\frac{a}{2}$ een veelvoud is van a, en dat is natuurlijk onmogelijk.

We hebben nu bewezen dat het veiligheidsbewijs van [36] niet van toepassing is, maar dat wil nog niet zeggen dat het systeem onveilig is voor $N = aq + 1$. Het enige waarvan we zeker zijn is dat het systeem onveilig is als N priem is en $N \equiv 3 \mod 4$. Voor alle andere gevallen hebben we de veiligheid bewezen noch tegengesproken. Een contact met Franklin en Haber zelf kon hieromtrent niet meer duidelijkheid scheppen. Het vinden van een bewijs voor de semantische veiligheid of onveiligheid van het systeem is voorlopig een open probleem en kan het onderwerp vormen van verder onderzoek.

Dit encryptiesysteem voldoet aan enkele eigenschappen waar we in het

protocol op zullen steunen.

Eigenschap (Compact). *De grootte van een geëncrypteerde bit is constant, namelijk twee elementen van Z_N^*, onafhankelijk van het aantal deelnemers aan de encryptie.*

Eigenschap (XOR-homomorf). *Eender wie kan gemakkelijk een encryptie berekenen van de XOR van twee geëncrypteerde bits.*

Bewijs. Inderdaad, als

$$E_S(b) = [\alpha,\ \beta] = \left[g^r \mod N,\ (-1)^b \left(\prod_{j\in S} g^{x_j}\right)^r \mod N\right]$$

en

$$E_S(b') = [\alpha',\ \beta'] = \left[g^{r'} \mod N,\ (-1)^{b'} \left(\prod_{j\in S} g^{x_j}\right)^{r'} \mod N\right]$$

dan is

$$[\alpha\alpha',\ \beta\beta'] = \left[g^{r+r'} \mod N,\ (-1)^{b\oplus b'} \left(\prod_{j\in S} g^{x_j}\right)^{r+r'} \mod N\right]$$

een geldige encryptie van $b \oplus b'$. □

Eigenschap (Vermombaar). [4] *Eender wie kan vanuit een gegeven groepsencryptie gemakkelijk een willekeurig tuple maken dat een geldige encryptie is voor dezelfde bit, zonder daarbij de waarde van de bit te weten te komen.*

Bewijs. Als $E_S(b) = [\alpha,\ \beta]$ en $r \in_R Z_N^*$, dan is

$$V(E_S(b)) = \left[\alpha g^r \mod N,\ \beta \left(\prod_{j\in S} g^{x_j}\right)^r \mod N\right]$$

een geldige groepsencryptie voor dezelfde waarde b. Voor deze bewerking hebben we enkel de publieke sleutels van de deelnemers nodig. We noemen $V(E_S(b)$ een *vermomming* van $E_S(b)$. □

[4] *Vermombaar* is *niet* hetzelfde als wat we in sectie 4.4 *blindeerbaar* genoemd hebben, alhoewel beide eigenschappen in hun oorspronkelijke papers de naam *blindable* hebben meegekregen. Om verwarring te vermijden gebruiken we in deze tekst verschillende vertalingen voor beide begrippen. Het belangrijkste verschil tussen vermombaar en blindeerbaar is dat bij toepassing van vermombaarheid de waarde van de verborgen bit ongewijzigd blijft, terwijl die bij blindeerbaarheid kan veranderen. Vermombaarheid verandert de verpakking zonder de inhoud te bekijken, blindeerbaarheid verandert de inhoud zonder de verpakking open te maken.

Eigenschap (Intrekbaar). *Een deelnemer kan zonder interactie met de andere deelnemers (d.w.z. door een enkele waarde naar de andere deelnemers te sturen) zich terugtrekken uit de encryptie.*

Bewijs. Formeel betekent dit dat er een functie W_i bestaat zodat $D_i(E_S(b))$ gemakkelijk te berekenen is uit $E_S(b)$ en $W_i(E_S(b))$. Inderdaad, als $E_S(b) = [\alpha,\ \beta]$, dan kan $D_i(E_S(b))$ gemakkelijk berekend worden uit

$$W_i([\alpha,\ \beta]) = \alpha^{-x_i} \mod N$$

want $D_i(E_S(b)) = [\alpha,\ \beta\alpha^{-x_i}]$ voor elke $i \in S$. We noemen $W_i(E_S(b))$ de *intrekking* van i voor $E_S(b)$. □

Alvorens met het protocol te beginnen moeten een geschikte N en g afgesproken worden. Iedere partij i kiest een private sleutel x_i zodat de publieke sleutel $[N, g, g^{x_1} \mod N, \ldots, g^{x_n} \mod N]$ kan bekend gemaakt worden. Om het protocol in te zetten zendt elke partij een groepsencryptie voor zijn invoerbits uit[5]. We zullen nu weer tonen hoe elke partij uit encrypties voor de invoerbits van een bepaalde poort een encryptie voor de uitvoer van die poort kan berekenen.

De lezer zal wel al weten wat gedaan bij een XOR-poort: gewoon het XOR-homomorfisme toepassen natuurlijk. Als de invoerbits geëncrypteerd zijn als $[\alpha,\ \beta]$ en $[\alpha',\ \beta']$ nemen we $[\alpha\alpha',\ \beta\beta']$ als uitvoer van de XOR-poort. De uitvoer van een NOT-poort kunnen we op een gelijkaardige manier berekenen door op de invoer het XOR-homomorfisme toe te passen met een vaste groepsencryptie van een één, bijvoorbeeld $[1,\ -1]$.

Opnieuw is het de AND-poort die moeilijk doet. Ons encryptiesysteem voldoet niet aan een AND-homomorfisme, dus zullen we onze fantasie moeten laten werken. Zij $\hat{u} = E(u)$ en $\hat{v} = E(v)$ de encrypties van de invoerbits. Het protocol moet de partijen toelaten een groepsencryptie $\hat{w} = E(u \wedge v)$ te berekenen. Ze doen dit als volgt:

1. Elke partij i kiest twee willekeurige bits b_i en c_i en zendt encrypties $\hat{b}_i = E(b_i)$ en $\hat{c}_i = E(c_i)$ uit.

2. Iedere partij berekent door herhaalde toepassing van het XOR-homomorfisme $\hat{u}' = E(u') = E(u \oplus b_1 \oplus \cdots \oplus b_n)$ en $\hat{v}' = E(v') = E(v \oplus c_1 \oplus \cdots \oplus c_n)$. Elke partij i zendt intrekkingen $W_i(\hat{u}')$ en $W_i(\hat{v}')$ voor $\hat{u}'$ en $\hat{v}'$ uit.

3. Alle partijen kunnen nu $\hat{u}'$ en $\hat{v}'$ decrypteren tot u' en v'. Beschouw u' en v' eens als vervormde versies van u en v. We zoeken nu een verband tussen $w' = u' \wedge v'$ en $w = u \wedge v$ zodat alle partijen uit een encryptie $\hat{w}' = E(w')$ een encryptie $\hat{w} = E(w)$ kunnen berekenen, zonder daarbij

[5] Met *uitzenden* bedoelen we in deze sectie steeds naar alle partijen tegelijk sturen, beter bekend als *broadcast*

iets te weten te komen over w, u of v. Wel, als we gebruik maken van de booleaanse eigenschap dat

$$(a \oplus b) \wedge c = (a \wedge c) \oplus (b \wedge c)$$

kunnen we w' schrijven als

$$\begin{aligned}
w' &= u' \wedge v' \\
&= (u \oplus b_1 \oplus \cdots \oplus b_n) \wedge (v \oplus c_1 \oplus \cdots \oplus c_n) \\
&= (u \wedge v) \quad \oplus \quad (u \wedge c_1) \quad \oplus \cdots \oplus \quad (u \wedge c_n) \\
&\qquad\quad \oplus \quad (b_1 \wedge c_1) \quad \oplus \cdots \oplus \quad (b_n \wedge c_1) \\
&\qquad\quad \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \\
&\qquad\quad \oplus \quad (b_1 \wedge c_n) \quad \oplus \cdots \oplus \quad (b_n \wedge c_n) \\
&\qquad\quad \oplus \quad \underbrace{(b_1 \wedge v)}_{w_1} \quad \oplus \cdots \oplus \quad \underbrace{(b_n \wedge v)}_{w_n} \\
&= (u \wedge v) \oplus w_1 \oplus \cdots \oplus w_n
\end{aligned}$$

Partij i is in staat een encryptie te berekenen van $w_i = (b_i \wedge c_1) \oplus \cdots \oplus (b_i \wedge c_n) \oplus (u \wedge c_i) \oplus (v \wedge b_i)$, ook al kent hij de waarde van c_j niet als $j \neq i$. Hij heeft immers wel encrypties $E(c_j)$ van de andere partijen gekregen. De bits b_i en c_i kent hij natuurlijk wel, want die heeft hij daarnet zelf gekozen. Om $E(b_i \wedge c_j)$ te berekenen, maakt hij onderscheid tussen volgende twee gevallen:

- Als $b_i = 0$, dan is $b_i \wedge c_j = 0$ en dus kan hij voor $E(b_i \wedge c_j)$ een willekeurige encryptie van een nul nemen, bijvoorbeeld $[1, 1]$.
- Als $b_i = 1$, dan is $b_i \wedge c_j = c_j$ en dus is $E(c_j)$ een geldige encryptie voor $E(b_i \wedge c_j)$.

Ook $E(u \wedge c_i)$ en $E(v \wedge b_i)$ kan hij op een analoge manier berekenen. Zo krijgt hij een encryptie voor alle termen van w_i en kan hij met het XOR-homomorfisme $E(w_i)$ berekenen. Deze encryptie vermomt hij nog even (de reden hiervoor zal later duidelijk worden) zodat $\hat{w}_i = V(E(w_i))$. Elke partij zendt de $\hat{w}_i$ die hij berekend heeft uit.

4. Iedere partij kent $w' = u' \wedge v'$ al sinds stap 2 en heeft net encrypties voor alle w_i's gekregen. Vermits nu $w = u \wedge v = w' \oplus w_1 \oplus \cdots \oplus w_n$ kan elke partij, opnieuw dankzij het XOR-homomorfisme, een encryptie $\hat{w} = E(w)$ voor de uitvoer van de AND-poort berekenen.

Als de uitvoer van de laatste poort in het circuit berekend is, kennen alle partijen een groepsencryptie van de uitvoer van het circuit. Alle partijen zenden intrekkingen voor de encrypties van de uitvoerbits uit zodat iedereen de uitvoer kan decrypteren.

Wat de correctheid van het protocol betreft zal het verhaal wel al bekend in de oren klinken: indien alle partijen de regels volgen berekenen ze voor elke draad een groepsencryptie van de bit die op die draad zou staan als de geheime inputs van alle partijen aan het circuit zouden aangelegd worden. Dit geldt ook voor de uitvoerdraden, die in de laatste stap van het protocol gedecrypteerd worden zodat de juiste uitvoer bekend wordt.

Voor een formeel bewijs van de veiligheid van het protocol verwijzen we de lezer naar de oorspronkelijke paper ([18]). Intuïtief kunnen we inzien dat bij de evaluatie van een NOT-poort of een XOR-poort geen interactie optreedt en er dus ook geen informatie kan lekken die voordien niet gekend was. Voor een AND-poort worden u' en v' bekend gemaakt, maar dat is geen probleem zolang de b_i's en c_i's geheim blijven. Die blijven inderdaad geheim, maar we moeten uiterst voorzichtig omspringen met intuïtieve veiligheidsbewijzen daarvoor want het gevaar kan in een klein hoekje schuilen. Stel dat we in stap 3 de vermomming achterwege laten, zodat deelnemer i gewoon $\hat{w}_i = E(w_i)$ uitzendt. Een andere deelnemer k heeft dezelfde encrypties voor de c_j's ($j = 1 \ldots n$) want die zijn in de eerste stap naar alle deelnemers tegelijk uitgezonden. Alle deelnemers hebben ook dezelfde encrypties $\hat{u} = E(u)$ en $\hat{v} = E(v)$ voor de invoerbits. Het enige wat deelnemer k nog in de weg staat om zelf $\hat{w}_i = E(w_i)$ te construeren op exact dezelfde manier als deelnemer i gedaan heeft, is dat hij de waarde van de bits b_i en c_i niet kent. Hij weet echter wel wat de uitkomst van die bewerkingen moet zijn, namelijk $\hat{w}_i$. Als hij alle combinaties voor b_i en c_i afgaat — lang zal het niet duren, het zijn er maar vier — zal hij onmiddellijk zien welke de juiste is: die combinatie die de $\hat{w}_i$ oplevert die deelnemer i uitgezonden heeft. Hij kan hetzelfde doen voor alle andere deelnemers zodat hij b_j en c_j, $j = 1 \ldots n$, te weten komt. En omdat $u = u' \oplus b_1 \oplus \cdots \oplus b_n$ en $v = v' \oplus c_1 \oplus \cdots \oplus c_n$ kan hij dan uit u' en v' de waarde van u en v zelf achterhalen! De vermomming van $E(w_i)$ voegt een vleugje willekeur aan de berekening van $\hat{w}_i$ toe, zodat de andere deelnemers een verwaarloosbaar kleine kans hebben hun eigen berekening zinvol aan $\hat{w}_i$ te kunnen toetsen.

Een belangrijke bijdrage van dit protocol vanuit theoretisch oogpunt is dat het n partijen in staat stelt een booleaans circuit met C poorten veilig te evalueren met slechts $O(nC)$ geëncrypteerde bits aan communicatie. Inderdaad, de evaluatie van een NOT- of XOR-poort vraagt helemaal geen communicatie, en voor de evaluatie van een AND-poort zendt elke partij vier geëncrypteerde bits uit (eigenlijk drie encrypties en twee intrekkingen, maar we tellen intrekkingen als een halve geëncrypteerde bit omdat die ook half zo groot zijn). In totaal geeft dit voor de evaluatie van een AND-poort een complexiteit van $4n$ geëncrypteerde bits. In het slechtste geval, namelijk als alle C poorten in het netwerk AND-poorten zijn, vraagt het protocol een netwerkbelasting van $4nC$ geëncrypteerde bits. Voor een willekeurig circuit kunnen we zeggen dat de netwerkbelasting inderdaad van orde $O(nC)$ is.

Laten we dit eens vergelijken met het protocol van Chaum, Damgård en van de Graaf uit sectie 7.2. In dat protocol verzendt speler i een waarheidstabel van de C poorten in het circuit met $O(i)$ bit commitments eraan naar speler $i+1$. Voor elke poort worden er gedurende het volledige protocol dus $O(1+2+\ldots+n) = O(n^2)$ bit commitments doorgestuurd, oftewel $O(n^2C)$ voor het volledige circuit.

De verbetering van een factor n die Franklin en Haber hebben kunnen doorvoeren is niet zozeer het resultaat van hun protocol op zich. De winst wordt uitsluitend verwezenlijkt door het groep-georiënteerde encryptiesysteem, meer bepaald door de eigenschap ervan die we daarnet compactheid hebben genoemd. Elke speler kan meehelpen in de encryptie van een bit zodat de andere partijen van hem afhankelijk worden voor de decryptie ervan, zonder daarbij de hoeveelheid data van de encryptie te moeten vergroten. Het is zelfs zo dat we ditzelfde groep-georiënteerd encryptiesysteem kunnen substitueren als bit commitment schema in het protocol van Chaum, Damgård en van de Graaf uit sectie 7.2. Het aantal bit commitments aan elke waarheidstabel die speler i doorstuurt naar speler $i+1$ is dan constant in plaats van $O(i)$, zodat de totale complexiteit van dit protocol ook neerkomt op $O(nC)$.

Eerlijkheidshalve moeten we deze lineaire efficiëntie wel nuanceren. Het gaat hier om een lineair aantal *broadcasts*, niet zonder meer om een lineair aantal boodschappen. Op een netwerk waar een broadcast niet meer belasting veroorzaakt dan een gewone boodschap, bijvoorbeeld een Ethernet LAN, is de complexiteit inderdaad lineair. Als een broadcast echter gerealiseerd wordt door naar elke partij een afzonderlijke boodschap te sturen, hervalt de complexiteit tot kwadratisch in het aantal partijen. Het protocol van Chaum e.a. blijft daarentegen genieten van een lineaire complexiteit op voorwaarde dat het in de semi-eerlijke versie wordt gebruikt (d.w.z. zonder de zero-knowledge proofs), want dan moet speler i enkel met speler $i+1$ communiceren en is er geen broadcast nodig.

Hoofdstuk 8

Toepassingen

8.1 Toepassingen met specifieke veiligheidseisen

Stel dat een werkgroep binnen de Verenigde Naties bestaat uit zeven leden die bij beslissingen elk een stem hebben. Elk lid kan ja of nee stemmen. De Verenigde Staten en Rusland hebben echter weer een uitzondering kunnen forceren zodat zij de enige leden zijn die de mogelijkheid hebben een "superstem" S-ja of S-nee uit te brengen. Ze zijn niet verplicht een superstem uit te brengen, ze kunnen ook een gewone stem uitbrengen als ze dat verkiezen. Als niemand een superstem uitbrengt beslist de meerderheid van de stemmen. In het geval van één enkele of twee dezelfde superstemmen worden alle gewone stemmen genegeerd. In het geval van twee tegenstrijdige superstemmen beslist opnieuw de meerderheid. Als de stemming publiek is vergen deze regels iets meer denkwerk dan een gewone stemming, maar doemen er geen onoverkomelijke problemen op. De Verenigde Staten en Rusland willen echter dat niemand weet wanneer zij hun superstem gebruiken, zelfs niet de klerk die de stemmen telt. Als reactie daarop eisen de andere leden een volledig anonieme stemming. Een degelijk Secure Distributed Computing protocol kan aan al hun eisen voldoen.

Er zijn ongelooflijk veel geavanceerde toepassingen te bedenken die specifieke veiligheidseisen hebben. Eén ervan zullen we in de komende paragrafen in detail bespreken: een Secret Query Database. Bovendien maken we een inschatting van de praktische haalbaarheid ervan. Deze resultaten zijn in het kader van deze thesis in een paper gepubliceerd ([29]).

8.1.1 Het probleem: Secret Query Database

Alice is het hoofd van de human resources afdeling van een groot bedrijf dat op zoek is naar nieuwe werknemers. Bob verzamelt CV's in een gegevensbank en verdient zijn boterham met het verkopen van die CV's aan geïnteresseerde bedrijven. Zijn gegevensbank bevat een grote verscheidenheid aan CV's, van mensen met alle mogelijke opleidingen, ervaringen en

leeftijden. Als het bedrijf van Alice op zoek is naar ingenieurs in de computerwetenschappen (zijn ze dat niet allemaal?) wil ze niet betalen voor het CV van een licentiaat wijsbegeerte. Ze is enkel bereid te betalen voor de CV's die haar interesseren. Bob ziet het natuurlijk niet zitten haar de volledige gegevensbank toe te sturen zodat zij zelf kan beslissen hoeveel er haar interesseren: Alice zal waarschijnlijk beweren dat er geen enkel CV bij was dat haar interesseerde, terwijl ze een uurtje later wel alle computerwetenschappers in Bobs gegevensbank gecontacteerd heeft en een week later een eigen bedrijfje opricht dat CV's aanbiedt net onder de prijs van Bob. Langs de andere kant is Alice ook niet bereid haar selectiecriteria aan Bob te geven zodat hij kan bepalen welke records haar zullen interesseren. Ze kan hiervoor verschillende redenen hebben: stel dat Alice voor een boekhandel werkt en plots op zoek is naar vijf computerwetenschappers met ervaring in e-commmerce, dan kan Bob voor veel geld aan een concurrerende boekhandel gaan vertellen dat Alice van plan is boeken te verkopen over het Internet. Misschien gebruikt ze ook wel compromitterende of zelfs illegale selectiecriteria, zoals geslacht of ras. Om uit deze impasse te geraken hebben Alice en Bob een *Secret Query Database* nodig, een concept dat we nu zullen uitleggen.

Bob heeft een gevenensbank met meerdere records, waarvan Alice enkel geïnteresseerd is in records die voldoen aan een bepaalde query. Laten we die query q noemen en één van Bobs records x. Het resultaat van $q(x)$ is één enkele bit: een 1 als x voldoet aan q, een 0 als dat niet het geval is. De Secret Query Database moet Alice en Bob toelaten $q(x)$ te berekenen zodat Bob q niet te weten komt en Alice x niet kan achterhalen. Als het resultaat van $q(x)$ een 1 is, stuurt Bob het record naar Alice en betaalt Alice ervoor. Als het resultaat een 0 is, houdt Bob het record voor zichzelf.

Wat we juist bedoelen met geheimhouding van de query is een delicate zaak. In deze tekst zullen we ons richten op het volgende niveau van veiligheid: als we mogen aannemen dat de query volledig willekeurig is[1] en als we stellen dat E de verzameling is van alle reeds geëvalueerde records (dus Bob kent $q(x)$ voor elke $x \in E$) en dat $y \notin E$, dan mag Bob niet in staat zijn $q(y)$ niet-verwaarloosbaar beter te voorspellen dan door willekeurig te raden. Deze definitie laat toe dat Bob te weten komt *welke* records Alice wil, maar niet *waarom* juist deze records haar interesseren. Als we q beschouwen als een gigantische waarheidstabel waarvan elke rij bestaat uit een mogelijk record x als invoer en het resultaat van de query $q(x)$ als uitvoer, dan betekent onze definitie van veiligheid dat Bob enkel de rijen van records die daadwerkelijk samen met Alice geëvalueerd worden te weten komt, maar dat hij niets kan zeggen over het resultaat van de andere rijen van de waarheidstabel.

[1] D.w.z. dat het resultaat van de query voor een bepaald record onafhankelijk is van het resultaat voor andere records.

Een sterkere definitie zou zijn dat Bob enkel het aantal geselecteerde records te weten komt (zodat hij weet welk bedrag Alice hem moet betalen), maar niet weet welke records precies geselecteerd zijn. Deze definitie zou ons leiden naar een veralgemening van *Private Information Retrieval (PIR)*, aangebracht in [12]. Het probleem dat daar onder handen wordt genomen is het volgende: een gegevensbank bezit een grote bitstring x waarvan een gebruiker de i^{e} bit wil ontvangen zonder daarbij i te verraden aan de gegevensbank en zonder de volledige bitstring x te moeten downloaden. Dit blijkt informatietheoretisch onmogelijk te zijn als er maar één enkele gegevensbank in het spel is. PIR gaat op zoek naar oplossingen met meerdere gegevensbanken waarvan verondersteld wordt dat ze onderling geen informatie met mekaar uitwisselen. In de oorspronkelijke definitie van PIR is het geen probleem als de gebruiker gedurende het hele proces ook nog andere bits dan de i^{e} bit te weten kan komen. Bovendien is het niet mogelijk een ander selectiecriterium op te geven dan de index van de gewenste bit. Private Information Retrieval is dus een fundamenteel verschillend probleem dan het probleem dat we met onze Secret Query Database willen oplossen. Voor meer informatie over PIR kan de geïnteresseerde lezer terecht in [12, 20, 21, 25, 9].

Om de netwerkbelasting te beperken zouden we geneigd zijn een protocol te gebruiken dat zo weinig mogelijk interactie vereist. Een minimale hoeveelheid aan interactie is echter onontbeerlijk om de veiligheid van de Secret Query Database te garanderen. Geen van beide partijen mag in staat zijn een query te evalueren zonder enige hulp van de ander. In het bijzonder mag Bob na de evaluatie van $q(x)$ tezamen met Alice niet in staat zijn op eigen houtje $q(x')$ te berekenen, net zoals Alice niet in staat mag zijn zonder hulp van Bob $q'(x)$ te berekenen. Het eerste geval zou recht ingaan tegen onze definitie van veiligheid van de query. Als de laatste regel geschonden wordt, kan Alice achtereenvolgens de queries "is de i^{e} bit van x een 1?" evalueren voor alle bits van x en op die manier het volledige record x reconstrueren zonder ervoor te betalen.

We zullen in deze sectie een inschatting trachten te maken van de praktische haalbaarheid van een Secret Query Database op basis van protocols voor Secure Distributed Computing. We moeten daartoe eerst een realistische schatting maken van de grootte van een record en het aantal poorten in het circuit dat de query implementeert. De meeste protocols voor Secure Distributed Computing gaan er echter van uit dat de te evalueren functie bekend is aan beide partijen, terwijl hun invoer geheim moet blijven. Als Alice haar query geheim wil houden moet het circuit een soort universeel circuit zijn dat naast een record van Bob ook een codering voor de query van Alice als invoer neemt. Om echt elke willekeurige functie als query van Alice toe te laten, moet dat universeel circuit onhaalbaar groot zijn. We gaan daarom de vrijheid van Alice een beetje beknotten. We zien haar query eerder als een soort formulier met de verschillende velden van het record, waarin ze de

voorwaarden waaraan dat veld moet voldoen in kan invullen. Het universeel circuit moet dus een eenvoudige query-taal kunnen interpreteren. Met het concrete voorbeeld van Bobs gegevensbank van CV's voor ogen, zien we de volgende waarden als optimistisch maar realistisch: een record bestaat uit $r = 4000$ bits, oftewel 500 bytes. Het circuit dat een typische query voorstelt heeft $C = 1000$ poorten. Een universeel circuit dat de meest voorkomende queries kan interpreteren gebruikt $C_u = 4000$ poorten. De codering van de query neemt dan ongeveer $c = 1000$ bits in beslag.

8.1.2 Implementatie met behulp van Goldreich, Micali en Wigderson

Het nagaan van de bruikbaarheid van een protocol voor een concrete toepassing beslaat meer dan enkel het invullen van enkele waarden. Voor elke toepassing moet bekeken worden welke veiligheid er precies vereist is en of er mogelijkheden zijn om communicatie uit te sparen door veiligheidsmaatregelen die niet relevant zijn voor die specifieke toepassing te laten wegvallen. Een inschatting voor alle protocols uit de vorige twee hoofdstukken zou ons te ver leiden. We zullen er slechts twee als voorbeeld volledig uitwerken.

Laten we eerst eens het protocol van Goldreich, Micali en Wigderson uit sectie 6.2 inbouwen in een Secret Query Database. Zowel Alice als Bob kunnen het circuit zien, dus hebben we een universeel circuit nodig. Laat Bob de rol spelen van de "circuit-encryptor", terwijl Alice het geëncrypteerde circuit zal evalueren. Misschien kunnen we netwerkbelasting uitsparen door verschillende records op één enkele encryptie van het circuit te evalueren. Daarmee breken we echter het grondprincipe van het protocol dat degene die het circuit evalueert voor elke draad slechts één codering te weten komt, namelijk die van de bit die op die draad staat. Als deze regel niet gerespecteerd wordt kan Alice meer informatie over Bobs record te weten komen dan enkel het resultaat van haar query. Voor elke evaluatie moet Bob dus een nieuwe encryptie van het circuit maken en naar Alice doorsturen.

De query van Alice is niet van record tot record verschillend. Ze heeft ook geen enkele reden om een andere query uit te voeren op een ander record, want ze heeft toch geen idee welk record ze precies aan het evalueren is, elk record ziet er in gecodeerde vorm voor haar eender uit: een hoop willekeurige bits. Het heeft dan ook geen zin dat Bob voor elk record een nieuwe reeks oblivious transfers met Alice uitvoert om haar van de codering van haar invoer te voorzien. Het is een beter idee deze oblivious transfer maar één keer uit te voeren. Bob moet dan wel consequent in alle volgende encrypties van het circuit dezelfde r_i^0 en r_i^1 gebruiken op alle invoerdraden van Alice.

De netwerkbelasting kan verder gedrukt worden als Bob herbruikbare data op CD-ROM publiceert. Met *herbruikbare* data bedoelen we dat het mogelijk moet zijn die data verschillende keren opnieuw te gebruiken in

meerdere evaluaties zonder daarbij de veiligheid van de records of de query in het gedrang te brengen, ook als deze evaluaties gebeuren met klanten die onder mekaar informatie uitwisselen of — hetgeen op het zelfde neerkomt — met één en dezelfde klant. Deze methode is vooral bruikbaar voor langzaam evoluerende gegevensbanken, zoals bijvoorbeeld Bobs gegevensbank met CV's. Bob kan zijn CD-ROM dan op regelmatige doch niet al te frequente basis uitgeven.

Bob mag zeker niet bij elk record dezelfde codering voor zijn eigen invoerdraden gebruiken. Alice heeft dan nog maar de keuze uit twee mogelijkheden om zijn volledige gegevensbank te weten te komen, waar ze al snel de juiste uit zal vinden. Het kan echter geen kwaad als Bob bij een andere klant dezelfde codering gebruikt voor de evaluatie van dezelfde records, ook als die klant gaat roddelen met Alice. Bob kan dus de codering voor zijn eigen invoerbits gerust op CD-ROM uitgeven. Dit brengt de veiligheid van zijn gegevens zeker niet in gevaar, want eigenlijk is dit gewoon een CD-ROM vol willekeurige bits zonder enige betekenis op zich.

We moeten ook nog concrete afspraken over de grootte van de encryptiesleutels in het protocol. Laten we voor de encryptiefunctie E een 64-bit symmetrisch algoritme gebruiken (IDEA bijvoorbeeld). Ook R en de r_i's nemen we 64 bits groot. De codering van een record beslaat dan $64r = 256000$ bits oftewel 32Kbytes per record op CD-ROM. De geëncrypteerde versie van een NOT-poort neemt $4 \cdot 64$ bits of 32 bytes in beslag, die van een AND-poort beslaat 64 bytes. Gesteld dat er ongeveer even veel NOT-poorten als AND-poorten in het circuit aanwezig zijn, neemt een poort gemiddeld 48 bytes in beslag en kan het volledige geëncrypteerde circuit voorgesteld worden door $48C_u$ bytes oftewel 192 Kbytes per record. Tenslotte moet voor elke invoerbit van Alice een one-out-of-two oblivious bitstring transfer uitgevoerd worden. Als we de implementatie gebruiken zoals uiteengezet in sectie 4.5 moeten er in totaal zes getallen modulo p over het netwerk verzonden worden, namelijk β_0, β_1, a_0, a_1, r_0 en r_1. Als we voor p een priemgetal van 512 bits kiezen, komt dit uit op een extra belasting van $\frac{1}{8} \cdot 6 \cdot 512c = 384$ Kbytes, onafhankelijk van het aantal records.

8.1.3 Implementatie met behulp van Abadi en Feigenbaum

Het protocol van Abadi en Feigenbaum dat we in sectie 6.3 besproken hebben lijkt geknipt voor onze Secret Query Database. De probleemstelling is niet helemaal het oorspronkelijke Secured Distributed Computing probleem: het gaat ervan uit dat één van beide partijen geheime gegevens heeft en de ander een geheime functie wil evalueren op die gegevens. Als we Bobs record x als de geheime gegevens nemen en Alices query q als de geheime functie, hebben we niet eens een universeel circuit nodig. Dit bespaart ons een factor vier op het aantal poorten van het circuit.

Bob moet een getal $n = pq$ kiezen met p en q priemgetallen congruent

met 3 modulo 4. De grootte van n is een belangrijke veiligheidsparameter: als Alice in staat is n te factoriseren kan ze Bobs record decrypteren zonder ervoor te betalen. Alle gegevens die gedurende het protocol doorgestuurd worden zijn echter getallen modulo n. Het verdubbelen van de lengte van n verdubbelt dus meteen ook de netwerkbelasting. Op dit ogenblik is het mogelijk een 512-bit getal op een paar maanden tijd te factoriseren met een machine van enkele miljoenen dollars. De aankoop van deze machine kan de moeite lonen als we daarmee Bobs volledige gegevensbank kunnen kraken, maar om de klaartekst van één enkel record in handen te krijgen doet Alice er waarschijnlijk beter aan het op een eerlijke manier van Bob te kopen. Als Bob voor alle records met dezelfde n werkt brengt hij Alice in de verleiding zo'n machine te kopen. Als hij echter voor elk record ook een nieuwe n gebruikt is er geen haar op Alices hoofd dat daaraan denkt. Voor de meeste gegevensbanken zal een sleutellengte van 512 bits dus volstaan.

Ondanks de relatief kleine sleutellengte, veroorzaakt de encryptie van een record nog steeds een gigantische data-expansie: elke bit wordt voorgesteld door 512 bits, dus een vermenigvuldiging met factor 512. Eén record neemt dan $512r$ bits of 256 Kbytes in beslag. Voor een redelijk aantal records is het natuurlijk onmogelijk een dergelijke hoeveelheid data over het Internet te versassen. Bob maakt echter op geen enkel ogenblik in het protocol de factorisatie van n bekend, zodat de geëncrypteerde records herbruikbaar zijn. Hij kan dus net als daarnet regelmatig een CD-ROM uitgeven met zijn geëncrypteerde records erop zonder daarbij de veiligheid van zijn gegevens in gevaar te brengen.

Voor de evaluatie van een AND-poort moeten drie geëncrypteerde bits over het netwerk worden verstuurd (twee van Alice naar Bob en één terug van Bob naar Alice). Als we veronderstellen dat de helft van de poorten in het circuit NOT-poorten en XOR-poorten zijn die geen interactie vereisen en dat de andere helft AND-poorten zijn, moeten er gemiddeld $\frac{1}{2} \cdot 3 \cdot 512 \cdot C$ bits of 96 Kbytes over het netwerk gezonden worden per record-evaluatie.

Protocols voor meerdere partijen werken a fortiori ook voor twee partijen. Hoewel deze protocols in vele opzichten sterker en algemener zijn dan hun broertjes voor twee partijen, blijken ze minder geschikt voor de concrete toepassing van een Secret Query Database. Vergeleken met rasechte twee-partijen protocols veroorzaken ze allemaal een veel grotere netwerkbelasting.

8.1.4 Vergelijking van de resultaten en optimizaties

We leggen de prestaties van de twee protocols die we net besproken hebben even naast mekaar in tabel 8.1. We veronderstellen dat de de volledige gegevensbank $n_r = 1024$ records bevat. We maken in de tabel een onderscheid tussen

- de hoeveelheid gegevens die op CD-ROM kunnen gepubliceerd worden

	CD-ROM	per query	per record
Goldreich, Micali en Wigderson	$8n_r r$ bytes = 32 Mbytes	$384c$ bytes = 375 Kbytes	$48C_u$ bytes/record = 190 Kbytes/record
Abadi en Feigenbaum	$64n_r r$ bytes = 250 Mbytes	0 bytes	$96C$ bytes/record = 94 Kbytes/record

Tabel 8.1: Overzicht van de netwerkbelasting

- de hoeveelheid data die voor elke query moet uitgewisseld worden, maar die wel voor alle records bruikbaar blijft
- de hoeveelheid data die voor elk record opnieuw over het netwerk moet

De netwerkbelasting is inderdaad hoog, maar we moeten hier toch enkele opmerkingen bij maken. Er is altijd een afweging tussen netwerkbelasting en veiligheid. In de tweede oplossing hebben we bijvoorbeeld een 512-bit encryptiesysteem gebruikt. Als de waarde van één enkel record eerder klein is, kunnen we eventueel nog kleinere getallen gebruiken. Zo is er ook een afweging tussen netwerkbelasting en complexiteit van de query. Als we een eenvoudigere query-taal hanteren, kan de omvang van een universeel circuit, en daarmee de netwerkbelasting, verder gedrukt worden.

Door gebruik te maken van agent-technologie kan de communicatie beter haalbaar worden: de agenten kunnen zich op knopen (Engels: hosts) nestelen die een verbinding met hoge bandbreedte tussen mekaar hebben. Zo vermijden we dat een al te grote hoeveelheid aan data de halve aardbol rond moet. Natuurlijk veronderstellen we daarbij dat er vertrouwde knopen zijn waarop de agenten rustig kunnen uitgevoerd worden. Als er een knoop is die we onvoorwaardelijk vertrouwen, wordt het probleem van een Secret Query Database natuurlijk triviaal om op te lossen: Bob stuurt zijn gegevensbank naar de host, Alice stuurt haar query naar de knoop en de knoop bepaalt welk record aan Alices vereisten voldoet en welk niet. Het vertrouwen dat beide partijen moeten hebben in dit soort vertrouwde derde partij is echter niet vergelijkbaar met het vertrouwen dat nodig is voor de oplossing met agenten. Bovendien moet het uitvoerende systeem niet eens op de hoogte zijn van een Secret Query Database protocol, het moet enkel een veilig uitvoeringsplatform aanbieden. Een dergelijke knoop kan dus ook voor andere toepassingen nuttige diensten leveren. Dit systeem is *generisch* in die zin dat om het te gebruiken voor een totaal ander protocol, er helemaal geen server moet geherprogrammeerd worden. Op de mogelijkheden van deze technologie wordt dieper ingegaan in [32].

Met deze beschouwingen in het achterhoofd, en gezien het feit dat de beschikbare bandbreedte op het Internet gestaag stijgt, denken we dat het

juist is dat een toepassing als een Secret Query Database zeker voor de nabije toekomst kan zijn, misschien zelfs al voor vandaag.

8.2 Privacy voor mobiele code

Mobiele code is een belangrijke nieuwe techniek in computernetwerken waarbij volledige programma's of stukken programmacode kunnen uitgewisseld worden tussen computers. De grote verscheidenheid aan uitvoeringsplatforms op het Internet wordt verborgen door een gemeenschappelijke programmeertaal, zoals bijvoorbeeld Java, te gebruiken. Mobiele agenten zijn stukjes mobiele code die zelfstandig over het Internet zwerven, informatie verzamelen, zelf beslissen wanneer waar naartoe te gaan en uiteindelijk de vergaarde informatie aan hun afzender afleveren.

Als er over de veiligheid van mobiele code gesproken wordt, heeft men het meestal over de bescherming van de uitvoerende machine tegen agressieve code. De mobiele code kan zich bijvoorbeeld toegang verschaffen tot private bestanden of de machine besmetten met een computervirus. Over dit probleem is veel nagedacht en er zijn ook degelijke oplossingen voor gevonden (bijvoorbeeld het zandbakprincipe).

Tot nog toe heeft het onderzoek naar mobiele code zich vooral gefocust op het verbeteren van de efficiëntie en op de veiligheid van de uitvoerende machine. Een ander punt is de veiligheid van de mobiele agenten zelf: hun code en data zijn volledig overgeleverd aan de goede wil van de knoop waarop ze uitgevoerd worden. Er bestaat een folklore in de onderzoeksgemeenschap van mobiele code die zegt dat mobiele code niet kan beschermd worden tegen het uitvoerende systeem zonder toevlucht te zoeken tot bijzondere ("tamperproof") hardware. Dit klinkt ook aannemelijk omdat het uitvoerende systeem volledige toegang heeft tot de code en de data van het programma. Er gaan echter stemmen op die zeggen dat privacy voor mobiele code wel degelijk mogelijk is. Maar daarvoor moeten we afstappen van het idee-fixe dat een mobiele agent bestaat uit klaartekst code en data omdat hij moet kunnen uitgevoerd worden. In het algemeen is het mogelijk een geëncrypteerde functie te evalueren zonder daarbij de functie te decrypteren. Dit principe werd reeds schematisch voorgesteld in Figuur 6.1 op pagina 53. We hebben in sectie 6.4 uiteengezet hoe Sander en Tschudin een eerste stap in die richting hebben gezet met een protocol dat toelaat veeltermen op die manier te evalueren.

Stel dat iemand een mobiele agent lanceert die alle virtuele CD-winkels van het Internet afloopt om de laagste prijs voor een bepaalde CD te vinden en bij die winkel de CD koopt en betaalt. Als de uitvoerende host toegang heeft tot de programmacode en -data, ziet hij onmiddellijk wat de bedoeling van de agent is. Hij kan dan de gegevens die de agent tot dan toe vergaard heeft vervalsen of de code zodanig wijzigen dat de selectie in zijn eigen voor-

deel zal gebeuren. Als er voor de betaling van de CD een credit-card nummer ingebouwd zit in het programma, kan hij ook dat nummer achterhalen en de bankrekening van de eigenaar leegplunderen.

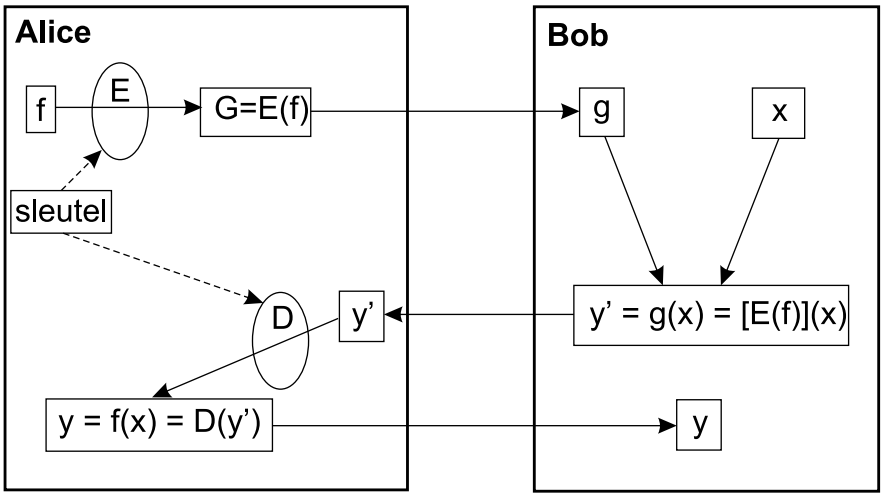
De bescherming die we wensen te bereiken voor mobiele agenten tegen hun uitvoerende hosts is tweevoudig:

- *Privacy van code en data:* De uitvoerende machine mag niet in staat zijn de code of de data die de agent met zich meedraagt te inspecteren.
- *Integriteit van code en data:* Het moet onmogelijk zijn voor de uitvoerende machine de code of de data van de agent op een doelgerichte manier te wijzigen.

Er zijn reeds enkele manieren voorgesteld om, zonder gebruik te maken van echte cryptografie, de programmacode en -data zodanig door mekaar te hutsen en te vermommen dat reverse-engineering gewoon moeilijker wordt gemaakt. De veiligheid van deze systemen is echter zeer moeilijk in te schatten, zodat ze voor echt gevoelige toepassingen zoals e-commerce een te vage oplossing vormen. De bescherming aangeboden aan de mobiele code moet cryptografisch bewijsbaar zijn door aan te tonen dat het achterhalen van de code of het doelgericht wijzigen van de code equivalent is met het oplossen van een gekend moeilijk probleem uit de wiskunde.

Cryptografische bescherming van mobiele agenten kan gerealiseerd worden door Secure Distributed Computing protocols te gebruiken. Het te evalueren circuit moet dan een universeel circuit zijn, een beetje vergelijkbaar met het universeel circuit van een Secret Query Database maar dan niet voor een eenvoudige query-taal maar voor een volwaardige programmeertaal. De geheime invoer aan het circuit is dan langs de ene kant een codering van het uit te voeren programma, en langs de andere kant de te verwerken gegevens van de host. Gezien de cijfervoorbeelden van de Secret Query Database moeten we ons geen illusies maken: met de huidige bandbreedte en de huidige protocols is dit praktisch onhaalbaar vanwege de immense netwerkbelasting die het zou teweegbrengen.

Bovendien zijn de meeste Secure Distributed Computing protocols te interactief om een efficiënte toepassing te krijgen in mobiele code. In het ideale geval is er na het lanceren geen interactie meer nodig tussen de agent en zijn thuishaven. Een groot voordeel van mobiele code is net dat de afzender na de lancering van de agent zich met andere zaken kan bezighouden en eventueel off-line kan gaan terwijl de agent over het Internet zwerft. Om dit voordeel volledig te behouden hebben we dus efficiënte niet-interactieve Secure Distributed Computing protocols nodig.



Figuur 8.1: Software bescherming d.m.v. Secure Distributed Computing

8.3 Software bescherming

Mits een kleine wijziging kan hetzelfde principe voor beveiliging van software zorgen. Stel dat Alice, een geniale professor in de wiskunde, een algoritme bedenkt dat een belangrijk wiskundig probleem (bijvoorbeeld discrete logaritmen berekenen in een eindig veld) veel sneller oplost dan elk ander gekend algoritme. Haar heel leven lang heeft ze volledig ten dienste gestaan van de wetenschap, maar uit deze ontdekking wil ze winst slaan. Wettelijk gezien is het heel moeilijk een patent te nemen op een algoritme. Als ze het algoritme implementeert en de software verkoopt, zal haar algoritme door reverse-engineering vrijwel zeker uitlekken. Ze wil ook niet de rol van een "orakel" spelen waar iedereen een getal naartoe kan sturen opdat zij het discrete logaritme ervan berekent en terugstuurt, want haar algoritme blijft toch vrij zware rekenkracht vereisen. Ze is niet van plan twintig werkstations aan te kopen om de hele aardbol van rekenkracht te voorzien. Alice wil eigenlijk haar software op een zodanige manier verkopen dat haar klanten op een zo autonoom mogelijke manier kunnen werken maar dat haar algoritme toch geheim blijft en dat zij kan blijven controleren wie de software mag gebruiken en wie niet.

De techniek om aan al deze voorwaarden te voldoen is schematisch voorgesteld in Figuur 8.1. Alice encrypteert haar algoritme f tot een uitvoerbare maar geëncrypteerde functie g en deelt dit aan iedereen uit die denkt ooit een discreet logaritme te berekenen, onder andere aan Bob. Als Bob het discreet logaritme van x wil berekenen, evalueert hij de functie $g(x) = y'$. Met deze uitvoer kan hij echter niets beginnen, want y' is nog geëncrypteerd. Om de klaartekst te bekomen moet hij y' naar Alice sturen, want zij is de enige die de sleutel kent. Alice laat Bob betalen voor het gebruik van haar

algoritme, berekent $D(y') = y$ en stuurt de klaartekst y naar Bob.

Met behulp van een dergelijk protocol kan Alice haar klanten laten betalen per gebruik van het algoritme, zonder dat zij voor de rekenkracht moet instaan. Het decrypteren van de oplossing vraagt natuurlijk ook rekenkracht, maar daar gaat het waarschijnlijk slechts om een fractie van de rekenkracht die nodig is om het algoritme zelf uit te voeren.

Hoofdstuk 9

Besluit

We hebben een heel aantal Secure Distributed Computing protocols besproken, zowel voor twee als voor meerdere partijen. Deze protocols onderscheiden zich van mekaar qua cryptografische principes die aan de grondslag ervan liggen, qua netwerkbelasting die ze veroorzaken en qua soort en aantal tegenstanders dat ze kunnen weerstaan. Wat betreft de aanpak van het probleem kunnen we de protocols indelen in twee grote categoriën: de *circuit-encryptie* (Engels: *circuit scrambling*) protocols en de *poort-per-poort* protocols. In circuit-encryptie protocols dragen alle partijen op hun beurt bij aan de vervorming van het circuit tot het voor iedereen onherkenbaar is. Dan wordt dit vervormde circuit geëvalueerd, ofwel door één bepaalde partij, ofwel door alle partijen tezamen. Tot deze categorie kunnen we de protocols rekenen die we uiteengezet hebben in secties 6.1, 6.2, 6.4 en 7.2. Deze protocols benaderen nog het best het ideaal van niet-interactieve protocols die we nodig hebben voor de bescherming van mobiele code, zoals besproken in het vorige hoofdstuk. Poort-per-poort protocols daarentegen beginnen met het uitwisselen van de geëncrypteerde invoer van alle partijen. Dan propageren ze poort per poort doorheen het volledige circuit door van elk tussenresultaat een encryptie te berekenen, totdat ze een encryptie hebben van de uitvoer. Tezamen decrypteren ze dan de uitvoer. Tot deze categorie behoren de protocols uiteengezet in secties 6.3, 7.1 en 7.3, en eigenlijk ook het protocol met de speelkaarten uit hoofdstuk 5.

Het leeuwendeel van deze thesis bestond, zoals voorzien, uit literatuurstudie. De meerwaarde van deze tekst ten opzichte van de literatuur ligt vooral in het aanschouwelijk voorstellen van de protocols en het aanbrengen van de manier van denken die eigen is aan het ontwerp van cryptografische protocols. Toch heb ik ook enkele resultaten geboekt door zelfstandig voort te bouwen op de protocols uit de literatuur. Zo is de uitbreiding op het protocol van Goldreich, Micali en Wigderson uit sectie 6.2 die de veiligheid ervan versterkt zodat het protocol ook kwaadaardige tegenstanders aankan het resultaat van eigen denkwerk. Ook de verbetering in het protocol van

Abadi en Feigenbaum uit sectie 6.3 die de communicatie voor de evaluatie van een AND-poort halveert is een eigen bijdrage. Verder heb ik de bruikbaarheid van de protocols ingeschat door de communicatiecomplexiteit van een Secret Query Database te berekenen. Over dit onderwerp heb ik in samenwerking met Prof. Dr. ir. Frank Piessens en Prof. Dr. ir. Bart De Decker een paper geschreven die aanvaard is ter publicatie voor het IFIP World Computer Congress 2000 ([29]). Ook heb ik meegewerkt aan een paper die het idee mobiele agenten te gebruiken om bandbreedte-verslindende cryptografische protocols praktisch haalbaar te maken verder uitwerkt ([32]).

Het onderzoek naar Secure Distributed Computing is duidelijk verre van afgerond. Op gebied van efficiëntie zijn er zeker nog bergen te verzetten. Ook is er nog steeds geen dubbel homomorf encryptieschema (d.w.z. dat zowel $E(x+y)$ als $E(x \cdot y)$ gemakkelijk te berekenen zijn uit $E(x)$ en $E(y)$) gevonden. Een dergelijk encryptieschema zou de weg banen voor volledig niet-interactieve protocols. Enkele punten waar ik zelf graag dieper had op ingegaan, mocht er meer tijd beschikbaar geweest zijn, zijn de volgende:

- Ook protocols die bestand zijn tegen kwaadaardige tegenstanders kunnen voorlopig niet verhinderen dat een oneerlijke partij één bit meer van de uitvoer kan kennen dan de andere partijen door het protocol vroegtijdig af te breken. Dit voordeel volledig teniet doen is waarschijnlijk onmogelijk, maar misschien is het wel mogelijk met behulp van technieken als "simultaneous secret exchange" dit voordeel te beperken tot bijvoorbeeld een halvering van het rekenwerk om die bit op eigen houtje te berekenen.

- Over de veiligheid van het encryptieschema gebruikt in het protocol van Franklin en Haber uit sectie 7.3 bestaat blijkbaar nog onduidelijkheid. Misschien is het veiligheidsbewijs van Tsiounis en Yung ([36]) uitbreidbaar tot de veiligheid van het voorgestelde schema, of misschien is het schema zelfs bewijsbaar onveilig.

- De vergelijking van ditzelfde encryptiesysteem met ElGamal-encryptie roept het idee op meer dan één bit aan informatie in één encryptie te proppen door meer boodschappen te toe te laten dan enkel $M = 1$ en $M = -1$. De belangrijkste drempel die hiervoor moet overwonnen worden is het behoud van het homomorfisme.

Met het schrikbeeld van de deadline voor ogen heb ik me echter niet meer in deze problemen kunnen verdiepen. Een andere keer misschien ...

Bibliografie

[1] M. Abadi en J. Feigenbaum, "Secure Circuit Evaluation, a Protocol Based on Hiding Information from an Oracle," *Journal of Cryptology*, Vol. 2, Nr. 1, 1990, p. 1–12.

[2] M. Abadi, J. Feigenbaum en J. Kilian, "On Hiding Information from an Oracle," *Proceedings of the 19th ACM Symposium on the Theory of Computing (STOC)*, 1987, p. 195–203.

[3] D. Barrington, "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1," *Proceedings of the 18th ACM Symposium on the Theory of Computing (STOC)*, 1986, p. 1–5.

[4] D. Beaver, "Foundations of Secure Interactive Computing," *Advances in Cryptology—CRYPTO '91 Proceedings*, Springer-Verlag, 1988, p. 377-391.

[5] A. Beimel, T. Malkin en S. Micali, "The All-or-Nothing Nature of Two-Party Secure Computation," *Advances in Cryptology—CRYPTO '99 Proceedings*, Springer-Verlag, 1999, p. 80–97.

[6] S. Goldwasser en M. Bellare, "Lecture Notes on Cryptography," Augustus 1999.

[7] M. Ben-Or, S. Goldwasser en A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," *Proceedings of the 20th ACM Symposium on the Theory of Computing (STOC)*, 1988, p. 1–10.

[8] C. Cachin, "On the Foundations of Oblivious Transfer," *Advances in Cryptology-EUROCRYPT '98 Proceedings*, Springer-Verlag, 1998.

[9] C. Cachin, S. Micali en M. Stadler, "Computationally Private Information Retrieval with Polylogarithmic Communication," *Advances in Cryptology—EUROCRYPT '99 Proceedings*, Springer-Verlag, 1999, p. 402-414.

[10] D. Chaum, C. Crépeau en I. Damgård, "Multiparty Uncondtionally Secure Protocols," *Proceedings of the 20th ACM Symposium on the Theory of Computing (STOC)*, 1988, p. 11–19.

[11] D. Chaum, I. Damgård en J. van de Graaf, "Multiparty Computations Ensuring Privacy of each Party's Input and Correctness of the Result," *Advances in Cryptology—CRYPTO '87 Proceedings*, Springer-Verlag, 1988, p. 87–119.

[12] B. Chor, O. Goldreich, E. Kushilevitz en M. Sudan, "Private Information Retrieval," *Proceedings of the 36th Annual Symposium on the Foundations of Computer Science (FOCS)*, 1995, p. 41–50.

[13] R. Cramer, "Introduction to Secure Computation," *Lecture Notes in Computer Science, Vol. 1561*, Springer-Verlag, 1999, p. 16–62.

[14] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt en T. Rabin, "Efficient Multiparty Computations Secure Against an Adaptive Adversary," *Advances in Cryptology—EUROCRYPT '99 Proceedings*, Springer-Verlag, 1999, p. 311-326.

[15] S. Even, O. Goldreich, and A. Lempel, "A Randomizing Protocol for Signing Contracts," *Communications of the ACM*, Vol. 28, Nr. 6, juni 1985, p. 637–647.

[16] M. Fisher, S. Micali en C. Rackoff, "A Secure Protocol for the Oblivious Transfer," *Journal of Cryptology*, Vol. 9, 1996, p. 191–195.

[17] M. Franklin, "Complexity and Security of Distributed Protocols," Ph. D. thesis, Computer Science Department of Columbia University, New York, 1993.

[18] M. Franklin en S. Haber, "Joint encryption and message-efficient secure computation," *Journal of Cryptology*, Vol. 9, Nr. 4, 1996, p. 217–232.

[19] Z. Galil, S. Haber en M. Yung, "Cryptographic Computation: Secure Fault-Tolerant Protocols and the Public-Key Model," *Advances in Cryptology—CRYPTO '87 Proceedings*, Springer-Verlag, 1987, p. 135–155.

[20] Y. Gertner, Y. Ishai, E. Kushilevitz en T. Malkin, "Protecting Data Privacy in Private Information Retrieval Schemes," *Proceedings of the 30th Annual ACM Symposium on the Theory Of Computing (STOC)*, 1998, p. 151–160.

[21] Y. Gertner, S. Goldwasser en T. Malkin, "A Random Server Model for Private Information Retrieval," *Proceedings of 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 1998.

[22] O. Goldreich, "Secure Multi-Party Computation (Working Draft)", Weizmann Institute of Science, Israël, 1998.

[23] O. Goldreich, S. Micali en A. Wigderson, "How to Play Any Mental Game," *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, 1987, p. 218–229.

[24] S. Goldwasser en S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, Vol. 28, Nr. 2, april 1984, p. 270–299.

[25] E. Kushilevitz en R. Ostrovsky, "Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval," *Proceedings of 38th Annual Symposium on the Foundations of Computer Science (FOCS)*, 1997.

[26] B. Litow, "Introduction to cryptography," 1999, beschikbaar op URL: `http://www.cs.jcu.edu.au/ftp/web/teaching/Subjects/cp5030/1999/`.

[27] S. Loureiro en R. Molva, "Privacy for Mobile Code", *Proceedings of the workshop on Distributed Object Security (OOPSLA)*, 1999, p. 37–42.

[28] S. Micali, P. Rogaway, "Secure Computation," *Advances in Cryptology—CRYPTO '91 Proceedings*, Springer-Verlag, 1991, p. 392–404.

[29] G. Neven, F. Piessens en B. De Decker, "On the Practical Feasibility of Secure Distributed Computing: a Case Study," aanvaard ter publicatie voor het IFIP World Computer Congress 2000.

[30] V. Niemi en A. Renvall, "Secure Multiparty Computations Without Computers," *Theoretical Computer Science*, Vol. 191, Nr. 1, januari 1998, p. 173–183.

[31] N. Nisan, "Algorithms for Selfish Agents", *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, maart 1999, p. 1–15.

[32] F. Piessens, B. De Decker, E. Van Hoeymissen en G. Neven, "On the Trade-Off between Communication and Trust in Secure Computations," aanvaard voor de 6de ECOOP Workshop on Mobile Object Systems.

[33] T. Sander en C. Tschudin, "On Software Protection via Function Hiding", *Proceedings of the 2nd Workshop on Information Hiding*, Portland, Oregon, USA, april 1998.

[34] T. Sander en C. Tschudin, "Towards Mobile Cryptography", *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, Oakland, California, mei 1998.

[35] B. Schneier, "Applied Cryptography Second Edition: Protocols, Algorithms and Source Code in C," John Wiley & Sons, 1996.

[36] Y. Tsiounis en M. Yung, "On the Security of ElGamal based Encryption," *International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, 1998, Yokohama, Japan.