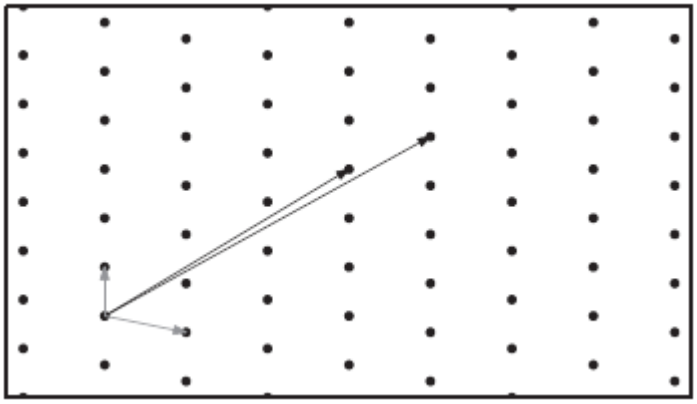
‘P=NP’ versus Crypto

Op roosters gebaseerde cryptografie

Marloes Venema

15 juli 2016



Bachelorscriptie Wiskunde
Begeleider: Wieb Bosma
Tweede lezer: Sebastiaan Terwijn

Inhoudsopgave

1 Inleiding

We denken er misschien niet elk moment van iedere dag aan, maar het is overal, en het is belangrijk: cryptografie. Van de technologie in onze pinpas tot onze eigen computer, van DigiD tot onze smartphone — ze hebben er allemaal mee te maken. In het tijdperk van de technologie is cryptografie erg belangrijk om onze privacy en veiligheid te waarborgen.

We hebben allemaal misschien wel eens de televisie aangezet en een televisieserie gekeken waarin een slimme hacker probeert in te breken in verscheidene computersystemen. In het ergste geval gaat het hier om zwaar beveiligde faciliteiten van de NSA of het Pentagon, maar zelfs simpelere systemen als je eigen computernetwerk zijn nooit veilig voor deze hackers.

Nu is dit natuurlijk een zwaar overdreven fenomeen. In de werkelijkheid kun je het Pentagon of de NSA nooit hacken vanaf je eigen laptop — of zelfs een goede computer. Het kost enorme computerrekenkracht om een verbinding die beveiligd wordt door Advanced Encryption Standard (AES) [7] of een andere versleuteling binnen te dringen. Ter illustratie, een doodgewone wifi-netwerkverbinding is beveiligd met WPA2, waarvan het versleutelingsprotocol gebruik maakt van AES. Om een AES-128 versleuteling te breken, moeten we gemiddeld $2^{64} \approx 1,8 \cdot 10^{19}$ sleutels uitproberen om de juiste te vinden. Zelfs met een supercomputer zou dit miljarden jaren kosten.

Het is dus onmogelijk om op een systeem als deze in te breken door simpelweg sleutels te raden. Het komt in de praktijk dus altijd aan op het menselijke aspect dat we kunnen 'hacken'. Hierbij kunnen we denken aan slecht gekozen wachtwoorden, of gewoon slordige omgang met onze wachtwoorden, of in een toenemende mate groter wordend probleem: phishing.

We kunnen dus intuïtief nagaan dat cryptografie op zichzelf staand en onafhankelijk van het menselijke aspect erg sterk is, maar waarom is dat eigenlijk zo?

In de cryptografie maken we onderscheid tussen twee soorten cryptografie: symmetrische en asymmetrische cryptografie. AES is een voorbeeld van symmetrische cryptografie, het soort cryptografie waarin we slechts één sleutel gebruiken. Van deze sleutel nemen we aan dat alleen de twee partijen waartussen gecorrespondeerd wordt deze weten. Dus voor ieder tweetal partijen hebben we één sleutel nodig om een dergelijke correspondentie te beveiligen, en deze sleutel moet geheim blijven voor de rest van de wereld.

Daarentegen maakt asymmetrische cryptografie gebruik van twee sleutels, maar dan kan iedereen corresponderen met één bepaalde partij, zonder dat andere partijen in kunnen breken op de correspondentie. Slechts een van deze twee sleutels is openbaar: de publieke sleutel. De andere sleutel

is alleen in het bezit van de eigenaar van deze twee sleutels en wordt de geheime sleutel genoemd. Het idee is dat de publieke sleutel wordt gebruikt om een bericht te versleutelen, en dat iedereen dit dus kan doen, maar enkel de eigenaar van de geheime sleutel kan dit versleutelde bericht ontsleutelen.

In de praktijk wordt vaak een combinatie van deze twee soorten cryptografie toegepast. Asymmetrische cryptografie is namelijk veilig, maar erg traag in het gebruik. Echter is symmetrische cryptografie juist weer heel snel, maar heeft als nadeel dat het delen van een sleutel die alleen bekend is bij de twee deelnemende partijen een probleem vormt, terwijl dit weer geen probleem is bij asymmetrische cryptografie. Een oplossing is dus om door middel van asymmetrische cryptografie een sleutel te versturen, en die weer te gebruiken bij de symmetrische cryptografie. Dan hebben we de veiligheid van de asymmetrische cryptografie, en de snelheid van de symmetrische cryptografie.

Nu kunnen we ons afvragen hoe asymmetrische cryptografie precies te werk gaat, en waarom we maar twee sleutels nodig hebben om communicatie naar één persoon mogelijk te maken zonder dat een derde partij 'mee kan lezen'. Een visualisatie hiervan zou zijn als we een hele stapel met hangsloten hebben waar we zelf geen sleutel bij hebben, maar de persoon naar wie we een bericht willen sturen juist wel. In dit scenario schrijven we een bericht, stoppen deze in een koffer en 'versleutelen' we deze koffer met het hangslot.

We kunnen dus zelf deze koffer niet meer openmaken, maar het voordeel is dat niemand anders dat kan, behalve de persoon naar wie we het bericht sturen. De koffer wordt verstuurd naar deze persoon, en daar aangekomen wordt de koffer opengemaakt met de sleutel.

Zo werkt het ook bij asymmetrische cryptografie. Hier hebben we een cryptosysteem, met de ruimtes waarin we de boodschappen en gecodeerde boodschappen definiëren, de sleutels en de algoritmes om boodschappen te versleutelen, en gecodeerde boodschappen te ontsleutelen. In een asymmetrisch cryptosysteem wordt de publieke sleutel gebruikt voor de versleuteling en de geheime sleutel voor de ontsleuteling.

Een van de meest gebruikte asymmetrische cryptosystemen is RSA, en is uitgevonden door Ron Rivest, Adi Shamir en Leonard Adleman [18].

Voor RSA hebben we twee grote priemgetallen p en q nodig waarvan we het product $n = pq$ uitrekenen. Deze p en q zijn geheim, maar n is publiek. Met p en q kunnen we de Euler indicator $\varphi(n) = (p-1)(q-1)$ berekenen. Het aantal inverteerbare elementen van $\mathbb{Z}/n\mathbb{Z}$, genoteerd met $(\mathbb{Z}/n\mathbb{Z})^*$, is gelijk aan $\varphi(n)$. Deze $\varphi(n)$ is ook geheim.

Dan nemen we als publieke sleutel een $e \in (\mathbb{Z}/n\mathbb{Z})^*$ zodat $\gcd(e, \varphi(n)) = 1$, oftewel e en $\varphi(n)$ zijn relatief priem. Dan is er een $d \in (\mathbb{Z}/n\mathbb{Z})^*$ zodat

geldt: $e \cdot d \equiv 1 \bmod \varphi(n)$, omdat e inverteerbaar is in $\mathbb{Z}/\varphi(n)\mathbb{Z}$. We nemen deze d voor onze geheime sleutel.

Dan zijn onze versleutelings- en ontsleutelingsalgoritmen de volgende functies: voor $m \in \mathbb{Z}/n\mathbb{Z}$ berekenen we $\mathrm{Enc}(m) = m^e \bmod n$ voor de versleuteling, en voor $c \in \mathbb{Z}/n\mathbb{Z}$ berekenen we $\mathrm{Dec}(c) = c^d \bmod n$ voor de ontsleuteling.

Dan willen we dat ook nog geldt dat voor boodschap $m \in \mathbb{Z}/n\mathbb{Z}$ de ontsleuteling van de versleutelde boodschap weer dezelfde boodschap teruggeeft, dus: $\mathrm{Dec}\left(\mathrm{Enc}(m)\right) = (m^e)^d = m^{ed}$. Omdat $e \cdot d \equiv 1 \bmod \varphi(n)$ geldt, kunnen we een $k \in \mathbb{Z}$ vinden zodat $ed = 1 + k\varphi(n)$. We weten ook met de Kleine Stelling van Fermat dat geldt dat $x^{\varphi(n)} \equiv 1 \bmod n$ voor iedere $x \in \mathbb{Z}/n\mathbb{Z}$. Dus we hebben dat $m^{ed} = m^{1+k\varphi(n)} = m^1 \cdot \left(m^{\varphi(n)}\right)^k = m \cdot 1^k \equiv m \bmod n$. Dus de ontsleuteling is de inverse van de versleuteling en is daarom correct gedefinieerd.

We kunnen dus deze inverse alleen berekenen als we d weten en deze kunnen we alleen uitrekenen als we p en q weten. Echter is alleen $n = pq$ publiek, dus als we dit systeem willen kraken, moeten we n factoriseren. Het is dus duidelijk dat RSA een cryptosysteem is dat gebaseerd is op de moeilijkheid van deze factorisatie. Voor kleine getallen kunnen we dit nog wel snel doen, bijvoorbeeld van 55 kunnen we snel zeggen dat $55 = 5 \cdot 11$, maar voor grotere getallen wordt dit te moeilijk om snel op te lossen. Stel bijvoorbeeld $n = 6557$, dan wat zijn p en q zodat $n = pq$?

Zo zijn alle asymmetrische cryptosystemen gebaseerd op een moeilijk probleem. RSA is gebaseerd op priemfactorisatie, terwijl bijvoorbeeld Diffie-Hellman weer gebaseerd is op het discrete log probleem[1].

Nu is er een probleem in de wiskunde en informatica dat het P versus NP probleem heet. Deze vraagt of alle makkelijk verifieerbare problemen ook makkelijk oplosbaar zijn. Dus als we bijvoorbeeld priemfactorisatie hebben, dan kunnen we dit makkelijk verifiëren, want stel dat we een $n = 6557$ hebben, en $p = 79$ en $q = 83$ zijn gegeven, dan kunnen we direct zien dat $n = pq$. Dus priemfactorisatie is makkelijk verifieerbaar.

Dan is de vraag: is priemfactorisatie ook makkelijk oplosbaar?

Stel dat we kunnen bewijzen dat alle wiskundige problemen die makkelijk verifieerbaar zijn ook makkelijk oplosbaar zijn, wat betekent dit dan voor de asymmetrische cryptografie? Want dan kunnen we een oplossing vinden voor $n = pq$ voor een gegeven n, en we hebben zojuist kunnen zien dat dan het hele RSA cryptosysteem gebroken is.

Bewijzen dat alle wiskundige problemen die makkelijk verifieerbaar zijn,

[1] Voor $g, h \in G$ voortbrengers van multiplicatieve groep G, bepaal $x \in \mathbb{Z}$ zodat $g^x = h$.

ook makkelijk oplosbaar zijn betekent natuurlijk niet dat er meteen ook een oplossing is voor ieder makkelijk verifieerbaar probleem. Het kan best zijn dat priemfactorisatie nog steeds lang duurt om op te lossen, en daardoor het systeem zijn veiligheid behoudt. Maar er rijst toch een vraag:

> Is er een asymmetrisch cryptosysteem gebaseerd op een wiskundige probleem dat niet gebroken wordt onder de aanname dat alle makkelijk verifieerbare problemen ook makkelijk oplosbaar zijn?

In deze bachelorscriptie zullen we bekijken wat 'makkelijk' precies betekent (hoofdstuk 2), laten we zien dat er een klasse van cryptosystemen bestaat dat gebaseerd is op roosters (hoofdstuk 3), introduceren en bekijken we een dergelijk cryptosysteem (hoofdstuk 4) en bekijken we een paar aanvallen op dit systeem (hoofdstuk 5).

Uiteindelijk zullen we zien dat deze vraag niet zo gemakkelijk te beantwoorden is, en dat we eigenlijk nooit zeker weten dat er geen makkelijke oplossing voor een probleem is. Maar we kunnen wél kijken of er een cryptosysteem bestaat dat gebaseerd is op een probleem dat op zijn minst moeilijker is dan priemfactorisatie.

(En dat is al moeilijk genoeg.)

2 P versus NP

In de wiskunde is een van de problemen die nog steeds niet opgelost zijn het P versus NP probleem. Om het kort uit te leggen betekent dat dat we niet weten of alles wat makkelijk te verifiëren is met een computer, ook makkelijk op te lossen is. In deze sectie wordt kort samengevat wat P en NP problemen precies zijn.

In de informatica zijn er twee manieren om P en NP te beschrijven: met behulp van Turingmachines, en door middel van de grote-O-notatie. In dit geval zal ik alleen ingaan op de tweede manier, en niet op de definitie die gebruik maakt van een Turingmachine, omdat de grote-O-notatie relevanter — en gemakkelijker in het gebruik — is in dit geval.

Voor een uitgebreidere uitleg, een eventuele nieuwsgierigheid naar Turingmachines en complexiteit, en de afkomst van deze informatie refereer ik naar [20, 6].

2.1 'Grote-O-notatie'

Eerst hebben we een definitie nodig die aangeeft wat de orde van een functie is.

Definitie 2.1
Zij $g : \mathbb{R} \to \mathbb{R}$ een functie. Een functie $f : \mathbb{R} \to \mathbb{R}$ is van orde g als er een constante $c \in \mathbb{R}_{>0}$ en een $x_0 \in \mathbb{R}$ bestaan zodat voor alle $x > x_0$ geldt dat $|f(x)| \leq cg(x)$. Dit wordt genoteerd als: $f \in O(g)$ (en wordt ook wel de grote-O-notatie genoemd).

In het bijzonder is dus een algoritme in $O(n^k)$ voor een zekere $k \in \mathbb{N}$ als de functie die de benodigde 'tijd' beschrijft van dit algoritme van orde n^k is. Met 'tijd' wordt hier het aantal elementaire stappen dat een computer moet uitvoeren om dit algoritme uit te voeren bedoeld. Als zo'n k bestaat, heet het algoritme **polynomiaal**.

2.2 Klasse NP

De klasse NP is de klasse van non-deterministisch polynomiale problemen. Makkelijk gezegd: een NP-probleem is een probleem dat efficiënt te verifiëren is. Dat betekent dat een computer relatief snel een eventuele oplossing van een probleem kan verifiëren.

Definitie 2.2
Een probleem behoort tot de klasse NP als er een polynomiaal algoritme bestaat dat een gegeven oplossing voor dit probleem verifieert.

Het woord 'non-deterministisch' komt van de definitie die gebruik maakt van Turingmachines. Dit houdt in dat er een *non-deterministische* Turingmachine is die de oplossing van het probleem vindt.

2.3 Klasse P

De klasse P is de klasse van polynomiale problemen. Makkelijk gezegd: een P-probleem is een probleem dat efficiënt op te lossen is. Dat betekent dus dat een computer relatief snel een oplossing voor dit probleem kan berekenen.

Definitie 2.3
Een probleem behoort tot de klasse P als er een polynomiaal algoritme bestaat dat een oplossing vindt voor dit probleem.

In tegenstelling tot de NP problemen is er voor P-problemen een *deterministische* Turingmachine die het probleem oplost.

2.4 NP-volledig

De klasse van NP-volledige problemen is de klasse van moeilijkste problemen in NP. In het bijzonder is een probleem NP-volledig als dit probleem een makkelijk verifieerbaar beslissingsprobleem is.

Definitie 2.4
Een probleem heet een beslissingsprobleem als het probleem een antwoord vereist dat ofwel 'ja' is, ofwel 'nee'.

Bijvoorbeeld: is n een priemgetal? Dan is het antwoord ofwel 'ja', ofwel 'nee'.

Definitie 2.5
Een beslissingsprobleem D heet NP-volledig als

i) dit probleem een klasse NP-probleem is;

ii) en voor ieder probleem H in NP er een polynomiale-tijdsreductie algoritme[2] *bestaat van D naar H.*

[2]Dit betekent dat er een polynomiaal algoritme bestaat dat H transformeert in een instantie van D, zodat als er een oplossing voor D is in polynomiale tijd, H ook opgelost kan worden in polynomiale tijd.

De definitie van NP-volledig gebruiken we om de definitie van NP-moeilijk te geven.

2.5 NP-moeilijk

De klasse NP-moeilijk is de klasse problemen die op zijn minst net zo moeilijk zijn als de moeilijkste problemen in de klasse NP. Deze problemen hoeven zelf dus niet in NP te liggen.

Definitie 2.6
Een probleem H heet NP-moeilijk als er een NP-volledig beslissingsprobleem B bestaat zodat er een polynomiale-tijdsreductie algoritme van B naar H bestaat.

Merk op: deze definitie is equivalent met de volgende uitspraak:

Een probleem H heet NP-moeilijk als ieder ander NP-probleem L met een polynomiale-tijdsreductie algoritme naar H getransformeerd kan worden.

Deze uitspraak wordt ook vaak gebruikt in de complexiteitstheorie, maar betekent precies hetzelfde (en legt ook een directe link uit met de 'globale' uitleg van NP-moeilijke problemen, in de zin dat deze problemen minstens net zo moeilijk als de moeilijkste problemen in NP).

Dit ligt aan het feit dat ieder probleem in NP met een polynomiale-tijdsreductie algoritme naar een NP-volledig probleem getransformeerd kan worden (want immers: de NP-volledige problemen zijn de moeilijkste problemen in NP).

Uit deze definitie volgt vrijwel direct:

Gevolg 2.7
Als een optimalisatieprobleem H een NP-volledige beslissingsversie L heeft, dan is H NP-moeilijk.

Om te illustreren wat een optimalisatieprobleem en een beslissingsversie van dit probleem precies is, bekijken we een van de beroemdste optimalisatieproblemen: het handelsreizigersprobleem (Engels: the Travelling Salesman Problem (TSP)).

Zij $n \in \mathbb{N}$ het aantal steden dat bezocht moet worden door een handelsreiziger en noteer de steden als $u_1, ..., u_n$. Noteer de afstand tussen punten u_i en u_j (voor $i, j \in \{1, ..., n\}$) als $d(u_i, u_j)$. Hier zijn $d(u_i, u_j) = d(u_j, u_i) > 0$ voor alle $i, j \in \{1, ..., n\}$ met $i \neq j$ en $d(u_i, u_j) = 0$ voor $i = j$.

Het handelsreizigersprobleem vraagt naar het kortste pad, genoteerd als permutatie $(u_1, ..., u_n)$, zodat $d(u_n, u_1) + \sum_{i=1}^{n-1} d(u_i, u_{i+1})$ minimaal is.

In andere woorden: het handelsreizigersprobleem zoekt naar het kortste pad door alle steden, dat iedere stad precies één keer bezoekt en weer eindigt in het beginpunt.

Stelling 2.8
Het handelsreizigersprobleem is NP-moeilijk.

Bewijs: In [6] wordt bewezen dat dit probleem NP-moeilijk is. ■

Dit is dus een optimalisatieprobleem, omdat er gezocht wordt naar een minimaal element (namelijk het kortste pad). De beslissingsversie van dit probleem zou zijn:

Zij $l \in \mathbb{R}_{>0}$ een gegeven lengte, is er een pad dat iedere stad precies één keer bezoekt en eindigt in het beginpunt, dat een lengte van minstens l heeft?

Bewijzen dat dit NP-volledig is komt dus op hetzelfde neer als bewijzen dat het handelsreizigersprobleem NP-moeilijk is.

2.6 Verband tussen de klassen

We zien al snel dat er een duidelijk verband is tussen de vier gedefinieerde klassen. De klassen NP-moeilijk en NP-volledig, en vanzelfsprekend NP, hebben duidelijk het volgende verband: NP-volledig $\subseteq$ NP (want alle NP-volledige problemen liggen in NP) en NP-volledig $\subseteq$ NP-moeilijk (NP-moeilijk zijn alle moeilijkste problemen die zowel in NP als buiten NP liggen). Maar niet NP-moeilijk $\subseteq$ NP-volledig (want niet alle problemen hoeven in NP te liggen, zoals bijvoorbeeld TSP is NP-moeilijk, maar hoeft niet in NP te liggen), en dan al helemaal niet NP-moeilijk $\subseteq$ NP (want NP-volledig $\subseteq$ NP-moeilijk).

Wat de relatie is tussen NP en P — en daarmee ook NP-volledig en NP-moeilijk — is wat moeilijker te zien. We kunnen wel zien dat P $\subseteq$ NP, want alle problemen die makkelijk op te lossen zijn, zijn ook makkelijk te verifiëren, maar andersom is al veel moeilijker om aan te tonen.

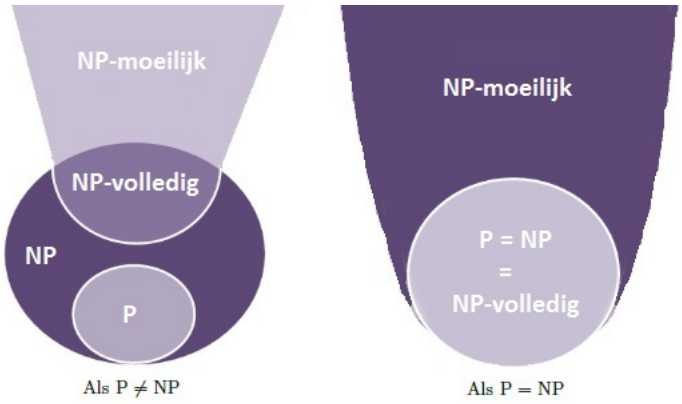
Sterker nog: het is maar de vraag of P = NP of P $\neq$ NP.

Het is dan ook een van de grootste problemen in de wiskunde en informatica en is niet voor niets een van de zeven Millennium Problemen [22], waarvoor overigens een prijs van een miljoen dollar wordt uitgereikt aan de persoon die een bewijs weet te vinden. Tot zover zijn er alleen vermoedens over deze gelijkheid. We kunnen wel bekijken wat de relaties tussen deze klassen zijn in het hypothetische geval dat P = NP, en in het geval dat P $\neq$ NP.

Stel P $\neq$ NP, dan zijn er problemen in NP die niet in P liggen. Vanzelfsprekend liggen dan de moeilijkste problemen van NP — dus de NP-volledige problemen — niet in P, want als een van deze problemen in P ligt, dan liggen ze allemaal in P, en alle problemen die makkelijker zijn liggen dan ook in P. Dus P $\cap$ NP-volledig $= \emptyset$ en dan ook P $\cap$ NP-moeilijk $= \emptyset$.

Stel P = NP, dan worden alle problemen die een polynomiaal algoritme hebben dat een antwoord verifieert ook makkelijk oplosbaar. En dan geldt dat alle problemen in NP-volledig ook in P liggen. NP-moeilijk is nog steeds een grotere klasse dan de NP-klasse, want deze bevat ook alle moeilijkere problemen.

Voor de duidelijkheid geeft deze figuur weer hoe de klassen zich tot elkaar verhouden:



2.7 Toepassing op cryptosystemen

Om een public key cryptosysteem te vinden dat toch moeilijk blijft onder de aanname dat P = NP, willen we dus een cryptosysteem vinden dat berust op een probleem dat niet in NP zit. Een goed begin is dus een wiskundig probleem te vinden dat NP-moeilijk is. Een mooie bijkomstigheid is dan ook dat, als toch wordt bewezen dat P $\neq$ NP — iets wat overigens verondersteld wordt — dat dan ook echt zeker is dat dit probleem niet in P zit, en dus altijd moeilijk op te lossen is met een computer.

We zoeken hier naar een probleem in NP-moeilijk, omdat het moeilijk is om aan te tonen een een probleem niet in NP zit (hoe laat je zien dat er geen algoritme is dat een probleem in polynomiale tijd verifieert?). Het is relatief makkelijker om aan te tonen dat een probleem NP-moeilijk is.

Bovendien is een cryptosysteem dat gebaseerd is op een NP-moeilijk probleem cryptografisch gezien veiliger — omdat onder de ene aanname

deze sterker is, en onder de andere aanname deze in het slechtste geval polynomiaal op te lossen is — dan een cryptosysteem dat gebaseerd is op een probleem dat in NP zit.

3 Cryptosystemen gebaseerd op roosters

3.1 Introductie

Een vrij recente vorm van cryptografie is cryptografie die gebaseerd is op roosters. Deze cryptografie is gebaseerd op de moeilijkheid van roosterproblemen. Met name in het post-kwantum tijdperk is er behoefte aan een cryptosysteem dat ook bij introductie van kwantumcomputers veilig blijft, anders dan de nu veelgebruikte varianten zoals RSA. Van RSA is bewezen dat deze gebroken kan worden in polynomiale tijd op een kwantumcomputer met Shor's algoritme [19]. Een veelbelovend type asymmetrisch cryptosysteem is het type cryptosysteem dat gebaseerd is op de moeilijkheid van problemen met roosters.

In eerste oogopslag zouden we ons kunnen afvragen wat kwantumcryptografie te maken heeft met wat wij willen doen: namelijk een cryptosysteem vinden waarvan de veiligheid gebaseerd is op een NP-moeilijk probleem. Van drie problemen in het bijzonder is bewezen dat deze NP-moeilijk zijn, wat dus juist van toepassing is op onze zoektocht naar een cryptosysteem dat gebaseerd is op een NP-moeilijk probleem.

In de op roosters gebaseerde cryptografie staan de volgende drie problemen voornamelijk centraal: het kortste vectorprobleem (SVP, afkorting van het Engelse 'Shortest Vector Problem'), het dichtstbijzijnde vectorprobleem (CVP, afkorting van het Engelse 'Closest Vector Problem') en het kortste onafhankelijke vectorenprobleem (SIVP, afkorting van het Engelse 'Shortest Independent Vectors Problem'). Wat deze problemen inhouden, wordt besproken in sectie 3.3, maar het probleem waar we in het bijzonder geïnteresseerd in zijn is CVP.

3.2 Definities en notaties

Om een duidelijk beeld te krijgen van wat nou eigenlijk een rooster is, zullen we eerst bekijken hoe dit precies wordt gedefinieerd. Daarna komen nog wat andere handige eigenschappen aan bod.

Definitie 3.1
Laat $b_1, ..., b_n \in \mathbb{R}^n$ lineair onafhankelijke vectoren zijn. Definieer het rooster met volledige rang als de volgende verzameling:

$$\mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \text{ voor } 1 \leq i \leq n \right\}$$

In dit geval wordt $b_1, ..., b_n$ de basis genoemd. Een basis kan ook gerepresenteerd worden door een matrix $B = [b_1, ..., b_n] \in \mathbb{R}^{n \times n}$, waar de basisvectoren de kolommen van de matrix zijn. Dus een alternatieve notatie voor het rooster is: $\mathcal{L}(B) = \{Bx : x \in \mathbb{Z}^n\}$ (waarbij $b_1, ..., b_n$ en x kolomvectoren zijn[3]).

Merk op: als de rang $< n$, dan heet $\mathcal{L}(b_1, ..., b_m)$ nog steeds een rooster, maar dan van rang $m < n$. In dit geval bekijken we slechts roosters van volledige rang.

Opmerking 3.2
De norm, genoteerd met $\|.\|$, wordt gedefinieerd als de Euclidische norm: Voor alle $x, y \in \mathbb{R}^n$ geldt:

$$\|x - y\| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

De afstand tussen een rooster met roosterbasis $B \in \mathbb{R}^{n \times n}$ en een doelvector $t \in \mathbb{R}^n$ wordt genoteerd als:

$$dist(t, \mathcal{L}(B)) = dist(t, B) = \min_{x \in \mathcal{L}(B)} \|t - x\|$$

Definitie 3.3
Definieer voor $x \in \mathbb{R}$ het symbool $\lfloor x \rceil$ als de afronding van x. Dus:

$$\lfloor x \rceil = z$$

voor een $z \in \mathbb{Z}$ zodat voor alle $z' \in \mathbb{Z}$ geldt: $\|x - z'\| \geq \|x - z\|$.

Merk op: definieer voor vector $x \in \mathbb{R}^n$ (of zelfs een matrix) het symbool $\lfloor x \rceil$ als volgt:

$$\lfloor x \rceil_i = \lfloor x_i \rceil$$

voor iedere $i \in \{1, ..., n\}$, dus de afronding van ieder element van de vector.

Deze definitie is vooral nodig bij de introductie van de algoritmes van Babai. Wat deze inhouden wordt behandeld in sectie 4.

Nu introduceren we de Gegeneraliseerde Regel van Cramer. Deze is nodig voor een belangrijke stelling die ook in deze sectie wordt geïntroduceerd.

[3]We laten vanaf nu alle vectoren kolomvectoren zijn, tenzij anders aangegeven.

Stelling 3.4 (Gegeneraliseerde Regel van Cramer)
Zij $A \in \mathbb{R}^{n\times n}$ een matrix met $\det(A) \neq 0$ en $AX = B$ voor $X, B \in \mathbb{R}^{n\times n}$. Laat $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ en $1 \leq j_1 < j_2 < \cdots < j_k \leq n$. Dan is:

$$X \begin{bmatrix} i_1 & i_2 & \cdots & i_k \\ \cdots & \cdots & \cdots & \cdots \\ j_1 & j_2 & \cdots & j_k \end{bmatrix}$$

de $k \times k$ deelmatrix van X gevormd door de rijen i_s en de kolommen j_s voor $s = 1, ..., k$. En:

$$A_B \begin{bmatrix} i_1 & i_2 & \cdots & i_k \\ \cdots & \cdots & \cdots & \cdots \\ j_1 & j_2 & \cdots & j_k \end{bmatrix}$$

is de $n \times n$ matrix die geconstrueerd wordt uit A door kolom i_s van A te vervangen door kolom j_s van B, voor $s = 1, ..., k$.

Dan zegt de gegeneraliseerde regel van Cramer:

$$\det\left(X \begin{bmatrix} i_1 & i_2 & \cdots & i_k \\ \cdots & \cdots & \cdots & \cdots \\ j_1 & j_2 & \cdots & j_k \end{bmatrix}\right) = \det\left(A_B \begin{bmatrix} i_1 & i_2 & \cdots & i_k \\ \cdots & \cdots & \cdots & \cdots \\ j_1 & j_2 & \cdots & j_k \end{bmatrix}\right) / \det(A)$$

Bewijs: Zie [9] voor het bewijs van deze stelling. ■

Lemma 3.5
Laat $U \in \mathbb{Z}^{n\times n}$ een inverteerbare geheeltallige matrix zijn met $|\det(U)| = 1$. Dan uit $UU^{-1} = I$ en de Gegeneraliseerde Regel van Cramer volgt: U^{-1} is geheeltallig.

Bewijs: Voor iedere $i, j \in \{1, ..., n\}$ geldt met de Gegeneraliseerde Regel van Cramer:

$$\det\left(U^{-1} \begin{bmatrix} i \\ \cdots \\ j \end{bmatrix}\right) = \det\left(U_I \begin{bmatrix} i \\ \cdots \\ j \end{bmatrix}\right) / \det(U)$$

Maar als we dan de linkerkant omschrijven:

$$\begin{aligned} \det\left(U^{-1} \begin{bmatrix} i \\ \cdots \\ j \end{bmatrix}\right) &= \det((U^{-1})_{ij}) \\ &= (U^{-1})_{ij} \end{aligned}$$

dan kunnen we laten zien dat ieder element van de inverse een geheel getal is. Alle elementen van U en I zijn geheel, dus:

$$U_I \begin{bmatrix} i \\ \cdots \\ j \end{bmatrix} \in \mathbb{Z}^{n \times n}$$

is een geheelgetallig matrix. Dan is de determinant van deze matrix ook geheeltallig. In het bijzonder geldt dan:

$$\left| \det \left(U_I \begin{bmatrix} i \\ \cdots \\ j \end{bmatrix} \right) / \det(U) \right| = \left| \det \left(U_I \begin{bmatrix} i \\ \cdots \\ j \end{bmatrix} \right) \right| \in \mathbb{Z}$$

omdat $|\det(U)| = 1$. En dus: $(U^{-1})_{ij} \in \mathbb{Z}$ voor alle $i, j \in \{1, ..., n\}$. Dus $U^{-1} \in \mathbb{Z}^{n \times n}$. ■

Dit lemma gebruiken we om het volgende lemma te bewijzen.

Een speciale eigenschap van roosters is dat ze gerepresenteerd kunnen worden door verschillende bases. Het is daarvoor belangrijk om te weten hoe deze bases eruitzien ten opzichte van elkaar. Daarvoor introduceren we het volgende lemma:

Lemma 3.6
Zij $U \in \mathbb{Z}^{n \times n}$ een unimodulaire matrix. Dat wil zeggen, zodat $|\det(U)| = 1$. Dan geldt voor alle $B \in \mathbb{R}^{n \times n}$

$$\mathcal{L}(B) = \mathcal{L}(BU)$$

Bewijs: ($\subseteq$) Zij $x \in \mathcal{L}(B)$. Dan zijn er $x_1, ..., , x_n \in \mathbb{Z}$ zodat $x = \sum_{i=1}^{n} x_i b_i$ waar b_i voor $1 \leq i \leq n$ de basisvectoren van B. Met andere woorden: $x = B(x_1, ..., x_n)$. Er geldt: $|\det(U)| = 1$, dus in het bijzonder is U inverteerbaar, en $U^{-1} \in \mathbb{Z}^{n \times n}$ met $|\det(U^{-1})| = 1$ (want: $\det(U) \cdot \det(U^{-1}) = \det(UU^{-1}) = \det(I) = 1$ dus $\det(U^{-1}) = \pm 1$, en met Lemma 3.5 volgt dat $U^{-1} \in \mathbb{Z}^{n \times n}$ een geheeltallige matrix is). Dan: $B(UU^{-1}) = B$, dus $B(x_1, ..., x_n) = B(UU^{-1})(x_1, ..., x_n)$.
Dus: $B(UU^{-1})(x_1, ..., x_n) = BU(U^{-1}(x_1, ..., x_n)) = BU(y_1, .., y_n)$ zodat $y_i \in \mathbb{Z}$ voor alle $1 \leq i \leq n$, want $y_i = \sum_{j=1}^{n} (U^{-1})_{ij} x_j$ en $U_{ij}, x_j \in \mathbb{Z}$. Dus $x \in \mathcal{L}(BU)$.

($\supseteq$) Zij $x \in \mathcal{L}(BU)$. Dan op dezelfde manier: er zijn $y_1, ..., y_n \in \mathbb{Z}$ zodat $x = BU(y_1, ...y_n)$. Dan $(x_1, ..., x_n) = U(y_1, ..., y_n) \in \mathbb{Z}^n$, omdat U

unimodulair. Dan volgt direct: $x = BU(y_1, ..., y_n) = B(x_1, ..., x_n)$ en dus $x \in \mathcal{L}(B)$. ■

Dit lemma gebruiken we om de volgende stelling te bewijzen:

Stelling 3.7
Voor alle bases $B, B' \in \mathbb{R}^{n\times n}$ geldt: $\mathcal{L}(B) = \mathcal{L}(B')$ dan en slechts dan als er een unimodulaire matrix U is zodat $B' = BU$.

Bewijs: ($\Leftarrow$) Volgt direct uit Lemma 3.6.

($\Rightarrow$) Stel $\mathcal{L}(B) = \mathcal{L}(B')$. Dus voor iedere $x \in \mathbb{Z}^n$ is er een $y \in \mathbb{Z}^n$ zodat $Bx = B'y$. In het bijzonder is er dan een $U' \in \mathbb{Z}^{n\times n}$ zodat $B = B'U'$, want: voor iedere $j \in \{1, ..., n\}$ is er een $y_j = (y_{1j}, ..., y_{nj}) \in \mathbb{Z}^n$ zodat $B(e_j) = B'(y_j)$ (waarvoor e_j de eenheidsvectoren), en dus $(B(e_j))_i = B_{ij}$ en $(B'(y_j))_i = \langle \hat{b}'_i, y_j \rangle$ voor alle $i \in \{1, ..., n\}$, waar $\hat{b}'_i$ de rijvectoren van B' zijn. Laat $U' \in \mathbb{Z}^{n\times n}$ zijn zodat $(U')_{ij} = y_{ij}$ voor iedere $i, j \in \{1, ..., n\}$. Dan $B = B'U'$, want voor alle $i', j' \in \{1, ..., n\}$ geldt dat

$$\begin{aligned}(B'U')_{i'j'} &= \sum_{i=1}^{n}\sum_{j=1}^{n}(B')_{i'j}(U')_{ij'} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}(B')_{i'j}y_{ij'} \\ &= \langle \hat{b}'_{i'}, y_{j'} \rangle \\ &= B_{i'j'}\end{aligned}$$

En op dezelfde manier is er een $U \in \mathbb{Z}^{n\times n}$ zodat $B' = BU$. Dan: $B'U' = (BU)U' = B(UU')$ en dus met $B(UU') = B$ volgt dat $UU' = I$ (dus: $U^{-1} = U'$). Dan: $\det(UU') = \det(I) = 1$, oftewel: $\det(U) \cdot \det(U') = 1$, maar zowel U als U' zijn geheeltallig, dus $\det(U), \det(U') \in \mathbb{Z}$. En dan volgt dat ofwel $\det(U) = \det(U') = 1$ ofwel $\det(U) = \det(U') = -1$. Dus $|\det(U)| = 1$, en dus U unimodulair. ■

Voor het rekenen met roosters is het handig om te weten wat de determinant van een rooster is.

Definitie 3.8
Zij $B \in \mathbb{R}^{n\times n}$ een basis, en $\mathcal{L} = \mathcal{L}(B)$ het bijbehorende rooster. We definiëren de determinant, genoteerd als $\det(\mathcal{L})$ *als volgt:*

$$\det(\mathcal{L}) = |\det(B)|$$

Merk op: omdat bewezen is dat voor bases B, B' zodat $\mathcal{L}(B) = \mathcal{L}(B')$ geldt dat er een unimodulaire matrix U is zodat $B' = BU$ weten we dus $\det(\mathcal{L}(B')) = |\det(B')| = |\det(BU)| = |\det(B) \cdot \det(U)| = |\det(B)| \cdot |\det(U)| = |\det(B)|$.

Herinnering: De Gram-Schmidtmethode is een algoritme dat gebruikt wordt om een basis $b_1, ..., b_n$ om te zetten in een georthogonaliseerde basis.

Algoritme 3.9 (Gram-Schmidtmethode)
Laat $b_1, ..., b_n \in \mathbb{R}^n$ lineair onafhankelijke vectoren zijn. Definieer b_i^* voor alle i als volgt:

- $b_1^* = b_1$ *en*
- *voor iedere $i \in \{2, ..., n\}$ is*

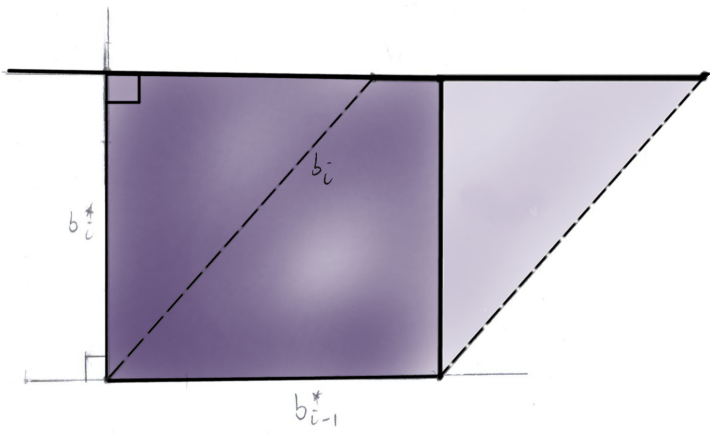
$$b_i^* = b_i - \sum_{j=1}^{i-1} \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} b_j^*$$

Dan is $b_1^*, ..., b_n^*$ de georthogonaliseerde projectie van $b_1, ..., b_n$.

Merk op: we noteren ook wel $\mu_{ij} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}$.

Ook geldt dat $\|b_i\| \geq \|b_i^*\|$, omdat b_i^* in tegenstelling tot b_i wel altijd een orthogonaal stelsel $b_1^*, ..., b_i^*$ vormt en de oppervlakte van het parallellogram dat opgespannen wordt door b_i en b_j^* is net zo groot als de oppervlakte van het parallellogram dat opgespannen wordt door b_i^* en b_j^* (zie Figuur 1).

Dan is de oppervlakte van dit parallellogram gelijk aan $\|b_i^*\| \cdot \|b_j^*\|$ en er geldt dat deze b_i^* de kortste vector is die met b_j^* een parallellogram opspant die dezelfde oppervlakte heeft als de parallellogram die opgespannen wordt door b_i en b_j^*. Dus daarom geldt $\|b_i\| \geq \|b_i^*\|$.



Figuur 1.

In sommige literatuur [14] wordt de determinant van een rooster iets anders gedefinieerd dan in Definitie 3.8. Deze definitie is handig om iets beter te begrijpen wat een orthogonaliteitsafwijking precies is (en in het bijzonder wordt het rekenen met de orthogonaliteitsafwijking gemakkelijker).

Definitie 3.10
Zij $B \in \mathbb{R}^{n\times n}$ een basis, en $\mathcal{L} = \mathcal{L}(B)$ het bijbehorende rooster. Dan is de determinant van dit rooster gelijk aan het volume van de ruimte die wordt ingesloten door de kolomvectoren van B. Dus:

$$\det(\mathcal{L}) = \prod_{i=1}^{n} \|b_i^*\|$$

Merk op: dit komt van het op de vorige pagina 'bewezen' feit dat b_i en b_j^* dezelfde oppervlakte opspannen als b_i^* en b_j^*. Echter, b_i^* en b_j^* staan loodrecht op elkaar, dus de oppervlakte van dit parallellogram is $\|b_i^*\| \cdot \|b_j^*\|$.

In het cryptosysteem dat we nog moeten definiëren in het volgende hoofdstuk is het belangrijk dat we weten hoe de bases van het rooster waar we in werken eruit moeten zien om te voldoen aan de eisen die versleuteling of juist alleen ontsleuteling mogelijk te maken.

Hiervoor introduceren we twee begrippen: de orthogonaliteitsafwijking en de duale orthogonaliteitsafwijking van een basis.

Definitie 3.11
Zij $B \in \mathbb{R}^{n \times n}$ een matrix met $\det(B) \neq 0$. De orthogonaliteitsafwijking van B is gedefinieerd als volgt:

$$OD(B) := \frac{\prod_i \|b_i\|}{|\det(B)|}$$

waarbij $\|b_i\|$ de Euclidische norm van de i-de kolom in B.

Dan kunnen we hierbij het volgende lemma bewijzen:

Lemma 3.12
$OD(B) = 1$ dan en slechts dan als de basisvectoren van B (dus de kolommen van B) een orthogonaal stelsel vormen.

Bewijs: $\mathrm{OD}(B) = 1$ dan en slechts dan als $\frac{\prod_i \|b_i\|}{|\det(B)|} = 1$ dan en slechts dan als $\prod_i \|b_i\| = |\det(B)| = \prod_i \|b_i^*\|$. Maar dan omdat $\|b_i\| \geq \|b_i^*\|$ geldt, moet dus wel gelden dat $b_i = b_i^*$ voor alle i. En andersom geldt ook: als $b_i = b_i^*$ voor alle i, dan $\prod_i \|b_i\| = \prod_i \|b_i^*\|$. ■

De definitie van duale orthogonaliteitsafwijking is gelinkt aan deze definitie, maar voordat we toe zijn aan deze definitie, moeten we eerst weten wat een duaal rooster is.

Als $B = [b_1, ..., b_n]$ een basis is voor een rooster, dan is het duale rooster van $\mathcal{L}(B)$ het rooster dat opgespannen wordt door de rijen van B^{-1}. Met andere woorden, $D = (B^{-1})^t$ is de roosterbasis van het duale rooster van $\mathcal{L}(B)$, genoteerd als $\mathcal{L}(D)$.

Dan definiëren we de duale orthogonaliteitsafwijking als volgt:

Definitie 3.13
Zij $B \in \mathbb{R}^{n \times n}$ een matrix met $\det(B) \neq 0$. De duale orthogonaliteitsafwijking van B is gedefinieerd als volgt:

$$OD^*(B) := \frac{\prod_i \|\hat{b}_i\|}{|\det(B^{-1})|} = |\det(B)| \prod_i \|\hat{b}_i\|$$

waar $\hat{b}_i$ de i-de rij van B^{-1} is.

Deze definitie is belangrijk bij de keuze van de bases die we kiezen in het cryptosysteem dat we definiëren in het volgende hoofdstuk.

3.3 Veelvoorkomende problemen met roosters

De meest-voorkomende problemen met betrekking tot roosters zijn de volgende — al eerder genoemde — problemen.

Shortest Vector Problem (SVP): Gegeven een roosterbasis B, vind de kortste vector $v \neq 0$ in $\mathcal{L}(B)$.

Closest Vector Problem (CVP): Gegeven een roosterbasis B en een doelvector t (die meestal niet in het rooster ligt), vind het roosterpunt $v \in \mathcal{L}(B)$ dat het dichtst bij t ligt.

Shortest Independent Vectors Problem (SIVP): Gegeven een roosterbasis $B \in \mathbb{Z}^{n \times n}$, vind n lineair onafhankelijke roostervectoren $s_1, ..., s_n$ zodat $S = [s_1, ..., s_n]$ en $s_i \in \mathcal{L}(B)$ voor alle $1 \leq i \leq n$ zodat $\|S\| = \max_i \|s_i\|$ minimaal.

In dit geval gaan we vooral kijken naar CVP, omdat het GGH cryptosysteem gebaseerd is op de moeilijkheid van CVP.

3.4 Moeilijkheid van CVP

Om het GGH cryptosysteem in te leiden, is het handig om te weten waarom CVP nou precies zo moeilijk is. Deze sectie gaat over de bewijsbaarheid van de moeilijkheid van CVP. In [14] laten Micciancio en Goldwasser zien dat we voor elk van de volgende drie formuleringen een instantie van de daaropvolgende formulering kunnen maken met behulp van een polynomiale-tijdsreductie. Deze formuleringen zijn:

- **Beslissingsversie**: Gegeven een roosterbasis $B \in \mathbb{Z}^{n \times n}$ en een doelvector $t \in \mathbb{R}^n$ en een $r \in \mathbb{Q}$, beslis of $\text{dist}(t, B) \leq r$ of $\text{dist}(t, B) > r$.
- **Optimalisatieversie**: Gegeven een roosterbasis $B \in \mathbb{Z}^{n \times n}$ en een doelvector $t \in \mathbb{R}^n$, bereken $\text{dist}(t, B)$.
- **Zoekversie**: Gegeven een roosterbasis $B \in \mathbb{Z}^{n \times n}$ en een doelvector $t \in \mathbb{R}^n$, vind een roostervector Bx zodat $\|Bx - t\|$ minimaal.

We zijn al bekend met de laatste versie: de zoekversie. Dit is CVP zoals we deze gedefinieerd hebben in de vorige subsectie.

Er is een polynomiale-tijdsreductie algoritme van de optimalisatieversie naar de zoekversie: stel er is een oplossing voor de zoekversie, dus Bx zodat $\|Bx - t\|$ minimaal, dan kunnen we $\text{dist}(t, B) = \|Bx - t\|$ uitrekenen.

Zo ook voor beslissingsversie naar optimalisatieversie: stel er is een oplossing voor de optimalisatieversie, namelijk $d = \text{dist}(t, B)$. Dan kunnen we bepalen of $d \leq r$ of $d > r$.

Stelling 3.14
De beslissingsversie van CVP is NP-volledig in de Euclidische norm.

Bewijs: In [14] wordt er met behulp van polynomiale-tijdsreductie laten zien dat het deelsomprobleem (Engels: the subset sum problem) getransformeerd kan worden naar een instantie van de beslissingsversie van CVP in de ℓ_p-norm voor alle $p \in \mathbb{N}$. Dus er geldt dat CVP minstens net zo moeilijk is als het deelsomprobleem. Van het deelsomprobleem is bewezen dat deze NP-volledig is [6].

Het deelsomprobleem vraagt voor een gegeven verzameling van gehele getallen of er een niet-lege deelverzameling is waarvan de sommatie over deze verzameling gelijk is aan nul. Dus: voor een $H \subseteq \mathbb{Z}$, is er een $G \subseteq H$, $G \neq \emptyset$, zodat $\sum_{g \in G} g = 0$?

Omdat de beslissingsversie van CVP NP-volledig is voor iedere ℓ_p-norm is de beslissingsversie van CVP ook NP-volledig in de ℓ_2-norm, oftewel de Euclidische norm.

Op pagina 48 en 49 van [14] staat het volledige bewijs. ■

Dan volgt met Gevolg 2.7 dat de optimalisatieversie van CVP NP-moeilijk is, en in het bijzonder dat CVP zelf NP-moeilijk is:

Gevolg 3.15
CVP is NP-moeilijk.

Overigens wordt in [14] ook bewezen dat SVP NP-moeilijk is, en bovendien dat CVP minstens net zo moeilijk is als SVP. Dit gaan we gebruiken in sectie 5.1.

4 Introductie van het GGH cryptosysteem

In deze sectie gaan we het GGH cryptosysteem, geintroduceerd door Oded Goldreich, Shafi Goldwasser en Shai Halevi in 1997, bekijken. De moeilijkheid van dit cryptosysteem is gebaseerd op de moeilijkheid van CVP. GGH is waarschijnlijk — van alle cryptosystemen die op dit moment bekend zijn — een van de meest intuïtieve cryptosystemen die gebruik maken van roosters, wat het aantrekkelijk maakt om dit cryptosysteem te bekijken.

CVP wordt toegepast in GGH door gebruik te maken van een zogenaamde 'trapdoor function'. Dit is een functie — ook wel eenwegsfunctie genoemd[4] — waarvan alleen de ene kant op de uitkomst gemakkelijk te berekenen is, maar de andere kant op computationeel moeilijk is zonder de hulp van speciale informatie, en waarvan geen algoritmes bekend zijn die computationeel sneller zijn dan van willekeurige invoeren testen of ze als uitkomst gelijk zijn aan het gezochte antwoord. Dit 'oplossen' van een probleem zonder gebruik te maken van een algoritme heet *brute force*.

Bij een 'trapdoor function' kunnen we bijvoorbeeld denken aan RSA. Deze maakt gebruik van het feit dat een product van twee priemgetallen moeilijk te factoriseren is, en in het bijzonder dat we m niet kunnen achterhalen als we $m^e \bmod n$ uitgerekend hebben. De 'trapdoor function' is in dit geval dan: $f(m, e) = m^e \bmod n$. Hier is $n = pq$ een product van twee grote priemgetallen p en q. De 'trapdoor' in dit geval is $d \in (\mathbb{Z}/n\mathbb{Z})^*$ zodat $e \cdot d \equiv 1 \bmod \varphi(n)$, omdat we deze m alleen kunnen terughalen met behulp van deze d.

Een manier om deze d te achterhalen is om deze uit te rekenen (met bijvoorbeeld het uitgebreide algoritme van Euclides), dus om een d te vinden zodat $e \cdot d \equiv 1 \bmod \varphi(n)$. Echter moet je wel deze $\varphi(n)$ kunnen uitrekenen om deze inverse te bepalen, wat nu net het probleem is. Namelijk: $\varphi(n) = (p-1)(q-1)$, en dus kunnen we deze alleen uitrekenen als we p en q weten, oftewel n kunnen factoriseren[5].

Op een zelfde manier zouden we graag willen dat CVP toegepast kan worden als 'trapdoor function' in een cryptosysteem. In 1997 werd er al een cryptosysteem geïntroduceerd waarvan de veiligheid afgeleid was van

[4] Een 'trapdoor function' is een speciale variant van een eenwegsfunctie, waar omkering mogelijk is onder bepaalde voorwaarden.

[5] Voor zover we weten is $\varphi(n)$ alleen te berekenen met behulp van p en q. Als we aannemen dat we toch $\varphi(n)$ kunnen achterhalen, kunnen we ook p en q achterhalen, en is dus het systeem gebroken. Want: $n = pq$ en $\varphi(n) = (p-1)(q-1)$. Dus we kunnen $p = n/q$ invullen: $\varphi(n) = (n/q - 1)(q - 1)$ en dit is een tweedegraads vergelijking met één onbekende die we gewoon op kunnen lossen met de abc-formule.

een variant van SVP [1]. Echter gebruikte deze geen 'trapdoor function' en was daarom niet echt geschikt om als asymmetrisch cryptosysteem gebruikt te worden. Ook was deze variant van SVP makkelijker dan SVP zelf, en gebruikte het systeem erg grote sleutels en de benodigde tijd om iets te versleutelen was erg lang. Het systeem wordt dus als erg onpraktisch en inefficiënt ervaren.

Wel inspireerde dit Oded Goldreich, Shafi Goldwasser en Shai Halevi om een asymmetrisch cryptosysteem te ontwikkelen dat gebaseerd is op een probleem met roosters, maar in tegenstelling tot Ajtai en Dwork wel een 'trapdoor function' gebruikt om de versleuteling en ontsleuteling te definiëren.

4.1 Het originele GGH cryptosysteem

In deze subsectie wordt het originele GGH cryptosysteem zoals voorgesteld door Goldreich, Goldwasser en Halevi, beschreven.

De geheime sleutel: De geheime sleutel is een 'goede' roosterbasis $R \in \mathbb{R}^{n \times n}$, zodat ontsleuteling alleen mogelijk is met de geheime sleutel.

De publieke sleutel: De publieke sleutel is daarentegen een 'slechte' roosterbasis $B \in \mathbb{R}^{n \times n}$, waarvoor geldt dat er een unimodulaire matrix $U \in \mathbb{Z}^{n \times n}$ bestaat zodat $BU = R$. Met andere woorden, B en R representeren hetzelfde rooster: $\mathcal{L}(B) = \mathcal{L}(R)$ (zie Stelling 3.7).

Versleuteling: Voor de versleuteling willen we een manier om de boodschap die we willen versturen versleutelen als een punt in het rooster $\mathcal{L}(B)$. Daarna tellen we er een korte 'verstoringsvector' e bij op voor de 'echte' versleuteling.

Ontsleuteling: Voor de ontsleuteling keren we deze functie om. We gebruiken hierbij een van de algoritmes die CVP kunnen oplossen. Laat c de gecodeerde boodschap zijn. Dan kunnen we een instantie van CVP vormen die we kunnen oplossen met onze 'trapdoor' (we laten later zien wat dit precies betekent en wat deze 'trapdoor' precies is). Dan kunnen we verstoringsvector e terugvinden door $e = c - Bv$ uit te rekenen, waarbij v de oplossing is die we gevonden hebben met het algoritme dat CVP oplost.

Nu we een idee hebben van hoe dit cryptosysteem er precies uit zou moeten zien, kunnen we specificeren wat 'goede' en 'slechte' bases betekenen, en hoe

de versleuteling en ontsleuteling precies werken, en misschien nog wel belangrijker: waarom. Allereerst kijken we naar wat een basis een 'goede' basis voor een rooster als hulpmiddel van een cryptosysteem inhoudt. Daarvoor introduceren we het begrip 'Lovász-gereduceerde basis'.

4.2 Lovász-gereduceerde bases

In de op roosters gebaseerde cryptografie, en met name de cryptografie die gerelateerd is aan CVP, is het belangrijk dat we weten wat een Lovász-gereduceerde basis is.

Dit is belangrijk, omdat een Lovász-gereduceerde matrix het mogelijk maakt om een oplossing te vinden voor CVP.

In [11] wordt gesproken van een Lovász-reduceerde basis als voor een $B \in \mathbb{R}^{n\times n}$ met $B = [b_1, ..., b_n]$ voor lineair onafhankelijke vectoren $b_1, ..., b_n$ geldt dat

$$|\mu_{ij}| \leq \frac{1}{2} \text{ voor alle } 1 \leq j < i \leq n \tag{1}$$

en

$$\|b_i^* + \mu_{ii-1}b_{i-1}^*\|^2 \geq \tau\|b_{i-1}^*\|^2 \text{ voor alle } 1 < i \leq n \tag{2}$$

waar voor τ elke willekeurige $\frac{1}{4} < \tau < 1$ gekozen kan worden.

Ter herinnering: we hebben in sectie 3.9 deze μ_{ij} gedefinieerd.

In [2] wordt (2) geschreven als:

$$\|b_i^*\| \geq \frac{\|b_{i-1}^*\|}{\sqrt{2}} \text{ voor alle } 1 < i \leq n. \tag{3}$$

Dit komt op hetzelfde neer omdat voor alle $1 < i \leq n$:

$$\begin{aligned}
\|b_i^* + \mu_{ii-1}b_{i-1}^*\|^2 &= \sum_{j=1}^{n}(b_i^* + \mu_{ii-1}b_{i-1}^*)_j^2 \\
&= \sum_{j=1}^{n}(b_i^*)_j^2 + (\mu_{ii-1}(b_{i-1}^*)_j)^2 + 2\mu_{ii-1}(b_i^*)_j(b_{i-1}^*)_j) \\
&= \|b_i^*\|^2 + \|\mu_{ii-1}b_{i-1}^*\|^2 + 2\langle b_i^*, \mu_{ii-1}b_{i-1}^*\rangle \\
&= \|b_i^*\|^2 + |\mu_{ii-1}|^2\|b_{i-1}^*\|^2 + 2\mu_{ii-1}\langle b_i^*, b_{i-1}^*\rangle.
\end{aligned}$$

Maar $\langle b_i^*, b_{i-1}^*\rangle = 0$, omdat b_i^* en b_{i-1}^* orthogonaal zijn. En als we (1) invullen voor μ_{ii-1} krijgen we:

$$\|b_i^*\|^2 + |\mu_{ii-1}|\|b_{i-1}^*\|^2 + 2\mu_{ii-1}\langle b_i^*, b_{i-1}^*\rangle \leq \|b_i^*\|^2 + \frac{1}{4}\|b_{i-1}^*\|^2.$$

Dus we krijgen de vergelijking:

$$\|b_i^* + \mu_{ii-1} b_{i-1}^*\|^2 \leq \|b_i^*\|^2 + \frac{1}{4}\|b_{i-1}^*\|^2.$$

En dus geldt met (2):

$$\|b_i^*\|^2 + \frac{1}{4}\|b_{i-1}^*\|^2 \geq \tau \|b_{i-1}^*\|^2$$
$$\|b_i^*\|^2 \geq (\tau - \frac{1}{4})\|b_{i-1}^*\|^2.$$

Dan voor $\tau = 3/4$ hebben we:

$$\|b_i^*\|^2 \geq \left(\frac{3}{4} - \frac{1}{4}\right)\|b_{i-1}^*\|^2 = \frac{1}{2}\|b_{i-1}^*\|^2.$$

Dan:

$$\|b_i^*\| \geq \frac{1}{\sqrt{2}}\|b_{i-1}^*\|.$$

En dat is precies (3). Dus we hebben laten zien dat vergelijking (2) vergelijking (3) impliceert. Echter is omgekeerd niet per se het geval. We kunnen dus zeggen dat (3) een zwakkere eis is dan (2). We laten voor een Lovász-gereduceerde basis (1) en (3) gelden, omdat onder deze aannames de algoritmes van Babai gebruikt kunnen worden.

4.2.1 Orthogonaliteitsafwijking

In Definitie 3.11 hebben we gedefinieerd wat orthogonaliteitsafwijking is, en dat dit impliciet een belangrijk begrip is dat gerelateerd is aan Lovász-gereduceerde bases. In de introductie van GGH [8] wordt in plaats van de vergelijkingen die hierboven gegeven zijn, het begrip orthogonaliteitsafwijking gebruikt. Daar wordt de geheime sleutel gekozen als een roosterbasis met een kleine duale orthogonaliteitsafwijking, en de publieke sleutel juist een grote.

Omdat we gebruik maken van de algoritmes van Babai, en deze alleen kunnen gebruiken als er de vergelijkingen gelden voor de geheime sleutel, willen we dus wel aannemelijk maken dat deze vergelijkingen gelden voor een basis met kleine duale orthogonaliteitsafwijking.

Eerst gaan we bekijken wat dat precies betekent: 'een kleine duale orthogonaliteitsafwijking'. Laat $B \in \mathbb{R}^{n \times n}$ een matrix zijn met kleine duale orthogonaliteitsafwijking. Dus: $\mathrm{OD}^*(B) = |\det(B)| \prod_i \|\hat{b}_i\|$ is klein. We

weten: $|\det(B)| = \prod_i \|b_i^*\|$ ligt vast – want de determinant van een rooster is onafhankelijk van de keuze van de basis, dus als $\mathrm{OD}^*(B)$ klein is, dan komt dat omdat $\prod_i \|\hat{b}_i\|$ klein is.

Nu geldt dat $\|\hat{b}_i\|$ minimaal is als $\|\hat{b}_i\| = \|\hat{b}_i^*\|$. Dus als $\hat{b}_1, ..., \hat{b}_n$ een orthogonaal stelsel vormt dan is $\prod_i \|\hat{b}_i\| = \prod_i \|\hat{b}_i^*\|$ minimaal. Dus hoe meer geldt dat $\hat{b}_1, ..., \hat{b}_n$ een 'orthogonaal stelsel vormt', hoe kleiner $\prod_i \|\hat{b}_i\|$ en dus $\mathrm{OD}^*(B)$ is.

Merk op: R heeft een kleine duale orthogonaliteitsafwijking dan en slechts dan als $\left(R^{-1}\right)^t$ een kleine orthogonaliteitsafwijking heeft. We willen dat R^{-1} CVP oplost en dus Lovász-gereduceerd is, dus het volstaat om te laten zien dat voor een Lovász-gereduceerde matrix B geldt dat de orthogonaliteitsafwijking klein is.

We willen dus dat met de vergelijkingen (1) en (3) ook volgt dat $b_1, ..., b_n$ 'bijna een orthogonaal stelsel vormen'. Dus laat $b_1, ..., b_n$ de kolommen van de basis B zijn waarvoor vergelijkingen (1) en (3) gelden. Dan:

$$|\mu_{ij}| \leq \frac{1}{2} \text{ voor alle } 1 \leq j < i \leq n$$

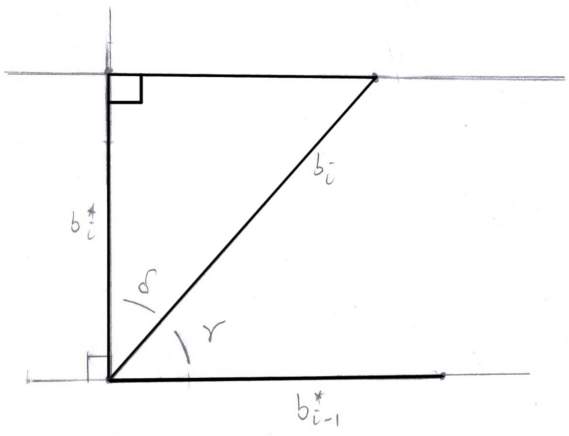
We kiezen $j = i - 1$. We gebruiken dat $\langle b_i, b_{i-1}^* \rangle = \|b_i\| \|b_{i-1}^*\| \cos(\gamma)$, waarvoor γ de hoek die b_i met b_{i-1} maakt is. Dan:

$$\begin{aligned} |\mu_{ii-1}| &= \left| \frac{\langle b_i, b_{i-1}^* \rangle}{\langle b_{i-1}^*, b_{i-1}^* \rangle} \right| \\ &= \left| \frac{\|b_i\| \|b_{i-1}^*\| \cos(\gamma)}{\|b_{i-1}^*\|^2} \right| \\ &= \left| \frac{\|b_i\| \cos(\gamma)}{\|b_{i-1}^*\|} \right| \end{aligned}$$

Dus: $\left| \frac{\|b_i\| \cos(\gamma)}{\|b_{i-1}^*\|} \right| \leq \frac{1}{2}$. Dan, als we $\|b_{i-1}^*\|$ naar de andere kant brengen en $\|b_i^*\| \geq \frac{\|b_{i-1}^*\|}{\sqrt{2}}$ invullen krijgen we:

$$\begin{aligned} |\|b_i\| \cos(\gamma)| &\leq \frac{\|b_{i-1}^*\|}{2} \\ &\leq \frac{\|b_i^*\| \sqrt{2}}{2} \end{aligned}$$

Nu kunnen we kijken hoe b_i, b_{i-1}^* en b_i^* zich tot elkaar verhouden, en met name wat de hoeken tussen b_i en b_{i-1}^* en b_i en b_{i-1}^* zijn.



Figuur 2.

Merk op: γ is de hoek die b_i maakt met b^*_{i-1}. Van b^*_i en b^*_{i-1} weten we dat de hoek $\frac{1}{2}\pi$, oftewel 90°, is. Dus dan is $\delta = \frac{1}{2}\pi - \gamma$ de hoek die b_i maakt met b^*_i (zie Figuur 2.). Er geldt: $\cos(\delta) = \frac{\|b^*_i\|}{\|b_i\|}$. Dan kunnen we deze vergelijking weer invullen:

$$\begin{aligned} |\|b_i\| \cos(\gamma)| &\leq \frac{\|b^*_i\|\sqrt{2}}{2} \\ |\cos(\gamma)| &\leq \frac{\|b^*_i\|\sqrt{2}}{2\|b_i\|} \\ |\cos(\gamma)| &\leq \frac{\sqrt{2}}{2}\cos(\delta) \end{aligned}$$

Dan kunnen we $\cos(\delta)$ vervangen door $\sin(\gamma)$, want $\cos(\delta) = \cos(\frac{1}{2}\pi - \gamma) = \sin(\gamma)$. Dus we krijgen een uitdrukking met slechts één onbekende:

$$|\cos(\gamma)| \leq \frac{\sqrt{2}}{2}\sin(\gamma)$$

We bekijken γ op het interval $[-\pi, \pi]$. Voor $\gamma \in [-\frac{1}{2}\pi, \frac{1}{2}\pi]$ geldt dat $\cos(\gamma) \geq 0$, en voor $\gamma \in [-\pi, -\frac{1}{2}\pi) \cup (\frac{1}{2}\pi, \pi]$ geldt dat $\cos(\gamma) < 0$.

Dus voor $\gamma \in [-\frac{1}{2}\pi, \frac{1}{2}\pi]$:

$$\begin{aligned}
\cos(\gamma) &\leq \frac{\sqrt{2}}{2}\sin(\gamma) \\
\frac{2}{\sqrt{2}} &\leq \frac{\sin(\gamma)}{\cos(\gamma)} \\
\sqrt{2} &\leq \tan(\gamma) \\
\arctan(\sqrt{2}) &\leq \gamma
\end{aligned}$$

En voor $\gamma \in [-\pi, -\frac{1}{2}\pi) \cup (\frac{1}{2}\pi, \pi]$:

$$\begin{aligned}
-\cos(\gamma) &\leq \frac{\sqrt{2}}{2}\sin(\gamma) \\
\cos(\gamma - \pi) &\leq -\frac{1}{\sqrt{2}}\sin(\gamma - \pi) \\
-\sqrt{2} &\geq \frac{\sin(\gamma - \pi)}{\cos(\gamma - \pi)} \\
-\sqrt{2} &\geq \tan(\gamma - \pi) \\
\arctan(-\sqrt{2}) &\geq \gamma - \pi \\
\arctan(-\sqrt{2}) + \pi &\geq \gamma
\end{aligned}$$

Dus voor γ met

$$\arctan(\sqrt{2}) \leq \gamma \leq \arctan(-\sqrt{2}) + \pi$$

geldt dat er voldaan wordt aan de vergelijkingen. Als we dit omzetten in graden, krijgen we, bij benadering:

$$55^\circ \leq \gamma \leq 125^\circ$$

En dus mag de hoek die b_i maakt met de georthogonaliseerde projectie b_i^* maximaal 35° zijn. Dus:

$$\delta \leq \arctan(-\sqrt{2}) + \pi - \frac{1}{2}\pi = \arctan(-\sqrt{2}) + \frac{1}{2}\pi$$

En dus vormen $b_1, ..., b_n$ een 'bijna orthogonaal' stelsel en is dus de orthogonaalafwijking klein.

Nu we weten wat de hoek die b_i maakt met b_i^*, kunnen we ook afschatten wat de orthogonaliteitsafwijking van een Lovász-gereduceerde basis B precies mag zijn.

4.2.2 Bovengrens voor de orthogonaliteitsafwijking

We weten dus wat de ondergrens van de orthogonaliteitsafwijking mag zijn voor een Lovász-gereduceerde basis (namelijk 1, in het geval dat alle kolomvectoren van minimale lengte zijn en dus een orthogonaal stelsel vormen[6]). Nu willen we nog een bovengrens bepalen, en in het bijzonder één die volgt uit de vergelijkingen die moeten gelden voor een basis B zodat deze als Lovász-gereduceerd beschouwd mag worden.

We weten we dat voor de hoek δ die b_i met b_i^* maakt geldt:

$$0 \leq \delta \leq \arctan(-\sqrt{2}) + \frac{1}{2}\pi$$

We weten ook dat geldt:

$$\frac{\|b_i^*\|}{\|b_i\|} = \cos(\delta)$$

en op het interval $[0, \arctan(-\sqrt{2}) + \frac{1}{2}\pi]$ is $\cos(\delta)$ dalend. Dus:

$$1 = \cos(0) \geq \frac{\|b_i^*\|}{\|b_i\|} \geq \cos(\arctan(-\sqrt{2}) + \frac{1}{2}\pi) \approx 0,816497..$$

En dus:

$$1 = \frac{1}{\cos(0)} \leq \frac{\|b_i\|}{\|b_i^*\|} \leq \frac{1}{\cos(\arctan(-\sqrt{2}) + \frac{1}{2}\pi)} \approx 1,22474..$$

Dan is $\mathrm{OD}(B)$ gelijk aan:

$$\begin{aligned}
\frac{\prod_i \|b_i\|}{\prod_i \|b_i^*\|} &= \prod_i \frac{\|b_i\|}{\|b_i^*\|} \\
&\leq \prod_i \frac{1}{\cos(\arctan(-\sqrt{2}) + \frac{1}{2}\pi)} \\
&= \left(\frac{1}{\cos(\arctan(-\sqrt{2}) + \frac{1}{2}\pi)}\right)^n \\
&\approx 1,2247^n
\end{aligned}$$

Dus nu hebben we een begrenzing van $\mathrm{OD}(B)$ zodat als $1 \leq \mathrm{OD}(B) \leq 1,2247^n$ geldt, dan is B een Lovász-gereduceerde basis (en vice versa).

[6] Want dan geldt $b_i = b_i^*$ voor alle i en dus $\mathrm{OD}(B) = \frac{\prod_i \|b_i\|}{\prod_i \|b_i^*\|} = \frac{\prod_i \|b_i^*\|}{\prod_i \|b_i^*\|} = 1$. We hadden al laten zien dat $b_i = b_i^*$ minimale lengte heeft, dus dan is ook $\mathrm{OD}(B)$ minimaal.

4.3 Reductiealgoritmes van Babai

In beide procedures die geïntroduceerd worden in deze subsectie nemen we het volgende aan:

Aanname: Laat $B \in \mathbb{R}^{n\times n}$ een Lovász-gereduceerde basis zijn van rooster $\mathcal{L}(B)$, en $x \in \mathbb{R}^n$ een punt in $\mathbb{R}^n$. Dan $x = \sum_{i=1}^{n} x_i b_i$ (waar b_i de kolomvectoren van B zijn) voor zekere $x_i \in \mathbb{R}$ (want $b_1, ..., b_n$ onafhankelijke vectoren en vormen dus een basis voor $\mathbb{R}^n$).

4.3.1 De afrondprocedure van Babai

Een van de twee algoritmes die Babai introduceerde is de afrondprocedure:

Algoritme 4.1 (Afrondprocedure van Babai)
Laat $y_i = \lfloor x_i \rceil$ voor alle $1 \leq i \leq n$. Dan $y = \sum_{i=1}^{n} y_i b_i \in \mathcal{L}(B)$ het roosterpunt dat bij benadering het dichtst bij x ligt.

Deze procedure vindt niet altijd het dichtstbijzijnde roosterpunt, maar in sommige gevallen vindt deze procedure een punt dat wel bijna het dichtstbijzijnde punt is. Babai bewees daarbij de volgende stelling:

Stelling 4.2
Als basis B gereduceerd is, dan vindt de afrondprocedure een roosterpunt $y \in \mathcal{L}(B)$ zodat

$$\|x - y\| \leq \left(1 + 2n\left(\frac{9}{2}\right)^{\frac{n}{2}}\right)\|x - y'\|$$

voor alle $y' \in \mathcal{L}(B)$.

Bewijs: Zie [2] voor het bewijs. ■

4.3.2 De dichtstbijzijnde-vlakprocedure van Babai

Het andere algoritme is iets minder simpel dan de eerste en vereist duidelijk meer werk dan de vorige methode.

Algoritme 4.3 (Dichtstbijzijnde-vlakprocedure van Babai)
Laat $U = \sum_{i=1}^{n-1} \mathbb{R}b_i$ de lineaire deelruimte gegenereerd door $b_1, ..., b_{n-1}$ zijn en laat $\mathcal{L}'([b_1, ..., b_{n-1}])$ het corresponderende deelrooster van $\mathcal{L}(B)$ zijn.

Vind $v \in \mathcal{L}(B)$ zodat de afstand tussen x en $U + v$ minimaal is, dus zodat voor alle $v' \in \mathcal{L}(B)$ geldt dat: $\|(U + v) - x\| \leq \|(U + v') - x\|$. Laat

x' de orthogonale projectie van x op $U + v$ zijn. Vind dan, met recursie, $y \in \mathcal{L}'([b_1, ..., b_{n-1}])$ zodat y het dichtst bij $x' - v$ ligt. En laat $z = y + v$ het roosterpunt in $\mathcal{L}(B)$ dat bij benadering het dichtst bij x ligt.

Merk op: om v en x' te vinden, moeten we x als lineaire combinatie van de georthogonaliseerde basisvectoren schrijven: $x = \sum_{i=1}^{n} \alpha_i b_i^*$. Laat $\beta = \lfloor \alpha_n \rceil$ de afronding van α_n zijn. Dan is $x' = \sum_{i=1}^{n-1} \alpha_i b_i^* + \beta b_n^*$ en $v = \beta b_n$.

Net zoals bij de vorige procedure vindt deze procedure niet altijd het dichtstbijzijnde roosterpunt.

Stelling 4.4
Als basis B gereduceerd is, dan vindt de dichtstbijzijnde-vlakprocedure een roosterpunt $z \in \mathcal{L}(B)$ zodat

$$\|x - z\| \ \leq 2^{n/2} \|x - z'\|$$

voor alle $z' \in \mathcal{L}(B)$. Bovendien geldt: $|x - z| < 2^{n/2-1} \|b_n^*\|$.

Bewijs: Zie [2] voor het bewijs. ■

4.4 De geheime sleutel en ontsleuteling

Zoals genoemd is de geheime sleutel $R \in \mathbb{R}^{n \times n}$ een 'goede' basis voor het rooster waarin we werken. Een 'goede' basis betekent dat deze genoeg informatie bevat om een inversie van de versleuteling mogelijk te maken (in tegenstelling tot de publieke sleutel, welke een basis zou moeten zijn die de omkering van de functie erg moeilijk maakt).

In dit geval is de omkering van de versleuteling equivalent aan het zoeken naar een punt $R^{-1}v \in \mathcal{L}(R^{-1}) = \mathcal{L}(B^{-1})$ dat het dichtst bij $R^{-1}c \in \mathbb{R}^n$ ligt. Dus CVP moet opgelost worden voor punt $R^{-1}c = R^{-1}(Bv + e)$. We gebruiken hierbij de afrondprocedure van Babai. De aanname is dat R^{-1} een Lovász-gereduceerde basis is, want de duale orthogonaliteitsafwijking van R is klein, en dus — zoals we hebben laten zien in subsectie 4.2.1 — is R^{-1} een Lovász-gereduceerde basis. Dus voor $R^{-1}c$ kunnen we het dichtstbijzijnde punt in $\mathcal{L}(R^{-1})$ vinden met de afrondprocedure. We schrijven daarvoor c als lineaire combinatie van vectoren $r_1, ..., r_n$ (dus $c = \sum_{i=1}^{n} \alpha_i r_i$ voor $\alpha_i \in \mathbb{R}$), waar deze r_i de kolomvectoren van R zijn. En met

$$\lfloor R^{-1}c \rceil = \begin{pmatrix} \lfloor \langle \hat{r}_1, c \rangle \rceil \\ \vdots \\ \lfloor \langle \hat{r}_n, c \rangle \rceil \end{pmatrix}$$

waarvoor $\hat{r}_1, ..., \hat{r}_n$ de rijvectoren van R^{-1} zijn, krijgen we

$$\begin{aligned}\lfloor R^{-1}c \rceil &= \left\lfloor R^{-1}\left(\sum_{i=1}^{n}\alpha_i r_i\right)\right\rceil \\ &= \begin{pmatrix} \lfloor \langle \hat{r}_1, \sum_{i=1}^{n}\alpha_i r_i\rangle \rceil \\ \vdots \\ \lfloor \langle \hat{r}_n, \sum_{i=1}^{n}\alpha_i r_i\rangle \rceil \end{pmatrix}.\end{aligned}$$

Dan $\langle \hat{r}_j, \sum_{i=1}^{n}\alpha_i r_i\rangle = \sum_{i=1}^{n}\alpha_i\langle \hat{r}_j, r_i\rangle$ en omdat $R^{-1}R = I$ weten we dat $(R^{-1}R)_{ji} = \sum_k (R^{-1})_{jk}R_{ki} = \langle \hat{r}_j, r_i\rangle = I_{ij}$. En daaruit volgt dat $\langle \hat{r}_j, r_i\rangle = 1$ als $j = i$ en $\langle \hat{r}_j, r_i\rangle = 0$ als $j \neq i$. Dus:

$$\langle \hat{r}_j, \sum_{i=1}^{n}\alpha_i r_i\rangle = \sum_{i=1}^{n}\alpha_i\langle \hat{r}_j, r_i\rangle = \alpha_j.$$

En dus:

$$\lfloor R^{-1}c \rceil = \begin{pmatrix} \lfloor \alpha_1 \rceil \\ \vdots \\ \lfloor \alpha_n \rceil \end{pmatrix} = \lfloor \alpha \rceil$$

waar $\alpha = (\alpha_1, ..., \alpha_n)$. Omdat $\mathcal{L}(B) = \mathcal{L}(R)$ hebben we een unimodulaire U zodat $BU = R$ en dus $U = B^{-1}R$. Dan kunnen we $U\lfloor R^{-1}c \rceil$ uitrekenen:

$$\begin{aligned}U\lfloor R^{-1}c \rceil &= U\lfloor \alpha \rceil \\ &= B^{-1}(R\lfloor \alpha \rceil) \\ &= B^{-1}\left(\sum_{i=1}^{n}\lfloor \alpha_i \rceil r_i\right)\end{aligned}$$

en omdat $\sum_{i=1}^{n}\lfloor \alpha_i \rceil r_i \in \mathcal{L}(R) = \mathcal{L}(B)$ kunnen we deze ook schrijven als een punt in $\mathcal{L}(B)$, dus over de basis B. Laat $\beta_i \in \mathbb{Z}$ zijn voor iedere $i \in \{1, ..., n\}$ zodanig dat $\sum_{i=1}^{n}\lfloor \alpha_i \rceil r_i = \sum_{i=1}^{n}\beta_i b_i$. Dan:

$$\begin{aligned}B^{-1}\left(\sum_{i=1}^{n}\lfloor \alpha_i \rceil r_i\right) = B^{-1}\left(\sum_{i=1}^{n}\beta_i b_i\right) &= \begin{pmatrix} \langle \hat{b}_1, \sum_{i=1}^{n}\beta_i b_i\rangle \\ \vdots \\ \langle \hat{b}_n, \sum_{i=1}^{n}\beta_i b_i\rangle \end{pmatrix} \\ = \begin{pmatrix} \sum_{i=1}^{n}\beta_i\langle \hat{b}_1, b_i\rangle \\ \vdots \\ \sum_{i=1}^{n}\beta_i\langle \hat{b}_n, b_i\rangle \end{pmatrix} &= \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix},\end{aligned}$$

omdat $\langle \hat{b}_j, b_i \rangle = 0$ als $j \neq i$ en $\langle \hat{b}_j, b_i \rangle = 1$ als $j = i$ (op dezelfde manier als bij $r_i, \hat{r}_j$). Dan $v = [\beta_1, ..., \beta_n]$ de v waar we naar zochten zodat $c = Bv + e$ en kunnen we e vinden door $e = c - Bv$ uit te rekenen.

4.4.1 De geheime sleutel

Omdat dit proces dus alleen werkt als R een 'goede' basis is, moet R^{-1} wel Lovász-gereduceerd zijn, oftewel R heeft een kleine duale orthogonaliteits-afwijking.

4.4.2 Ontsleuteling

Voor de ontsleuteling van $c \in \mathbb{R}^n$ berekenen we $v = U \left\lfloor R^{-1}c \right\rceil$, waar U de unimodulaire matrix is zodat $BU = R$. We hebben hierboven gezien dat dit werkt voor een 'goede' basis R.

4.5 De publieke sleutel en versleuteling

Voor de versleuteling hebben we een publieke sleutel nodig, die de versleuteling gemakkelijk maakt, maar toch geen informatie weggeeft over de geheime sleutel, of een manier geeft om de versleuteling toch te inverteren. Ook moet deze publieke sleutel een roosterbasis zijn die hetzelfde rooster representeert als de geheime sleutel, dus als $R \in \mathbb{R}^{n \times n}$ de geheime sleutel en $B \in \mathbb{R}^{n \times n}$ de publieke sleutel, dan moet gelden dat $\mathcal{L}(B) = \mathcal{L}(R)$.

Voor de geheime sleutel hebben we een R zodat de ontsleuteling mogelijk is, oftewel deze R^{-1} is een Lovász-gereduceerde basis, oftewel deze R heeft een kleine duale orthogonaliteitsafwijking. In [8] stelden Goldreich, Goldwasser en Halevi dat de publieke sleutel een basis moet zijn met een grote duale orthogonaliteitsafwijking, oftewel een basis die sterk niet-gereduceerd is. Vanzelfsprekend maakt het niet uit wat de vorm is van B om de versleuteling uit te kunnen voeren: $Bv + e$ waarvoor $v \in \mathbb{Z}^n$ de boodschap is en e de verstoringsvector.

Dus voor de publieke sleutel B willen we dat de rijvectoren van de inverse een grote hoek maken met hun Gram-Schmidt georthogonaliseerde projectie. In de introductie van GGH zelf wordt hier niet echt een specifieke methode voor aangegeven, behalve door uit een gegeven R — waarvan we de elementen uit de uniforme verdeling kiezen (zie sectie 4.7) — een B te genereren uit R. Hiervoor zijn twee manieren gegeven in [8] zelf. Echter introduceerde Micciancio [13] een eenduidige manier om deze publieke basis te genereren, en wel door middel van de Hermite normaalvorm (HNF, van het Engelse: Hermite Normal Form). Hierover meer in sectie 4.10.

4.6 Correctheid en verstoringsvector

Nu we weten hoe de sleutels eruit zien, en weten hoe de versleuteling en ontsleuteling werken, kunnen we kijken of deze correct zijn. Oftewel: geeft versleutelen van een boodschap, en daarna deze ontsleutelen wel de juiste boodschap terug, of kan het zijn dat deze anders is in sommige gevallen?

Allereerst zullen we kijken naar wat er gebeurt als een boodschap $v \in \mathbb{Z}^n$ versleuteld en daarna ontsleuteld wordt[7]. Dus:

$$c = \mathrm{Enc}(v) = Bv + e,$$

maar we laten hier nog even buiten beschouwing wat deze verstoringsvector e precies is. Dan ontsleutelen we c:

$$\begin{aligned}
\mathrm{Dec}(c) &= U \left\lfloor R^{-1}c \right\rceil \\
&= U \left\lfloor R^{-1}(Bv + e) \right\rceil \\
&= U \left\lfloor R^{-1}Bv + R^{-1}e \right\rceil \\
&= U \left\lfloor U^{-1}B^{-1}Bv + R^{-1}e \right\rceil \\
&= U \left\lfloor U^{-1}v + R^{-1}e \right\rceil,
\end{aligned}$$

met $BU = R$ voor de unimodulaire matrix U, en dus $R^{-1} = U^{-1}B^{-1}$. We weten ook dat $U^{-1} \in \mathbb{Z}^{n\times n}$ en $v \in \mathbb{Z}^n$, dus $U^{-1}v \in \mathbb{Z}^n$. Maar dan geldt $\left\lfloor U^{-1}v \right\rceil = U^{-1}v$, dus deze $U^{-1}v$ is niet relevant in de afronding, en dus kunnen we deze buiten de afrondingshaken halen:

$$\begin{aligned}
U \left\lfloor U^{-1}v + R^{-1}e \right\rceil &= U(U^{-1}v + \left\lfloor R^{-1}e \right\rceil) \\
&= v + U \left\lfloor R^{-1}e \right\rceil.
\end{aligned}$$

Dan geldt $\mathrm{Dec}(c) = v$ dan en slechts dan als $v + U \left\lfloor R^{-1}e \right\rceil = v$ dan en slechts dan als $U \left\lfloor R^{-1}e \right\rceil = 0$ de nulvector is.

Dus de correctheid van de versleuteling en ontsleuteling is afhankelijk van de keuze van verstoringsvector e.

4.6.1 Verstoringsvector

We willen de verstoringsvector e dus zodanig kiezen dat $\left\lfloor R^{-1}e \right\rceil = 0$ maar deze e toch lang genoeg is zodat CVP oplossen moeilijk is voor basis B. Dit

[7] We gebruiken hiervoor de afkortingen 'Enc' en 'Dec' van het Engelse 'encryption' en 'decryption'. Dit betekent respectievelijk 'versleuteling' en 'ontsleuteling'.

kunnen we doen door te kijken naar de ℓ_1-norm van de de rijen van R^{-1} en hiervoor de grootste te nemen. Laat ρ de grootste van deze normen zijn:

$$\rho = \max_{1 \leq i \leq n} \sum_{j=1}^{n} \left| (R^{-1})_{ij} \right|_1 .$$

Dan kiezen we $\sigma \in \mathbb{R}_{>0}$ zodat $\sigma < \frac{1}{2\rho}$. Voor de verstoringsvector e kiezen we voor ieder element $e_i = \pm\sigma$ zodat de kans[8] $\Pr[e_i = \sigma] = \Pr[e_i = -\sigma] = \frac{1}{2}$ voor ieder element van e. Voor deze keuze van e moet dus gelden dat $\left\lfloor R^{-1}e \right\rceil = 0$:

$$\begin{aligned} \left| (R^{-1}e)_i \right| = |\langle \hat{r}_i, e \rangle| &= \left| \sum_{j=1}^{n} (R^{-1})_{ij} e_i \right| \\ &\leq \sum_{j=1}^{n} \left| (R^{-1})_{ij} \right| |e_i| = \sum_{j=1}^{n} \left| (R^{-1})_{ij} \right| \sigma \leq \sigma\rho. \end{aligned}$$

Dus $\left| (R^{-1}e)_i \right| \leq \sigma\rho < \frac{\rho}{2\rho} = \frac{1}{2}$, dus $\left\lfloor (R^{-1}e)_i \right\rceil = 0$ voor iedere i, en dus is $\left\lfloor R^{-1}e \right\rceil = 0$.

Vaak wordt voor deze σ een groter getal gekozen dan die hierboven gekozen σ, zodat de instantie van CVP die we moeten oplossen in de ontsleuteling moeilijker wordt. Echter is er dan wel een kans dat de ontsleuteling niet de originele v geeft. Wel kunnen we deze kans afdwingen met een vooraf gekozen $\varepsilon > 0$, zodat je van tevoren weet wat de kans op een foute ontsleuteling is. Andersom kan natuurlijk ook: een σ kiezen en dan de kans op een ontsleutelingsfout ε berekenen.

Daarbij geldt ook nog dat als er een fout in de ontsleuteling is gemaakt, we kunnen checken of er een fout gemaakt is. We kunnen namelijk gewoon nagaan wat de verstoringsvector e is geweest, en als de elementen van e niet allemaal $\pm\sigma$ zijn, weten we dat er een fout is gemaakt bij de ontsleuteling.

In [8] wordt een $\sigma \approx 3$ gekozen. Deze σ hebben ze berekend voor parameters $n = 120$ en $\varepsilon = 10^{-5}$ en wordt verder gebruikt voor alle dimensies en de daarin gekozen geheime sleutels R.

4.7 Bases genereren

Ten eerste moeten we een dimensie $n \in \mathbb{N}$ bepalen. Hoe groter de n, hoe veiliger het systeem. Echter worden de sleutels kwadratisch groter en het

[8] Dit doen we met kans een half, zodat de verstoringsvector alleen met 'brute force' geraden kan worden. In dit geval zijn er 2^n mogelijkheden voor e en zullen we gemiddeld ongeveer 2^{n-1} mogelijkheden voor e moeten uitproberen om de juiste e te 'raden'.

versleutelingsproces duurt kwadratisch langer, dus we willen deze n niet té groot hebben.

In [8] wordt deze n tussen de 250 en 300 gekozen, omdat verwacht wordt dat alle mogelijke aanvallen op het systeem dan minstens net zo effectief als 'brute force' zijn.

4.7.1 De geheime sleutel

De eerste optie voor de geheime sleutel is om deze random te genereren. Dus ieder element van R wordt uniform gekozen uit $\{-l, ..., l\}$ voor een $l \in \mathbb{N}$. Merk op dat hoe kleiner de l is, hoe minder opslagruimte de geheime sleutel nodig heeft. Er zijn voor R in dit geval zo'n $(2l)^{n^2}$ mogelijkheden, wat voor $l \geq 1$ een grote hoeveelheid mogelijke sleutels is, dus een 'brute force' aanval op de geheime sleutel zal sowieso onbegonnen werk zijn[9]. Dus l mag om die reden klein zijn. Er moet natuurlijk wel rekening gehouden worden het met feit dat R ook een kleine duale orthogonaliteitsafwijking moet hebben, maar dat gaat ook goed voor kleine l volgens de makers van GGH.

De tweede optie is om een R' op de bovenstaande manier te genereren, dus $R' \in \{-l, ..., l\}^{n \times n}$ uniform gekozen, en dan tellen we daar een vermenigvuldiging van de eenheidsmatrix bij op. Dus: $R = R' + kI$ voor een getal $k \in \mathbb{Z}$ (in [8] kiezen ze $k \approx \sqrt{n}l$).

4.7.2 De publieke sleutel

Zoals we al besproken hadden in sectie 4.5, wordt er behalve een 'random' vermenigvuldiging van unimodulaire matrices met R slechts één andere manier gegeven. Deze houdt in dat we een basisvector nemen, en deze optellen bij een random gehele lineaire combinatie van de rest van de vectoren. In [8] gebruiken ze voor de coëfficiënten in deze lineaire combinatie elementen uit $\{-1, 0, 1\}$ random gekozen met $\Pr[1] = \Pr[-1] = 1/7$, zodat 0 het vaakst gekozen zou worden. Dit deden ze om te voorkomen dat de elementen van B veel grotere waarden dan de elementen van R zouden aannemen. Het bleek voldoende te zijn om dit proces $2n$ keer te herhalen.

De tweede optie was om B te verkrijgen door R te vermenigvuldigen met unimodulaire matrices $T_1, T_2, T_3, ...$, waarvoor $T_i = L_i U_i$, en L_i een onderdriehoeksmatrix is met op de diagonaal ± 1 en de rest uniform gekozen uit $\{-1, 0, 1\}$ en U_i de bovendriehoeksmatrix op dezelfde manier gegenereerd

[9] Ter illustratie: AES-256 — een wereldwijd gebruikt *symmetrisch* cryptosysteem — heeft 2^{256} mogelijke sleutels en wordt op dit moment als een van de veiligste symmetrische cryptosystemen ervaren [12].

als L_i, maar dan elementen op de bovendriehoek in plaats van de onderdriehoek. Dit proces werkt in dit geval voldoende goed vanaf vier van zulke T_i.

In sectie 4.10 zullen we de Hermite normaalvorm introduceren — een betere keuze voor de publieke basis dan de vorige twee opties — die een expliciete methode geeft om publieke basis te genereren.

4.8 Bases representatie

Om gebruik te maken van dit cryptosysteem, moeten we de sleutels kunnen bewaren. Voor de publieke sleutel is dit niet zo moeilijk, want deze zal alleen uit gehele getallen bestaan. Afhankelijk van de grootte van de elementen van B, zal de ruimte die deze matrix nodig heeft kwadratisch zijn. Dus als alle elementen van B geschreven kunnen worden met b bits, dan zijn er bn^2 bit nodig om de hele matrix op te slaan.

Dan kan $Bv + e$ uitgerekend worden in kwadratische tijd. Afhankelijk van de grootte van de elementen van B en v zullen de vermenigvuldigingen van deze elementen 'snel' zijn, aangenomen dat deze elementen opgeslagen kunnen worden in een aantal bits dat een stuk kleiner is dan n. Verstoringsvector e heeft n elementen, dus er moeten n optellingen gedaan worden.

In de ontsleuteling berekenen we $U \left\lfloor R^{-1}c \right\rceil$, dus is het handiger om U en de inverse van R op te slaan. Dit gaat goed voor U, omdat deze net zoals B een matrix is die uit gehele getallen bestaat en dus kwadratische opslagruimte nodig zal hebben. Echter, voor R geldt dat R een matrix is die bestaat uit gehele getallen, maar R^{-1} is dat niet, en we willen deze juist opslaan.

Afhankelijk van hoe groot de elementen van R^{-1} zijn en de invloed die deze grootte heeft op de ontsleuteling, willen we het aantal bits kiezen dat gebruikt wordt om ieder element van R^{-1} te representeren.

Laat l het aantal bits zijn dat we voor R^{-1} kunnen bewaren. Laat hierbij $R' = R^{-1} - E$ zijn, waarvoor E overal de rest van de bits voorstellen en in het bijzonder geldt dat $|E_{ij}| < 2^{-l}$ voor alle i, j. Dus R' representeert hier de matrix R^{-1} waarvoor ieder element 'afgerond' is op l bits.

Dan kunnen we kijken wat deze R' doet met het ontsleutelingsproces, dus in plaats van $v = U \left\lfloor R^{-1}c \right\rceil$ voor gecodeerde boodschap c te berekenen,

berekenen we v' als volgt:

$$\begin{aligned} v' = U\left\lfloor R'c \right\rceil &= U\left\lfloor (R^{-1} - E)(Bv + e) \right\rceil \\ &= U\left\lfloor R^{-1}Bv + R^{-1}e - E(Bv + e) \right\rceil \\ &= U\left\lfloor U^{-1}B^{-1}Bv + R^{-1}e - E(Bv + e) \right\rceil \\ &= UU^{-1}v + U\left\lfloor R^{-1}e - E(Bv + e) \right\rceil \\ &= v + U\left\lfloor R^{-1}e - E(Bv + e) \right\rceil \end{aligned}$$

Dus in dit geval willen we weer dat $\left\lfloor R^{-1}e - E(Bv + e) \right\rceil = 0$. We weten al dat $e \in \{\pm\sigma\}^n$, dus we willen het gedrag van $E(Bv + e)$ bepalen, oftewel dit moet klein zijn om afgerond de nulvector te geven. Ee is hierin tamelijk irrelevant, omdat $|E_{ij}| < 2^{-l}$ en dus $|(Ee)_i| < \sigma \sum_j 2^{-l} = \sigma n 2^{-l}$ dus voor grote l is dit heel klein (vanaf $l > \log(\sigma n) + 1$ geldt dat $|(Ee)_i| < 1/2$. Ter illustratie: voor $n = 200$ en $\sigma = 3$ geldt dit al vanaf 7 bits).

Dan willen we nog bekijken hoeveel invloed EBv heeft op deze afronding. Stel dat ieder element van B gerepresenteerd wordt door k bits, en ieder element van v door m bits. Dan wordt ieder element in Bv gerepresenteerd door maximaal $v + m$ bits. Dan: $|(EBv)_i| < \sum_j 2^{k+m-l} = n \cdot 2^{k+m-l}$ voor iedere i.

Ter illustratie: stel $n = 200$ en we gebruiken respectievelijk 16 en 8 bits voor de elementen van B en v, en we representeren de elementen van R^{-1} door 64 bits. Dan hebben we: $|(EBv)_i| < 200 \cdot 2^{16+8-64} \approx 2^{-32}$ voor iedere i. Dus deze heeft net als Ee haast geen invloed op het veel grotere $R^{-1}e$, dat maximaal een absolute waarde van een half mag hebben. Dus als het aantal bits dat we gebruiken om R^{-1} te representeren maar groot genoeg is (deze kunnen we bepalen afhankelijk van de grootte van de elementen in B en v), dan heeft dit bijna geen invloed op het ontsleutelingsproces.

Er is natuurlijk wel een kans op ontsleutelingsfouten, maar deze was al niet gelijk aan nul door onze keuze van σ. De kans op ontsleutelingsfouten ε wordt slechts iets groter in dit geval.

Merk op: in de introductie van GGH wordt gesteld dat de grootte van de publieke sleutel in $O(n^2)$ ligt. Echter laat de praktijk zien (zie bijvoorbeeld [13]) dat dit vaak $O(n^3 \log n)$ is.

4.9 Representatie van boodschap v

Er worden verscheidene voorstellen in de introductie van GGH [8] gedaan wat betreft het representeren van een boodschap v. Een manier om dit te doen is gewoonweg de boodschap in te bedden in de vector v.

Echter kan dit niet zomaar. Stel we hebben een boodschap v en deze is versleuteld tot een $c = Bv + e$. We kunnen $B^{-1}c = v + B^{-1}e$ berekenen en definiëren $d = B^{-1}e$.

We bekijken de rijen $\hat{b}_i$ in B^{-1} waarvan de norm $\|\hat{b}_i\|$ klein is. Dan zal d_i ook klein zijn, want stel $\sigma\|\hat{b}_i\| < 1$, dan volgt met $\langle \hat{b}_i, e\rangle = \sum_j (B^{-1})_{ij} e_i$ dat $|d_i| = \left|\sum_j (B^{-1})_{ij} e_i\right| \leq \sum_j |(B^{-1})_{ij}|\sigma = \sigma \sum_j |(B^{-1})_{ij}|$.

Omdat de kans op ontsleutelingsfouten ε heel klein is geldt dus met grote waarschijnlijkheid dat $\sigma \sum_j |(B^{-1})_{ij}|$ klein genoeg is om afgerond $\left\lfloor \sigma \sum_j |(B^{-1})_{ij}| \right\rceil = 0$ te zijn (want zie sectie 4.6.1: als $\sum_j |(B^{-1})_{ij}| \leq \rho$, dan geldt met kans van ongeveer $1 - \varepsilon$ dat $\sigma \sum_j |(B^{-1})_{ij}| \leq 1/2$).

Dus als geldt dat $\sigma\|\hat{b}_i\| < 1$, dan weten we ook dat het waarschijnlijk is dat $|d_i| < 1/2$, en kunnen we v_i berekenen door $\left\lfloor (B^{-1}c)_i \right\rceil = v_i$ uit te rekenen. Dus voor rijen in B^{-1} met een kleine norm kunnen we v_i berekenen.

Het is daarom verstandig om de boodschap in te bedden in de delen van v die corresponderen met de rijen van B^{-1} die een grote norm hebben (waarvoor dus geldt dat $\sigma\|\hat{b}_i\| > 1$).

In sommige literatuur (bijvoorbeeld [13]) wordt voorgesteld om de boodschap niet in v in te bedden, maar in de verstoringsvector. Dus bijvoorbeeld σ voor 1 en $-\sigma$ voor 0 voor ieder element van e. Dan wordt in plaats van de verstoringsvector juist het roosterpunt uniform gekozen.

4.10 Hermite normaalvorm

In [13] wordt een methode geïntroduceerd. Deze geeft onder andere een manier geeft om een publieke basis B te genereren vanuit R. Deze methode reduceert de grootte van B en tevens de gecodeerde boodschappen ten opzichte van het voorstel in het systeem dat Goldreich, Goldwasser en Halevi voorstelden. Micciancio's voorstel voor een publieke basis B is de Hermite normaalvorm.

Definitie 4.5
Laat $B \in \mathbb{Z}^{n\times n}$ een roosterbasis zijn. Dan is B in Hermite normaalvorm (HNF) als geldt:

(i) $b_{ij} = 0$ *voor alle* $i > j$,

(ii) $b_{ij} > 0$ *voor alle* $i = j$,

(iii) $0 \leq b_{ij} < b_{ii}$ *voor alle* $i < j$.

Dan kunnen we laten zien dat er voor iedere roosterbasis $B \in \mathbb{Z}^{n\times n}$ een matrix is die in de Hermite normaalvorm is die hetzelfde rooster representeert.

Lemma 4.6
Laat $B \in \mathbb{Z}^{n\times n}$ een roosterbasis zijn. Dan bestaat er een unieke $H \in \mathbb{Z}^{n\times n}$ die in Hermite normaalvorm is zodat $BU = H$ voor een unimodulaire matrix U.

Bewijs: Zie [5] voor het bewijs. Dit wordt op twee manieren bewezen door middel van een algoritme.

Het eerste algoritme werkt door een aantal elementaire kolomoperaties uit te voeren op B en daarmee H te verkrijgen in een eindig aantal stappen.

Het tweede algoritme werkt ongeveer hetzelfde, maar gebruikt ook het uitgebreide algoritme van Euclides.

Echter wordt hier niet de uniciteit van H bewezen. Laat $H = [h_1, ..., h_n]$ en $H' = [h'_1, ..., h'_n]$ matrices in de Hermite normaalvorm zijn, waarvoor geldt dat er unimodulaire matrices U, U' zijn zodat $BU = H$ en $BU' = H'$. Stel $H \neq H'$.

Laat i, j indices zijn zodat $H_{ij} \neq H'_{ij}$ en voor alle $i' > i$ geldt dat $H_{i'j} = H'_{i'j}$. Dan ofwel $H_{ij} > H'_{ij}$, ofwel $H_{ij} < H'_{ij}$. We nemen aan dat $H_{ij} > H'_{ij}$ (andersom gaat soortgelijk).

We weten dat $h_j, h'_j \in \mathcal{L}(H) = \mathcal{L}(H')$ en dus ook $h_j - h'_j \in \mathcal{L}(H)$. Dan zijn er $x_1, ..., x_n \in \mathbb{Z}$ zodat $h_j - h'_j = \sum_k x_k h_k$. Er geldt dat $(h_j - h'_j)_{i'} = 0$ voor $i' > i$ en $(h_j - h'_j)_i \neq 0$.

Ter verduidelijking: dus h_j, h'_j zijn de j-de kolommen van respectievelijk H en H'. En dus is $(h_j)_i$ de i-de rij van de j-de kolom van H en is dus gelijk aan H_{ij} (hetzelfde geldt voor H').

Dan moet gelden dat $x_{i+1} = \ldots = x_n = 0$, want H is een bovendriehoeksmatrix[10]. Dan geldt dat $(h_j - h'_j)_i = \sum_{k=1}^{n} x_k (h_k)_i = x_i (h_i)_i = x_i H_{ii}$. Dus:

$$H_{ij} - H'_{ij} = x_i H_{ii} \tag{4}$$

Merk op dat $x_i > 0$, omdat $(h_j - h'_j)_i = H_{ij} - (H')_{ij} > 0$.

Nu weten we dat $i \leq j$ geldt, omdat voor alle $i' > j$ geldt dat $H_{i'j} = H'_{i'j} = 0$. Dan zijn er twee mogelijkheden: $i = j$ en $i < j$.

[10] Omdat de laatste component van $(h_j - h'_j)$ gelijk is aan nul en in de lineaire combinatie alleen h_n van invloed is, moet $x_n = 0$ zijn. Dan geldt voor iedere $n \geq i' > i$ dat de i'-de component als enige meeweegt (de voorgangers $x_n, ..., x_{i'+1}$ waren met inductie gelijk aan nul) in de combinatie, dus $x_{i'} = 0$ om overeen te stemmen met $(h_j - h'_j)_{i'} = 0$.

Stel $i = j$. Dan $(h_j - h'_j)_i = H_{ij} - H'_{ij} = H_{ii} - H'_{ii} > 0$ per aanname. Maar met (4) geldt ook $H_{ii} - H'_{ii} = x_i H_{ii}$ en dus $(1 - x_i)H_{ii} = H'_{ii}$. Omdat $x_i > 0$ en dus $x_i \geq 1$ geldt dat $0 \geq (1 - x_i)H_{ii} = H'_{ii}$. Echter, $H'_{ii} > 0$, dus hebben we een tegenspraak. Dus moet wel $i < j$ gelden.

Echter, als $i < j$, dan geldt dat $H_{ij} < H_{ii}$ en dan volgt uit (4) dat $x_i H_{ij} < H_{ij} - H'_{ij}$ en dus $(x_i - 1)H_{ij} < -H'_{ij}$. Dan $(x_i - 1)H_{ij} < 0$ omdat $-H'_{ij} \leq 0$. En omdat $x_i > 0$ geldt dat $(x_i - 1)H_{ij} \geq 0$. Maar dat is een tegenspraak, dus $i < j$ kan ook niet gelden.

Dus $H = H'$. ■

Merk op: we noteren dit als $\mathrm{HNF}(B) = H$.

Voor iedere basis $B \in \mathbb{Z}^{n \times n}$ noemen we $\mathrm{HNF}(B)$ de Hermite normaalvorm van B (zie [16] voor een methode om deze efficiënt te berekenen).

We zien direct dat H hetzelfde rooster representeert als B. We hadden namelijk al bewezen in Stelling 3.7 dat als $B = HU$ voor U unimodulaire matrix, dan $\mathcal{L}(B) = \mathcal{L}(H)$. We willen natuurlijk ook dat deze Hermite normaalvorm uniek is voor ieder rooster. Daarvoor is het volgende lemma:

Lemma 4.7
Laat $B, B' \in \mathbb{Z}^{n \times n}$ bases zijn die hetzelfde rooster representeren, dus $\mathcal{L}(B) = \mathcal{L}(B')$. Dan geldt: $HNF(B) = HNF(B')$.

Bewijs: Laat B, B' roosterbases zijn zodat $\mathcal{L}(B) = \mathcal{L}(B')$ en laat $H, H' \in \mathbb{Z}^{n \times n}$ zodat $\mathrm{HNF}(B) = H$ en $\mathrm{HNF}(B') = H'$. Dan met Stelling 3.7 hebben we een unimodulaire matrix U zodat $BU = B'$, en met Lemma 4.6 hebben we unimodulaire matrices T, T' zodat $BT = H$ en $B'T' = H'$. Dus: $B'T' = BUT' = H'$. En $BUT' = HT^{-1}UT' = H'$. Laat $U' = T^{-1}UT'$ een unimodulaire matrix zijn (want T^{-1}, U, T' zijn unimodulair). Dan $HU' = H'$. Dus ook $B(TU') = (BT)U' = HU' = H'$. En omdat TU' unimodulair, geldt dus dat ook $\mathrm{HNF}(B) = H'$. Maar volgens Lemma 4.6 was de HNF uniek, dus moet wel gelden dat $H = H'$, oftewel $\mathrm{HNF}(B) = \mathrm{HNF}(B')$. ■

Dus voor ieder rooster $\mathcal{L}$ is er een unieke roosterbasis H in de Hermite normaalvorm zodat $\mathcal{L} = \mathcal{L}(H)$.

In [14] wordt voor de publieke basis de Hermite normaalvorm van R voorgesteld, omdat $\mathrm{HNF}(R)$ geen informatie lekt over R, en is R niet terug te verkrijgen vanuit de Hermite normaalvorm[11]. Ook is de duale orthogona-

[11] Voor iedere R' zodat $\mathcal{L}(R) = \mathcal{L}(R')$ weten we dat geldt dat $\mathrm{HNF}(R) = \mathrm{HNF}(R')$, dus zal het moeilijk worden om de exacte R te vinden die we gebruikt hebben als geheime sleutel zodat de publieke sleutel $B = \mathrm{HNF}(R)$. Hiervoor zijn geen methoden bekend anders dan 'brute force' toe te passen op bases R' en de HNF te berekenen.

liteitsafwijking van HNF(R) groot, zoals gesteld wordt. Dit komt omdat de orthogonaliteitsafwijking van HNF(R) klein is, omdat de helft van de elementen gelijk is aan nul, op de diagonaal de maximale elementen van iedere rij staan, en het product van alle elementen op de diagonaal gelijk is aan de determinant van het rooster.

In [13] wordt een aangepaste versie van het GGH cryptosysteem voorgesteld door Micciancio die ook gebaseerd is op de keuze van een publieke basis door middel van de HNF, maar deze maakt ook wat andere aanpassingen in de ver- en ontsleuteling, en daarmee in het cryptosysteem in het algemeen.

Echter zijn er veel voordelen aan dit cryptosysteem. Het systeem is namelijk minstens net zo moeilijk als het GGH cryptosysteem, en veel efficiënter en praktischer in het gebruik. Zowel de publieke sleutel als de gecodeerde boodschap in het 'nieuwe cryptosysteem' is met een factor n kleiner dan de publieke sleutel en de gecodeerde boodschap in het GGH cryptosysteem. Dit geldt tevens voor het versleutelingsproces.

Een mooie bijkomstigheid hiervan is dus ook dat we grotere dimensies voor n kunnen gebruiken dan bij het originele GGH cryptosysteem, zonder dat dit extreem grote sleutels geeft, maar toch veilig is in het gebruik én als praktisch wordt ervaren.

Voor meer informatie over dit cryptosysteem, verwijs ik naar [13].

5 Aanvallen op het GGH cryptosysteem

Om te zien of een cryptosysteem ook echt veilig is, is het belangrijk om te analyseren welke soort aanvallen goed werkt, of goed zou kunnen werken, te zien wat het gedrag van deze aanvallen is en of ze ook wel degelijk beter werken dan simpelweg 'brute force' te gebruiken — want in het geval dat alle aanvallen slechter werken dan 'brute force', wordt een systeem als veilig ervaren[12].

In dit hoofdstuk bekijken we twee soorten aanvallen, namelijk de in [8] geïntroduceerde 'insluitaanval' en de cryptoanalyse die gepubliceerd werd door Phong Nguyen [17] in 1999.

5.1 Insluitaanval

In de introductie van GGH (zie [8]) wordt een aantal voorstellen gedaan voor aanvallen op het GGH cryptosysteem. Een van de succesvolste is de insluitaanval (Engels: the Embedding Attack) en wordt gebruikt om CVP te benaderen.

Laat $b_1, ..., b_n$ de basisvectoren zijn en c een punt waarvoor we een roosterpunt in een $(n+1)$-dimensionaal rooster willen vinden dat dichtbij ligt:

$$B' = \begin{pmatrix} | & | & | & & | \\ c & b_1 & b_2 & \dots & b_n \\ | & | & | & & | \\ 1 & 0 & 0 & & 0 \end{pmatrix}.$$

Dan kunnen we een roosterreductie algoritme gebruiken om de kortste vector $v' \in \mathcal{L}(B')$ die niet nul is te vinden in $\mathcal{L}(B')$. Merk op dat dit equivalent is aan een instantie van SVP oplossen.

Dan vinden we bij benadering: $\|v'\| \leq \alpha \|u'\|$ voor alle $u' \in \mathcal{L}(B')$ zodat $u' \neq 0$ voor een constante α.

Laat dan $v = (v'_1, ..., v'_n)$ de vector zijn die overeenstemt met de eerste n componenten van v'. Dan zijn er twee mogelijkheden:

- $\|v - c\| \leq \|u - c\|$ voor alle $u \in \mathcal{L}(B')$;
- Of v is niet de oplossing voor deze instantie van CVP.

[12] In de praktijk zijn voor de meeste cryptosystemen wel aanvallen bekend die beter werken dan 'brute force', en toch nog als veilig worden beschouwd. Er is bijvoorbeeld een aanval bekend dat het al eerder genoemde AES kan breken met ongeveer de helft van het aantal mogelijkheden dat 'brute force' nodig zou hebben [4].

In het eerste geval zijn we klaar, want dan is v de oplossing voor deze instantie van CVP. En in het tweede geval kunnen we het punt v dat we gevonden hebben gebruiken voor een 'exhaustive search' naar de vector $\hat{v}$ die het dichtst bij c ligt, door alle roosterpunten die dicht bij v liggen te proberen en daar de dichtstbijzijnde van te nemen.

5.2 Cryptoanalyse van het GGH cryptosysteem

Twee jaar na de introductie van GGH, publiceerde Nguyen een cryptoanalyse van GGH [17]. Hierin liet hij zien wat de zwakte was die hij ontdekte in het systeem, en gebruikte hij deze om de uitdaging die Goldreich, Goldwasser en Halevi hadden gedaan bij de introductie van GGH te volbrengen. Zij plaatsten vijf gecodeerde boodschappen, voor dimensie $n \in \{200, 250, 300, 350, 400\}$, op hun website, met als opdracht deze te ontsleutelen.

In 1999 lukte het Nguyen om vier van de vijf gecodeerde boodschappen te decoderen, behalve die van dimensie 400, welke hij alleen gedeeltelijk kon terughalen (met gedeeltelijk wordt hier bedoeld: de oorspronkelijke boodschap modulo 6. Uiteindelijk werd deze gecodeerde boodschap ook ontsleuteld door Lee en Hahn in 2010 [10]).

De zwakte van GGH die in deze cryptoanalyse vooral aan bod komt, is de keuze van de verstoringsvector. De vorm van deze verstoringsvector zorgt er namelijk voor dat er informatie over de versleuteling weglekt, en dat is iets wat we niet willen bij een cryptosysteem als we veiligheid willen waarborgen. De veiligheid van GGH zoals deze is voorgesteld in het vorige hoofdstuk is terug te brengen tot het oplossen van CVP, maar Nguyen liet zien dat, omdat de verstoringsvector de voorgestelde vorm heeft, deze teruggebracht kan worden naar een makkelijkere instantie van CVP.

5.2.1 Rekenen modulo 2σ

Laat $B \in \mathbb{Z}^{n \times n}$ de publieke basis zijn voor een zekere $n \in \mathbb{N}$ en σ gekozen zoals in het vorige hoofdstuk is uitgelegd (met of zonder ontsleutelingsfouten is in de cryptoanalyse irrelevant). We nemen $v \in \mathbb{Z}^n$ een boodschap en $e \in \{\pm\sigma\}^n$ de verstoringsvector. Dan is de versleuteling:

$$c = Bv + e.$$

Laat nu $s = (\sigma, ..., \sigma) \in \mathbb{Z}^n$ zijn. Dan weten we $e + s \equiv 0 \bmod 2\sigma$, en dus:

$$c + s \equiv Bv \bmod 2\sigma.$$

We definiëren $v_{2\sigma} = v \bmod 2\sigma$. Nu willen we weten hoeveel oplossingen er zijn voor de vergelijking

$$y = Bv_{2\sigma} \bmod 2\sigma$$

waar $y = Bv \bmod 2\sigma$. We weten in dit geval de waarde van y, B en σ. Het aantal oplossingen voor y willekeurig is gelijk aan het aantal oplossingen voor de vergelijking

$$Bv_{2\sigma} \equiv 0 \bmod 2\sigma.$$

Laat $K = \{x \in \mathbb{Z}^n : Bx \equiv 0 \bmod 2\sigma\}$ de kern van deze vergelijking zijn. Er zijn twee mogelijkheden: $\gcd(\det(B), 2\sigma) = 1$ of $\gcd(\det(B), 2\sigma) \neq 1$. In het eerste geval is B inverteerbaar modulo 2σ en in het tweede geval niet. Als B wel inverteerbaar modulo 2σ, dan kunnen we direct $y = Bx \bmod 2\sigma$ oplossen, namelijk: $x = B^{-1}y \bmod 2\sigma$. Zo niet, dan willen we de kardinaliteit van K weten (want dan weten we het aantal oplossingen voor $y = Bx \bmod 2\sigma$).

De kardinaliteit van K is afhankelijk van n en σ. In [17] wordt de kardinaliteit van deze K berekend voor grote n en $\sigma = 3$, de in [8] voorgestelde σ. In slechts $0,3\%$ van de gevallen is de kardinaliteit van K groter dan 18, dus in het merendeel van de gevallen geldt dat voor een matrix B ofwel B inverteerbaar is, ofwel de kern van B een kleine kardinaliteit heeft.

5.2.2 Versimpelen van CVP

Stel we hebben een oplossing gevonden voor $y = Bx \bmod 2\sigma$ in $x = v_{2\sigma}$. Dan kunnen we $c = Bv + e$ omschrijven tot:

$$c - Bv_{2\sigma} = B(v - v_{2\sigma}) + e.$$

Dan is $v - v_{2\sigma} = 2\sigma v'$ voor een $v' \in \mathbb{Z}^n$. Dan:

$$\frac{c - Bv_{2\sigma}}{2\sigma} = Bv' + \frac{e}{2\sigma}.$$

We kunnen $c' = \frac{c - Bv_{2\sigma}}{2\sigma} \in \mathbb{Q}^n$ berekenen, en $e' = \frac{e}{2\sigma} \in \{\pm\frac{1}{2}\}$. Nu krijgen we dus een instantie van CVP van de vorm:

$$c' = Bv' + e' \tag{5}$$

waarvoor $\|e'\| = \sqrt{\sum_i (e')_i^2} = \sqrt{n \cdot \frac{1}{4}} = \frac{1}{2}\sqrt{n}$. De lengte van de originele verstoringsvector was $\|e\| = \sqrt{\sum_i e_i^2} = \sqrt{n \cdot \sigma^2} = \sigma\sqrt{n}$. Maar er geldt dat $\frac{1}{2} < \sigma$, dus CVP is voor (5) een stuk makkelijker op te lossen dan voor de originele instantie.

En als we een oplossing hebben voor (5), hebben we ook een oplossing voor $c = Bv + e$, want $v = v_{2\sigma} + 2\sigma v'$ en $e = 2\sigma e'$.

Dus in het geval dat B inverteerbaar modulo 2σ is, hoeven we dit maar één keer te doen, en stel dat dat niet het geval is, dan moeten we dit proces een aantal keer herhalen, namelijk $k = \#K$ keer (voor iedere oplossing die we vinden). Maar de kans dat de kern van B klein is, is heel groot, dus in de meeste gevallen hoeven we dit proces niet zo vaak te herhalen, en kunnen we dus de boodschap hiermee toch achterhalen.

5.2.3 Oplossen met de insluittechniek

In sectie 5.1 hebben we al gezien hoe de insluitaanval te werk gaat. Deze zorgt ervoor dat gecodeerde boodschappen ontsleuteld kunnen worden tot dimensie $n = 200$. Dus om deze techniek te laten falen, of in ieder geval langzamer te laten werken dan een 'brute force attack', moet de dimensie van n al groter dan 200 zijn.

In [17] werd dezelfde techniek ook gebruikt om de versimpelde instanties van CVP zoals hierboven beschreven op te lossen.

5.2.4 Evaluatie

Door de keuze van de verstoringsvector kunnen we dus de veiligheid van het GGH cryptosysteem terugbrengen tot een veel gemakkelijkere CVP-instantie, waardoor het systeem niet veilig is, en in redelijk snelle tijd voor dimensies tot 400 de boodschap teruggevonden kan worden.

Zelfs onder de aanname dat voor $n > 400$ de instantie van CVP die we verkregen hebben toch niet opgelost kan worden, zou het volgens Nguyen nog niet praktisch zijn om voor zo'n grote dimensie het systeem te gebruiken. De publieke sleutel is dan erg groot (al 1,8MB in dimensie 400) en de gecodeerde boodschap ook (al 6,4KB in dimensie 400). Daarbij kost de versleuteling 400^2 vermenigvuldigingen van getallen met maximale bit-lengte 129 (in dimensie 400). Alles wat een grotere dimensie heeft, zal dus nóg groter zijn.

Een voorstel voor lagere dimensies zou zijn om de vorm van de verstoringsvector aan te passen. In plaats van $e_i \in \{\pm\sigma\}$ kunnen we bijvoorbeeld $e_i \in \{-\sigma, ..., \sigma\}$ uniform kiezen voor iedere i, dus zodat het gemiddelde nul

is. In dat geval is de verwachte lengte van e:

$$\begin{aligned}\|e\| &= \sqrt{\sum_i e_i^2} = \sqrt{\frac{n}{2\sigma+1} 2 \sum_{j=1}^{\sigma} j^2} \\ &= \sqrt{\frac{2n}{6(2\sigma+1)} \sigma(\sigma+1)(2\sigma+1)} = \sqrt{\frac{n}{3}(\sigma^2+\sigma)}\end{aligned}$$

Dus dat is ongeveer $\sigma\sqrt{n/3}$ en is dus een stuk kleiner dan $\sigma\sqrt{n}$ en maakt het misschien weer te makkelijk om CVP op te lossen (met behulp van de insluittechniek), omdat de verstoringsvector niet lang genoeg is.

Een andere keuze voor e zou kunnen zijn om $e_i \in \{\pm\sigma, \pm(\sigma-1)\}$ te kiezen, zodat e iets langer is, maar weer meer lijkt op de originele e.

Dit was dan ook een van de redenen van Nguyen om gebruik van dit systeem niet aan te raden, omdat hij voorspelde dat het nooit echt veilig én praktisch kon zijn.

5.3 Afsluiting

Uiteindelijk werd GGH door een van de makers, Goldreich, 'dood' verklaard (in een privégesprek met Nguyen [17]). Echter zijn er nog steeds ontwikkelingen op het gebied van cryptosystemen die afgeleid zijn van GGH. In sectie 4.10 hebben we al gezien dat we voor een publieke basis de Hermite normaalvorm kunnen kiezen, en dat er zelfs een aangepaste versie van GGH bestaat, ontwikkeld door Micciancio, die een stuk praktischer en veiliger is dan GGH.

Daarnaast zijn er twee nog recentere publicaties gedaan op het gebied van cryptosystemen die afgeleid zijn van GGH. Één daarvan werd gepubliceerd in 2012 door Yoshino en Kunihiro [21], en haakt tevens ook in op Micciancio's geïntroduceerde 'verbetering' van GGH. Het idee van Yoshino en Kunihiro's heeft grofweg te maken met de vorm en lengte van de verstoringsvector.

Eerder dit hoofdstuk hebben we al laten zien dat de vorm van de verstoringsvector het mogelijk maakt om de originele boodschap terug te halen vanuit de publieke basis (voor grotere dimensies), én het feit dat deze vector relatief kort is ten opzichte van de roostervectoren (voor kleinere dimensies). Hun oplossing is dus om voor deze verstoringsvector de elementen uit een grotere verzameling te pakken, én daarbij sommige elementen van de verstoringsvector in absolute waarde een veel groter element laten zijn. Dan is de lengte van de verstoringsvector groter dan in het originele cryptosysteem,

en heeft deze vector toch niet dezelfde vorm die het mogelijk maakt om de boodschap te achterhalen.

Vooralsnog is dit systeem niet gebroken. Echter publiceerden De Barros en Menasché Schechter een artikel [3] in 2014 waarin ze stelden dat het implementeren van GGH-YK, zoals we de aangepaste versie van Yoshino en Kunihiro noemen, een paar problemen had. Ten eerste was het onmogelijk om een geheime sleutel te genereren met de in GGH-YK gegeven condities, en ten tweede bleek dat de afgeronde verstoringsvector die door de ontsleutelingsprocedure wordt verkregen toch vaak de nulvector is.

Het eerste probleem kon opgelost worden door de condities iets te verzwakken, maar het tweede probleem is wel ernstig in de zin dat daarom het systeem toch erg veel op GGH leek en daarmee dus met dezelfde problemen kampte als het originele GGH cryptosysteem.

Daarom kwamen ze met een nog weer iets aangepaste versie van GGH-YK, deze noemden ze GGHYK-M, die dit probleem weer zou moeten verhelpen. Zij gebruikten voor de geheime sleutel een M-matrix. Een M-matrix is een vierkante matrix van de vorm $M = \gamma I + P$, waar $\gamma > \max\{|\lambda_i| : \lambda_i \text{ een eigenwaarde van } P\}$ en $p_{ij} \leq 0$ voor alle i, j.

Als geheime sleutel gebruikten ze een matrix $R = \gamma I + P \in \mathbb{Z}^{n \times n}$ waar $\gamma \in \mathbb{Z}$ en $p_{ij} \in \{-1, 0\}$. Voor de rest deden ze bijna alles hetzelfde als in GGH-YK, behalve dat ze voor de verstoringsvector in het versleutelingsproces alleen met positieve waarden versleutelden. Ze pasten ook de manier van het inbedden van de boodschap m in de versleuteling aan zodat ze min of meer een simpelere instantie kregen van het GGH-YK systeem, met een werking die ongeveer hetzelfde is.

Het implementeren van het systeem gaf hun het gewenste resultaat, en loste de problemen van GGH-YK op. Ze testten ook verscheidene aanvallen op sleutels en gecodeerde boodschappen, maar konden hieruit niet de originele sleutels of boodschappen verkrijgen. Daarom moedigden ze juist de lezers aan om toch nog verder onderzoek te doen naar asymmetrische cryptografie die gebaseerd is op roosterproblemen.

Referenties

[1] M. Ajtai; C. Dwork: *A Public-Key Cryptosystem with Worst-Case/ Average-Case Equivalence*, in 29th ACM Symposium on Theory of Computing, p. 284–293, 1997.

[2] L. Babai: *On Lovász' Lattice Reduction and the Nearest Lattice Point Problem*, in Combinatorica **6** (1), p. 1–13, 1986.

[3] C. F. de Barros; L. Menasché Schechter: *GGH May Not Be Dead After All*. CNMAC 2014, 2014.

[4] A. Bogdanov; D. Khovratovich; C. Rechberger: *Biclique Cryptanalysis of the Full AES*, in D. H. Lee; X. Wang (eds): *Advances in Cryptology – ASIACRYPT 2011*, p. 344–371. Springer, 2011.

[5] H. Cohen: *A Course in Computational Algebraic Number Theory*, p. 66–67. Springer, 1993.

[6] T. H. Cormen; C. E. Leiserson; R. L. Rivest; C. Stein: *Introduction to Algorithms*, Second Edition. The MIT Press, 2001.

[7] J. Daemen; Vincent Rijmen: *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, 2002.

[8] O. Goldreich; S. Goldwasser; S. Halevi: *Public-Key Cryptosystems from Lattice Reduction Problems*. Springer, 1997.

[9] Z. Gong; M. Aldeen; L. Elsner: *A Note on a Generalized Cramer's Rule*, in Linear Algebra and its Applications **340**, p. 253–254. Elsevier, 2002.

[10] M. S. Lee; S. G. Hahn: *Cryptanalysis of the GGH Cryptosystem*, in Mathematics in Computer Science **3**, p. 201–208. Springer, 2010.

[11] A. K. Lenstra; H. W. Lenstra, Jr.; L. Lovász: *Factoring Polynomials with Rational Coefficients*, in Mathematische Annalen **261**, p. 515-534. Springer, 1982.

[12] K. M. Martin: *Everyday Cryptography: Fundamental Principles and Applications*. The Australian Mathematical Society, 2012.

[13] D. Micciancio: *Improving Lattice Based Cryptosystems Using the Hermite Normal Form*, in *Cryptography and Lattices*, p. 126–145. Springer, 2001.

[14] D. Micciancio; S. Goldwasser: *Complexity of Lattice Problems: A Cryptographic Perspective*. Springer, 2002.

[15] D. Micciancio; O. Regev: *Lattice-Based Cryptography*, in D.J. Bernstein; J. Buchmann; E. Dahmen (eds): *Post Quantum Cryptography*, p. 147–191. Springer, 2009.

[16] D. Micciancio; B. Warinschi: *A linear space algorithm for computing the Hermite Normal Form*, in B. Mourrain (editor): *International Symposium on Symbolic and Algebraic Computation*. ACM, 2001.

[17] P. Q. Nguyen: *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto'97*, in M. Wiener (editor): *Advances in Cryptology – CRYPTO '99*. Springer, 1999.

[18] R. L. Rivest; A. Shamir; L. Adleman: *A method for obtaining digital signatures and public-key cryptosystems*, in Communications of the ACM, **21**, p. 201–224. ACM, 1978.

[19] P. W. Shor: *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, p. 303-332. SIAM, 1999.

[20] T. A. Sudkamp: *Languages and Machines: An Introduction to the Theory of Computer Science*, Third Edition. Pearson, 2005.

[21] M. Yoshino; N. Kunihiro: *Improving GGH Cryptosystem for Large Error Vector*, in International Symposium on Information Theory and its Applications, p. 416–420, 2012.

[22] Website: http://www.claymath.org/millennium-problems. Laatst bezocht op: 26 juni 2016.