EINDHOVEN UNIVERSITY OF TECHNOLOGY

Digitale handtekeningen gebaseerd op elliptische krommen

Citation for published version (APA):
Sas, M., Tilborg, van, H. C. A., & Verheul, E. R. (1998). *Digitale handtekeningen gebaseerd op elliptische krommen*. (IWDE Report; Vol. 98-03). IWDE.

Document status and date:
Gepubliceerd: 01/01/1998

Document Version:
Uitgevers PDF, ook bekend als Version of Record

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Technische
Universiteit
Eindhoven

Instituut
Wiskundige Dienstverlening
Eindhoven

Rapport IWDE 98 - 03

Digitale handtekeningen gebaseerd op elliptische krommen

Ir. M. Sas (primaire auteur)
Prof.dr. H.C.A. v. Tilborg
Dr. E.R. Verheul

(With an Engish summary at the end)

oktober 1998

Den Dolech 2
Postbus 513
5600 MB Eindhoven

Rapport IWDE 98 - 03

Digitale handtekeningen gebaseerd op elliptische krommen

Ir. M. Sas (primaire auteur)
Prof.dr. H.C.A. v. Tilborg
Dr. E.R. Verheul

(With an Engish summary at the end)

oktober 1998

Management Samenvatting

Dit rapport geeft een beschrijving en een analyse van een relatief nieuw type digitale handtekeningen: deze zijn gebaseerd op elliptische krommen. Deze samenvatting bestaat uit twee onderdelen: het eerste onderdeel bevat basiskennis inzake digitale handtekeningen, het tweede bevat de resultaten van een vergelijking van drie systemen.

1 Introductie digitale handtekeningen

1.1 Wat zijn digitale handtekeningen?

Digitale handtekeningen maken het mogelijk dat een (rechts-)persoon een koppeling, een "digitale handtekening", maakt tussen zijn identiteit en bepaalde informatie (bijv. een contract) op een zodanige manier dat:
- alleen de persoon deze koppeling kan maken; de koppeling is dus sterk persoonsafhankelijk.
- de koppeling met de identiteit door iedereen vastgesteld kan worden (d.w.z. zonder dat daarbij geheime informatie benodigd is).

Een digitale handtekening kan worden voorgesteld als een getal (van zo'n 100 tot 300 cijfers) dat toegevoegd wordt aan informatie.

1.2 Wat maken digitale handtekeningen mogelijk?

Bovenstaande twee punten maken het mogelijk om met digitale handtekeningen de belangrijke eigenschap van *onweerlegbaarheid* (non-repudiation in het Engels) van informatie te realiseren: indien een persoon iets getekend heeft, kan hij dat daarna niet ontkennen. Mocht daarover toch onduidelijkheid bestaan, dan kan de handtekening in principe getoetst worden door een rechter. Digitale handtekeningen wijken daarom aanzienlijk af van Message Authentication Codes (MAC's) die veel bij financiële transacties worden gebruikt. Voor het maken en het verifiëren van MAC is namelijk dezelfde geheime informatie nodig; een MAC is dus niet herleidbaar tot slechts één (rechts-)persoon.

1.3 Waar en hoe worden digitale handtekeningen gebruikt?

Digitale handtekeningen beschermen de integriteit en authenticiteit van informatie; zij bieden geen bescherming voor de vertrouwelijkheid van informatie. Zij worden onder meer gebruikt voor het betrouwbaar opslaan en versturen van informatie waarvan de integriteit en onweerlegbaarheid belangrijk zijn (bijv. bij contracten bij Electronic Commerce). Digitale handtekeningen worden ook gebruikt voor toegangsbeveiliging bij informatiesystemen. Om een handtekening op informatie te plaatsen maakt een persoon gebruik van zijn *private* (geheime) sleutel; een bij deze sleutel behorende *publieke* sleutel maakt de verificatie van de handtekening mogelijk. Deze publieke sleutel moet aan iedereen bekend zijn; voor de distributie van de publieke sleutels zijn daarom structuren nodig (o.m. *Public Key Infrastructures*).

1.4 Wat is van belang bij een digitale-handtekeningsysteem?

Bij een digitale-handtekeningsysteem zijn de volgende zaken met name van belang:
- Veiligheid.
  De mate van zekerheid dat de onweerlegbaarheid van digitale handtekening niet gecompromitteerd kan worden.
- Complexiteit van het opzetten van het systeem.
  Hierbij wordt onderscheid gemaakt tussen de eenmalige generatie van systeemparameters (gebruikt door alle gebruikers) en de generatie van sleutelmateriaal (gebruiker specifiek).
- Benodigde rekencapaciteit bij gebruikers.
  De complexiteit van de benodigde hardware bij gebruikers (e.g. een eenvoudige smartcard, Pentium processor).
- Benodigde systeemopslag bij gebruiker.
  De benodigde opslagcapaciteit voor systeemparameters, publieke sleutels en geheime sleutels.
- Benodigde tijd voor het plaatsen van een digitale handtekening.
- Benodigde tijd voor het verifiëren van een digitale handtekening.
- Benodigde ruimte voor het opslaan van de handtekening (i.e. de grootte van de handtekening).

Management samenvatting van dit rapport

1.5 Waar gaat dit rapport (niet) over?

Dit rapport gaat in op de onderliggende cryptografische techniek van digitale handtekeningen: de relatief nieuwe digitale-handtekeningsystemen gebaseerd op Elliptische Krommen worden uitgelegd en vergeleken met twee andere, meer bekende, systemen (RSA, Klassieke Discrete Log). Voor het gebruik van digitale handtekeningen zijn behalve cryptografische, ook nog andere beveiligingstechnieken nodig (e.g. hashing, timestamping, smartcards). Daarbij zijn ook niet-technische maatregelen (e.g. procedurele, organisatorisch, juridische) en diensten (e.g. TTP's) noodzakelijk; daarop gaat dit rapport niet in.

1.6 Hoe werken de onderzochte systemen?

RSA is het oudste systeem en de veiligheid is gebaseerd op de moeilijkheid van het factorizeren van grote getallen: bij een product van twee priemgetallen is het moeilijk de priemgetallen terug te vinden. Hiervan gebruikmakend is het eenvoudig een situatie te creëren waarin het bepalen van "wortels" moeilijk is tenzij men de priemgetallen kent. Het "machtsverheffen" (de omgekeerde bewerking van "worteltrekken") daarentegen is niet moeilijk, ook zonder kennis van de priemgetallen. In de context van RSA bestaat de publieke sleutel uit het product van twee priemgetallen en de private sleutel uit deze priemgetallen. De digitale handtekening van een boodschap bestaat uit zijn "wortel": een "bewijs" dat men de priemgetallen kent.

De veiligheid van Klassieke Discrete Log systemen is gebaseerd op de moeilijkheid van het kunnen bepalen van logaritmen in "klassieke" groepen (multiplicatieve groep van lichamen): gegeven een onbekende macht van, zeg, 2 (het "basis element"), dan is het in deze context moeilijk de bijbehorende exponent (de logaritme) te bepalen. In de context van Klassieke Discrete Log systemen bestaat de private sleutel uit de exponent en de publieke sleutel uit de macht van 2. De digitale handtekening bestaat uit een bewijs dat men de geheime exponent "kent". Omdat zulke bewijzen op diverse manieren gemaakt kunnen worden, zijn er ook diverse digitale handtekening systemen gebaseerd op het Klassieke Discrete Log probleem. Een van de bekendste is de Digital Signature Algorithm (DSA) een standaard van de Amerikaanse Overheid.

De moeilijkheid van het discrete logaritme probleem doet zich ook voor in andere "groepen" (d.w.z. een situatie waarin men kan vermeningvuldigen en dus machtsverheffen). Met name in de groepen die men krijgt door op een Elliptische Kromme (denk aan een kromme in het platte vlak) een vermenigvuldiging te definiëren. De veiligheid van Elliptische Discrete Log systemen is gebaseerd op de moeilijkheid van het kunnen bepalen van logaritmen in dit type groepen. Ook in deze context bestaat de private sleutel uit de exponent en de publieke sleutel uit de macht van een basiselement. De digitale handtekening bestaat wederom uit een bewijs dat men de geheime exponent "kent". In principe zijn alle digitale handtekening systemen gebaseerd op het Klassieke Discrete Log probleem "vertaalbaar" naar het Elliptische Discrete Log probleem. De vertaling van DSA wordt ECDSA genoemd.

1.7 Hoe worden de drie systemen vergeleken

Als uitgangspunt in de vergelijking van de drie systemen, zijn de conventionele digitale-handtekeningsystemen RSA en DSA beschouwd van een relatief hoge veiligheid (modulus 1024 bit), welke vervolgens worden vergeleken met ECDSA (de elliptische variant van DSA) van gelijkwaardige veiligheid (modulus 160 bit). Met andere woorden, het eerste aspect genoemd onder punt 4 is gefixeerd in de vergelijking.

Hierbij wordt de kanttekening gemaakt dat de vergelijking van de veiligheid gebaseerd is op de bekende cryptanalytische aanvallen. Het voordeel van ECDSA is daarbij dat er eigenlijk geen goede aanvallen bekend zijn: de bekende technieken ("index- en factor-bases"), gebruikt voor relatieve snelle aanvallen op het klassieke discrete log probleem en het factorizeringsprobleem, lijken zich niet te laten generaliseren tot het discrete log probleem op elliptische krommen. Hoewel experts er vertrouwen in hebben dat zulke (gegeneraliseerde) aanvallen in het algemeen niet bestaan, is de studie van Elliptische Krommen op dit punt nog vrij jong. Ter illustratie: in het recente verleden zijn heel bepaalde

deelklassen van elliptische krommen gevonden waarvoor wel efficiënte aanvallen bestaan. Dit soort elliptische krommen zijn echter eenvoudig te detecteren en te vermijden.

1.7.1 Vergelijking complexiteit opzetten van het systeem

Het genereren van de systeemparameters ("goede" priemgetallen) bij RSA en DSA is eenvoudiger dan het genereren van systeemparameters bij ECDSA (vinden van een "goede" kromme); bij de eerste generatie moet men denken aan minuten, bij de tweede aan uren.

Het genereren van sleutelmateriaal valt bij RSA vrijwel samen met de generatie van de systeemparameters; de generatie van sleutelmateriaal bij DSA en ECDSA is verwaarloosbaar.

1.7.2 Vergelijking benodigde rekencapaciteit bij gebruikers

De benodigde rekencapaciteit bij gebruikers is bij RSA en DSA aanzienlijk groter dan bij ECDSA. Hoewel bij smartcard toepassingen, bij alle drie de systemen een smartcard benodigd is met een mathematische coprocessor, is deze coprocessor bij ECDSA aanzienlijk eenvoudiger (en dus goedkoper).

1.7.3 Vergelijking benodigde systeemopslag bij gebruikers

Ook de benodigde opslagcapaciteit bij gebruikers is bij RSA en DSA aanzienlijk groter dan bij ECDSA: voor elke publieke sleutel is bij ECDSA slechts 160 bit benodigd (20 karakters) en bij RSA en DSA zo'n 1024 bit (128 karakter). Ook dit geeft ECDSA een aanzienlijk voordeel boven RSA en DSA bij smartcard toepassingen.

1.7.4 Vergelijking benodigde tijd voor het plaatsen van een digitale handtekening

Zoals bovenstaand reeds is aangegeven, is voor het gebruik van RSA en DSA meer rekenkracht benodigd dan voor ECDSA. Indien men uitgaat van de rekencapaciteit benodigd voor RSA of DSA, dan kost het plaatsen van een digitale handtekening bij ECDSA het minste tijd, voor het plaatsen van een DSA handtekening is ongeveer 4 maal meer tijd nodig en voor het plaatsen van een RSA handtekening zo'n 6 maal meer tijd.

1.7.5 Vergelijking benodigde tijd voor het verifiëren van een digitale handtekening

Zoals bovenstaand reeds is aangegeven, is voor het gebruik van RSA en DSA meer rekenkracht benodigd dan voor ECDSA. Indien men uitgaat van de rekencapaciteit benodigd voor RSA of DSA, dan kost het verifiëren van een digitale handtekening bij RSA het minste tijd, voor het verifiëren van een ECDSA handtekening is ongeveer 7 maal meer tijd nodig en voor het verifiëren van een DSA handtekening zo'n 28 maal meer tijd.

1.7.6 Vergelijking benodigde ruimte voor het opslaan van de digitale handtekening

Voor het opslaan voor een DSA of ECDSA handtekening is 320 bit benodigd (zo'n 40 karakters), voor het opslaan van een RSA handtekening is 1024 bit benodigd (zo'n 128 karakters).

1.8 De conclusie van de vergelijking

In onderstaande tabel staan de resultaten van de vergelijking opgesomd: ▼ betekent minpunt, ▲ betekent pluspunt.

	ECDSA	RSA	DSA
Veiligheid	Gelijkgesteld	Gelijkgesteld	Gelijkgesteld
Complexiteit Systeem Generatie	▼ (< Uren)	▲ (<minuten)	▲ (< minuten)
Rekencapaciteit bij gebruikers	▲ (eenvoudige chip)	▼ (complexere chip)	▼ (complexere chip)
Opslagcapaciteit bij gebruikers	▲ (160 bit per sleutel)	▼ (1024 bit per sleutel)	▼ (1024 bit per sleutel)
Plaatsingstijd digitale handtekening	▲	▼ (6 maal ECDSA)	▼ (4 maal ECDSA)
Verificatietijd digitale handtekening	▼ (7 maal RSA)	▲	▼ (28 maal RSA)
Grootte digitale handtekening	▲ (320 bit per hand.)	▼ (1024 bit per hand.)	▲ (320 bit per hand.)

Geconcludeerd kan worden dat ECDSA grote voordelen biedt ten opzichte van RSA en DSA bij toepassingen waar slechts een beperkte reken- of opslag capaciteit beschikbaar is bij gebruikers, zoals bijvoorbeeld bij smartcard toepassingen.

Hierbij wordt (nogmaals) de kanttekening gemaakt dat de vergelijking van de veiligheid gebaseerd is op de bekende cryptanalytische aanvallen. Het voordeel van ECDSA is daarbij dat er eigenlijk geen goede aanvallen bekend zijn. Hoewel experts er vertrouwen in hebben dat zulke aanvallen in het algemeen niet bestaan, is de studie van Elliptische Krommen op dit punt nog vrij jong. Vertrouwenswekkend is het feit dat het Duitse *Bundesamt für Sicherheit in der Informationstechnik* - dat een adviserende/normenstellende rol heeft voor (digitale) informatiebeveiliging bij overheid en burger - in een onderzoek naar digitale handtekening systeem stelt dat "Aus Sicht des BSI sind folgende Signaturalgorithmen geeignet: [...] DSA-Varianten, basierend auf elliptischen Kurven"

Echter, indien men elliptische krommen gaat gebruiken, verdient het in ieder geval aanbeveling om zich aan te sluiten bij gerenommeerde standaarden (zoals bijvoorbeeld IEEE P1363 en ANSI X9.62) en om de "stand der cryptoanalytische techniek" nauwlettend te blijven volgen. Dit is echter een aanbeveling die voor vrijwel alle cryptografische toepassingen geldt.

Voorwoord

De geschiedenis van de asymmetrische cryptografie, of public key cryptografie, begint in 1976 met het artikel "New Directions in Cryptography" door W. Diffie en M. Hellman. Het fundamentele begrip in deze vorm van cryptografie is dat van de "trapdoor one-way" problemen. Bij dit soort (wiskundige) problemen kiest men in feite eerst een antwoord en werkt vervolgens terug naar een bijbehorend probleem ("public key") dat het gekozen antwoord ("private key") volledig impliciet bepaalt. Daarbij is het echter wel de bedoeling dat het praktisch onmogelijk moet zijn het antwoord te bepalen slechts op basis van het probleem: het probleem moet "moeilijk" zijn. Zodra het bovendien mogelijk is om derden te overtuigen dat men het antwoord weet, zonder het antwoord te hoeven geven, dan kunnen trapdoor one-way problemen ondermeer gebruikt worden voor authenticiteit toepassingen zoals digitale handtekeningen: een persoon kan onweerlegbaar worden gekoppeld aan het antwoord op een moeilijke probleem ("private key").

Hoewel het concept van het trapdoor one-way probleem elegant is, kan het als zwaktepunt worden gezien dat er eigenlijk maar betrekkelijk weinig wiskundig problemen bekend zijn die aanleiding geven tot trapdoor one-way problemen. In feite is het onbewezen dat er überhaupt trapdoor one-way problemen bestaan, slechts problemen waarvan men het verwacht. Een van de oudste en bekendste van zulke problemen is het factorisatie probleem: het is eenvoudig twee grote priemgetallen te genereren en hun product te nemen. Het terugvinden van de priemgetallen (=vraag) op basis van hun product (=probleem) staat bekend als erg "moeilijk". Deze opzet geeft aanleiding tot de RSA public key cryptosystemen. Andere problemen waarop trapdoor one-way problemen op gebaseerd zijn, zijn:

- het kunnen vinden van dichtstbijzijnde punten in bepaalde verzamelingen (lineaire codes, tralies);
- het kunnen vinden van "discrete logaritmen in eindige lichamen".

Er zijn ook problemen geformuleerd waarvan men verwachtte dat zij "trapdoor" one-way waren, maar het niet bleken te zijn. Het bekendste voorbeeld hiervan is het "knapzak" systeem van Merkle en Hellman (een van de uitvinders van public key cryptografie). Dit illustreert dat men moet oppassen met verwachtingen rond de hardheid van problemen.

Een tweede zwaktepunt, van de tot nu genoemde asymmetrische systemen is dat de daarbij gebruikte sleutellengten nogal verschillen van de sleutellengen bij, qua veiligheid, vergelijkbare symmetrische systemen (zoals DES en IDEA). Ruwweg

moeten de asymmetrische sleutels zo'n 10 maal groter zijn dan de symmetrische om een vergelijkbare veiligheid te verkrijgen. Ter illustratie, een symmetrische sleutel van zo'n 80 bit correspondeert met de veiligheid van een RSA sleutel van zo'n 768 bit. In sommige toepassingen (bijv. toepassingen waarin smartcards worden gebruikt) is de hoeveelheid opslagruimte echter beperkt.

Toen elliptische kromme cryptosystemen werden geïntroduceerd in 1985, speelden zij in op beide genoemde zwaktepunten: zij zijn gebaseerd op nieuwe klassen trapdoor one-way systemen (varianten van het eerdergenoemde discrete logaritme probleem) en zij gaven aanleiding tot kortere sleutels (ruwweg twee maal de lengte van symmetrische sleutels). De reden hiervoor is dat men (in de open wetenschappelijke wereld) tot nu toe de onderliggende "structuur" van elliptische krommen nauwelijks weet te gebruiken bij cryptanalystische aanvallen: de meeste aanvallen zijn "canoniek" van aard. In feite gaat het hier om geavanceerde vormen van "exhaustive key search". Echter, reeds een aantal keer zijn klassen elliptische krommen (bijv. die van de super-singuliere elliptische krommen) gevonden waar de onderliggende structuur een cryptanalytische aanval aanzienlijk efficiënter maakt dan de canonieke.

Het risico bestaat dat men eens in staat zal voor alle elliptische krommen de onderliggende structuur te gebruiken in een cryptanalytische aanval die aanzienlijk efficiënter is dan de canonieke; volgens experts is dit risico echter niet zo groot. Illustratief hiervoor is het feit dat het Duitse *Bundesamt für Sicherheit in der Informationstechnik* - dat een adviserende/normenstellende rol heeft voor (digitale) informatiebeveiliging bij overheid en burger - in een onderzoek naar digitale handtekening systemen stelt dat "Aus Sicht des BSI sind folgende Signaturalgorithmen geeignet: [...] DSA-Varianten, basierend auf elliptischen Kurven" (uit : "Maßnahmenkatalog für digitale Signaturen", BSI, 18.11.1997, Version 1.0).

In ieder geval kan geconcludeerd worden dat de keuze van parameters bij elliptische kromme cryptosystemen van essentieel belang is en dat men daarbij op de hoogte moet zijn van de laatste ontwikkelingen op het gebied. Dit rapport bevat een waardevolle beschrijving van elliptische kromme cryptosystemen, toegespitst op authenticiteit toepassingen zoals digitale handtekeningen. Op basis hiervan krijgt men een goed beeld van de plus- en minpunten, de praktische implementeerbaarheid alsmede de laatste (cryptanalytische) ontwikkelingen rond dit soort toepassingen. Het onderliggende rapport maakte de eerder genoemde keuze van parameters daarom goed mogelijk.

Inhoudsopgave

Hoofdstuk 1 Inleiding public key-cryptografie ... 1
1.1 Public key-cryptografie ... 1
1.2 Het integer factorisatie-probleem en RSA ... 3
1.2.1 RSA ... 3
1.2.2 Aanvallen voor RSA ... 4
1.3 Het discrete logaritme-probleem en handtekeningen ... 4
1.3.1 Discrete logaritme handtekeningen-systemen ... 5
1.3.1.1 Gegeneraliseerd model ... 5
1.3.1.2 ElGamal ... 6
1.3.1.3 DSA ... 7
1.3.1.4 Schnorr ... 8
1.3.1.5 Nyberg-Rueppel ... 8
1.3.2 Aanvallen voor het 'gewone' discrete logaritme-probleem ... 9
1.3.2.1 Exhaustive search ... 9
1.3.2.2 Baby-step Giant-step ... 9
1.3.2.3 Pollard ρ ... 10
1.3.2.4 Pohlig-Hellman ... 10
1.3.2.5 Index-Calculus ... 10
1.4 Samenvatting en verdere hoofdstukindeling ... 11
Hoofdstuk 2 Beschrijving van systemen gebaseerd op elliptische krommen ... 13
2.1 Inleiding elliptische krommen ... 14
2.2 Het elliptische kromme discrete logaritme-probleem en handtekeningen ... 15
2.2.1 ElGamal met elliptische krommen ... 16
2.2.2 ECDSA ... 17
2.2.3 Schnorr met elliptische krommen ... 17
2.2.4 Nyberg-Rueppel met elliptische krommen ... 17
Hoofdstuk 3 De cryptografische sterkte van elliptische kromme-systemen ... 19
3.1 Discrete logaritme-aanvallen ... 19
3.1.1 Exhaustive search ... 20
3.1.2 Baby-step Giant-step ... 20
3.1.3 Pollard ρ ... 20
3.1.4 Pohlig-Hellman ... 21
3.1.5 Index-Calculus ... 21
3.2 Aanvallen voor speciale krommen ... 21
3.2.1 Singuliere krommen ... 21
3.2.2 MOV-reductie ... 22
3.2.3 Smart-Satoh-Araki ... 23

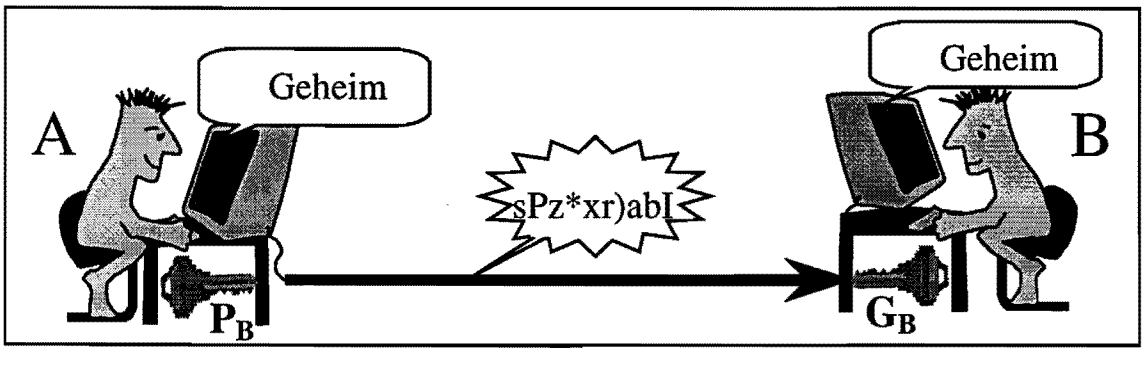
3.3 Conclusie ... 23
Hoofdstuk 4 Veilig te achten sleutellengtes voor de eerstkomende 10 jaar ... 24
4.1 Sleutellengtes ... 24
4.2 Software en hardware aanvallen ... 25
4.2.1 Software aanvallen ... 25
4.2.2 Hardware aanvallen ... 26
4.3 De Certicom challenge ... 26
4.4 Conclusie ... 27
Hoofdstuk 5 Parameters en kwaliteitscriteria ... 28
5.1 De keuze van het lichaam en de representatie van zijn elementen ... 28
5.1.1 Het lichaam $GF(p)$ ... 29
5.1.2 Het lichaam $GF(2^m)$ ... 29
5.1.2.1 Polynoombasis ... 29
5.1.2.2 Normaalbasis ... 30
5.2 De keuze van de kromme ... 31
5.3 De keuze van het genererend punt ... 34
5.4 Samenvatting: parametergeneratie ... 34
Hoofdstuk 6 De praktische implementeerbaarheid ... 36
6.1 Vergelijking ECDSA en DSA/RSA ... 36
6.1.1 De rekentijd ... 37
6.1.2 De geheugenopslag ... 39
6.1.3 De bandbreedte ... 39
6.1.4 Andere opmerking ... 40
6.2 Standaards in ontwikkeling ... 40
6.3 Commerciële elliptische kromme-produkten ... 41
Bibliografie ... 43
Appendix A Complexiteit ... 46
A.1 Ordesymbolen ... 46
A.2 Subexponentiële algoritmen ... 46
Appendix B Modulo rekenen, lichamen, groepen en de discrete logaritme ... 47
B.1 Gehele getallen modulo n ... 47
B.2 Lichamen en groepen ... 48
B.3 Discrete logaritme ... 48
Appendix C Elliptische krommen ... 50
C.1 De optelling op een elliptische kromme ... 50
C.2 De discriminant van een Weierstraß-vergelijking en (super)singulariteit ... 52
C.3 Het aantal punten op een elliptische kromme ... 53
C.4 Point compressie ... 54
Appendix D RSA gebaseerd op elliptische krommen ... 55

Hoofdstuk 1
Inleiding public key-cryptografie

In dit hoofdstuk wordt eerst kort uitgelegd wat public key-cryptografie precies is. Vervolgens wordt besproken wanneer een wiskundig probleem geschikt is voor gebruik in public key-cryptosystemen. De twee meest gebruikte problemen worden besproken: hun definitie, voorbeelden van public key-cryptosystemen die op deze problemen gebaseerd zijn en een overzicht van de aanvallen die bekend zijn voor deze dit soort systemen, voor zover deze van belang zijn voor de rest van het document. Tot slot wordt de verdere hoofdstukindeling van dit document kort samengevat.

1.1 Public key-cryptografie

In de cryptografie worden twee soorten systemen gebruikt: symmetrische en asymmetrische. Het kenmerk van symmetrische cryptosystemen is dat, als twee gebruikers het willen gebruiken voor de beveiliging van informatie tussen hen, ze beiden over één en dezelfde sleutel moeten beschikken. Bij asymmetrische cryptosystemen daarentegen hebben alle gebruikers twee, aan elkaar gerelateerde, sleutels: een geheime, alleen aan de eigenaar bekende, sleutel en een publieke, voor iedereen beschikbare, sleutel. Nu heeft een gebruiker A alleen de publieke sleutel P_B van de andere gebruiker B nodig om een bericht te vercijferen voor B; gebruiker B kan dit bericht dan ontcijferen met behulp van zijn eigen geheime sleutel G_B (zie Figuur 1-1). Of A gebruikt zijn eigen geheime sleutel om een boodschap te signeren, zodat B deze handtekening kan checken met behulp van de publieke sleutel van A. De asymmetrische systemen behoren tot de **public key-cryptografie**.



Figuur 1-1 Encryptie met behulp van een public-key systeem: A stuurt een geheim naar B, vercijferd met de publieke sleutel P_B van B, zodat B dit kan ontcijferen met zijn geheime sleutel G_B.

In 1976 werd public key-cryptografie voor het eerst beschreven door Whitfield Diffie en Martin Hellman [10] en sindsdien zijn er verscheidene public-key systemen voorgesteld, zoals bijvoorbeeld RSA en ElGamal. Al deze cryptosystemen hebben met elkaar gemeen dat ze wat betreft hun veiligheid zijn gebaseerd op de moeilijkheid van een wiskundig probleem. De meest gebruikte problemen zijn het factoriseren van grote getallen en het discrete logaritme-probleem (voor een nadere uitleg van deze problemen: zie paragrafen 1.2, 1.3, 2.2 en appendix B.3).

Wiskundige problemen worden moeilijk genoemd, als er geen algoritmen bestaan die deze problemen snel genoeg kunnen oplossen. De vraag hierbij is natuurlijk wat snel genoeg is.

De snelheid wordt altijd uitgedrukt in termen van de grootte van de input. Hoe groter de input is, des te langer zal een bepaalde berekening duren in het algemeen. Een voorbeeld is het optellen van twee getallen: de som van 11 + 12 is sneller te bepalen dan de som van 1234567 + 2345678. In sommige algoritmen neemt de rekentijd echter sneller toe dan in andere.

Een algoritme wordt nu snel met betrekking tot de grootte van de input genoemd, als de tijd die het nodig heeft om het probleem op te lossen naar boven **polynomiaal** begrensd is. Dit betekent dat deze tijd kleiner of gelijk is aan cn^d, waarbij c en d willekeurige constanten zijn en n de grootte van de input is. Een algoritme is langzaam, als de looptijd naar beneden **exponentieel** begrensd is, dat wil zeggen dat de tijd groter of gelijk is aan een e-macht met de grootte van de input in de exponent. Dus de tijd wordt naar beneden begrensd door $ce^{p(n)}$, waarbij c weer een constante is en $p(n)$ een expressie in de inputgrootte n.

Voor sommige problemen bestaan er algoritmen die **subexponentieel** genoemd worden. De tijd die deze algoritmen nemen ligt tussen polynomiale en exponentiële tijd in. Deze tijd wordt beschreven door een bepaalde expressie $L[x,c,\alpha]$, die afhangt van de grootte van de inputruimte x en twee constanten c en α. De constante α ligt tussen 0 en 1 en geeft aan hoe goed het algoritme is: het is polynomiaal als α gelijk is aan 0 en exponentieel als α gelijk is aan 1. (Zie ook appendix A.2.)

Voor de beoordeling van een probleem is verder ook van belang welke inputruimte er precies gebruikt wordt (gehele getallen, reële getallen, etcetera) en met welke bewerking (optelling, vermenigvuldiging, en dergelijke). Soms bestaan er namelijk efficiënte algoritmen in de ene ruimte+bewerking maar in een andere combinatie niet. (Zie appendix B.3 voor een voorbeeld van dit verschijnsel bij het discrete logaritme-probleem.)

Kortom, een wiskundig probleem wordt moeilijk genoemd in een ruimte met een bepaalde bewerking, als er alleen exponentiële algoritmen hiervoor bekend zijn; het wordt als makkelijk beschouwd als er polynomiale algoritmen voor bestaan. Als het beste algoritme een subexponentieel algoritme is, wordt het probleem eenvoudiger naarmate de α in de expressie $L[x,c,\alpha]$ voor de looptijd van dit algoritme kleiner is. In de cryptografie gebruikt men daarom het liefst problemen waarvoor alleen exponentiële algoritmen gevonden zijn.

De cryptografisch meest gebruikte, moeilijke, wiskundige problemen zijn het factoriseren van grote getallen en het (gewone) discrete logaritme-probleem. Het beste algoritme dat voor deze problemen bekend is, is echter subexponentieel (met α gelijk aan $1/3$). Een derde probleem, dat op dit moment steeds populairder wordt omdat er tot nu toe alleen exponentiële algoritmen voor bestaan, is het discrete logaritme-probleem op de punten van een elliptische kromme.

In de volgende twee paragrafen worden de eerste twee problemen beschreven. Van beide wordt er een toepassing gegeven. Ook wordt de veiligheid van dit soort systemen besproken, waarbij speciale aandacht uitgaat naar het discrete logaritme-probleem, aangezien dit probleem veel analogie vertoont met het elliptische kromme discrete logaritme-probleem. In de overige hoofdstukken zal dieper op dit laatste probleem en zijn toepassingen ingegaan worden.

1.2 Het integer factorisatie-probleem en RSA

Zoals de naam al impliceert, heeft het integer factorisatie-probleem te maken met het factoriseren van gehele getallen: het factoriseren van grote getallen die het produkt zijn van grote priemfactoren blijkt over het algemeen moeilijk te zijn. De exacte definitie van het integer factorisatie-probleem is als volgt. Stel dat er een positief geheel getal n gegeven is, welke het produkt is van twee grote priemgetallen p en q, bepaal dan deze p en q.

Aangezien er efficiënte algoritmen bestaan die grote priemgetallen vinden, is het 'eenvoudig' om zo'n getal n te construeren, terwijl het factoriseren van n een moeilijk probleem is. Dit probleem zou daarom voor een cryptosysteem gebruikt kunnen worden.

1.2.1 RSA

Een bekend systeem dat is gebaseerd op de moeilijkheid van het factoriseren is RSA. Dit cryptosysteem, genaamd naar de uitvinders ervan, Ron Rivest, Adi Shamir en Len Adleman, werd in 1978 gepubliceerd [36] en was het eerste praktische public key-cryptosysteem dat werd voorgesteld.

Bij RSA bestaat de publieke sleutel uit een paar (n,e), waarbij n het produkt van twee grote priemgetallen p en q is en e een getal tussen 1 en $n-1$, zodanig dat e geen gemeenschappelijke delers met $(p-1)(q-1)$ heeft. Als nu een boodschap m, welke een getal tussen 1 en $n-1$ is, met deze sleutel vercijferd wordt, berekent men $m^e \pmod{n}$, oftewel de uitkomst die men krijgt, als eerst m tot de e-de macht wordt verheven en vervolgens van dit getal de rest na deling met n wordt genomen (zie appendix B.1).

De corresponderende geheime sleutel bij het paar (n,e) is het getal d dat voldoet aan:

$$de \equiv 1 \pmod{\lambda(n)},$$

waarbij $\lambda(n)$ het kleinste gemene veelvoud van $p-1$ en $q-1$ is. Deze d heeft namelijk de eigenschap dat

$$(m^e)^d \equiv m \pmod{n} \text{ (en ook } (m^d)^e \equiv m \pmod{n}).$$

Hieruit volgt dat de ontcijfering neerkomt op het verheffen van de vercijferde boodschap tot de macht d gerekend modulo n.

	RSA
Modulus	$n = pq$, p en q priem
Basisbewerking	Vermenigvuldiging
Publieke sleutel	e, met $1 < e < n-1$
Geheime sleutel	d, met $ed \equiv 1 \pmod{\lambda(n)}$

Het vinden van de geheime sleutel d, gegeven n en e, is net zo moeilijk als het factoriseren van n. Verder denkt men dat, als iemand de boodschap m wil vinden uit $m^e \pmod{n}$, hij de factorisatie van n ook echt nodig heeft; dit is echter niet bewezen. Daarom zegt men dat RSA gebaseerd is op het integer factorisatie-probleem.

1.2.2 Aanvallen voor RSA

Het beste algoritme dat tot nu toe bekend is voor het factoriseren van RSA-moduli is de **general number field sieve**. Dit algoritme neemt subexponentiële tijd, en wel $L[x,c,1/3]$, voor een bepaalde constante c. Bovendien zijn er een aantal voorwaarden waaraan p en q moeten voldoen om het integer factorisatie-probleem zo moeilijk mogelijk te maken. Er is bijvoorbeeld een snel factorisatie-algoritme, als $p-1$ en $q-1$ alleen kleine priemfactoren hebben. Op dit moment moet de n in RSA bovendien minstens 512 bits lang zijn om enige veiligheid te bieden en zelfs 1024 bits of meer voor lange termijn-veiligheid.

1.3 Het discrete logaritme-probleem en handtekeningen

Het discrete logaritme-probleem kan op verschillende wiskundige groepen gedefinieerd worden. In dit document wordt met het 'gewone' discrete logaritme-probleem die variant bedoeld die als groep een eindig lichaam $GF(q)$ heeft met als bewerking de vermenigvuldiging in deze groep. Als q gelijk is aan een zeker priemgetal p, is deze groep gelijk aan de gehele getallen modulo p. Aangezien deze laatste gebruikt wordt in de cryptosystemen die in dit hoofdstuk aan bod komen, definiëren we hier het 'gewone' discrete logaritme-probleem aan de hand van deze groep. (Voor meer informatie over groepen en dergelijke zie Appendix B.)

Neem daarom de gehele getallen modulo een priemgetal p met uitzondering van 0: $\{1,2,\ldots, p-1\}$. Deze getallen vormen een wiskundige groep met de gewone vermenigvuldiging van getallen. Deze groep is cyclisch, dat wil zeggen dat er een getal α in deze groep te vinden is, zodat elk getal uit de groep te schrijven is als een macht van deze α (modulo p).

Voorbeeld

Neem p gelijk aan 7 en α gelijk aan 3. Dan geldt het volgende:

$$\alpha \equiv 3 \bmod 7$$
$$\alpha^2 \equiv 9 \equiv 2 \bmod 7$$
$$\alpha^3 \equiv 2 \cdot 3 \equiv 6 \bmod 7$$
$$\alpha^4 \equiv 6 \cdot 3 \equiv 18 \equiv 4 \bmod 7$$
$$\alpha^5 \equiv 4 \cdot 3 \equiv 12 \equiv 5 \bmod 7$$
$$\alpha^6 \equiv 5 \cdot 3 \equiv 15 \equiv 1 \bmod 7$$

Dus elk element van de groep $\{1,2,3,4,5,6\}$ kan geschreven worden als een macht van α.

De groep $\{1,2,..., p-1\}$ is dus gelijk aan $\{1, \alpha, \alpha^2, \ldots, \alpha^{p-1}\}$ (α^p is weer gelijk aan 1); de groep gegenereerd door α zal genoteerd worden als $\langle\alpha\rangle$. Het **'gewone' discrete logaritme-probleem** op deze groep wordt nu als volgt gedefinieerd: vind het getal l tussen 0 en p, zodat $\beta \equiv \alpha^l$ (modulo p), wanneer het grondtal α, de modulus p en een getal β gegeven zijn. Deze l wordt de logaritme van β naar α genoemd en wordt ook wel genoteerd als $l = \log_\alpha \beta$. Aangezien bij dit probleem β snel te berekenen is uit α, l en p, terwijl het vinden van l moeilijk is gegeven α, β en p, is dit probleem geschikt voor toepassing in een cryptosysteem.

1.3.1 Discrete logaritme handtekeningen-systemen

In 1984 stelde Taher ElGamal als eerste voor om het discrete logaritme-probleem te gebruiken voor public key-systemen. In [11] definieerde hij een systeem voor encryptie en een andere voor digitale handtekeningen. Nadien zijn er vele varianten op ElGamal's handtekeningen-systeem bedacht, waaronder DSA [29], Schnorr [40] en Nyberg-Rueppel [30][31].

Al deze systemen kunnen gegeneraliseerd worden naar één model, dat hieronder eerst beschreven zal worden. Vervolgens worden nog kort de systemen ElGamal, DSA, Schnorr en Nyberg-Rueppel gespecificeerd met enkele opmerkingen over de veiligheid van deze systemen.

1.3.1.1 Gegeneraliseerd model

In deze paragraaf wordt een gegeneraliseerd model van discrete logaritme-systemen behandeld (zie ook [39]). Allereerst wordt er beschreven hoe een sleutelpaar tot stand komt. Daarna volgt een uitleg over hoe het signeren van een boodschap met behulp van dit sleutelpaar in zijn werk gaat en hoe deze handtekening geverifieerd kan worden.

Voor het genereren van een sleutelpaar zijn een drietal parameters p, q en α nodig, welke publiek gemaakt mogen worden. Kies p gelijk aan een groot priemgetal en neem q gelijk aan $p-1$ of een grote priemfactor van $p-1$. Vervolgens moet α tussen 1 en p zo gekozen worden, dat q het kleinste getal is waarvoor $\alpha^q \equiv 1 \pmod p$; q is de **orde** van α. Een tweede eis aan α is dat het discrete logaritme probleem in de groep $\langle\alpha\rangle$ moeilijk is.

De geheime sleutel is nu een random getal x kleiner dan q; de bijbehorende publieke sleutel is $y = \alpha^x \bmod q$.

Voor het signeren van een boodschap m wordt er een geheime, random k ongelijk aan 1 gekozen, welke kleiner is dan q en geen delers anders dan 1 gemeenschappelijk heeft met q (i.e. $ggd(k,q) = 1$). (Als q een priemgetal is, voldoet elke k kleiner dan q.) Met deze k wordt eerst $r = \alpha^k \bmod p$ berekend (de zogenaamde **commitment**).

Het tweede deel s van de handtekening over een boodschap m wordt opgelost uit de **signature equation**, welke gelijk is aan $ak \equiv b + cx \pmod q$, waarbij s, m en r in de coëfficiënten a, b en c bevat zijn. In onderstaande tabel worden alle mogelijkheden voor a, b en c opgesomd. (In de tabel is $r' = r \bmod q$.)

Mogelijke permutaties voor a, b en c		
±r'	±s	m

±r'm	±s	1
±r'm	±ms	1
±mr'	±r's	1
±ms	±r's	1

Elke regel in de tabel geeft zes mogelijkheden, als de '±''en buiten beschouwing worden gelaten. De eerste regel geeft dan bijvoorbeeld de volgende mogelijke signature equations:

1. r'k = s + mx mod q
2. r'k = m + sx mod q
3. sk = r' + mx mod q
4. sk = m + r'x mod q
5. mk = s + r'x mod q
6. mk = r' + sx mod q

Om de handtekening te verifiëren moet de ontvanger van de boodschap m met handtekening (r,s) checken dat $r^a \equiv \alpha^b y^c \pmod{p}$. Deze vergelijking wordt de **verificatievergelijking** genoemd.

Het voorgaande wordt samengevat in de volgende tabel:

	Discrete logaritme-systeem
Parameters	p, q en α met p priem, q gelijk aan $p\text{-}1$ of een priemfactor van $p\text{-}1$, $1 < \alpha < p$ met de orde van α gelijk aan q en d.l.-probleem in $\langle\alpha\rangle$ moeilijk
Geheime sleutel	random getal $0 < x < q$
Publieke sleutel	$y = \alpha^x \bmod p$
Handtekening voor m	(r,s), waarbij $r = \alpha^k \bmod p$ met k random, $1 < k < q$, ggd$(k, q) = 1$, en s opgelost uit $ak \equiv b + cx \pmod{q}$
Verificatie handtekening	controleer of $r^a \equiv \alpha^b y^c \pmod{p}$

ElGamal en DSA zijn in essentie gebaseerd op de vierde signature vergelijking die uit de eerste regel van het tabelletje voor a, b en c volgde. Schnorr en Nyberg-Rueppel zijn nauw verwant aan de vijfde van deze vergelijkingen. Voor een aantal andere signature vergelijkingen die bevat zijn in dit model zijn ook handtekening-systemen gedefinieerd, maar voor verreweg de meeste vergelijkingen is dit niet gedaan.

1.3.1.2 ElGamal

ElGamal's handtekeningensysteem [11] volgt precies het stramien dat in paragraaf 1.3.1.1 beschreven is. In de volgende tabel wordt hiervan een overzicht gegeven.

	ElGamal handtekeningen
Parameters	p, q en α, waarbij p priem, q gelijk aan $p\text{-}1$ en $1 < \alpha < p$ met de orde van α gelijk aan q en d.l.-probleem in $\langle\alpha\rangle$ moeilijk
Geheime sleutel	random getal $0 < x < q$
Publieke sleutel	$y = \alpha^x \bmod p$
Handtekening voor m	(r,s), waarbij $r = \alpha^k \bmod p$ met k random, $1 < k < q$,

	ggd$(k, q) = 1$, en s zodanig dat $s = k^{-1}(m - xr) \bmod q$
Verificatie handtekening	controleer of $0 < r < q$ en $y^r r^s \pmod p \equiv \alpha^m \pmod p$

In de tabel is er sprake van een handtekening over de boodschap m. In werkelijkheid wordt er echter eerst een **hashfunctie** h op m toegepast. Een hashfunctie is, eenvoudig gezegd, een functie die een input van willekeurige lengte afbeeldt op een output van vaste (kortere) lengte; deze functie is bovendien eficiënt toe te passen op een input, maar gegeven de output is het moeilijk een bijbehorende input te vinden. In de tabel moet dan overal m vervangen worden door $h(m)$.

Wat betreft de veiligheid van dit systeem kunnen de volgende opmerkingen geplaatst worden (zie ook [26]):

1. Als het discrete logaritme-probleem in $\langle\alpha\rangle$ moeilijk is, kan een aanvaller met een zelfgegenereerde commitment r voor een boodschap m de bijbehorende s slechts door random te kiezen vinden. De kans van slagen is hierbij $1/p$, wat klein is als p groot is.
2. Voor elke boodschap moet een andere random k gebruikt worden. Als er namelijk twee verschillende boodschappen getekend zijn met behulp van dezelfde k, kan deze k berekend worden. Zodra k bekend is, kan vervolgens de geheime sleutel x bepaald worden.
3. Het gebruik van een hashfunctie is noodzakelijk om te voorkomen dat er willekeurig paren (r,s) gegenereerd kunnen worden, die een handtekening vormen voor een boodschap m. (Al is er geen garantie dat deze boodschap leesbare tekst bevat.)
4. De controle $0 < r < p$ voorkomt dat een aanvaller boodschappen naar keuze kan signeren, wanneer hij bovendien beschikt over een boodschap met een goede handtekening.

1.3.1.3 DSA

In 1991 werd DSA voorgesteld als "Federal Information Processing Standard"; het is in 1994 geaccepteerd als "Digital Signature Standard" [29]. Ook in DSA wordt niet de boodschap m maar de hashwaarde $h(m)$ gebruikt, waarbij als hashfunctie h het algoritme SHA-1 [12] wordt gebruikt. In onderstaande tabel wordt DSA samengevat (lees $h(m)$ in plaats van m).

	DSA handtekeningen
Parameters	p, q en α, waarbij p priem, q een priemfactor van $p-1$, $1 < \alpha < p$ met de orde van α gelijk aan q en d.l.-probleem in $\langle\alpha\rangle$ moeilijk
Geheime sleutel	random getal $0 < x < q$
Publieke sleutel	$y = \alpha^x \bmod p$
Handtekening voor m	(r,s), waarbij $r = (\alpha^k \bmod p) \bmod q$ met k random, $1 < k < q$, en $s = (k^{-1}(m + xr)) \bmod q$
Verificatie handtekening	controleer of $0 < r < q$, $0 < s < q$ en $r \equiv ((\alpha^{u1} y^{u2}) \bmod p) \pmod q$ met $u1 = mw \bmod q$, $u2 = rw \bmod q$ en $w = s^{-1} \bmod q$

Aan p en q zijn nog enkele beperkingen opgelegd. Het priemgetal q wordt eerst gegenereerd en heeft een lengte van 160 bits. Vervolgens wordt p bepaald met een lengte tussen de 512 en 1024 bits.

DSA is gebaseerd op twee verschillende discrete logaritme problemen: het discrete logaritme-probleem in $\mathbf{Z}_p^*$ en die in $\langle\alpha\rangle$. Wat betreft de veiligheid zijn de opmerkingen uit paragraaf 1.3.1.2 ook van toepassing op DSA.

1.3.1.4 Schnorr

Het Schnorr-handtekeningensysteem is weer een andere variant van ElGamal. De sleutelgeneratie is hetzelfde als bij DSA met het verschil dat er geen beperkingen aan de lengtes van p en q zijn opgelegd. Ook Schnorr maakt gebruik van een hashfunctie h, maar op een andere manier: als input van h wordt de concatenatie van m en een ander getal e genomen (zie de overzichtstabel).

	Schnorr handtekeningen
Parameters	p, q en α, waarbij p priem, q een priemfactor van $p-1$, $1 < \alpha < p$ met de orde van α gelijk aan q en d.l.-probleem in $\langle\alpha\rangle$ moeilijk
Geheime sleutel	random getal $0 < x < q$
Publieke sleutel	$y = \alpha^x \bmod p$
Handtekening voor m	(r,s), waarbij $r = h(m\|\|e)$ met $e = (\alpha^k \bmod p) \bmod q$ en k random, $1 < k < q$, en $s = (xr+k) \bmod q$
Verificatie handtekening	controleer of $r = h(m\|\|v)$, waarbij $v = (\alpha^s y^{-r}) \bmod p$

1.3.1.5 Nyberg-Rueppel

De laatste variant op ElGamal die hier gegeven wordt, is het Nyberg-Rueppel-scheme, een handtekeningensysteem waarbij ook "message recovery" mogelijk is. Dit wil zeggen dat uit de handtekening (r,s) de boodschap m teruggevonden kan worden. De sleutelgeneratie gaat hetzelfde als bij het Schnorr-systeem uit de vorige paragraaf. (De beschrijving zoals die hier gegeven wordt, is overgenomen uit [26]).

	Nyberg-Rueppel handtekeningen
Parameters	p, q en α, waarbij p priem, q een priemfactor van $p-1$, $1 < \alpha < p$ met de orde van α gelijk aan q en d.l.-probleem in $\langle\alpha\rangle$ moeilijk
Geheime sleutel	random getal $0 < x < q$
Publieke sleutel	$y = \alpha^x \bmod p$
Handtekening voor m	(r,s), waarbij $r = me$ met $e = \alpha^{-k} \bmod p$ met k random, $1 < k < q$, en $s = k + rx \bmod q$
Verificatie handtekening	controleer of $0 < e < p$, $0 \leq s < q$ en $m = vr$ met $v = \alpha^s y^{-r} \bmod p$

De opmerkingen met betrekking tot de veiligheid die in paragraaf 1.3.1.2 zijn gemaakt zijn ook hier weer van toepassing. Een extra opmerking is dat $0 < e < p$ zeker gecheckt moet worden om te voorkomen dat een aanvaller boodschappen naar keuze kan signeren, als hij één boodschap met bijbehorende handtekening tot zijn beschikking heeft. (Zie bijvoorbeeld [26].)

1.3.2 Aanvallen voor het 'gewone' discrete logaritme-probleem

Discrete logaritme-systemen kunnen in principe op twee manieren gekraakt worden. Enerzijds kan het achterliggende discrete logaritme-probleem opgelost worden, zodat de geheime sleutel berekend kan worden uit de publieke sleutel. Anderzijds kan er geprobeerd worden handtekeningen te genereren zonder het discrete logaritme-probleem direct op te lossen.

Deze laatste aanvallen zijn al kort besproken bij de beschrijving van de discrete logaritme-systemen in paragraaf 1.3.1.Van de eerste soort aanvallen zijn er tot nu toe de volgende vijf bekend:

1. Exhaustive search
2. Baby-step Giant-step
3. Pollard ρ
4. Pohlig-Hellman
5. Index-Calculus

In de volgende deelparagrafen zal elk van deze vijf aanvallen kort besproken worden. Er wordt daarbij vanuit gegaan dat er de groep $\langle\alpha\rangle$ gegenereerd wordt door een α van orde n (dus $\alpha^n = 1$) met n niet noodzakelijk een priemgetal. Er is verder een β gegeven welke gelijk is aan α^l; het gehele getal l wordt gezocht.

Het blijkt dat de eerste vier aanvallen elk exponentiële rekentijd nemen, terwijl de laatste subexponentiële rekentijd vergt. Deze laatste aanval is daarom de efficiëntste aanval op het 'gewone' discrete logaritme-probleem.

1.3.2.1 Exhaustive search

Zoals de naam al zegt, is de exhaustive search-aanval, niets anders dan achtereenvolgens alle mogelijkheden uitproberen. Dus bij het discrete logaritme-probleem komt het neer op (met dezelfde notatie als eerder) het achtereenvolgens berekenen van $\alpha, \alpha^2, \alpha^3, \ldots$, totdat $\alpha^l = \beta$ is bereikt.

Daar α orde n heeft, ligt de rekentijd van exhaustive search in de orde van n bewerkingen. De totale rekentijd is daarom exponentieel in n, zodat deze aanval niet bruikbaar is voor grote n.

1.3.2.2 Baby-step Giant-step

De baby-step giant-step aanval is een slimmere manier om een exhaustive search uit te voeren: er wordt geheugen gebruikt om de rekentijd te verlagen. Dit gaat als volgt in zijn werk.

Neem s gelijk aan het gehele getal $\lceil\sqrt{n}\rceil$ en stel dat het getal l, dat gezocht wordt, geschreven wordt als $l = is + j$, waarbij i en j gehele getallen zijn met $0 \leq j < s$. Het getal j is dus de rest na deling van l door s.

In het geval van het discrete logaritme-probleem geldt dan dat $\beta(\alpha^{-s})^i = \alpha^j$. Maak nu een tabel van de paren (j, α^j), voor $0 \leq j < s$ (de baby-steps), waarbij de tabel gesorteerd wordt op de tweede coördinaat om snel hierin te kunnen zoeken. Bereken

daarna achtereenvolgens β, $\beta\alpha^{-s}$, $(\beta\alpha^{-s})\alpha^{-s} = \beta(\alpha^{-s})^2$, ... (de giant-steps), totdat $\beta(\alpha^{-s})^i$ voorkomt in de tabel. Dan zijn i en j bekend en dus ook l.

Bij deze aanval zijn orde $\sqrt{n}$ bewerkingen nodig en worden orde $\sqrt{n}$ elementen in het geheugen opgeslagen. Ook hier is de rekentijd daarom exponentieel.

1.3.2.3 Pollard ρ

De Pollard ρ-methode [34] neemt net als de baby-step giant-step aanval in de orde van $\sqrt{n}$ bewerkingen, maar gebruikt nagenoeg geen geheugen.

Bij deze methode worden op een bepaalde manier drie rijen $(x_i)_{i \geq 0}$, $(a_i)_{i \geq 0}$ en $(b_i)_{i \geq 0}$ gegenereerd die de volgende eigenschappen hebben: (in verklarende woordenlijst de bovenstaande gebruikte notatie ...)

$$x_i = \alpha^{a_i} \beta^{b_i}, i \geq 0$$

Deze generatie van rijen gaat net zolang door totdat er een element x_j gevonden is die gelijk is aan een eerder element x_i. Zodra zo'n element gevonden is, is het discrete logaritme-probleem opgelost. Dan geldt namelijk:

$$x_i = \alpha^{a_i} \beta^{b_i} = \alpha^{a_j} \beta^{b_j} = x_j$$

oftewel, als alle termen met β naar de ene kant worden gebracht en alle termen met α naar de andere:

$$\beta^{b_i - b_j} = \alpha^{a_j - a_i}.$$

Door aan beide zijden de logaritme naar α te nemen volgt hier dan uit dat

$$(b_i - b_j)\, l = (b_i - b_j) \log_\alpha \beta = a_j - a_i \bmod n.$$

(De "mod n" volgt uit het feit dat α^n gelijk is aan 1 in het gebruikte lichaam.) Aldus wordt de volgende expressie voor de discrete logaritme verkregen:

$$l = (a_j - a_i)/(b_i - b_j) \bmod n.$$

1.3.2.4 Pohlig-Hellman

De aanval van Pohlig-Hellman maakt gebruik van de factorisatie van de orde n van het element α. Als deze factorisatie van n gelijk is aan

$$n = \prod_{i=1}^{t} p_i^{\lambda_i},$$

ligt het aantal benodigde bewerkingen van het Pohlig-Hellman-algoritme in de orde van

$$\sum_{i=1}^{t} \lambda_i (\log n + \sqrt{p_i} \log p_i).$$

De werking van het algoritme komt neer op het berekenen van $(l \bmod p_i\lambda_i)$ voor alle i van 1 tot t. Vervolgens kan met de Chinese Reststelling $(l \bmod n)$ berekend worden (zie appendix B.1).

Aan de expressie voor de rekentijd is te zien, dat het algoritme alleen efficiënt is, als n het produkt is van kleine priemgetallen. Daarom is het van cryptografisch belang om een α te kiezen waarvan de orde n een groot priemgetal bevat.

1.3.2.5 Index-Calculus

In tegenstelling tot de voorgaande algoritmen is de Index-Calculus-methode niet op elke willekeurige groep toepasbaar. Het is echter wel een subexponentieel algoritme.

Bijvoorbeeld voor het discrete logaritme-probleem in het eindige lichaam van p^N elementen zijn dit de beste algoritmen met rekentijden van $L[p^N, 1.526, 1/3]$, wanneer $(\log p)^2 < N$, en $L[p^N, 1.923, 1/3]$, wanneer $N < (\log p)^{1/2}$; deze algoritmen heten de function field sieve en de number field sieve.

Een Index-Calculus-algoritme maakt gebruik van een kleine verzameling elementen uit de groep, de zogenaamde **factor base**, waarmee zo veel mogelijk elementen uit de hele groep gerepresenteerd kunnen worden (als produkt van de elementen uit de factor base). Zo'n factor base zou bijvoorbeeld uit priemgetallen kunnen bestaan in het geval van de gewone gehele getallen.

Stel dat de factor base gegeven wordt door $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_t\}$. In het algoritme worden random exponenten s gezocht zodat geldt:

$$\alpha^s = \prod_{i=1}^{t} \gamma_i^{a_i},$$

voor positieve, gehele exponenten a_i. Door aan beide kanten de logaritme naar α te nemen wordt dan de volgende vergelijking verkregen:

$$s \equiv \sum_{i=1}^{t} a_i \log_\alpha \gamma_i \pmod{n}.$$

Als voldoende van deze exponenten s gegenereerd worden, wordt een stelsel van vergelijkingen gevonden, waaruit de $\log_\alpha \gamma_i \pmod{n}$ uit opgelost kunnen worden. Vervolgens worden dan weer random exponenten s genomen, totdat $\alpha^s\beta$ geschreven kan worden als het produkt van machten van de γ_i's:

$$\alpha^s \beta = \prod_{i=1}^{t} \gamma_i^{b_i},$$

voor positieve, gehele machten b_i. Als weer aan beide zijden de logaritme wordt genomen, geeft dit de volgende vergelijking:

$$sl \equiv s \log_\alpha \beta \equiv \sum_{i=1}^{t} b_i \log_\alpha \gamma_i \pmod{n}.$$

Daar de $\log_\alpha \gamma_i \pmod{n}$ gevonden waren uit het stelsel van vergelijkingen, kan l nu ook bepaald worden.

1.4 Samenvatting en verdere hoofdstukindeling

Public-key systemen zijn gebaseerd op een moeilijk wiskundig probleem. In dit hoofdstuk zijn twee problemen genoemd die op dit moment in cryptosystemen gebruikt worden: het integer factorisatie-probleem en het 'gewone' discrete logaritme-probleem. Een voorbeeld van een cryptosysteem gebaseerd op het eerste probleem is RSA; in ElGamal, DSA, Schnorr en Nyberg-Rueppel wordt gebruikt gemaakt van de moeilijkheid van het tweede probleem.

De discrete logaritme-systemen zijn uitgebreid onder de loep genomen, omdat deze vooral van belang zijn voor de rest van dit document. Van deze systemen zijn twee soorten aanvallen besproken, waarbij de algoritmes die het discrete logaritme-probleem proberen op te lossen het belangrijkst zijn. Deze laatste algoritmen worden in onderstaande tabel opgesomd tezamen met de bijbehorende complexiteiten (n is hierbij de orde van het genererende element α).

Algoritme	Complexiteit
Exhaustive Search	$O(n)$ berekeningen
Baby-step Giant-step	$O(\sqrt{n})$ berekeningen en $O(\sqrt{n})$ geheugen
Pollard ρ	$O(\sqrt{n})$ berekeningen
Pohlig-Hellman	$O(\sum_{i=1}^{t} \lambda_i (\log n + \sqrt{p_i} \log p_i))$ berekeningen als $n = \prod_{i=1}^{t} p_i^{\lambda_i}$
Index-Calculus	$L[p^N, c, 1/3]$, c een constante

Uit deze tabel volgt dat van de genoemde aanvallen de subexponentiële Index-Calculus-algoritmen het efficiëntst zijn. Verder volgt uit de Pohlig-Hellman-methode dat de orde van de parameter α een groot priemgetal moet zijn.

In de volgende hoofdstukken zal een derde moeilijk probleem geïntroduceerd worden: het discrete logaritme-probleem op de punten van een elliptische kromme (Hoofdstuk 2). Het blijkt dat dit probleem volledig analoog aan het 'gewone' discrete logaritme-probleem gedefinieerd kan worden; ook de 'gewone' discrete logaritme-cryptosystemen hebben een analoge elliptische kromme-variant.

In Hoofdstuk 3 zal vervolgens ingegaan worden op de cryptografische sterkte van deze elliptische kromme-systemen. Hierbij zal eerst gekeken worden naar de toepasbaarheid van de aanvallen voor het 'gewone' discrete logaritme-probleem op het elliptische kromme discrete logaritme-probleem. Daarna worden een aantal andere aanvallen aangestipt, die alleen toepasbaar zijn op speciale elliptische krommen.

Hoofdstuk 4 brengt het laatste nieuws op het gebied van 'gebroken elliptische kromme-systemen'. Dit wordt onder andere gedaan aan de hand van de Certicom challenge en de tot nu gerealiseerde of realiseerbare software en hardware attacks. De nadruk ligt hier op de vraag welke sleutellengtes veilig zijn met de huidig computersnelheden en –ontwikkelingen.

Voor de implementatie van een elliptische kromme-cryptosysteem moeten een aantal keuzes gemaakt worden wat betreft de keuze van de parameters, de te gebruiken lichamen en representaties daarvan en dergelijke. In Hoofdstuk 5 worden alle aspecten waar rekening mee gehouden moeten worden opgesomd. De praktische implementeerbaarheid van deze aspecten wordt vervolgens besproken in Hoofdstuk 6. In dit hoofdstuk wordt ook aandacht besteed aan welke cryptosystemen gebaseerd op elliptische krommen al op de markt verkrijgbaar zijn.

Hoofdstuk 2
Beschrijving van systemen gebaseerd op elliptische krommen

In 1985 stelden tegelijkertijd Neil Koblitz [18] en Victor Miller [27] als eersten voor om elliptische krommen te gebruiken in cryptosystemen. Zij lieten zien hoe systemen gebaseerd op het 'gewone' discrete logaritme-probleem ook met behulp van elliptische krommen gedefinieerd zouden kunnen worden; deze zijn dan gebaseerd op het elliptische kromme discrete logaritme-probleem. Aangezien het tweede probleem moeilijker leek bij dezelfde sleutellengte, zouden op deze manier net zo veilige of zelfs veiligere cryptosystemen verkregen kunnen worden met een kleinere sleutellengte. Dit laatste zou de efficiëntie van bepaalde cryptosystemen kunnen vergroten.

In 1991 stelden Koyama, Maurer, Okamoto en Vanstone [20] een RSA-achtig systeem voor met behulp van elliptische krommen, die echter gebaseerd is op het integer factorisatie-probleem. Later volgden nog enkele van dergelijke systemen, zoals in 1993 van Demytko [9], alle gebaseerd op het integer factorisatie-probleem (zie ook Appendix D voor een beschrijving van het Demytko-systeem, als voorbeeld van een RSA-achtig systeem). Ook hier dacht men dat het gebruik van elliptische krommen de veiligheid van RSA-achtige systemen zou kunnen vergroten. Er is echter gebleken dat de aanvallen op 'gewone' RSA ook varianten hebben die toepasbaar zijn op deze elliptische kromme-RSA, zodat deze laatste geen praktisch voordeel opleveren ten opzichte van 'gewone' RSA.

In de praktijk zijn dus vooral de elliptische kromme-systemen gebaseerd op het elliptische discrete logaritme-probleem in de belangstelling. Daarom wordt in dit document ook alleen deze klasse van cryptosystemen beschreven. In het vervolg zullen dan met elliptische kromme-cryptosystemen die systemen bedoeld worden, waarvan de veiligheid gebaseerd is op het discrete logaritme-probleem op een elliptische kromme.

Alvorens te beginnen met de beschrijving van deze cryptosystemen volgt eerst een kort stukje theorie over elliptische krommen en de definitie van het discrete logaritme-probleem op een elliptische kromme. Vervolgens wordt een beschrijving gegeven van systemen gebaseerd op dit probleem.

2.1 Inleiding elliptische krommen

Eenvoudig gezegd is een elliptische kromme een verzameling punten, welke bij elkaar opgeteld en van elkaar afgetrokken kunnen worden. Als twee willekeurige, mogelijk dezelfde, punten op de kromme worden opgeteld of afgetrokken, levert dit weer een punt op de kromme op.

Meer concreet kan elke elliptische kromme E beschreven worden door een zogenaamde **affine Weierstraß-vergelijking**:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

waarbij de a_i's in een bepaald getallenlichaam K zitten. Meestal is K gelijk aan $GF(2^m)$ of $GF(p)$, waarbij p een priemgetal is, i.e. de eindige lichamen met 2^m respectievelijk p elementen (m is hierbij een willekeurig positief geheel getal). De punten op de kromme zijn de paren (x,y) die aan deze vergelijking voldoen, waarbij x en y in K zitten, plus nog een extra punt O, het 'punt in het oneindige':

$$E = \{(x,y) \in K^2 \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup O.$$

Afhankelijk van welk lichaam er gebruikt wordt, kan deze vergelijking vereenvoudigd worden. Voor het lichaam $GF(p)$ is deze vereenvoudigde vergelijking gelijk aan:

$$y^2 = x^3 + ax + b,$$

waarbij a en b in $GF(p)$ zitten. Voor het lichaam $GF(2^m)$ zijn er twee soorten vergelijkingen mogelijk, maar de krommen over dit lichaam die in de praktijk gebruikt worden hebben alle de volgende vorm:

$$y^2 + xy = x^3 + ax^2 + b.$$

Als K gelijk is aan een eindig lichaam met q elementen, is het duidelijk dat de verzameling punten op de kromme eindig is: er zijn hoogstens q^2 mogelijkheden voor de coördinaten (x,y). Hasse heeft bewezen dat het aantal punten op de kromme over het lichaam van q elementen gelijk is aan $q + 1 - t$, waarbij $|t| < 2\sqrt{q}$. Het aantal punten op een kromme E wordt ook wel de **orde** van E genoemd en wordt hier genoteerd als $|E|$.

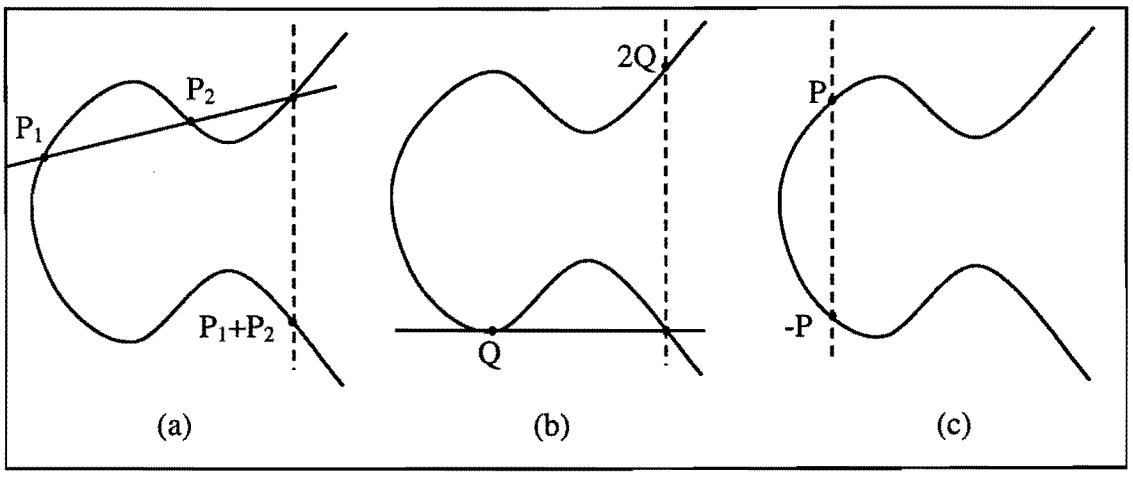
Op de punten van zo'n kromme kan een optelling gedefinieerd worden. Met deze optelling vormen de punten op de kromme een groep, waarbij het punt O hierbij het nulpunt is. (Zie ook appendix C.1.)

Het optellen van twee punten gaat als volgt in zijn werk, als het lichaam K gelijk aan de reële getallen genomen zou worden:

- Trek een rechte lijn door de twee punten; als deze punten gelijk zijn, neem dan de raaklijn aan de kromme.
- Neem het derde snijpunt van deze lijn met de kromme en spiegel dit in de symmetrieas van de kromme: dit is de uitkomst van de optelling.

In Figuur 2-1 wordt geïllustreerd hoe op een elliptische kromme E een voorstelling gemaakt kan worden van:

(a) de optelling $P_1 + P_2$ van twee verschillende punten P_1 en P_2;
(b) de optelling $2Q$ van twee dezelfde punten Q;
(c) de negatieve $-P$ van een punt P (i.e. $-P + P = O$).



Figuur 2-1 Het optellen van punten op een elliptische kromme.

Aan de hand van bovenstaande constructie van de som van twee punten kunnen expliciete vergelijkingen voor de coördinaten van deze som opgesteld worden in termen van de coördinaten van de twee punten. (Zie appendix C.1.) Deze vergelijkingen zijn niet alleen waar als K gelijk is aan de reële getallen, maar ook voor alle andere lichamen K.

Als het punt P nu n keer bij zichzelf wordt opgeteld, wordt dit geschreven als nP. Bijvoorbeeld: $4P = P + P + P + P$. Verder kan $-4P$ gezien worden als $-(4P)$ (de negatieve van $P+P+P+P$) of als $4(-P)$ (vier keer de negatieve van P bij zichzelf opgeteld); beide berekeningen geven dezelfde uitkomst. Als verder $0P$ gedefinieerd wordt als zijnde O, wordt de berekening van nP daarom ook wel de **scalaire vermenigvuldiging** van P met n genoemd, waarbij n elk geheel getal kan zijn.Voor elk punt P op de kromme bestaat er een positief geheel getal n zodat $nP=O$; het kleinste getal n die hieraan voldoet, wordt de **orde** van het punt P genoemd. Dit getal is een deler van het aantal punten op de kromme.

2.2 Het elliptische kromme discrete logaritme-probleem en handtekeningen

Stel nu dat een punt P op de kromme orde n heeft. Beschouw de groep gegenereerd door P, i.e. de verzameling $\{O, P, 2P, \ldots, (n-1)P\}$ van alle veelvouden van P; deze groep wordt genoteerd door $\langle P \rangle$. Neem verder Q gelijk aan lP, waarbij l een geheel getal tussen 0 en n is. Dan is l de discrete logaritme van Q naar P op de elliptische kromme oftewel de **elliptische kromme discrete logaritme van Q naar P**. Het probleem van het vinden van l gegeven P, Q en de kromme wordt **het elliptische kromme discrete logaritme-probleem** genoemd. Vergelijk dit met het 'gewone' discrete logaritme-probleem: gegeven een grondtal α, een modulus n en een getal β, vind het getal α zodanig dat $\beta \equiv \alpha^{l} \pmod{n}$. De discrete logaritme op een elliptische kromme is dus analoog gedefinieerd aan de 'gewone' discrete logaritme, alleen is de modulaire vermenigvuldiging vervangen door de optelling van punten op een elliptische kromme.

Bij het elliptische kromme discrete logaritme-probleem is Q relatief snel te berekenen uit l en P, maar het vinden van l is moeilijk, als P en Q gegeven zijn, precies zoals β en l bij het 'gewone' discrete logaritme-probleem. Dus ook in het geval van elliptische

krommen kan deze eigenschap gebruikt worden voor de constructie van asymmetrische cryptosystemen.

Gegeven het punt P van orde n op de kromme kan een sleutelpaar op de volgende manier gedefinieerd worden. De geheime sleutel is een random geheel getal l tussen 1 en n. De corresponderende publieke sleutel is het punt Q op de kromme gedefinieerd door lP. Zie ook onderstaande tabel.

	'gewoon'	Elliptisch
Basisbewerking	Vermenigvuldiging	Optelling
Basiselement	Genererend element α	Genererend punt P
Orde	Positief geheel getal n	Positief geheel getal n
Groep	$\langle\alpha\rangle = \{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$	$\langle P\rangle = \{O, P, 2P, \ldots, (n-1)P\}$
Geheime sleutel	random getal l (mod n)	random getal l (mod n)
Publieke sleutel	$\beta = \alpha^l$	$Q = lP$

Algemeen wordt aangenomen dat als iemand de geheime sleutel wil bemachtigen, deze het elliptische discrete logaritme-probleem zal moeten oplossen voor de publieke sleutel. Daarom zegt men dat de veiligheid van het type cryptosystemen dat dit soort sleutelparen gebruikt, gebaseerd is op de moeilijkheid van het elliptische discrete logaritme-probleem.

Daar het elliptische kromme discrete logaritme-probleem volledig analoog gedefinieerd is aan het gewone discrete logaritme-probleem hebben alle systemen die gebaseerd zijn op het discrete logaritme-probleem een elliptische kromme-variant. Voorbeelden van zulke cryptosystemen zijn de systemen die in het vorige hoofdstuk zijn behandeld: ElGamal, DSA, the Schnorr signature scheme, and the Nyberg-Rueppel signature scheme. In de volgende deelparagrafen zullen in vergelijkbare tabelletjes als in het vorige hoofdstuk al deze systemen beschreven worden.

Op dit moment wordt ook een aantal standaards ontwikkeld die betrekking hebben op elliptische kromme-systemen. Zo is er bijvoorbeeld ANSI X9.62 [4], waarin ECDSA, de elliptische kromme-variant van DSA, wordt beschreven. Een tweede standaard is IEEE P1363 [13], waarin tot nu toe als voorbeelden van signature schemes gebaseerd op elliptische krommen DSA en Nyberg-Rueppel zijn opgenomen. (Het kan zijn dat Nyberg-Rueppel uiteindelijk toch niet in de standaard komt vanwege een patentaanvraag hierop.)

2.2.1 ElGamal met elliptische krommen

ElGamal zou volgens onderstaande tabel getransformeerd kunnen worden tot een systeem gebaseerd op het elliptische kromme discrete logaritme-probleem. Hierbij moet weer in plaats van de boodschap m zelf de hashwaarde $h(m)$ gebruikt worden. Wat betreft de veiligheid kunnen dezelfde opmerkingen gemaakt worden als bij gewone ElGamal (zie paragraaf 1.3.1.2).

	ElGamal handtekeningen met elliptische krommen
Parameters	lichaam $GF(q)$, coëfficiënten a en b, genererend punt P met orde n, zodat het e.k.d.l.-probleem in $\langle P\rangle$ moeilijk is.

Geheime sleutel	random getal $0 < d < n$
Publieke sleutel	$Q = dP$
Handtekening voor m	(r,s), waarbij r gelijk is aan de x-coördinaat van kP met k random, $1 < k < n-1$, en $s = k^{-1}(m - dr) \bmod n$
Verificatie handtekening	controleer of $0 < r < n-1$, $0 < s < n-1$ en $r \equiv v \pmod n$, waarbij v de x-coördinaat van $u_1P - u_2Q$ is met $u_1 = mw \bmod n$, $u_2 = rw \bmod n$ en $w = s^{-1} \bmod n$

2.2.2 ECDSA

De elliptische kromme variant van DSA (zie ook paragraaf 1.3.1.3), genaamd ECDSA, wordt gebruikt in de ANSI-standaard X9.62 [4], die op dit moment ontwikkeld wordt. Ook hier is de analogie tussen DSA en ECDSA duidelijk te zien. In ECDSA wordt weer in plaats van m de hashwaarde $h(m)$ gebruikt, waarbij h de hashfunctie SHA-1 [12] is.

	ECDSA handtekeningen
Parameters	lichaam $GF(q)$, coëfficiënten a en b, genererend punt P met orde n, zodat het e.k.d.l.-probleem in $\langle P \rangle$ moeilijk is.
Geheime sleutel	random getal $0 < d < n$
Publieke sleutel	$Q = dP$
Handtekening voor m	(r,s), waarbij r gelijk is aan de x-coördinaat mod n van kP met k random, $1 < k < n-1$, en $s = (k^{-1}(m + dr)) \bmod n$
Verificatie handtekening	controleer of $0 < r < n$, $0 < s < n$ en $r \equiv v \pmod n$ waarbij v de x-coördinaat van $u_1P + u_2Q$ is met $u_1 = mw \bmod n$, $u_2 = rw \bmod n$ en $w = s^{-1} \bmod n$

2.2.3 Schnorr met elliptische krommen

Een analoge variant van het Schnorr-systeem is het volgende (zie ook paragraaf 1.3.1.4:

	Schnorr handtekeningen met elliptische krommen
Parameters	lichaam $GF(q)$, coëfficiënten a en b, genererend punt P met orde n, zodat het e.k.d.l.-probleem in $\langle P \rangle$ moeilijk is.
Geheime sleutel	random getal $0 < d < n$
Publieke sleutel	$Q = dP$
Handtekening voor m	(r,s), waarbij $r = h(m\|\|e)$ met e gelijk aan de x-coördinaat mod n van kP met k random, $1 < k < n-1$, en $s = (dr + k) \bmod n$
Verificatie handtekening	controleer of $r = h(m\|\|v)$, waarbij v gelijk is aan de x-coördinaat mod n van $sP-rQ$

2.2.4 Nyberg-Rueppel met elliptische krommen

Tot slot nog het Nyberg-Rueppel-systeem met behulp van elliptische krommen. Hier kunnen ook weer dezelfde veiligheidsopmerkingen gemaakt worden als eerder (zie paragraaf 1.3.1.5).

	Nyberg-Rueppel handtekeningen met elliptische krommen
Parameters	lichaam $GF(q)$, coëfficiënten a en b, genererend punt P met orde n, zodat het e.k.d.l.-probleem in $\langle P \rangle$ moeilijk is.
Geheime sleutel	random getal $0 < d < n$

Publieke sleutel	$Q = dP$
Handtekening voor m	(r,s), waarbij $r = me \bmod n$ met e gelijk aan de x-coördinaat mod n van kP, k random en $1 < k < n$, en $s = k + rd \bmod n$
Verificatie handtekening	controleer of $0 < e < p$, $0 \leq s < n$ en $m = rv$ met v gelijk aan de x-coördinaat mod n van $sP - rQ$

Hoofdstuk 3
De cryptografische sterkte van elliptische kromme-systemen

De aanvallen op de elliptische kromme-systemen zijn te verdelen in twee categorieën: aanvallen die het elliptische kromme discrete logaritme-probleem proberen op te lossen en aanvallen die gebruik maken van de eigenschappen van het specifieke handtekeningen-systeem, bijvoorbeeld Nyberg-Rueppel. De aanvallen van de tweede categorie hangen niet af van het feit dat het systeem gebruik maakt van elliptische krommen of niet en zijn al besproken in paragraaf 1.3.1 bij de afzonderlijke beschrijvingen van de systemen. Daarom zal hier ingegaan worden op de eerste categorie aanvallen; deze zijn weer onder te verdelen in de aanvallen die ook op het 'gewone' discrete logaritme-probleem besproken zijn (paragraaf 3.1) en aanvallen die toepasbaar zijn als er speciale krommen worden gebruikt (paragraaf 3.2).

3.1 Discrete logaritme-aanvallen

Aangezien het elliptische kromme discrete logaritme-probleem analoog gedefinieerd is aan het 'gewone' discrete logaritme-probleem, is er eerst geprobeerd de aanvallen die voor deze laatste bekend waren toe te passen op dit 'nieuwe' probleem. Voor de discrete logaritme-systemen zijn er tot nu toe vijf soorten aanvallen bekend (paragraaf 1.3.2):

1. Exhaustive search
2. Baby-step Giant-step
3. Pollard ρ
4. Pohlig-Hellman
5. Index-Calculus

In de volgende paragrafen zal de toepasbaarheid van deze aanvallen op het elliptische kromme discrete logaritme-probleem kort besproken worden. De eerste vier, exponentiële algoritmen kunnen direct vertaald worden naar het elliptische kromme discrete logaritme-probleem; het vijfde algoritme is niet toepasbaar [27]. Als de elliptische kromme bovendien aan bepaalde voorwaarden voldoet, wordt de vierde aanval inefficient; deze voorwaarden zijn vergelijkbaar met de voorwaarden die deze aanval onbruikbaar maakt voor het 'gewone' discrete logaritme-probleem. Daarom is de beste aanval die bekend is voor elliptische krommen de Pollard ρ-methode, terwijl er

voor het 'gewone' discrete logaritme-probleem subexponentiële algoritmen beschikbaar zijn. Dit verklaart dus ook waarom men denkt dat de elliptische kromme-variant van het discrete logaritme-probleem moeilijker is dan het 'gewone' discrete logaritme-probleem.

In dit hoofdstuk wordt de notatie gebruikt, zoals die in het vorige hoofdstuk geïntroduceerd is. Gegeven zijn dus een punt P van orde n en een punt Q, welke gelijk is aan lP. Gezocht wordt het getal l, $1 \leq l \leq n$.

3.1.1 Exhaustive search

De exhaustive search-aanval houdt in achtereenvolgens alle mogelijkheden uitproberen. Deze aanval is altijd toepasbaar, dus ook op de elliptische kromme-variant van het discrete logaritme-probleem: bereken P, $2P$, $3P$, ... , totdat $mP = Q$ is bereikt.

Daar P orde n heeft, ligt de rekentijd van exhaustive search net als bij het 'gewone' discrete logaritme-probleem in de orde van n bewerkingen. De totale rekentijd is daarom exponentieel in n, zodat deze aanval niet bruikbaar is voor grote n.

3.1.2 Baby-step Giant-step

De baby-step giant-step aanval is eenvoudig te transformeren naar elliptische krommen. Dit gaat als volgt in zijn werk.

Neem, precies zoals bij het 'gewone' discrete logaritme-probleem, s gelijk aan het gehele getal $\lceil \sqrt{n} \rceil$ en m, het getal dat gezocht wordt, gelijk aan $m = is + j$, waarbij i en j gehele getallen zijn met $0 \leq j < s$. Het getal j is dus de rest na deling van m door s.

Geheel analoog geldt dan dat $Q - i\,(sP) = jP$. Construeer een tabel bestaande uit de paren (j, jP), voor $0 \leq j < s$, gesorteerd op bijvoorbeeld de x-coördinaat van jP, leidt de berekening van achtereenvolgens Q, $Q - sP$, $(Q - sP) - sP = Q\ 2(sP)$, ..., tot het vinden van i en j en dus de gevraagde m.

Bij deze aanval zijn weer orde $\sqrt{n}$ bewerkingen nodig en worden orde $\sqrt{n}$ elementen in het geheugen opgeslagen. Dus ook hier is de rekentijd exponentieel.

3.1.3 Pollard ρ

De Pollard ρ-methode heeft ook een variant voor elliptische krommen en neemt ongeveer $\sqrt{(\pi n/2)}$ bewerkingen (optellingen op de kromme).

Er worden op een bepaalde manier drie rijen $(X_i)_{i \geq 0}$, $(a_i)_{i \geq 0}$ en $(b_i)_{i \geq 0}$ gegenereerd die de volgende eigenschappen hebben:

$$X_i = a_iP + b_iQ.$$

Deze generatie van rijen gaat weer net zolang door totdat er een element X_j gevonden is dat gelijk is aan een eerder element X_i. Zodra zo'n element gevonden is, is het discrete logaritme-probleem opgelost. Dan geldt namelijk:

$$X_i = a_iP + b_iQ = a_jP + b_jQ = X_j$$

oftewel, als alle termen met Q naar de ene kant worden gebracht en alle termen met P naar de andere:

$$(b_i - b_j)Q = (a_j - a_i)P.$$

Er geldt dat $Q=mP$ en $nP=O$, dus hieruit volgt:

$$(b_i - b_j)\, m = a_j - a_i \pmod{n}.$$

Aldus wordt de volgende expressie voor de elliptische kromme discrete logaritme van Q naar P verkregen:

$$m = (a_j - a_i)/(b_i - b_j) \pmod{n}.$$

Paul Van Oorschot and Michael Wiener [33] hebben een manier gevonden om het Pollard ρ algoritme te versnellen door m processoren parallel te laten rekenen. De rekentijd voor dit parallelle algoritme is van de orde $(\sqrt{(\pi n/2)})/m$.

3.1.4 Pohlig-Hellman

De aanval van Pohlig-Hellman maakt gebruik van de factorisatie van de orde n van het punt P. Net als in het geval van de 'gewone' discrete logaritme, heeft dit algoritme in de orde van

$$\sum_{i=1}^{t} \lambda_i (\log n + \sqrt{p_i} \log p_i)$$

bewerkingen nodig, als als de factorisatie van n gelijk is aan

$$n = \prod_{i=1}^{t} p_i \lambda_i .$$

De werking van het algoritme komt weer neer op het berekenen van $(m \bmod p_i\lambda_i)$ voor alle i van 1 tot t. Vervolgens kan met de Chinese Reststelling $(m \bmod n)$ berekend worden (zie appendix B.1).

Ook hier geldt weer dat het algoritme alleen efficiënt kan zijn, gezien de expressie voor de rekentijd, als n het produkt is van kleine priemgetallen. Daarom kan het beste een elliptische kromme genomen worden, waarvan de orde (=het aantal punten op de kromme) een groot priemgetal is of het produkt van een klein geheel getal en een groot priemgetal. Vervolgens moet het punt P zo gekozen worden, dat het als orde dit grote priemgetal heeft. Dan is deze aanval niet meer toepasbaar.

3.1.5 Index-Calculus

Bij het 'gewone' discrete logaritme-probleem zijn de Index-Calculus-algoritmen het efficiëntst (subexponentieel). Deze algoritmen zijn echter niet in elke groep toepasbaar. Ook voor het discrete logaritme-probleem op de punten van een elliptische kromme is er geen Index-Calculus-algoritme [27].

3.2 Aanvallen voor speciale krommen

In deze paragraaf worden drie aanvallen besproken die efficiënt zijn voor bepaalde krommen. De drie soorten krommen die respectievelijk bij deze aanvallen horen, zijn singuliere krommen, supersinguliere krommen en 'anomalous curves'. Als deze krommen vermeden worden bij het gebruik in cryptosystemen, zijn deze aanvallen niet toepasbaar.

3.2.1 Singuliere krommen

Voor de definitie van singuliere krommen wordt de affine Weierstraß-vergelijking uit paragraaf 2.1 eerst op een andere manier geschreven. Vervang in deze vergelijking x

door X/Z, y door Y/Z en vermenigvuldig het geheel met Z^3. Dan wordt de zogenaamde **homogene, projectieve Weierstrass-vergelijking** verkregen:

$$Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0$$

(voor het vervolg zijn alle termen naar één kant gehaald). Noem de expressie aan de linkerkant van het gelijkteken $F(X,Y,Z)$.

De kromme voorgesteld door deze vergelijking is **niet-singulier in P** (waarbij P op de kromme ligt en dus aan de Weierstraß-vergelijking voldoet), als ten minste één van de partiële afgeleiden $\partial F/\partial X$, $\partial F/\partial Y$, $\partial F/\partial Z$ ongelijk aan 0 is in het punt P. Als al deze partiële afgeleiden gelijk aan 0 zijn in P, wordt P een **singulier punt** genoemd. Verder heet de kromme **niet-singulier**, wanneer alle punten op de kromme niet-singulier zijn. Anders wordt de kromme singulier genoemd.

Elke singuliere kromme heeft slechts één singulier punt. De verzameling van alle niet-singuliere punten op de kromme, inclusief het punt in het oneindige O, wordt het **niet-singuliere gedeelte** van de kromme genoemd; dit zijn dus alle punten op de kromme behalve het ene singuliere punt. Op dit niet-singuliere gedeelte kan op dezelfde manier als eerder een optelling gedefinieerd worden; met deze optelling vormt het niet-singuliere gedeelte van de kromme weer een groep. Er kan dus ook een elliptische kromme discrete logaritme-probleem op deze verzameling punten gedefinieerd worden. Dit discrete logaritme-probleem is echter niet moeilijk.

Het elliptische kromme discrete logaritme-probleem op het niet-singuliere gedeelte van een singuliere kromme kan namelijk getransformeerd worden naar het discrete logaritme probleem in andere groepen, waar het discrete logaritme-probleem efficiënt is op te lossen. Zo'n transformatie van een probleem van de ene groep naar een andere wordt een **reductie** genoemd. Voor een aanval op het discrete logaritme-probleem is het interessant, als zo'n reductie naar een groep waarin het probleem eenvoudiger is, in polynomiale tijd kan worden bewerkstelligd.

Het discrete logaritme-probleem op de niet-singuliere punten van een singuliere kromme is in polynomiale tijd reduceerbaar tot het 'gewone' discrete logaritme-probleem (gedefinieerd met de vermenigvuldiging) in het eindige lichaam met q elementen of q^2 elementen of tot het discrete logaritme-probleem gedefinieerd met de optelling in het lichaam met q elementen (zie bijvoorbeeld in Menezes' boek [24]). In het eerste geval is er dan een subexponentieel algoritme die het discrete logaritme-probleem oplost en in het tweede geval is er zelfs een polynomiaal algoritme.

Er kunnen dus beter geen singuliere krommen gebruikt worden voor cryptosystemen gebaseerd op het discrete logaritme-probleem. Een singuliere kromme kan eenvoudig herkend worden aan de hand van de bijbehorende Weierstraß-vergelijking (zie ook appendix C.2).

3.2.2 MOV-reductie

"MOV" staat voor Menezes, Okamoto en Vanstone, degenen die deze aanval publiceerden [25]. Zij lieten zien hoe het discrete logaritme-probleem op een elliptische kromme gereduceerd kan worden tot het 'gewone' discrete logaritme-probleem in $GF(q^k)$, het eindige lichaam van q^k elementen, voor een zekere k.

Deze reductie bestaat voor elke kromme, maar kost in het algemeen exponentiële tijd (in ln q), omdat k in het algemeen exponentieel groot is. Alleen voor een speciale klasse van krommen, de zogenaamde **supersinguliere** krommen, kan de reductie in polynomiale tijd volbracht worden. (Voor supersinguliere krommen is bekend dat k alleen de waarden $1, 2, 3, 4$ en 6 aanneemt.) Dit resulteert dan in een subexponentiële aanval voor het discrete logaritme-probleem op deze krommen (omdat er een subexponentieel algoritme bestaat voor het 'gewone' discrete logaritme-probleem).

Het is dus aan te raden geen supersinguliere krommen of andere krommen waarvoor k klein is voor cryptosystemen te gebruiken. Deze krommen zijn met een algoritme eenvoudig te onderscheiden van andere krommen (zie ook paragraaf 5.2).

3.2.3 Smart-Satoh-Araki

Onlangs hebben Nigel Smart [45] en de Japanners Takakazu Satoh en Kiyomichi Araki [38] een derde klasse van zwakke krommen gevonden, de zogenaamde **'anomalous curves'**. Dit zijn krommen over $GF(p)$, het eindige lichaam met p elementen, die precies p punten hebben. Deze krommen zijn dus ook eenvoudig te herkennen aan de hand van het aantal punten op de kromme.

Smart, Satoh en Araki laten zien hoe het discrete logaritme probleem op deze krommen in polynomiale tijd opgelost kan worden. Zij construeren een bepaalde afbeelding λ van de punten op de kromme naar het eindige lichaam van p elementen, zodat uiteindelijk de logaritme m door een eenvoudige deling in $GF(p)$ verkregen kan worden ($m=\lambda(Q)/\lambda(P)$).

Ditzelfde resultaat was al eerder gevonden door Semaev [44], maar op een meer algebraïsch-meetkundige manier. Verder is dit resultaat gegeneraliseerd door Rück [37].

3.3 Conclusie

Als een aantal voorzorgsmaatregelen worden genomen, is het beste algoritme voor het discrete logaritme-probleem op een elliptische kromme de exponentiële Pollard ρ-methode. De voorzorgsmaatregelen houden ten eerste in dat de gekozen kromme niet tot één van de volgende soorten krommen behoort:

1. Singuliere krommen
2. Supersinguliere krommen
3. Anomalous curves

Bovendien is het van belang dat het aantal punten op de kromme een groot priemgetal als deler heeft; het genererende punt P moet dan zo genomen worden dat het als orde dit grote priemgetal heeft.

Hoofdstuk 4
Veilig te achten sleutellengtes voor de eerstkomende 10 jaar

Zoals uit het vorige hoofdstuk blijkt is het grote voordeel van elliptische kromme-systemen de grote cryptografische sterkte in relatie tot de sleutellengte. Daar het beste algoritme exponentieel is, kan er een kleinere sleutellengte gebruikt worden dan bij gewone discrete logaritme-systemen, waar het beste algoritme subexponentieel is. Met het sneller worden van de computers moet ook de sleutellengte vergroot worden. In dit opzicht zijn de elliptische kromme-systemen dan ook aantrekkelijker: de sleutels zullen minder snel hoeven te stijgen dan bij discrete logaritme-systemen.

In dit hoofdstuk is het laatste nieuws wat betreft elliptische kromme discrete logaritme-systemen geïnventariseerd. Hieruit kan afgeleid worden welke sleutellengtes als veilig geacht worden voor de korte en lange termijn. In de eerste paragraaf wordt een eerste aanzet gegeven voor ondergrenzen aan de lengte van n in elliptische kromme-systemen. In de tweede paragraaf worden een korte analyse gedaan naar het verband tussen de sleutellengte en rekentijd voor software en hardware attacks. Vervolgens wordt ingegaan op de Certicom-challenge, waaruit op te maken valt hoe ver men al is in het breken van elliptische kromme-cryptosystemen. In de laatste paragraaf worden dan vervolgens de conclusies getrokken over veilige sleutellengtes.

4.1 Sleutellengtes

In [6] hebben Blaze e.a. een uitspraak gedaan over de minimum sleutellengtes die nodig zijn voor veilige (i.e. bestand tegen aanvallen van goedgefinancieerde bedrijven en geheime diensten van overheden), symmetrische encryptiealgoritmen zoals DES en IDEA. Zij kwamen tot de conclusie dat tegenwoordig ten minste 75 bits-lange sleutels nodig zijn en dat voor de komende twintig jaar, gezien de verwachte ontwikkelingen in de computerwereld (de wet van Moore: de computer worden elk anderhalf jaar twee keer zo snel), de sleutels ten minste 90 bits lang zouden moeten zijn.

Een exhaustive search op een symmetrisch algoritme neemt ongeveer het zelfde aantal vergelijkbare bewerkingen als het Pollard ρ-algoritme voor een elliptische kromme met een n, die een twee keer zo lange lengte heeft als de symmetrische sleutel. Als het bovenstaande dan vertaald wordt naar elliptische kromme-cryptosystemen, resulteert dit

in de eis dat n ten minste 150 bits lang moet zijn voor korte termijn-veiligheid en ten minste 180 bits voor veiligheid op langere termijn.

Deze afleiding kan verder gerechtvaardigd worden door de volgende eigenschappen. Exhaustive search bij een symmetrisch algoritme en het Pollard ρ-algoritme voor elliptische krommen zijn verder vergelijkbaar, omdat ze beiden geparallelliseerd kunnen worden met een lineaire versnelling. Bovendien heeft het vinden van de geheime sleutel in beide gevallen tot gevolg dat er slechts één sleutel gevonden is en dat dit geen gevolgen heeft voor andere sleutels. Een aspect waarin deze algoritmen verschillen, maar wat de schatting alleen maar betrouwbaarder maakt, is dat de basisbewerking bij elliptische krommen (een optelling van twee punten) in het algemeen meer tijd kost dan een basisbewerking in een symmetrisch algoritme (één blokvercijfering).

4.2 Software en hardware aanvallen

In deze paragraaf zal gekeken worden naar software en hardware implementaties van het beste algoritme dat voor het elliptische kromme discrete logaritme-probleem bekend is: het Pollard ρ-algoritme. Met name wordt hier dan gekeken naar het verband tussen de lengte van n en de computersnelheden.

4.2.1 Software aanvallen

In het vorige hoofdstuk is uiteengezet, dat de beste aanval voor het elliptische discrete logaritme-probleem de Pollard ρ-methode is, welke ongeveer $\sqrt{\pi n/2}$ stappen neemt. Stel dat de grootte van n gelijk is aan m bits. Dan kan n geschreven worden als $n=2^m$. Als de rekensnelheid van de computers elk anderhalf jaar twee keer zo snel zou worden, met hoeveel bits b zou n dan vergroot moeten worden? Hiervoor moet b uit de volgende vergelijking worden opgelost:

$$\frac{1}{2}\sqrt{\pi 2^{m+b}/2} = \sqrt{\pi 2^m/2}\,.$$

Hieruit volgt dat b gelijk is aan 2. Er hoeven dus slechts twee bits extra genomen te worden om een twee keer zo snelle rekenkracht te niet te doen.

Een andere manier om de sterkte van elliptische kromme systemen te beschouwen is met gebruikmaking van MIPS-aannames. MIPS staat voor "Million Instructions Per Second". Een veilige aanname is dat een 1 MIPS-machine $4 \cdot 10^4$ elliptische optellingen per seconde kan uitvoeren. (Dat dit een veilige aanname is, kan gezien worden aan het feit dat er bijvoorbeeld een 'application specific' hardware implementatie is voor elliptische kromme-berekeningen over het lichaam $GF(2^{155})$ met een 40 MHz clock-rate die 40000 elliptische optellingen per seconde kan uitvoeren [2] en een software implementatie [43] op een 25 MIPS-machine die 2000 elliptische optellingen per seconde uitvoert.) Een kleine berekening laat dan zien dat zo'n 1 MIPS-machine ongeveer 2^{40} elliptische optellingen per jaar kan uitvoeren; deze rekenkracht per jaar voor een 1 MIPS-machine wordt MIPS-jaar genoemd. Verder wordt als benchmark 10^{12} MIPS-jaren behoorlijk veilig genoemd, omdat dit gelijk is aan de computerkracht van de hele planeet voor een behoorlijke tijd te laten werken.

In de volgende tabel wordt dan samengevat hoeveel rekenkracht in MIPS-jaren er nodig is om de discrete logaritme op een elliptische kromme te berekenen met behulp van de Pollard ρ-methode.

Lichaamsgrootte $\|K\|$ ($\log_2 \|K\|$)	Grootte van n ($\log_2 n$)	Rekentijd ($\sqrt{\pi n/2}$)	MIPS-jaren
163	160	2^{80}	$9{,}6 \cdot 10^{11}$
191	186	2^{93}	$7{,}9 \cdot 10^{15}$
239	234	2^{117}	$1{,}6 \cdot 10^{23}$
359	354	2^{177}	$1{,}5 \cdot 10^{41}$
431	426	2^{213}	$1{,}0 \cdot 10^{52}$

4.2.2 Hardware aanvallen

In paragraaf 3.1.3 is al aan de orde gekomen, dat Van Oorschot en Wiener een manier hebben gevonden om het Pollard ρ-algoritme met behulp van een parallelle zoekmethode uit te voeren. In [33] beschrijven zij hoe dit gedaan zou kunnen worden met behulp van special-purpose-hardware, als er een bedrag van $10 miljoen Amerikaanse dollars ter beschikking staat. Zij schatten dat voor n ter grootte van 36 decimale getallen oftewel 120 bits, een machine met 325000 parallelle processoren gebouwd kan worden voor bovenstaand bedrag; deze machine heeft dan 35 dagen nodig om één elliptische discrete logaritme te berekenen.

4.3 De Certicom challenge

Certicom, een Canadees bedrijf dat cryptosystemen gebaseerd op elliptische krommen op de markt brengt, heeft een 'challenge' uitgebracht om het onderzoek in dit gebied te vergroten. Voor deze challenge zijn een aantal krommen gegenereerd met bijbehorende sleutelparen; de challenge is om de geheime sleutels te vinden.

Er zijn drie moeilijkheidsgraden gedefinieerd: exercises, level I en level II. De moeilijkheid hangt af van de grootte van de orde n van het genererende punt P, welke in bits wordt gegeven. Verder zijn de challenges gesplitst in opgaven over $GF(p)$ en over $GF(2^m)$; ook is er onderscheid gemaakt in willekeurig gekozen krommen en zogenaamde Koblitz-krommen, waarop efficiënt gerekend kan worden (zie ook paragraaf 5.2).

Van de exercises wordt gedacht dat deze redelijk eenvoudig zijn, variërend van op te lossen in een paar uur (79 bits) tot in een paar weken (97 bits) (gebruikmakend van netwerken van computers). De kleine level I-opgaven van 109 bits zouden nog oplosbaar moeten zijn met een zeer groot netwerk van computers, maar de level I-opgaven van 131 bits worden onbreekbaar geacht met de huidige realiseerbare hardware en software attacks. De level II-opgaven, variërend van 163 bits tot 359 bits, zijn het moeilijkst; aangenomen wordt dat deze de komende twintig jaar niet gebroken zullen worden (zie ook paragraaf 4.1).

Op dit moment zijn alle exercises van 79 en 89 bits gebroken. Ook is pas geleden de exercise van 97 bits over $GF(p)$ opgelost. De aanvallen gebruikten de parallelle vorm

van het Pollard ρ algoritme [33]; een detail dat opvalt is dat de oplossingen van de exercises in *GF(p)* eerder gevonden zijn dan die van de exercises in $GF(2^m)$ met de zelfde lichaamsgrootte.

Om een indicatie te geven over de omvang van zo'n aanval: bij de recentste aanval op 97 bits over *GF(p)* deden 588 mensen en 1288 machines mee. De berekening werd volbracht in 53 dagen en kostte ongeveer $200 \cdot 10^{12}$ iteraties.

4.4 Conclusie

Gezien de huidige ontwikkelingen is voor n een sleutellengte van 160 bits voldoende voor korte termijn-veiligheid. Met deze lengte zijn de huidige software en hardware aanvallen nog niet toepasbaar. Als de computers elk anderhalf jaar twee keer zo snel worden, moet de lengte van n vergroot worden met 2 bits om dezelfde veiligheid te bereiken. Daarom zal een schatting van de sleutellengte voor langere termijn, bijvoorbeeld de komende tien jaar, richting 180 bits moeten gaan. Dit alles geldt natuurlijk onder het voorbehoud dat er zich geen nieuwe aanvallen voordoen.

Hoofdstuk 5
Parameters en kwaliteitscriteria

In dit hoofdstuk worden de parameters van een elliptische kromme systeem besproken. Deze parameters zijn zogenaamde systeemparameters: ze kunnen gelijk zijn voor een grote groep gebruikers en hoeven niet vaak veranderd te worden. (Een andere optie zou zijn om per gebruiker (per sleutelpaar) andere systeemparameters te kiezen.) De veiligheid van het elliptische kromme-systeem hangt niet af van het geheimhouden van deze parameters, dus ze mogen publiek gemaakt worden. De veiligheid is wel 'af te lezen' aan de parameter n: als n groter wordt, vergroot de veiligheid van het systeem ook.

In het algemeen is het genereren van de systeemparameters niet eenvoudig en kost redelijk veel tijd (een paar uur). Dit is eens te meer een reden om de systeemparameters voor een grote groep gebruikers en voor gebruik voor langere tijd te definiëren.

Achtereenvolgens zullen aan bod komen:
1. de keuze van het onderliggende lichaam en de representatie van zijn elementen;
2. de keuze van de kromme;
3. de keuze van het genererend punt.

In de laatste paragraaf wordt de parametergeneratie kort samengevat.

5.1 De keuze van het lichaam en de representatie van zijn elementen

Voor het lichaam K waarover de kromme gedefinieerd wordt, worden meestal de eindige lichamen met 2^m of p elementen genomen, omdat voor deze lichamen de beste implementaties voor berekeningen hierin bestaan. Maar theoretisch zou elk lichaam gekozen kunnen worden. Wat betreft de veiligheid is er geen reden om aan te nemen dat een van deze twee lichamen minder veilig zou zijn dan de andere. Er zijn echter wel andere criteria die de doorslag kunnen geven om het ene lichaam boven het andere te prefereren.

In de volgende twee deelparagrafen zal elk van deze twee lichamen besproken worden, waarbij er aandacht besteed zal worden aan de mogelijke representaties van de elementen, de te gebruiken rekenkundige bewerkingen en eventuele eisen aan het betreffende lichaam.

5.1.1 Het lichaam *GF(p)*

GF(p) is het eindige lichaam met p elementen, waarbij p een priemgetal is; dit lichaam is equivalent aan $\mathbf{Z}_p$. De elementen kunnen dus gerepresenteerd worden als de getallen $\{0, 1, ..., p-1\}$. Het optellen en vermenigvuldigen in dit lichaam gebeurt mod p. Het nulelement is het getal 0 en het eenheidselement is het getal 1.

Uit Hoofdstuk 4 volgt dat voor korte termijn-veiligheid de grootte $\log_2 p$ van p ten minste 163 bits moet zijn en voor lange termijn-veiligheid ongeveer 191 bits.

5.1.2 Het lichaam *GF(2^m)*

$GF(2^m)$ is het lichaam met 2^m elementen. De elementen kunnen op verschillende manieren gerepresenteerd worden, bijvoorbeeld met behulp van een polynoombasis of een normaalbasis. De keuze van de basis heeft geen effect op de veiligheid van het systeem maar kan wel invloed hebben op de efficiëntie.

Ook hier geldt dat $\log_2 2^m = m$ ten minste 163 moet zijn voor korte termijn-veiligheid en ongeveer 191 bits voor lange termijn-veiligheid (zie ook Hoofdstuk 4). Verder zijn in principe geen beperkingen op m, al vermoeden experts als Schnorr en Frey dat het minder veilig is om m ongelijk aan een priemgetal te nemen; zij hebben hier geen bewijs voor.

5.1.2.1 Polynoombasis

De verzameling polynomen $\{1, x, ..., x^{m-2}, x^{m-1}\}$ vormt een polynoombasis van $GF(2^m)$ over $GF(2)$, i.e. elk element van $GF(2^m)$ kan geschreven worden als een lineaire combinatie van elementen uit deze verzameling $\alpha_0+\alpha_1 x+ ...+\alpha_{m-2}x^{m-2}+\alpha_{m-1}x^{m-1}$, waarbij de coëfficiënten α_i, $i=0,1,...m-1$, gelijk zijn aan 0 of 1 (elementen van $GF(2)$). Deze basis wordt ook wel **standaardbasis** genoemd. Een element a uit $GF(2^m)$ dat correspondeert met het polynoom

$$a(x) = a_0 + a_1 x + ... + a_{m-2}\, x^{m-2} + a_{m-1}\, x^{m-1}$$

kan dan gerepresenteerd worden door de bitstring $(a_0\, a_1 ...\, a_{m-2} a_{m-1})$ ter lengte m.

Het nulelement is 0, gerepresenteerd door de bitstring $(00...00)$, en het eenheidselement is 1, welke correspondeert met de bitstring $(100...00)$. De optelling van twee elementen komt neer op het bitgewijs optellen mod 2 van de bitstrings die deze elementen representeren (ook wel "XOR" = "exclusive or" genoemd).

De vermenigvuldiging in dit lichaam gebeurt modulo het zogenaamde **reductiepolynoom**, een onontbindbaar (ook wel aangeduid met de term 'irreducibel') polynoom $f(x)$ van graad m met coëfficiënten uit $GF(2)$. De uitkomst van de vermenigvuldiging van twee elementen a en b, welke corresponderen met de polynomen $a(x)$ en $b(x)$, is dan het restpolynoom $r(x)$ (gerepresenteerd door de bitstring horend bij $r(x)$) dat verkregen wordt na deling van het produkt $a(x)b(x)$ door $f(x)$.

Het kwadrateren in dit lichaam is ongeveer gelijk aan de tijd van een modulo-reductie en voor het berekenen van een inverse is er een efficiënt algoritme te vinden in [43].

Om de efficiëntie van de berekeningen in dit lichaam te vergroten, kan voor het reductiepolynoom het beste een polynoom gekozen worden met zo min mogelijk

termen. Het kleinste aantal termen zodat een polynoom nog irreducibel kan zijn, is drie; polynomen met drie termen worden **trinomials** genoemd en zien er dus als volgt uit:

$$x^m + x^k + 1, \quad 1 \le k \le m-1.$$

Een gewenste eigenschap van deze trinomials, weer in verband met de efficiëntie maar ook voor een maximale kans op interoperabiliteit, is bovendien dat k zo klein mogelijk is. Een lijst van irreducibele trinomials voor m, $160 \le m \le 2000$, en zo klein mogelijke k wordt gegeven in de ANSI-standaard voor ECDSA [4].

Helaas bestaan niet voor elke m dergelijke irreducibele trinomials. In dat geval kan dan een vijfterm gebruikt worden genaamd **pentanomial**:

$$x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1, \quad 1 \le k_1 < k_2 < k_3 \le m-1.$$

Ook hier geldt weer dat vanuit efficiëntieoogpunt de k_i's, $i=1,2,3$, zo klein mogelijk moeten zijn. In de ANSI-standaard [4] wordt een lijst irreducibele pentanomials voor m, $160 \le m \le 2000$, gegeven, waarbij eerst k_1, dan k_2 en als laatste k_3 zo klein mogelijk zijn.

Opmerking: Als m geen priemgetal is en dus het produkt van ten minste twee gehele getallen, bijvoorbeeld $m=de$, dan kan er ook een basis $\{1, x, x^2, \ldots, x^{d-1}\}$ van $GF(d)$ over $GF(e)$ genomen worden voor de representatie van $GF(2^m)$. De elementen van $GF(2^m)$ zijn dan lineaire combinaties van de basiselementen, waarbij de coëfficiënten in $GF(e)$ liggen. Met behulp van deze methode kan in software efficiënt gerekend worden [49].

5.1.2.2 Normaalbasis

Een normaalbasis voor $GF(2^m)$ over $GF(2)$ is een basis van de vorm

$$\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\},$$

waarbij β een element van $GF(2^m)$ is. Een element a uit $GF(2^m)$ dat gelijk is aan

$$a = a_0\beta + a_1\beta^2 + \ldots + a_{m-2}\beta^{2^{m-2}} + a_{m-1}\beta^{2^{m-1}} = \sum_{i=0}^{m-1} a_i \beta^{2^i}$$

kan dan gerepresenteerd worden door de bitstring $(a_0a_1\ldots a_{m-2}a_{m-1})$ ter lengte m.

Het nulelement wordt gerepresenteerd door de bitstring $(00\ldots00)$, terwijl het eenheidselement gelijk is aan de bitstring $(11\ldots11)$. De optelling gaat hetzelfde als bij de polynoombasis: de bitstrings worden bitgewijs modulo 2 opgeteld (XOR).

De vermenigvuldiging van twee termen $a = (a_0a_1\ldots a_{m-2}a_{m-1})$ en $b = (b_0b_1\ldots b_{m-2}b_{m-1})$ heeft iets meer aandacht nodig. Stel het produkt ab is gelijk aan $c = (c_0c_1\ldots c_{m-2}c_{m-1})$. Dan geldt dus:

$$c = \sum_{k=0}^{m-1} c_k \beta^{2^k} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i} \beta^{2^j}.$$

Het produkt $\beta^{2^i}\beta^{2^j}$ is een element van $GF(2^m)$ en kan daarom ook geschreven worden als een lineare combinatie van de basiselementen:

$$\beta^{2^i}\beta^{2^j} = \sum_{k=0}^{m-1} \lambda_{ij}^{(k)} \beta^{2^k}, \quad \lambda_{ij}^{(k)} \in \{0,1\}.$$

Er kan verder nog afgeleid worden dat alle coëfficiënten $\lambda_{ij}^{(k)}$ voor $k > 0$ geschreven kunnen worden in termen van de coëfficiënten $\lambda_{ij}^{(0)}$, en wel op de volgende manier:

$$\lambda_{ij}^{(k)} = \lambda_{i-k,j-k}^{(0)} \text{ voor alle } 0 \le i, j, l \le m-1.$$

Uit dit alles volgt dan dat de coëfficiënten c_k gelijk zijn aan

$$c_k = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \lambda_{i-k,j-k}^{(0)} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{ij}^{(0)} ,$$

waarbij de laatste expressie verkregen wordt door hernummering van de indices.

De coëfficiënten $\lambda_{ij}^{(0)}$ kunnen van te voren berekend worden en opgeslagen in een matrix. Om de vermenigvuldiging te versnellen kan er dan gezocht worden naar een normaalbasis die zo min mogelijk enen in deze matrix heeft. Het minimum aantal enen is gelijk aan $2m-1$; een normaalbasis met dit aantal enen in de matrix heet een **optimale normaalbasis**.

Een voordeel van normaalbases is dat het kwadrateren zeer efficiënt volbracht kan worden. Het is namelijk slechts een cyclische verschuiving van de bitrepresentatie (oftewel van de coëfficiënten van het polynoom). Voor het berekenen van een inverse is er niet zo'n snel algoritme als bij de standaardbasis.

Er bestaat altijd een normaalbasis, maar een optimale normaalbasis is niet altijd aanwezig. In de ANSI-standaard [4] staat een tabel die alle waarden van m, $160 \leq m \leq 2000$, opsomt waarvoor er een optimale normaalbasis is voor $GF(2^m)$.

5.2 De keuze van de kromme

In paragraaf 2.1 is al verteld dat een kromme beschreven kan worden door een Weierstraß-vergelijking. Afhankelijk van welk lichaam er gebruikt wordt, kan deze vergelijking vereenvoudigd worden. Voor het lichaam $GF(p)$ is deze vereenvoudigde vergelijking gelijk aan:

$$y^2 = x^3 + ax + b ,$$

waarbij a en b in $GF(p)$ zitten. Voor het lichaam $GF(2^m)$ zijn er twee vergelijkingen mogelijk, maar alle niet-supersinguliere krommen (zie ook paragraaf 3.2.2) hebben de volgende vereenvoudigde Weierstraß-vergelijking:

$$y^2 + xy = x^3 + ax^2 + b .$$

Er wordt aangenomen dat het het beste is om een kromme zo willekeurig mogelijk te kiezen. Als er namelijk een speciaal soort kromme gekozen wordt, kan een aanvaller zich daarop richten en wellicht gebruik maken van de speciale eigenschappen die deze krommen bezitten. Dit blijkt wel uit paragraaf 3.2, waar voor specifieke krommen aanvallen beschreven zijn.

In paragraaf 3.2.1 is aangetoond dat de kromme niet-singulier moet zijn om het discrete logaritme-probleem op de elliptische kromme niet te vereenvoudigen. Om niet-singulariteit te garanderen moeten de coëfficiënten op de kromme aan de volgende voorwaarde voldoen:

1. $\begin{cases} 4a^3 + 27b^2 \neq 0 \text{ als het lichaam } GF(p) \text{ is en de kromme } y^2 = x^3 + ax + b \\ b \neq 0, \text{ als het lichaam } GF(2^m) \text{ is en de kromme } y^2 + xy = x^3 + ax^2 + b \end{cases}$ (zie paragraaf 3.2.1 en appendix C.2).

Om de MOV-reductie uit paragraaf 3.2.2 inefficiënt te maken, moet er gecontroleerd worden of de k die hierbij hoort groot genoeg is; wanneer dit het geval is, wordt er gezegd dat de kromme aan de 'MOV-condition' voldoet. De ondergrens die als 'groot genoeg' wordt beschouwd, wordt in ANSI [4] en IEEE [13] gelijk genomen aan 20. Dit betekent dat, als het onderliggende lichaam bijvoorbeeld $GF(2^{191})$ is, alle elliptische krommen waarop het elliptische kromme probleem gereduceerd kan worden tot het gewone discrete logaritme-probleem in lichamen ter grootte van ongeveer 2^{3800}, afgekeurd worden. Volgens de ANSI-standaard is dit een 'conservatieve' keuze, die voldoende is om tegen de MOV-reductie bestand te zijn.

Wanneer de kromme gedefinieerd is over $GF(q)$ en de ondergrens B wordt genoemd, voldoet deze kromme aan de MOV-condition als het volgende geldt:

2. $q^i \bmod n \neq 1$ voor alle $i = 1, \ldots, B$.

Uit Hoofdstuk 3 volgt ook, dat het belangrijk is om te weten wat het aantal punten op de kromme is, om te kunnen checken of het geen zwakke kromme is met betrekking tot het elliptische kromme logaritme-probleem. Om er voor te zorgen dat de exponentiële Pollard ρ-methode de beste aanval is met de huidige kennis moet het aantal punten $|E|$ op de kromme E daarom ook aan de volgende voorwaarden voldoen:

3. $|E| = kl$, waarbij k een klein, positief, geheel getal is en l een groot priemgetal (zie paragraaf 3.1.4); k wordt ook wel de **cofactor** genoemd;
4. $|E| \neq p$, als het gebruikte lichaam gelijk aan $GF(p)$ is (zie paragraaf 3.2.3);

De vraag is dan hoe het aantal punten op een kromme bepaald kan worden. Als de Weierstraß-vergelijking bekend is, kan met behulp van het algoritme van Schoof het aantal punten op de kromme bepaald worden [41][42]. Dit algoritme maakt onder andere gebruik van de zogenaamde **karakteristieke vergelijking** (zie appendix C.3). Dit algoritme is in zijn oorspronkelijke vorm echter niet efficiënt genoeg voor de berekening van het aantal punten van krommen over grote lichamen. Er zijn daarom een aantal versnellingen en uitbreidingen op dit algoritme bedacht, onder andere door Lercier en Morain [23].

Het kiezen van een Weierstraß-vergelijking kan op twee verschillende manieren gebeuren: er kunnen speciale krommen gekozen worden (zoals de zogenaamde Koblitz-krommen [19][46] die tot een efficiënte implementatie kunnen leiden), waar het aantal punten zonder gebruikmaking van het algoritme van Schoof eenvoudig te bepalen is, of de coëfficiënten a en b van de vergelijking kunnen random gekozen worden, waarna er meestal niets anders op zit dan het algoritme van Schoof toe te passen om het aantal punten te bepalen.

In het geval dat a en b random worden gekozen, kan daarbij ook een hashfunctie gebruikt worden, zodat verifeerbaar is dat a en b random gekozen zijn. In de ANSI-standaard paragraaf F3.3 [4] wordt beschreven hoe dit in zijn werk gaat; daar wordt als hashfunctie SHA-1 gekozen. Als het lichaam $GF(2^m)$ is komt het erop neer dat er een willekeurige input SEED ter lengte van 160 bits voor de hashfunctie wordt gekozen waarmee b gegenereerd wordt; vervolgens wordt a willekeurig gekozen. Als het

gebruikte lichaam $GF(p)$ is, wordt met de SEED een zeker getal r gegenereerd, waarna a en b zo gekozen worden dat geldt

$$rb^2 \equiv a^3 \pmod{p}.$$

Deze methode van het verifeerbaar random genereren van parameters wordt gebruikt in gewone DSA, om zwakke keuzes zo veel mogelijk te voorkomen. In ECDSA zijn er geen zwakke keuzes bekend naast de speciale krommen die al genoemd zijn in Hoofdstuk 3. Daarom is er voor ECDSA geen veiligheidseis die zegt dat deze methode gebruikt moet worden bij de parametergeneratie, zodat ook speciale krommen kunnen worden gekozen waarop efficiëntere implementaties mogelijk zijn.

De kromme kan ook op een andere manier gegenereerd worden. In plaats van een Weierstraß-vergelijking te kiezen en daarbij het aantal punten te berekenen, kan er ook eerst een geschikt aantal punten gekozen worden (zodat aan de eerder genoemde voorwaarden wordt voldaan) en daarbij een Weierstraß-vergelijking te bepalen. Deze methode wordt ook wel de **complexe vermenigvuldiging-methode** genoemd [47][21], omdat hierbij de eigenschap complexe vermenigvuldiging wordt gebruikt. Ook deze methode is niet eenvoudig, maar in de IEEE P1363-standaard [13] wordt een poging gedaan om de benodigde kennis voor het opstellen van een complexe vermenigvuldiging-algoritme te verschaffen.

Een derde mogelijkheid is om het aantal punten eerst over een kleiner deellichaam van het eigenlijke lichaam K te bepalen en daaruit vervolgens het aantal punten over K te bepalen met behulp van de zogenaamde **ζ-functie** (zie appendix C.3). In het geval dat K gelijk is aan $GF(2^m)$ kan m dan geen priemgetal getal zijn; zoals al eerder is opgemerkt (zie paragraaf 5.1.2) bestaat er een vermoeden dat dit minder veilig is dan wanneer m wel een priemgetal is.

Van bovengenoemde drie mogelijkheden om krommen met het bijbehorende aantal punten op de kromme te genereren is niet bewezen welke nu veiliger zijn. Er zijn wel meningen over. De eerste methode, het kiezen van een kromme (random of niet) en dan het aantal punten tellen met het algoritme van Schoof, wordt gezien als de veiligste methode; het random kiezen wordt overigens als veiliger gezien dat het selecteren van een speciale kromme. De tweede methode met behulp van complexe vermenigvuldiging wordt als iets minder veilig gezien. De laatste methode die gebruik maakt van deellichamen wordt als het minst veilig gezien.

Verder kunnen deze drie methoden ook nog onderverdeeld worden naar efficiëntie. De eerste methode is het minst en de derde is het meest efficiënt. Ook hier kan de eerste methode weer onderverdeeld worden naar random en geselecteerde krommen: de laatste is efficiënter dan de eerste.

In onderstaande tabel worden alle opmerkingen over de genoemde drie methoden om een kromme te genereren met het bijbehorende aantal punten erop samengevat.

	Veiligheid (Hoe meer *'s, des te veiliger.)	Efficiëntie (Hoe meer #'s, des te efficiënter.)

Tel het aantal punten van een 1. (verifieerbaar) random of 2. speciale kromme meteen bv. met het algoritme van Schoof.	1. **** 2. ***	1. # 2. ##
Kies het aantal punten en construeer hierbij een kromme m.b.v. complexe vermenigvuldiging.	**	###
Tel het aantal punten op de kromme eerst over een deellichaam en leid hieruit het aantal punten over het gehele lichaam af.	*	####

5.3 De keuze van het genererend punt

Het genererende punt P op de kromme van orde n moet zo gekozen worden, dat het elliptische kromme discrete logaritme-probleem in de groep $<P>$ voortgebracht door P moeilijk is.

Als er willekeurig een punt op de kromme gekozen wordt, heeft deze een orde die het aantal punten op de kromme deelt. Om het discrete logaritme-probleem zo moeilijk mogelijk te maken moet dit punt een zo hoog mogelijke orde n hebben. In paragraaf 5.2 punt 3 is al opgemerkt dat het aantal punten op de kromme gelijk moet zijn aan kl waarbij l een groot priemgetal is en de cofactor k een klein geheel getal. Dit impliceert dat het punt P orde $n = l$ moet hebben.

In de praktijk wordt van het punt P eerst de x-coördinaat random gegenereerd, waarna er een bijbehorende y-coördinaat gezocht wordt, zodat P op de kromme ligt (als dit mogelijk is; anders wordt er een nieuwe x-coördinaat gegenereerd). Vervolgens wordt $Q=kP$ berekend. Als Q gelijk aan O is, heeft P niet de gevraagde priemorde; er moet dan een nieuw random punt P gegenereerd worden. Als Q ongelijk aan O is, is het punt P een mogelijke kandidaat. Vervolgens wordt nog gecontroleerd of $rQ=rkP=nP$ wel gelijk aan O is; als dit klopt, is P het gezochte generende punt van orde r.

In Hoofdstuk 4 is uiteengezet dat n minstens grootte $\log_2 n = 160$ bits moet hebben (voor korte termijn-veiligheid); de ondergrens voor n wordt aangeduid met r_{min}. Naarmate het onderliggende lichaam K groter wordt, is het beter een grotere n te nemen. Maar hoe hoger de ondergrens voor n des te moeilijker een bijbehorende kromme hiervoor te vinden is. Als maatstaf hiervoor wordt daarom naast een 'absolute' ondergrens voor n een bovengrens l_{max} gedefinieerd voor de priemfactoren van de cofactor k; als al deze priemfactoren kleiner zijn dan l_{max} wordt n 'relatief' als groot genoeg gezien.

5.4 Samenvatting: parametergeneratie

Voor de parametergeneratie moeten de volgende stappen doorlopen worden:

a. Kies een lichaamsgrootte q:
   1. $q = p$, $p>3$ priem of

2. $q = 2^m$, waarbij een indicatie gegeven wordt van de gebruikte basis en een reductiepolynoom van graad m over $GF(2)$, als de gebruikte basis een polynoombasis is.

(Zie paragraaf 5.1 en Hoofdstuk 4.)

b. Kies een ondergrens r_{min} voor de orde n die gewenst is voor het genererende punt P; hiervoor moet gelden dat $\log_2 r_{min}$ minstens 160 bits is.
(Zie paragraaf 5.3 en Hoofdstuk 4.)

c. Kies een bovengrens l_{max} voor de priemfactoren van k.
(Zie paragraaf 5.3 en Hoofdstuk 4.)

d. Kies twee lichaamselementen a en b die de vergelijking van de kromme E definiëren.
(Zie de laatste tabel in paragraaf 5.2.)

e. Bereken het aantal punten op de kromme (als dit nog niet bekend is).

f. Verifieer punt 1 uit paragraaf 5.2.

g. Verifieer punt 3 uit paragraaf 5.2. Hierbij moeten alle priemfactoren van k kleiner zijn dan l_{max} en l groter zijn dan r_{min}.

h. Verifeer punt 2 uit paragraaf 5.2.

i. Verifieer punt 4 uit paragraaf 5.2.

j. Zoek een punt $P \neq O$ op de kromme van orde n $(=l)$.
(Zie paragraaf 5.3.)

k. De parameters worden dan gevormd door: a, b, P, n en k. Als a en b verifieerbaar random gekozen zijn, moet ook de input SEED van de hashfunctie aan de parameters toegevoegd worden.

Opmerking: als één van de verificaties f, g, h of i 'false' oplevert, moet opnieuw bij punt d. begonnen worden met het genereren van een nieuwe kromme, waarna alle volgende punten weer doorlopen moeten worden.

Hoofdstuk 6
De praktische implementeerbaarheid

Uit de vorige hoofdstukken zal duidelijk zijn geworden dat het belangrijkste voordeel van elliptische kromme-systemen de schijnbaar grote cryptografische sterkte ten opzichte van de sleutellengte is. Zoals in Hoofdstuk 4 wordt duidelijk gemaakt, kan dit voordeel groeien ten opzichte van andere cryptosystemen met het groter worden van de computerkracht: bij deze laatste systemen zal de sleutellengte sneller moeten groeien dan bij elliptische kromme-systemen om dezelfde veiligheid te behouden. (Dit kan natuurlijk veranderen als er nieuwe sterkere aanvallen op elliptische kromme-systemen gevonden worden.)

De kortere sleutels kunnen resulteren in kortere systeemparameters en handtekeningen. Dit heeft weer gunstige gevolgen voor de benodigde rekentijd, geheugenopslag en bandbreedte. Dit alles impliceert dat elliptische kromme-systemen de meeste voordelen bieden in toepassingen waar bijvoorbeeld de bandbreedte, verwerkingscapaciteit en geheugenopslag gelimiteerd zijn, zoals draadloze telefoons, broadcast-toepassingen en smart cards.

Om de praktische implementeerbaarheid van elliptische kromme-systemen te kunnen beoordelen wordt in dit hoofdstuk als eerste ECDSA vergeleken met DSA en RSA (paragraaf 6.1), voornamelijk op de volgende, bovengenoemde, drie punten:
1. rekentijd;
2. geheugenopslag;
3. bandbreedte.

Uit deze (ruwe) vergelijking blijkt in welke opzichten ECDSA verschilt van DSA en RSA en wat de voor- of nadelen zijn van dit systeem. Vervolgens worden in paragraaf 6.2 de standaards genoemd die op dit moment in ontwikkeling zijn en elliptische kromme-toepassingen bevatten. Tot slot worden in de laatste paragraaf van dit hoofdstuk op de markt verkrijgbare elliptische kromme-produkten beschreven.

6.1 Vergelijking ECDSA en DSA/RSA

Een aantal mensen heeft geprobeerd een vergelijking te maken tussen ECDSA en DSA of RSA. Dit is niet eenvoudig, omdat er dan bepaald moet worden welke systemen vergelijkbaar zijn: welke sleutellengtes leveren min of meer equivalente systemen op; welke implementaties kunnen eerlijkerwijs met elkaar vergeleken worden?

Een antwoord op de eerste vraag kan gevonden worden aan de hand van de resultaten uit Hoofdstuk 4. Met de huidige kennis blijkt dat ECDSA met n (zie Hoofdstuk 2) ter grootte van 160 bits (i.e. $\log_2 n = 160$) vergelijkbare veiligheid levert als RSA met een modulus n (zie paragraaf 1.2.1) van 1024 bits of DSA met een modulus p (zie paragraaf 1.3.1.3) van 1024 bits. In het volgende overzichtje worden nog enkele waarden van bovengenoemde parameters vergeleken; voor de duidelijkheid is er een kolom met de verhouding van deze groottes toegevoegd. Deze waarden zijn gebaseerd op de gegevens uit het artikel van Odlyzko [32].

ECDSA ($\log_2 n$)	DSA ($\log_2 p$) & RSA ($\log_2 n$)	verhouding
106	512	1 : 4,83
132	768	1 : 5,82
160	1024	1 : 6,40
211	2048	1 : 9,70
320	5120	1 : 16,0
600	21000	1 : 35,0

Vanwege deze cijfers worden in de praktijk voornamelijk 160 bits-implementaties van elliptische kromme-systemen vergeleken met 1024 bits-RSA/DSA-implementaties.

Een antwoord op de tweede vraag is moeilijker te geven. Aan de ene kant kan men de meest eenvoudige implementaties vergelijken waar geen gebruik gemaakt is van allerlei optimalisatietechnieken om de implementatie zo snel mogelijk te maken. Aan de andere kant kunnen natuurlijk juist wel de snelste (voor zover bekend) implementaties vergeleken worden (hardware of software). Een ander probleem bij software-implementaties is de keuze van de machine: de resultaten kunnen heel anders uitvallen op de ene machine dan de andere. Daarom zijn vergelijkingen tussen elliptische kromme-systemen en andere cryptosystemen allen "ruw" te noemen.

6.1.1 De rekentijd

Het eerste aspect dat bekeken kan worden is de benodigde rekentijd. In [17] worden ECDSA en DSA met elkaar vergeleken. In dit artikel wordt een ruwe schatting gemaakt van de tijd die nodig is voor één bewerking, i.e. om kP te berekenen in ECDSA met k ter grootte van 160 bits en om $g^k \bmod p$ te berekenen in DSA met q een 160 bits-getal en p een 1024 bits-getal. Hierbij worden geen mogelijke versnellingen in acht genomen.

Voor deze vergelijking wordt ten eerste aangenomen dat een vermenigvuldiging in $\mathbf{Z}_p$ l^2 bewerkingen neemt, als $\log_2 p = l$. Hieruit volgt dan dat een lichaamsvermenigvuldiging bij DSA (i.e. een vermenigvuldiging in $\mathbf{Z}_p$ met $\log_2 p = 1024$ bits) $(1024/160)^2 \approx 41$ keer meer tijd neemt dan een lichaamsvermenigvuldiging in ECDSA (i.e. een vermenigvuldiging in $\mathbf{Z}_p$ met $\log_2 p = 160$ bits).

Het berekenen van kP met de methode van herhaald verdubbelen en optellen neemt gemiddeld 160 verdubbelingen en 80 optellingen op de kromme. Uit de optelformules (zie appendix C.1) blijkt dat een verdubbeling of optelling op een elliptische kromme 1 lichaamsinversie en 2 lichaamsvermenigvuldigingen gebruikt. (De kosten van de lichaamsoptellingen en -kwadrateringen worden buiten beschouwing gelaten, omdat deze verwaarloosbaar zijn ten opzichte van de lichaamsinversies en -vermenigvuldigingen.) Er wordt in dit artikel aangenomen dat een lichaamsinversie

ongeveer vergelijkbare tijd neemt als drie lichaamsvermenigvuldigingen. (Dit aantal wordt in de praktijk genoemd, wanneer het gebruikte lichaam $GF(2^m)$ is voor zekere m.) Dan neemt de berekening van kP ongeveer $5 \cdot (160 + 80) = 1200$ lichaamsvermenigvuldigingen. Dit correspondeert met $1200/41 \approx 29$ 1024-bit lichaamsvermenigvuldigingen.

Voor de berekening van $g^k \bmod p$ met de methode van herhaald kwadrateren en vermenigvuldigen (deze methode is dezelfde als 'de methode van herhaald verdubbelen en optellen', alleen nu met de vermenigvuldiging in plaats van de optelling) zijn gemiddeld 160 kwadrateringen en 80 vermenigvuldigingen in het lichaam $\mathbf{Z}_p$ nodig; dit komt dus overeen met ongeveer 240 1024-bit modulaire vermenigvuldigingen.

In de volgende tabel worden bovenstaande schattingen samengevat. De conclusie die hieruit valt te trekken is dat een bewerking in ECDSA ongeveer 8 keer zo snel is als in DSA, bij de gegeven grootes van de parameters en zonder gebruikmaking van extra versnellingen.

DSA	ECDSA
Bewerking: $g^k \bmod p$	Bewerking: kP
$\log_2 p = 1024$ bits	$E(\mathbf{Z}_p)$ waarbij $\log_2 p = 160$ bits
$\log_2 q = 160$ bits	$\log_2 k = 160$ bits
160 kwadrateringen in $\mathbf{Z}_p$ en	160 verdubbelingen op $E(\mathbf{Z}_p)$ en
80 vermenigvuldingen in $\mathbf{Z}_p$	80 optellingen op $E(\mathbf{Z}_p)$
240 1024-bit lichaamsvermenigvuldigingen	1200 160-bit lichaamsvermenigvuldigingen
240 1024-bit lichaamsvermenigvuldigingen	29 1024-bit lichaamsvermenigvuldigingen

In [35] wordt een ruwe vergelijking gemaakt tussen ECDSA (of ElGamal encryptie met elliptische krommen als er sprake is van encryptie/decryptie) over $GF(q)$ met q ter groote van 160 bits, RSA met n ter grootte van 1024 bits en e gelijk aan $2^{16}+1$ en systemen als DSA met p een 1024 bits-getal. Ook hier wordt er geen rekening gehouden technieken om de snelheid te optimaliseren.

In dit laatstgenoemde artikel wordt geen analyse beschreven zoals hierboven is uiteengezet, maar gezien de getallen die worden gegeven is er een soortgelijke methode gevolgd. Er wordt hier echter aangenomen dat een optelling op een elliptische kromme niet 5 maar 10 1024-bit lichaamsvermenigvuldigingen neemt. Dit leidt tot de volgende tabel (de getallen beschrijven het aantal benodigde 1024-bit lichaamsvermenigvuldigingen):

	ECDSA	RSA ($e=2^{16}+1$)	DSA
encryptie	120	17	480
decryptie	60	384	240
signeren	60	384	240
verifiëren	120	17	480

In de bovenstaande vergelijking werden versnellingen niet meegenomen. De genoemde systemen RSA, DSA en ECDSA bieden toch een aantal mogelijkheden om de rekentijd te verkorten. Zo kan er in RSA een korte publieke sleutel e gebruikt worden om de

handtekeningverificatie of encryptie te versnellen. In DSA en ECDSA kan een groot deel van de handtekeninggeneratie en encryptietransformaties voorberekend worden. Verder kunnen bij DSA en ECDSA het onderliggende lichaam en de representatie voor de elementen zo gekozen worden dat de lichaamsberekeningen kunnen worden geoptimaliseerd (zie ook paragraaf 5.1), al moet hierbij wel enige voorzichtigheid in acht genomen worden met betrekking tot aanvallen voor speciale keuzes van de parameters. Implementaties die van dergelijke versnellingen gebruikmaken laten zien dat ECDSA ongeveer 10 keer sneller is dan RSA of DSA, volgens de schrijvers van [8].

6.1.2 De geheugenopslag

Een tweede apect waarop implementaties met elkaar vergeleken kunnen worden is de benodigde geheugenopslag voor de sleutel en parameters: hoeveel bits zijn er nodig om een sleutelpaar en de systeemparameters op te slaan?

Bij DSA en ECDSA zijn er systeemparameters nodig (p, q en α bij DSA en q, a, b en P bij ECDSA). Bij beide systemen is het relatief moeilijk is om deze parameters te genereren. Zodra ze echter eenmaal bepaald zijn, kunnen ze voor langere tijd en voor meerdere gebruikers gebruikt worden. Merk op dat er bij RSA geen systeemparameters benodigd zijn.

De volgende tabel geeft een overzicht van het aantal bits geheugen waarmee de systeemparameters en sleutelparen kunnen worden beschreven.

	ECDSA	RSA ($e=2^{16}+1$)	DSA
Parameters	$(4 \cdot 160) + 1 = 641$	0	$160 + (2 \cdot 1024) = 2208$
Publieke sleutel	$160 + 1 = 161$	$1024 + 17 = 1041$	1024
Geheime sleutel	160	$2 \cdot 1024 = 2048$	160

Het feit dat er bij ECDSA bij de publieke sleutel 161 staat in plaats van de 320 bits die men zou verwachten, zit hem in het volgende. In ECDSA kan met behulp van 'point compression' de punt op een elliptische kromme gerepresenteerd worden door één lichaamselement en één extra bit, in plaats van twee lichaamselementen. Dus, als q een 160 bits getal is, kunnen de publieke sleutels met 161 bits gerepresenteerd worden. (Deze eigenschap is ook gebruikt voor de grootte van de systeemparameter P.) (Zie ook appendix C.4.)

6.1.3 De bandbreedte

Een derde aspect van de implementatie is de bandbreedte van het systeem, i.e. hoeveel bits moeten er verzonden worden om een vercijferde boodschap of handtekening uit te wisselen.

Alle drie de systemen RSA, DSA en ECDSA hebben vergelijkbare bandbreedtes wanneer ze gebruikt worden voor encryptie of het signeren van lange boodschappen (als encryptie-variant van DSA kan het ElGamal-encryptiesysteem genomen worden en dus van ECDSA de elliptische kromme-variant van het ElGamal-encryptiesysteem). Het geval waar korte boodschappen worden verzonden (bijvoorbeeld voor sleuteltransport) zijn er grotere verschillen. Bijvoorbeeld voor het signeren van een boodschap ter grootte van 2000 bits wordt de handtekeninggrootte gegeven in de volgende tabel [8]:

	Handtekeninggrootte
RSA	1024 bits
DSA	320 bits
ECC	320 bits

en voor de encryptie van een boodschap ter grootte van 100 bits [8]:

	Vercijferde boodschap
RSA	1024 bits
ElGamal	2048 bits
ECC	321 bits

De conclusie is hier dat er een besparing op de bandbreedte is bij elliptische kromme systemen voor korte boodschappen bij encryptie. De handtekeningen bij DSA en ECDSA hebben dezelfde grootte.

6.1.4 Andere opmerking

Een belangrijke consequentie van het gebruik van een kleinere groep in elliptische kromme systemen is dat goedkopere hardware-implementaties mogelijk zijn voor bijvoorbeeld smart cards. Zo is er bijvoorbeeld een ASIC (Application Specific Integrated Circuit) voor bewerkingen op een elliptische kromme over het lichaam $GF(2^{155})$ (zie [2]) met slechts 12.000 gates, terwijl een chip voor modulaire vermenigvuldigingen van 512 bits getallen [16] ongeveer 50000 gates heeft en een chip voor lichaamsvermenigvuldigingen in $GF(2^{593})$ (zie [3]) ongeveer 90000 gates heeft.

6.2 Standaards in ontwikkeling

Op dit moment worden er al verschillende standaards ontwikkeld waar elliptische kromme-systemen in voorkomen. Enkele daarvan zijn al bijna volledig. Hieronder volgt een korte opsomming:

1. IEEE P1363 (Standard Specifications for Public Key Cryptography) [13]
2. ANSI X9 American National Standards Institute (ANSI) ASC X9 (Financial Services: ANSI X9.62, the Elliptic Curve Digital Signature Algorithm (ECDSA) [4] en ANSI X9.63, Elliptic Curve Key Agreement and Transport Protocols [5]).
3. ISO/IEC 14888: Digital signature with appendix-Part 3: Certificate-based mechanisms [15].
4. IETF The OAKLEY Key Determination Protocol of the Internet Engineering Task Force (IETF) [14].
5. ATM Forum The ATM forum Technical Committee's Phase I ATM Security Specification.

De eerste twee standaards zijn de belangrijkste standaards waarin elliptische kromme-systemen genoemd worden. De IEEE-standaard bevat beschrijvingen van public key-systemen: naast elliptische kromme-systemen worden ook gewone discrete logaritme-systemen en integer factorisatie-systemen beschreven. De handtekeningen-systemen gebaseerd op elliptische krommen die hierin voorkomen zijn ECDSA en Nyberg-Rueppel. Deze standaard bevat expliciete algoritmen die gebruikt kunnen worden voor de implementatie van de genoemde public key-systemen, mits deze redelijk te begrijpen zijn. (Het algoritme van Schoof voor het tellen van het aantal punten op een elliptische

kromme is bijvoorbeeld niet in de standaard opgenomen, omdat dit te ingewikkeld was. In plaats daarvan is er een andere manier beschreven om een elliptische kromme te genereren waarvan het aantal punten bekend is.)

De ANSI-standaard bevat alleen elliptische kromme-systemen; de X9.62 gaat over ECDSA. NIST (National Institute of Standards and Technology) verwacht dat de X9.62-standaard als een FIPS (U.S. Federal Information Processing Standard) zal worden aangenomen. Een draft van de X9.62 en X9.63 standaards kan gevonden worden op de pagina waar ook de IEEE P1363 te vinden is (zie bibliografie).

De ISO-standaard specificeert enkele elliptische kromme-varianten van ElGamal-achtige handtekeningen-systemen.

De IETF-standaard bevat de key agreement-variant van het Diffie-Hellman protocol, onder andere over elliptische krommen. In het bijzonder worden de lichamen $GF(2^{155})$ en $GF(2^{210})$ genoemd.

De ATM Forum-standaard beschrijft systemen voor confidentialiteit, authenticatie, data-integriteit en 'access control' voor ATM (Asynchronous Transfer Mode) netwerken. Onder deze systemen bevinden zich ook elliptische kromme systemen.

6.3 Commerciële elliptische kromme-produkten

Er zijn tot nu toe eigenlijk twee bedrijven die elliptische kromme produkten aanbieden:

- Certicom (http://www.certicom.com/)
- RSA (http://www.rsa.com/)

Certicom bracht als eerste produkten met elliptische krommen op de markt. Daar een veelgehoord argument om elliptische krommen niet te gebruiken is dat er niet genoeg onderzoek naar gedaan zou zijn, probeert Certicom het onderzoek naar deze systemen te stimuleren door een challenge uit te loven (zie Hoofdstuk 4). Certicom is verder nauw betrokken bij de totstandkoming van de twee belangrijkste standaards waar elliptische krommen in voorkomen [4] [13]. Op dit moment bieden ze de volgende produkten:

- Certicom Security Builder SDK: een cryptografische toolkit voor verschillende toepassingen en platforms. Wat betreft digitale handtekeningen bevat deze toolkit DSA en Nyberg-Rueppel gebaseerd op elliptische krommen. Maar de toolkit bevat ook keymanagementfuncties, encryptiefuncties (waaronder ook gewone DES of triple DES), hashfuncties en random-number-generatoren. Platforms die ondersteund worden, zijn onder andere Windows NT en UNIX.
- Certicom Cryptographic Plug-ins: cryptografische service modules (dus kant en klaar). Er is bijvoorbeeld een plug-in voor Microsoft toepassingen (zoals Windows 95), die soortgelijke functies bevat als bij de Security Builder worden genoemd, maar ook plug-in die alleen voor digitale handtekeningen is gemaakt.
- Certicom Embedded System Designs: geoptimiliseerde cryptografische oplossingen voor 'resource-constrained devices', zoals smart cards.
- Certicom Integrated Circuit Designs: cryptografische technologie voor hardware implementaties.

Al deze produkten bevatten Certicom's Elliptic Curve Engine.

RSA kon toch niet achterblijven, maar houdt ook een slag om de arm met het argument dat er nog meer onderzoek gedaan moet worden. RSA heeft onlangs BSAFE 4.0 op de markt gebracht. Dit is ook een cryptografische toolkit, waarin enkele elliptische kromme-implementaties aangeboden worden (zoals parametergeneratie en ECDSA) naast de 'gewone' keymanagementsystemen (zoals Diffie-Hellman), digitale handtekeningen-systemen (bijvoorbeeld DSA), encryptiesystemen (RSA, DES, Triple-DES), hashfuncties (bijvoorbeeld MD5 en SHA-1) en random-number-generatoren. Ook is er een BSAFE API beschikbaar.

Bibliografie

[1] G. Agnew, T. Beth, R. Mullin and S. Vanstone, "*Arithmetic operations in $GF(2^m)$*", Journal of Cryptology, 6, (1993), 3 -13.

[2] G. Agnew, R. Mullin and S. Vanstone, "*An implementation of elliptic curve cryptosystems over $F_{2^{155}}$*", IEEE Journal on Selected Areas in Communications, 11 (1993), 804-813.

[3] G. Agnew, R. Mullin, I. Onyszchuk and S. Vanstone, "*An implementation for a fast public-key cryptosystem*", Journal of Cryptology, 3 (1991), 63-79.

[4] ANSI X9.62, "*Public key cryptography for the financial services industry - the Elliptic Curve Digital Signature Algorithm (ECDSA)*", draft, 1997.

[5] ANSI X9.63, "*Public key cryptography for the financial services industry - Elliptic Curve Key Agreement and Transport Protocols*", draft, 1997.

[6] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson, and M. Wiener, "*Minimal key lengths for symmetric ciphers to provide adequate commercial security*", January 1996.

[7] E. Brickell, D. Gordon, K. McCurley and D. Wilson, "*Fast Exponentiation with precomputation*", Advances in cryptology - EUROCRYPT'92, LNCS 658 (1993), Springer-Verlag, 200-207.

[8] Certicom, ECC Whitepapers: "*Current Public-Key Cryptographic Systems*", http://www.certicom.com/ecc/wecc2.htm

[9] N. Demytko, "*A New Elliptic Curve Based Analogue of RSA*", Advances in Cryptology - Eurocrypt'93, LNCS 765 (1993), Springer-Verlag, 40-49.

[10] W. Diffie and M.E. Hellman, "*New directions in cryptography*" , IEEE Transactions in Information Theory, volume IT-22, November 1976, 644-654.

[11] ElGamal, "*A public key cryptosystem and a signature scheme based on discrete logarithms*", Advances in Cryptology - CRYPTO'84, LNCS 196 (1985), Springer-Verlag, 10-18, 1985.

[12] FIPS 180-1, "*Secure hash standard*", Federal Information Processing Standards Publication 180-1, U. S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield , Virginia, April 17, 1995 (supersedes FIPS PUB 180).

[13] IEEE P1363, Standard for Public-Key Cryptography, draft standard, 1997, http://stdsbbs.ieee.org/groups/1363/index.html

[14] IETF The OAKLEY Key Determination Protocol of the Internet Engineering Task Force (IETF), http://www.ietf.cnri.reston.va.us/

[15] ISO/IEC 14888, "*Digital signature with appendix – Part 3: Certificate-based mechanisms*", draft, 1997.

[16] P. Ivey, S. Walker, J. Stern and S. Davidson, "*An ultra-high speed public key encryption processor*", Proceedings of IEEE Custom Integrated Circuits Conference, Boston, 1992, 19.6.1-19.6.4.

[17] Don B. Johnson en Alfred J. Menezes, "*Elliptic Curve DSA (ECDSA): An Enhanced DSA*", Certicom whitepaper, March 1997, http://www.certicom.com/ecc/wecdsa.htm

[18] N. Koblitz, "*Elliptic curve cryptosystems*", Mathematics of Computation, number 48, 1987, 203-209.

[19] N. Koblitz, "*CM curves with good cryptographic properties*", Advances in Cryptology - CRYPTO'91, LNCS 576 (1992), Springer-Verlag, 279-287.

[20] K. Koyama, U. Maurer, T. Okamoto, S. Vanstone, "*New Public-Key Schemes Based on Ellitic Curves over the Ring* Z_n", Advances in Crytpology - CRYPTO'91, LNCS 576 (1991), Springer-Verlag, 252-266.

[21] Georg-Johann Lay and Horst G. Zimmer, "*Constructing Elliptic Curves with Given Group Order over Large Finite Fields*", Algorithmic Number Theory: First International Symposium, LNCS 877 (1994), Springer-Verlag, 250-263.

[22] R. Lercier, "*Finding good random elliptic curves for cryptosystems defined over* F_{2^n}", Advances in Cryptography - EUROCRYPT'97, LNCS 1233 (1997), Springer-Verlag, 379-392.

[23] R. Lercier and F. Morain, "*Counting the number of points on elliptic curves over finite fields: strategies and performances*", Advances in Crytpology - EUROCRYPT'95, LNCS 921 (1995), Springer-Verlag, 79-94.

[24] A.J. Menezes, "*Elliptic Curve Public Key Cryptosystems*", Kluwer Academic Publishers, 1993.

[25] A. Menezes, T. Okamoto and S. Vanstone, "*Reducing elliptic curve logarithms to logarithms in a finite field*", IEEE Transactions on Information Theory, 39, (1993), 1639-1646.

[26] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, "*Handbook of Applied Cryptography*", CRC Press, Boca Raton, Florida, 1997.

[27] V.S. Miller, "*Use of elliptic curves in cryptography*", Advances in Cryptology - CRYPTO'85, LNCS 218 (1986), Springer Verlag, 417-426.

[28] R. Mullin, I. Onyszchuk, S. Vanstone and R. Wilson, "*Optimal normal bases in* $GF(p^n)$" Discrete Applied Mathematics, 22 (1988/89), 149-161.

[29] National Institute of Standards and Technology, "*Digital Signature Standard*", FIPS Publication 186, 1993.

[30] K. Nyberg and R. Rueppel, "*A new signature scheme based on the DSA giving message recovery*", 1st ACM Conference on Computer and Communications Security, ACM Press, 1993, 58-61.

[31] K. Nyberg and R. Rueppel, "*Message recovery for signature schemes based on the discrete logarithm problem*", Designs, Codes and Cryptography, 7 (1996), 61-81.

[32] A. Odlyzko, "*The Future of Integer Factorization*", Cryptobytes - The technical newsletter of RSA Laboratories, volume 1, number 2, Summer 1995, 5-12.

[33] P. Van Oorschot and M. Wiener, "*Parallel Collision Search With Application To Hash Functions And Discrete Logarithms*", in Proceedings of the 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, November 2-4, 1994, 210-218 and to appear in the Journal of Cryptology.

[34] J. Pollard, "*Monte Carlo methods for index computation mod p*", Mathematics of Computation, 32 (1978), 918-924.

[35] M.J.B. Robshaw and Yiqun Lisa Yin, "*Elliptic Curve Cryptosystems*", RSA Laboratories Technical Note, http://www.rsa.com/ecc/html/elliptic_curve.html

[36] R.L. Rivest, A. Shamir and L.M. Adleman, "*A method for obtaining digital signatures and public-key cryptosystems*", Communications of the ACM, volume 21, February 1978, 120-126.

[37] H. G. Rück, "*On the Discrete Logarithm in the Divisor Class Group of Curves*", preprint (1997).

[38] T. Satoh and K. Araki, "*Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves*", preprint, 1997.

[39] Bruce Schneier, "*Applied Cryptography: Protocols, Algorithms, and Source Code in C*", John Wiley & Sons, second edition, 1996.

[40] C.P. Schnorr, "*Efficient signature generation by smart cards*", Journal of Cryptology, volume 4, 1991, 161-174.

[41] R. Schoof, "*Elliptic curves over finite fields and the computation of square roots mod p*", Mathematics of Computation, 44 (1985), 483-494.

[42] R. Schoof, "*Counting points on elliptic curves over finite fields*", Journal de Theéorie des Nombres de Bordeaux 7 (1995), 219-254.

[43] Richard Schroeppel, Hilarie Orman, Sean O'Malley, Oliver Spatscheck, "*Fast Key Exchange with Elliptic Curve Systems*", Advances in Cryptology – CRYPTO'95, LNCS 963 (1995), pp. 43-56.

[44] I. A. Semaev, "*Evaluation of discrete logarithms in a group of p-torsion points of ellliptic curves in characteristic p*", Math. Comp. 67, 1998, 353-356.

[45] Smart, Posting to sci.math.research, September 30, 1997.

[46] Jerome A. Solinas, "*An Improved Algorithm for Arithmetic on a Family of Elliptic Curves*", Advances in Cryptology – CRYPTO'97, LNCS1294 (1997), Springer-Verlag, 357-371.

[47] Anne-Monika Spallek, "*Konstruktion elliptischer Kurven über endlichen Körpern zu gegebener Punktegruppe*", Diplomarbeit, Essen 1993.

[48] S. Vaudenay, "*Hidden collisions on DSS*", Advances in Cryptology - CRYPTO'96, LNCS 1109 (1996), Springer -Verlag, 83-88.

[49] Erik De Win, Antoon Bosselaers, Servaas Vandenberghe, Peter De Gersem, Joos Vandewalle, "*A Fast Sofware Implementations for Arithmetic Operations in $GF(2^n)$*", Advances in Cryptology - Asiacrypt'96, LNCS 1163 (1996), Springer-Verlag, 65-76.

Appendix A Complexiteit

A.1 Ordesymbolen

In dit document worden twee ordesymbolen gebruikt: O en o. Hun definities zijn als volgt (f en g zijn hierbij functies die gedefinieerd zijn op de positieve gehele getallen) [26]:

- $f(n) = O(g(n))$ als er een positieve constante c en een positief geheel getal n_0 bestaan, zodat voor alle $n \geq n_0$ geldt:
$$0 \leq f(n) \leq cg(n).$$
- $f(n) = o(g(n))$ als voor alle positieve constanten c er een positieve constante n_0 bestaat, zodat voor alle $n \geq n_0$ geldt:
$$0 \leq f(n) < cg(n).$$

Dus, $f(n) = O(g(n))$ betekent dat f asymptotisch niet sneller groeit dan een constante veelvoud van $g(n)$ en $f(n) = o(g(n))$ betekent dat $g(n)$ een bovengrens voor $f(n)$ is die niet asymptotisch dicht is, oftewel $f(n)$ wordt zeer klein ten opzichte van $g(n)$ als n groter wordt. De expressie $o(1)$ wordt vaak gebruikt om een functie $f(n)$ aan te duiden waarvan de limiet naar 0 gaat als n naar oneindig gaat.

A.2 Subexponentiële algoritmen

Voor sommige wiskundige problemen bestaan er algoritmen die **subexponentieel** genoemd worden. De tijd die deze algoritmen nemen ligt tussen polynomiale en exponentiële tijd in. De looptijd wordt beschreven door de volgende expressie:
$$L[x,c,\alpha] = O(\exp((c + o(1))(\ln x)^{\alpha} (\ln \ln x)^{1-\alpha})),$$
waarbij 'exp(y)' hetzelfde als e^y is, x de grootte van de inputruimte, c een willekeurige constante en α een constante waarvoor geldt $0 < \alpha < 1$.

Merk op dat x de grootte van de inputruimte is, zodat $\ln x$ een maat voor de grootte n van een bepaalde input is. Als α gelijk aan 0 wordt genomen, wordt dit gelijk aan
$$L[x,c,\alpha] = O(\exp((c + o(1))(\ln \ln x))) = O((\ln x)^{c + o(1)}).$$
Dus dan is het algoritme polynomiaal in $\ln x$ (oftewel polynomiaal in n). Als α gelijk is aan 1, is de expressie gelijk aan:
$$L[x,c,\alpha] = O(\exp((c + o(1))(\ln x))),$$
oftewel het algoritme is exponentieel in $\ln x$ (en dus exponentieel in n).

Appendix B
Modulo rekenen, lichamen, groepen en de discrete logaritme

B.1 Gehele getallen modulo *n*

Laat n een positief geheel getal zijn. Een geheel getal a heet **congruent** aan een geheel getal b **modulo** n als a-b deelbaar is door n. Dit wordt genoteerd als:

$$a \equiv b \pmod{n}.$$

Bij een geheel getal a hoort dan een **equivalentieklasse** welke bestaat uit alle gehele getallen die congrent aan a modulo n zijn.

Voorbeeld:
Neem n gelijk aan 7. Dan is $35 \equiv 0 \pmod{7}$ en $15 \equiv 1 \pmod{7}$; de equivalentieklasse die hoort bij het getal 1 zijn de getallen : $\{\ldots,-13, -6, 1, 8, 15, \ldots\}$.

De **gehele getallen modulo** n, genoteerd als $\mathbf{Z}_n$, is de verzameling van alle (equivalentieklassen van) gehele getallen $\{0, 1, 2, \ldots, n-1\}$. Optellen, aftrekken en vermenigvuldiging in $\mathbf{Z}_n$ gebeurt allemaal modulo n.

Voorbeeld:
Neem n weer gelijk aan 7. In $\mathbf{Z}_7$ geldt: $3 + 5 \equiv 1 \pmod{7}$, $3 - 5 \equiv 5 \pmod{7}$ en $3 \cdot 5 \equiv 1 \pmod{7}$.

Een belangrijke stelling die gebruik maakt van modulo rekenen is de **Chinese Reststelling**, welke het volgende zegt:

Als de gehele getallen $n_1, n_2, \ldots, n_k$ paarsgewijs geen gemeenschappelijke delers hebben (i.e. 'relatief priem' zijn), dan heeft het volgende systeem van congruentie-vergelijkingen een unieke oplossing modulo $n = n_1 \cdot n_2 \cdot \ldots \cdot n_k$:

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

B.2 Lichamen en groepen

Bekende voorbeelden van **lichamen** zijn de verzamelingen van de reële getallen, de complexe getallen, de rationale getallen (breuken) en de gehele getallen modulo een priemgetal (deze laatste is een voorbeeld van een **eindig lichaam**, i.e. een lichaam met een eindig aantal elementen). Een verzameling is een lichaam als er een optelling en een vermenigvuldiging aanwezig is met de bijbehorende inverse operaties (dat wil zeggen aftrekken respectievelijk delen); deze bewerkingen moeten altijd een uitkomst geven, behalve delen door 0, wat niet gedefinieerd is.

Iets abstracter maar exact geformuleerd is een lichaam een verzameling met een optelling + en een vermenigvuldiging ·, waarvoor het volgende geldt:

(i) De optelling en vermenigvuldiging zijn associatief: $a + (b + c) = (a + b) + c$ en $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ voor alle a, b en c in het lichaam.

(ii) De optelling heeft een nulelement 0 en de vermenigvuldiging heeft een eenheidselement 1; voor deze elementen geldt: $0 + a = a + 0 = a$ en $1 \cdot a = a \cdot 1 = a$ voor alle elementen a in het lichaam.

(iii) Alle elementen hebben een inverse element voor de optelling en de vermenigvuldiging; voor de vermenigvuldiging is er één uitzondering: 0 heeft geen multiplicatieve inverse. Dus voor alle elementen a in het lichaam bestaat er een $-a$ in het lichaam waarvoor geldt: $a + (-a) = 0$; voor alle elementen a ongelijk aan 0 bestaat er een a^{-1} waarvoor geldt $a \cdot a^{-1} = 1$.

(iv) De distrubutieve axioma's gelden voor de optelling en de vermenigvuldiging: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ en $(b + c) \cdot a = (b \cdot a) + (c \cdot a)$ voor alle a, b en c in het lichaam.

(v) Zowel de optelling als de vermenigvuldiging is commutatief: $a + b = b + a$ en $a \cdot b = b \cdot a$ voor alle elementen a en b in het lichaam.

Naast lichamen worden in dit document ook **groepen** gebruikt, welke eigenlijk een primitievere vorm van lichamen zijn (een lichaam is een groep, maar een groep is geen lichaam). Een groep is een verzameling met een operatie, die bijvoorbeeld aangeduid kan worden met een *, waarvoor de volgende eigenschappen gelden:

(i) * is associatief: $a * (b * c) = (a * b) * c$ voor alle elementen a, b en c in de groep.

(ii) Er is een eenheidselement 1 in de groep: $1 * a = a * 1 = a$ voor alle elementen a in de groep.

(iii) Voor alle element a in de groep is er een inverse a^{-1} in de groep waarvoor geldt: $a*a^{-1} = a^{-1}*a = 1$.

B.3 Discrete logaritme

Laat G een groep zijn met bewerking * en laat g een element in G zijn. Stel dat we

$$\underbrace{g * g * \ldots * g * g}_{l \text{ keer}}$$

schrijven als g^l, dat wil zeggen met een **multiplicatieve** schrijfwijze. (Een alternatief is de **additieve** schrijfwijze lg.) Verder wordt g^0 gedefinieerd als 1, het eenheidselement van de groep.

Stel verder dat elk element uit G geschreven kan worden als g^i voor een zeker i. Dan heet G **cyclisch** en g een **generator** (of voortbrenger) van G. De groep G kan dan geschreven worden als $\{g^i \mid i = 0, 1, \ldots, n-1\}$, waarbij n de kleinste exponent i is waarvoor geldt dat $g^i = 1$; n wordt de **orde** van g genoemd.

Stel nu dat een element h in de cyclische groep G gelijk is aan g^l voor een zekere l. Dan wordt l de **discrete logaritme** van h naar g genoemd en het vinden van l gegeven h en g het **discrete logaritme-probleem** in de groep G. Het discrete logaritme-probleem kan dus in elke cyclische groep gedefinieerd worden. De moeilijkheid van het discrete logaritme-probleem verschilt echter per groep. Als voorbeeld geven we hier de gehele getallen modulo het priemgetal 7 met achtereenvolgens de optelling en de vermenigvuldiging als bewerking: $(\mathbf{Z}_7, +)$ en $(\mathbf{Z}_7^*, \cdot)$; dit zijn twee verschillende cyclische groepen.

Voorbeeld: $(\mathbf{Z}_7, +)$
$\mathbf{Z}_7$ is gelijk aan de verzameling $\{0, 1, 2, 3, 4, 5, 6\}$. De optelling + is gelijk aan de optelling modulo 7 en het eenheidselement (= nulelement) is het element 0. Een voortbrenger van deze groep is het element 1: elk element uit $\mathbf{Z}_7$ kan geschreven worden als een veelvoud van 1 (met de additieve schrijfwijze). De orde van 1 is gelijk aan p, want $p\,1 = p \equiv 0 \pmod{p}$.
Het discrete logaritme-probleem naar 1 in deze groep is triviaal: de discrete logaritme van bijvoorbeeld 2 naar 1 is gelijk aan 2.

Voorbeeld: $(\mathbf{Z}_7^*, \cdot)$
$\mathbf{Z}_7^*$ bestaat uit alle elementen uit $\mathbf{Z}_7$ ongelijk aan 0: $\{1, 2, 3, 4, 5, 6\}$. De vermenigvuldiging $\cdot$ is gelijk aan de vermenigvuldiging modulo 7 en het eenheidselement is het element 1. Een voortbrenger van deze groep is het element 3: elk element uit $\mathbf{Z}_7^*$ kan geschreven worden als een macht van 3 (met de multiplicatieve schrijfwijze):
3
$3 \cdot 3 = 3^2 \equiv 2 \pmod{7}$
$3 \cdot 3 \cdot 3 = 3^3 \equiv 6 \pmod{7}$
$3 \cdot 3 \cdot 3 \cdot 3 = 3^4 \equiv 4 \pmod{7}$
$3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 = 3^5 \equiv 5 \pmod{7}$
$3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 = 3^6 \equiv 1 \pmod{7}$
Het discrete logaritme-probleem naar 3 in deze groep is niet triviaal: het is bijvoorbeeld niet meteen duidelijk wat de discrete logaritme van 4 naar 3 is. Daar deze verzameling klein is, kan er echter wel een tabel worden gemaakt, waar de uitkomst in opgezocht kan worden. Maar het moge duidelijk zijn dat voor zeer grote groepen slimmere manieren gezocht moeten worden.

Appendix C
Elliptische krommen

C.1 De optelling op een elliptische kromme

Een elliptische kromme kan beschreven worden door een (affine) Weierstraß-vergelijking:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (*)$$

waarbij de a_i's in een bepaald getallenlichaam K zitten. De punten op de kromme zijn de paren (x,y) die aan deze vergelijking voldoen, waarbij x en y in K zitten, plus nog een extra punt O, het 'punt in het oneindige':

$$E = \{(x,y) \in K^2 \mid y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup O.$$

In paragraaf 2.1 zijn de negatieve en de optelling van punten beschreven met behulp van een stukje meetkunde. De negatieve van een punt wordt verkregen door het punt te spiegelen in de symmetrieas van de kromme. De optelling gaat als volgt: trek de lijn door de twee punten die opgeteld moeten worden (of neem de raaklijn aan de kromme als de twee punten gelijk zijn), spiegel het derde snijpunt in de symmetrieas van de kromme en beschouw dit laatste punt als de som van de twee punten.

Er kunnen ook een expliciete vergelijkingen gegeven worden van de coördinaten van de negatieve en de som van twee punten in termen van de coördinaten van de oorspronkelijke punten; deze vergelijkingen volgen uit het expliciet maken van bovenstaande meetkunde.

Stel dat we de negatieve willen weten van het punt $P = (x_1, y_1)$. Uit de Weierstraß-vergelijking (*) volgt dat er slechts één punt is met dezelfde x-coördinaat x_1; dit is het gezochte punt $-P$ en is dus gelijk aan

$$-P = (x_1, -y_1 - a_1x_1 - a_3).$$

Stel verder dat de punten $P = (x_1, y_1)$ en $Q = (x_2, y_2)$ opgeteld moeten worden en dat l de lijn door P en Q is, als $P \neq Q$, of de raaklijn aan de kromme in P als $P = Q$. Dan wordt l gegeven door de vergelijking $y = \lambda x + \beta$ met λ en β gegeven door de volgende vergelijkingen.

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{als } P \neq Q, \\ \dfrac{3x_1^{\ 2} + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, & \text{als } P = Q. \end{cases}$$

$$\beta = y_1 - \lambda\, x_1$$

Door deze lijn met de kromme te snijden (met behulp van de Weierstraß-vergelijking (*)) kan het derde snijpunt van de lijn met de kromme gevonden worden. Het spiegelen van dit punt in de symmetrieas volgt uit de formule voor de negatieve van een punt (zie boven). Dit resulteert in de volgende vergelijkingen voor de coördinaten van het punt $P+Q = (x_3, y_3)$ (met λ en β als boven):

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$$
$$y_3 = -(\lambda + a_1)\, x_3 - \beta - a_3$$

Deze voorstelling van de optelling gaat er eigenlijk van uit dat het lichaam K uit de reële getallen bestaat. Het blijkt echter dat dezelfde formules blijven gelden wanneer voor K een ander lichaam, zoals bijvoorbeeld een eindig lichaam, wordt genomen (de bewerkingen in de formules moeten dan alleen in dit lichaam uitgevoerd worden in plaats van in de reële getallen).

In paragraaf 2.1 is ook opgemerkt dat de Weierstraß-vergelijking (*) vereenvoudigd kan worden (door 'vervanging van variabelen', zie bijvoorbeeld [24]). De formules van de negatieve en de optelling kunnen dan ook vereenvoudigd worden en volgen direct uit de oorspronkelijke formules.

Voor het lichaam $GF(p)$ $(p>3)$ is de vereenvoudigde Weierstraß-vergelijking bijvoorbeeld gelijk aan:

$$y^2 = x^3 + ax + b\,,$$

waarbij a en b in $GF(p)$ zitten. De formules voor de negatieve $-P$ van een punt $P=(x_1,y_1)$ en de optelling $P+Q = (x_3, y_3)$ van twee punten $P=(x_1, y_1)$ en $Q=(x_2, y_2)$ worden dan gegeven door de volgende vergelijkingen ($a_1 = a_3 = a_2 = 0$, $a_4 = a$, $a_6 = b$):

$$-P = (x_1,-y_1)$$
$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda\,(x_1 - x_3) - y_1$$

waarbij

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{als } P \neq Q, \\ \dfrac{3x_1^{\ 2} + a}{2y_1}, & \text{als } P = Q. \end{cases}$$

Voor het lichaam $GF(2^m)$ zijn er twee soorten vergelijkingen mogelijk, maar de krommen over dit lichaam die in de praktijk gebruikt worden hebben alle de volgende vorm:

$$y^2 + xy = x^3 + ax^2 + b\,.$$

De formules voor de negatieve $-P$ van een punt $P=(x_1,y_1)$ en de optelling $P+Q = (x_3, y_3)$ van twee punten $P=(x_1, y_1)$ en $Q=(x_2, y_2)$ worden dan gegeven door de volgende vergelijkingen ($a_3 = a_4 = 0$, $a_1 = 1$, $a_2 = a$, $a_6 = b$):

$$-P = (x_1, y_1 + x_1)$$

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a_2, & \text{als } P \neq Q, \\ x_1^{\,2} + \dfrac{a_6}{x_1^{\,2}}, & \text{als } P = Q. \end{cases}$$

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1, & \text{als } P \neq Q, \\ x_1^{\,2} + \left(x_1 + \dfrac{y_1}{x_1}\right)x_3 + x_3, & \text{als } P = Q. \end{cases}$$

C.2 De discriminant van een Weierstraß-vergelijking en (super)singulariteit

Uit de Weierstraß-vergelijking van een elliptische krommen kan men ook afleiden of de kromme die door deze vergelijking wordt gedefinieerd singulier of niet-singulier is. Om dit te kunnen bepalen, is de **discriminant** benodigd. De discriminant van een kromme gegeven door de Weierstraß-vergelijking (*) wordt als volgt gedefinieerd:

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6$$

waarbij

$$d_2 = a_1^2 + 4a_2$$
$$d_4 = 2a_4 + a_1 a_3$$
$$d_6 = a_3^2 + 4a_6$$
$$d_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2$$

Een kromme is nu niet-singulier als de discrimant Δ van de bijbehorende Weierstraß-vergelijking ongelijk aan 0 is.
Bij de vereenvoudigde Weierstraß-vergelijkingen uit paragraaf C.1 horen ook eenvoudigere formules voor de discriminant. Met $a_1 = a_3 = a_2 = 0$, $a_4 = a$ en $a_6 = b$ volgt dat voor krommen over $GF(p)$ waarvoor de Weierstraß-vergelijking vereenvoudigd kan worden tot

$$y^2 = x^3 + ax + b$$

de discriminant gelijk is aan

$$\Delta = -16(4a^3 + 27b^2)$$

Een kromme gegeven door deze vergelijking is dus niet-singulier als $4a^3 + 27b^2$ ongelijk aan 0 is.

Voor krommen over het lichaam $GF(2^m)$ kan de Weierstraß-vergelijking omgezet worden in één van de volgende vergelijkingen:

$$y^2 + xy = x^3 + a_2 x^2 + a_6 \qquad (1)$$

of

$$y^2 + a_3 y = x^3 + a_4 x + a_6. \qquad (2)$$

De bijbehorende discriminanten zijn a_6 respectievelijk a_3^4. De krommen zijn dus niet-singulier als a_6 ongelijk aan 0 is of a_3 ongelijk aan 0 is.

Zoals al een paar keer is opgemerkt, wordt van deze vergelijkingen in de praktijk vergelijking (2) niet gebruikt. Dit is een gevolg van het feit dat krommen met een dergelijke Weierstraß-vergelijking supersingulier zijn.

Supersingulariteit voor krommen over het lichaam $GF(2^m)$ is af te lezen uit de zogenaamde **j-invariant**. De j-invariant $j(E)$ kan uitgedrukt worden in termen van de discriminant en de waarden d_2 en d_4 die bij de definitie van de discriminant werden gebruikt, namelijk:

$$j(E) = (d_2^2 - 24d_4)^3/\Delta \quad (\text{mits } \Delta \neq 0).$$

Voor vergelijking (1) is af te leiden dat $j(E)$ gelijk aan $1/a_6$ en voor vergelijking (2) is $j(E)$ gelijk aan 0.

Een kromme over $GF(2^m)$ is nu supersingulier dan en slechts dan als de j-invariant gelijk aan 0 is. Dat is het geval voor vergelijking (2) en niet voor vergelijking (1), dus alleen vergelijking (1) kan worden gebruikt voor cryptosystemen.

C.3 Het aantal punten op een elliptische kromme

Het aantal punten op een kromme waarvan de Weierstraß-vergelijking is gegeven, kan geteld worden met behulp van het algoritme van Schoof. Alhoewel het algoritme zelf behoorlijk inefficient is, zijn er een aantal dramatische verbeteringen en uitbreidingen van deze methode ontdekt in de afgelopen jaren. Op dit moment kunnen de aantallen punten van krommen over $GF(p)$ met p met een grootte van ongeveer 10^{499} en over $GF(2^m)$ met m rond de 1300 berekend worden. Het genereren van een 'goede' ellitpische krommen over lichamen ter grootte van $GF(2^{196})$ kost ongeveer 5 uur op een workstation (zie [22] en [23]).

In paragraaf 5.2 is er ook een methode genoemd voor het bepalen van het aantal punten op een kromme, waarbij eerst het aantal punten over een deellichaam wordt berekend en vervolgens het aantal over het gehele lichaam wordt afgeleid. Deze methode maakt gebruik van de **stelling van Weil**, welke zegt hoe het aantal punten over een deellichaam afhangt van het aantal punten over het gehele lichaam.

Stel dat E een elliptische kromme is gedefinieerd over het lichaam $GF(q)$, waarvan het aantal punten $|E(GF(q))|$ bekend is. Neem t gelijk aan $t = q + 1 - |E(GF(q))|$. Dan is de **karakteristieke vergelijking** bij deze kromme gelijk aan $T^2 - tT + q$. Deze vergelijking kan ontbonden worden in $(T - \alpha)(T - \beta)$, waarbij α en β complexe getallen zijn. (In dit verband wordt ook vaak de zogenaamde **ζ-functie** genoemd, welke gelijk is aan

$$Z(T) = \frac{1 - tT + qT^2}{(1-T)(1-qT)} .$$

Er geldt dat $1 - tT + qT^2 = (1 - \alpha T)(1 - \beta T)$.)

De stelling van Weil zegt dan dat het aantal punten op de elliptische kromme over $GF(q^k)$ gelijk is aan $|E(GF(q^k))| = q^k + 1 - \alpha^k - \beta^k$ voor alle $k \geq 1$.

Voorbeeld:
Stel we willen een kromme over $GF(2^m)$ kiezen, waarbij m deelbaar is door een klein getal l. Daar m gedeeld wordt door l, is $GF(2^l)$ een deellichaam van $GF(2^m)$.
Kies daarom random een elliptische kromme E: $y^2 + xy = x^3+ax^2+ b$, $b \neq 0$, waarbij a, b, in $GF(2^l)$ en bereken het aantal punten op deze kromme $|E(GF(2^l))|$. Omdat l klein is, kan dit laatste bijvoorbeeld met behulp van exhaustive search gedaan worden: probeer alle combinaties voor x en y uit $GF(2^l)$ in te vullen in de Weierstraß-vergelijking. Het aantal punten $|E(GF(2^m))|$ op de kromme over $GF(2^m)$ volgt dan uit de stelling van Weil met $q = 2^l$ en $k = m/l$.

C.4 Point compressie

In paragraaf 6.1.2 is opgemerkt dat voor een punt op een elliptische kromme niet beide coördinaten opgeslagen hoeven worden, maar dat één coördinaat plus één extra bit voldoende is. Hieronder wordt uitgelegd hoe dit bewerkstelligd kan worden. Er moet hierbij onderscheid gemaakt worden tussen krommen over $GF(p)$ en krommen over $GF(2^m)$.

Noem het meest rechtse bit van een getal de rest die men overhoudt na deling van dit getal met 2.

Een punt op een kromme over $GF(p)$ met Weierstraß-vergelijking $y^2 = x^3 + ax + b$ kan gerepresenteerd worden door de x-coördinaat van dit punt plus het meest rechtse bit van de y-coördinaat. Terugrekenen gaat dan als volgt: bereken y^2 door de x-coördinaat te substitueren in de Weierstraß-vergelijking en bereken een wortel van y^2; als het meest rechtse bit van deze wortel gelijk is aan het extra bit van de representatie, dan is de bijbehorende y-coördinaat deze wortel en anders is deze gelijk aan p – deze wortel.

Een punt op een kromme over $GF(2^m)$ met Weierstraß-vergelijking $y^2 + xy = x^3 + ax^2 + b$ kan gerepresenteerd worden door de x-coördinaat van dit punt plus één extra bit dat gelijk is aan:

- 0, als de x-coördinaat gelijk is aan 0.
- het meest rechtse bit in yx^{-1}, als de x-coördinaat ongelijk is aan 0.

Terugrekenen gaat dan als volgt:

- als de x-coördinaat gelijk is aan 0, dan is de y-coördinaat de wortel van b.
- als de x-coördinaat ongelijk is aan 0, bereken dan $\beta = x + a + bx^{-2}$ en vind een z zodat $z^2 + z = \beta$. Neem het meest rechtse bit van z en vergelijk dit met het extra bit van de representatie. Als deze twee bits niet gelijk zijn, vervang z dan door $z + 1$. De bijbehorende y-coördinaat wordt dan gegeven door xz.

Appendix D
RSA gebaseerd op elliptische krommen

In 1993 introduceerde Demytko een RSA-achtig cryptosysteem gebaseerd op elliptische krommen [9]. Het systeem is net zoals RSA gebaseerd op het integer factorisatie-probleem (zie paragraaf 1.2). Voor de beschrijving van dit systeem is het begrip "complementaire kromme" van een elliptische kromme nodig.

Stel dat er een elliptische kromme E gegeven is over het lichaam $GF(p)$, $p>3$, met Weierstraß-vergelijking:

$$y^2 = x^3 + ax + b,$$

met a en b in $GF(p)$. Laat verder v een vast element van $GF(p)$ zijn dat geen kwadraat is in dit lichaam (dus er is geen element in $GF(p)$ waarvan het kwadraat modulo p gelijk is aan v). Dan wordt de **complementaire kromme** van E ten opzichte van v, genoteerd door $\overline{E}_v$, gedefinieerd als de verzameling van alle paren (x, y) met x in $GF(p)$ en y gelijk aan $u\sqrt{v}$ (mod p), waarbij u in $GF(p)$, die voldoen aan dezelfde Weierstraß-vergelijking als E, i.e.:

$$y^2 = x^3 + ax + b.$$

Verder behoort het punt O ook tot de verzameling punten op de complementaire kromme.

Op de complementaire kromme kan ook een optelling gedefinieerd worden, analoog aan de oorspronkelijke optelling; de punten op de complementaire kromme vormen met deze optelling weer een groep. Stel dat de punten $P = (x_1, y_1) = (x_1, u_1\sqrt{v})$ en $Q = (x_2, y_2) = (x_2, u_2\sqrt{v})$ gegeven zijn op de complementaire kromme met bovenstaande Weierstraß-vergelijking. Dan zijn de optelregels als volgt gedefinieerd:

- $P + O = P$
- $-P = (x_1, -u_1\sqrt{v})$
- Als $P + Q = (x_3, y_3)$, worden x_3 en y_3 gegeven door

$$x_3 = (\lambda')^2 v - x_1 - x$$
$$y_3 = (\lambda'(x_1 - x_3) - u_1)\sqrt{v}$$

met

$$\lambda' = \begin{cases} \dfrac{u_2 - u_2}{x_2 - x_1}, & \text{als } P \neq Q, \\ \dfrac{3x_1^2 + a}{2u_1\sqrt{v}}, & \text{als } P = Q. \end{cases}$$

Bij het Demytko-cryptosysteem kiest elke gebruiker twee priemgetallen p en q zodat het factoriseren van $n = pq$ moeilijk is. Verder kiest elke gebruiker gehele getallen a en b, zodat geldt $ggd(4a^3 + 27b^2, n) = 1$.

Met deze coëfficiënten a en b kunnen dan de Weierstraß-vergelijkingen van de krommen $E(GF(p))$ en $E(GF(q))$ over $GF(p)$ respectievelijk $GF(q)$ gedefinieerd worden met de bijbehorende complementaire krommen $\overline{E}(GF(p))_{v_1}$ en $\overline{E}(GF(q))_{v_2}$.

Vervolgens worden vier constanten N_i, $i = 1, 2, 3, 4$, gedefinieerd als de aantallen punten op de vier bovenstaande krommen:

$$N_1 = |E(GF(p))|,$$
$$N_2 = |\overline{E}(GF(p))_{v_1}|,$$
$$N_3 = |E(GF(q))|,$$
$$N_4 = |\overline{E}(GF(q))_{v_2}|.$$

Elke gebruiker kiest net als bij RSA een encryptieexponent e, waarvoor hier geldt dat $ggd(e, N_i) = 1$ voor $i = 1, 2, 3, 4$. Er zijn vier verschillende decryptieexponenten d_i gegeven door de volgende vergelijkingen:

$$ed_1 \equiv 1 \pmod{kgv(N_1, N_3)},$$
$$ed_2 \equiv 1 \pmod{kgv(N_1, N_4)},$$
$$ed_3 \equiv 1 \pmod{kgv(N_2, N_3)},$$
$$ed_4 \equiv 1 \pmod{kgv(N_2, N_4)}.$$

De publieke sleutel bij het Demytko-systeem bestaat dus uit de getallen n, e, a en b, terwijl de geheime sleutel bestaat uit de getallen p, q, d_1, d_2, d_3 en d_4.

Als voorbeeld van de werking van het Demytko-systeem wordt de encryptie van een boodschap m, welke een getal van 1 tot n is, gedemonstreerd. Maar dit systeem zou ook voor digitale handtekeningen gebruikt kunnen worden.

De boodschap m wordt als x-coördinaat van een punt P op de elliptische kromme over Z_n genomen. (Z_n is geen lichaam, maar toch kan hierover een kromme met een optelling gedefinieerd worden. De optelling is analoog aan de gewone optelling, er zijn alleen een paar uitzonderingsgevallen.) De cijfertekst wordt dan gelijk genomen aan de x-coördinaat van het punt $Q = eP$, die hier voor het gemak c genoemd wordt.

Voor de decryptie moet eerst het getal $w = c^3 + ac + b \pmod{n}$ berekend worden. Hiermee wordt dan op de volgende manier bepaald welke van de vier decryptieexponenten d_i gebruikt zal worden voor de decryptie:

$$i = \begin{cases} 1 \text{ als } \left(\frac{w}{p}\right) = \left(\frac{w}{q}\right) = 1, \\ 2 \text{ als } \left(\frac{w}{p}\right) = 1, \left(\frac{w}{q}\right) = -1, \\ 3 \text{ als } \left(\frac{w}{p}\right) = -1, \left(\frac{w}{q}\right) = 1, \\ 4 \text{ als } \left(\frac{w}{p}\right) = \left(\frac{w}{q}\right) = -1. \end{cases}$$

($\left(\frac{\cdot\cdot}{\cdot\cdot}\right)$ is het Legendre symbool, zie bijvoorbeeld in [26].)

Vervolgens bestaat de decryptie uit het berekenen van d_iQ; de boodschap is dan gelijk aan de x-coördinaat van dit laatste punt.

Management Summary

This report describes and analyses a specific digital signature scheme: the relatively new digital signature scheme based on Elliptic Curves. This summary consists of two parts: the first part introduces digital schemes; a prerequisite for understanding the actual summary of this report in the second part of this summary.

1 Introducing digital signatures

1.1 What are digital signatures?

This report is about the technique of digital signatures. Digital signatures enable a (legal) binding between a legal party (e.g. a person) and certain information in such a way that:
only the party can create this binding;
the binding can be verified (validated) by anyone (i.e. without the need of any secret information).
A digital signature can be envisioned as a number of about 100-300 digits attached to the information.

1.2 What is the purpose of digital signatures?

With a digital signature one can achieve the property of *non-repudiation*: if a (legal) party has signed certain information, then it can not deny that later on. In case of dispute, any third party (for instance a judge) can verify the digital signature. In this respect, digital signatures differ from Message Authentication Codes (MAC's), that are commonly used in financial transactions: in the creation and validation of a MAC, the same secret information is required. Therefore a MAC is not reducible to only one party.

1.3 Where and how are digital signatures used?

Digital signatures protect the integrity and authenticity of information; they do not protect the confidentiality of information. Digital signatures are used in applications where the integrity and non-repudiation of information is important (e.g. contracts in Electronic Commerce). Digital signatures are also used for controlling access to information systems. To create a digital signature a party uses its *private* key; a matching *public* key makes the verification of the digital signature possible. As this public key must be widely known, special structures (*Public Key Infrastructures*) are usually constructed.

1.4 What is of importance with respect to digital signature systems?

The following aspects are of importance with respect to digital signature schemes:

- Security.
  The certainty that the non-repudiation of the system can not be compromised.
- Complexity of setting up the system.
  Here a distinction is made between the one-time generation of the system parameters (used by all uses) and the generation of the key-material (user specific).
- Required computational capability of users hardware.
  The complexity of the hardware required for users to employ the system (e.g. a simple smartcard, or a Pentium processor).
- Required datastorage capability of users hardware.
  The required capability for storing the system parameters, public keys and private keys.
- Required time for creating a digital signature.
- Required time for verifying a digital signature.
- Required space for storing a digital signature (i.e. the size of a signature).

2 Summary of this Report

2.1 What is (not) the topic of this report?

This report describes and analyses a specific digital signature scheme: the relatively new digital signature scheme based on Elliptic Curves are explained and compared with two other, more common, systems (RSA, Classic Discrete Log). To employ digital signatures not only cryptographic techniques are required, but also other security techniques (e.g. hashing, timestamping, smartcards). Moreover, also non-technical measures (e.g. procedural, organizational, legal) and services (e.g. TTPs) are required. These are not discussed in this report.

2.2 The working of the studied systems

RSA is the oldest system and its security is based on the difficulty of factorizing large numbers: given the product of two (large) primes, it is difficult to determine the primenumbers. Using this fact it is easy to create a situation in which determining "square roots" is difficult unless one knows the primenumbers. The squaring operation (the inverse of taking square roots) however, is not difficult to perform, even if one does not know the primenumbers. In the context of RSA the digital signature of a messages consists of its "square root": a "proof" that one knows the primenumbers.

The security of the Classical Discrete Log system is based on the difficulty of determining logarithms in "classical" groups (multiplicative groups of finite fields): given a unknown power of, say, 2 (the "base" element), it is difficult to determine the accompanying exponent (the logarithm). In the context of Classical Discrete Log systems, the private key consists of the exponent and the public key consists of the power of 2. Moreover, the digital signature consists of a "proof" that one knows the secret exponent. As such "proofs" can be constructed in various ways, there exist various digital signature systems based on the classical discrete log problem, one of the well-known is the Digital Signature Algorithm (DSA) a standard of US government.

The difficulty of the discrete logarithm problem also occurs in other "groups" (i.e. situations where one can multiply and hence take powers). More in particular, this problem is difficult in the groups that are constructed by defining a "multiplication" on an Elliptic Curve (think of a curve in the plane). The security of Elliptic Discrete Log systems is based on the difficulty of determining logarithms in this type of groups. Also in this context, the private key consists of a secret exponent and the public key consists of the resulting power of a base element. The digital signature consists of a proof that one "knows" the secret exponent. All digital signature schemes based on the classical discrete log problem are "translatable" to the elliptic discrete log problem. The "translation" of DSA is usually called ECDSA.

2.3 How the three systems are compared

As a starting point in the comparison of the three systems, the conventional digital signature systems RSA and DSA are considered in a relatively high security setting (modulus 1024 bit), which is then compared with ECDSA (the elliptic variant of DSA) of equivalent security (160 bit modulus). In other words, the first aspect mentioned under point four above is fixed throughout the comparison.

However, we remark that this comparison of security is based on the known cryptanalytic attacks on these systems. The advantage of ECDSA is that no "good" (i.e. efficient) attacks are known: known techniques ("index- en factor-bases") used for relatively fast attacks on the classical discrete log problem and the factoring problem do not seem to generalize to the discrete log problem in elliptic curves. Although experts are confident that no good (generalized) attacks exist in general, one should also realize that the study of elliptic curves at this specific point is quite young. As an illustration, in the recent past certain very specific subclasses of elliptic curves were discovered where very efficient attacks do exist. However, such elliptic curves are easy to detect and hence to avoid.

2.3.1 Comparing the complexity of setting up the system

The generation of the system parameters (finding "good" primenumbers) with RSA en DSA is simpler than the generation of the system parameters with ECDSA (finding a "good" curve; the first generation is in the order of minutes, the second in the order of hours.

The generation of key-material in the RSA scheme more or less coincides with the generation of the system parameters; the generation of key-material in the RSA and ECDSA schemes is of negligible complexity.

2.3.2 Comparing the required computational capability of users hardware

The required computational capability of user's hardware with RSA and DSA is considerably larger than required with ECDSA. Although a mathematical coprocessor is required when implementing any of the three signature systems on a smartcard, the coprocessor with ECDSA is much simpler (and hence cheaper).

2.3.3 Comparing the required datastorage capability of users hardware

The required datastorage capability of users hardware with RSA and DSA is considerably larger than required with ECDSA: for a typical ECDSA public key only 160 bit is required (20 characters) while for a typical RSA and DSA public key 1024 bit is required (128 characters). Also this aspect gives ECDSA a considerable advantage above RSA and DSA in smartcard applications where storage capability is usually low.

2.3.4 Comparing the required time for creating a digital signature

As already mentioned above, the use of RSA and DSA requires more computational capability of user's hardware than is the case with ECDSA. If one takes the computational capability required for the use of RSA and DSA as a starting point, then the creation of an ECDSA digital signature takes least time: the creation of a DSA signature takes approximately 4 times as much time and the creation of a RSA signature takes approximately 6 times as much time.

2.3.5 Comparing the required time for verifying a digital signature

As already mentioned above, the use of RSA and DSA requires more computational capability of user's hardware than is the case with ECDSA. If one takes the computational capability required for the use of RSA and DSA as a starting point, then the creation of an RSA digital signature takes least time: the creation of a ECDSA signature takes approximately 7 times as much time and the creation of a DSA signature takes approximately 28 times as much time.

2.3.6 Comparing the required space for storing a digital signature

For storing a DSA or ECDSA signature 320 bit (40 characters) storage is required, for storing an RSA signature 1024 bit (128 characters) storage is required.

2.4 The conclusion of the comparison

In the table below, the results of the comparison are listed: ▼ stands for a negative point, while ▲ stands for positive point.

	ECDSA	RSA	DSA
Security	Chosen to be the same	Chosen to be the same	Chosen to be the same
Complexity System Set up	▼ (< Hours)	▲ (<Minutes)	▲ (< Minutes)
Users computational capacity	▲ (simpler chip)	▼ (more complex chip)	▼ (more complex chip)
Users storage capacity	▲ (160 bit per key)	▼ (1024 bit per key)	▼ (1024 bit per key)
Creation time digital signature	▲	▼ (6 times ECDSA)	▼ (4 times ECDSA)
Verification time digital signature	▼ (7 times RSA)	▲	▼ (28 times RSA)
Size digital signature	▲ (320 bit per sig.)	▼ (1024 bit per sig.)	▲ (320 bit per sig.)

One can conclude that ECDSA has big advantages over RSA and DSA in applications where the available computational or datastorage capacity in user's hardware is limited, e.g. in smartcard applications.

However, we remark – once again - that this comparison of security is based on the known cryptanalytic attacks on these systems. The advantage of ECDSA is that no "good" (i.e. efficient) attacks are known. Although experts are confident that no good attacks exist in general, one should also realize that the study of elliptic curves at this specific point is quite young. Confidence building is the fact that the German *Bundesamt für Sicherheit in der Informationstechnik* – an advising / standardizing governmental organization for governmental and civil information security - makes the following statement in a study on digital signature systems: "Aus Sicht des BSI sind folgende Signaturalgorithmen geeignet: [...] DSA-Varianten, basierend auf elliptischen Kurven".

Despite this, it is advisable to adapt to well-known standards (such as IEEE P1363 en ANSI X9.62) when using elliptic curve cryptosystems and – moreover – to keep track of the "state of art in cryptoanalytic techniques". Although the latter advise holds for nearly all cryptographic applications.