KoA
Ontwikkeling

Kiezen op Afstand

Hertellen stemmen

Klant	:	Ministerie van Binnenlandse Zaken en Koninkrijksrelaties (BZK)
Contract Number	:	ECF2651
Offerte Number	:	PN45.6122
Divisie/Project Nummer	:	45.2651
Project Manager	:	Aart van Erven
Lijn Manager	:	Eric Kreupeling
Project/Document referentie	:	45.ECF2651.015.Hertellen stemmen.doc
Uitgifte	:	A(3)
Uitgifte datum	:	27 januari 2004

Status	:	Concept	
Verspreiding	:	KoA Team	
Opgesteld door	:	Wim Kegel Ronald Bruins	Consultant Datum: 27 januari 2004
Goedgekeurd (LogicaCMG)	:	.....	*eg Project Manager* Datum: .....
Goedgekeurd (BZK)	:	.....	*eg Project Manager BZK* Datum: .....

Revisie overzicht

datum	versie	status	auteur
15 januari 2004	A(1)	Eerste versie, gebaseerd op notitie RB	RB, WK
16 januari 2004	A(2)	Naar aanleiding van de eerste versie zijn aan de ontwerpers enkele vragen gesteld. De antwoorden op deze vragen zijn verwerkt in deze versie.	WK
27 januari 2004	A(3)	Antwoorden op vragen BZK verwerkt: • Appendix B toegevoegd met een aantal fragmenten uit de KoA applicatie die encryptie en decryptie regelt; • Verwijzingen naar de SUN documentatie over JCE toegevoegd; • Appendix C toegevoegd met de DTD's uit het TO voor kandidaat-gegevens en een nieuwe DTD voor het export bestand met vercijferde stemmen. • Overige delen voorbeeld uitslag (appendix A) toegevoegd. • Scheidingsteken voor de velden in de ge-encrypte stem is een puntkomma, geen komma zoals in eerdere versies werd gemeld. • Beschrijving van het algoritme voor het bepalen van een vingerafdruk toegevoegd. • Maximale lengte toegevoegd in tabel 1.	WK

Inhoudsopgave

1 Inleiding

1.1 Doel

Eén van de eisen die wordt gesteld aan het systeem voor Kiezen op Afstand (KoA) is dat het mogelijk moet zijn om de uitgebrachte stemmen te hertellen. In het overleg tussen LogicaCMG en de opdrachtgever is vastgesteld dat de interpretatie die aan deze eis wordt gegeven is dat de stemmen in ruwe vorm worden geëxporteerd en dat ze vervolgens buiten de applicatie in een apart programma worden verwerkt en herteld. Dit programma wordt niet ontwikkeld door LogicaCMG omdat het hertellen onafhankelijk van het KoA systeem dient te gebeuren.

Dit document bevat een beschrijving van de manier waarop de stemmen in het KoA systeem worden opgeslagen en van de export die van de stemmen kan worden gemaakt. Deze informatie kan worden gebruikt door een derde partij om het onafhankelijke telprogramma te ontwikkelen.

1.2 Opbouw document

De opbouw van dit document is als volgt:

- Hoofdstuk 2 bevat een beschrijving van het algoritme dat in de KoA applicatie wordt gebruikt om de stemmen op te slaan.
- Hoofdstuk 3 beschrijft het uitvoerformaat van de KoA applicatie en de stappen die nodig zijn om deze uitvoer te verwerken.
- Bijlage A bevat een voorbeeld van het tellingen rapport zoals dat door de KoA applicatie wordt gegenereerd.
- Bijlage B bevat een aantal codefragmenten uit de KoA applicatie die de gebruikte algoritmes implementeren.
- Bijlage C bevat de Document Type Definitions van de XML bestanden.

1.3 Referenties

[1] **Specificatie Functionele Eisen KoA**. Versie D(2), 18/12/2003.
In dit document aangeduid als FO, bevat een beschrijving van het KOA systeem. Met name relevant zijn hoofdstuk 3. dat nadere informatie bevat over de manier waarop het KoA systeem omgaat met sleutels, en hoofdstuk 7, dat de specificatie bevat van de export van de kandidatenlijsten.

[2] **IAIK Java Cryptography Extension** at http://jce.iaik.tugraz.at/products/index.php. De toolkit die in de KoA applicatie wordt gebruikt voor het versleutelen en ontsleutelen van de stemmen. IAIK is een implementatie van JCE 1.3.

[3] **Java Cryptography Extensions (JCE)** at http://java.sun.com/j2se/1.4.2/docs/guide/security/index.html. De interface specificatie (API) van JCE, opgesteld door SUN. De IAIK toolkit 1.3 is een implementatie van JCE.

[4] **45.ECF2651.025.Technisch Ontwerp Applicatie.** Versie C(4)**,** 16 januari 2004.

[5] **Java Security Architecture** at http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html. De interface specificatie voor ondermeer de MessageDigest klasse, opgesteld door Sun, die onder andere gebruikt wordt voor het berekenen van vingerafdrukken in de KoA applicatie.

2Opslaan van stemmen in de KoA applicatie

2.1 Stemmen in het KOA systeem

Stemmen worden in de KoA applicatie uitgebracht met behulp van een kandidaatcode. Een kandidaatcode is een getal van 9 cijfers dat voldoet aan de 11-proef die ook voor SoFi nummers wordt gebruikt. De gegevens over de kandidaten worden aangeleverd door het Ministerie van Binnenlandse zaken in de vorm van een XML bestand. Dit bestand wordt ingelezen, waarbij de KoA applicatie aan iedere kandidaat een vooraf vastgesteld aantal (op dit moment 1000) kandidaatcodes toekent. De reden dat er meerdere codes per kandidaat worden toegekend is dat dit een zekere mate van beveiliging biedt voor het stemmen per telefoon; zie het FO 1.3 voor meer informatie.

De kandidaatcodes worden samen met een volledige kopie van de ingelezen gegevens weer teruggeleverd aan het Ministerie van Binnenlandse Zaken. Het formaat van dit XML bestand is functioneel beschreven in het FO 1.3 , hoofdstuk 7. Het Ministerie gebruikt deze kandidaatcodes om het Overzicht van Kandidatenlijsten te genereren dat aan de kiezers wordt toegestuurd. Op een dergelijk overzicht staat bij iedere kandidaat één van de toegekende codes. De kiezer gebruikt de code op het overzicht om de kandidaat van zijn keuze aan te geven.

Bij het uitbrengen van de stem controleert de KoA applicatie of de kandidaatcode geldig is. Als dat het geval is zoekt de applicatie de gegevens van de kandidaat (zoals achternaam en de naam van de lijst waarop de kandidaat voorkomt) op en voegt deze toe aan de kandidaatcode. Deze gegevens zijn formeel redundant; ze bieden een extra mogelijkheid tot controle op de stem en maken het mogelijk om een uitslag te bepalen ook als de volledige kandidatenlijst zelf niet beschikbaar is.

De toegevoegde gegevens zijn:

Veld	Max. Lengte	Beschrijving
achternaam	80	De achternaam van de kandidaat. Merk op dat de naam diakritische tekens kan bevatten.
voorletters	20	De voorletters van de kandidaat.
positienummer	2	De positie van de kandidaat op de kandidatenlijst.
districtnummer	5	Het districtnummer van de kiezer. Wordt op dit moment niet gebruikt; is toegevoegd om het mogelijk te maken een uitslag per district vast te stellen.
kieslijstnummer	2	Het nummer van de lijst waarop de kandidaat voorkomt.
lijstnaam	80	De naam van de lijst waarop de kandidaat voorkomt. De naam kan leeg zijn. Merk op dat er altijd één lijst is met één enkele kandidaat die de speciale betekenis heeft van “Blanco stem”.
kieskringnummer	2	Het nummer van de kieskring waarin de stem is uitgebracht. Voor het Experiment KoA is dit altijd dezelfde kieskring (12, ’s Gravenhage)

Tabel 1: Toegevoegde gegevens bij een stem

2.2 Het versleutelen van de stem

De stemmen worden vervolgens versleuteld. Hiervoor wordt een combinatie van 3DES en RSA encryptie toegepast. Het RSA sleutelpaar is vast; het is in beheer bij het stembureau dat verantwoordelijk is voor de stemming binnen KoA. Bij het openen van de stemming verstrekt het stembureau de publieke sleutel aan de KoA applicatie; na het sluiten van de stemming verstrekt het stembureau de bijbehorende privé sleutel aan de applicatie om het tellen van de stemmen binnen de applicatie mogelijk te maken. Dit mechanisme garandeert dat de stemmen pas kunnen worden geteld na een opdracht van het daartoe bevoegde stembureau.

Het versleutelen van de stemmen gebeurt als volgt:

1. De stemcode wordt met de toegevoegde gegevens in een tekstbuffer geplaatst waarbij alle velden gescheiden worden door een puntkomma. De tekstbuffer bevat dan de volgende informatie: “kandidaatcode;achternaam;voorletters;positienummer; districtnummer;kieslijstnummer;lijstnaam;kieskringnummer”.
2. Genereer een (3-keyed, 168 bits) 3DES sleutel. Hiervoor wordt de ingebouwde sleutel generator van de IAIK toolkit 1.3 gebruikt.
3. Versleutel de 3DES sleutel met behulp van de publieke sleutel van het RSA sleutelpaar. Er wordt gebruik gemaakt van het CBC algoritme.
4. Genereer een “salt” element (een array van 5 bytes met willekeurige waarden) en plaats dit voor de stem in de tekstbuffer. Deze toevoeging zorgt er voor dat twee overigens identieke stemmen altijd een verschillend record in de database opleveren.
5. Versleutel de stem (met salt) met de 3DES sleutel.
6. Voeg de vercijferde 3DES sleutel (resultaat van stap 2.2) en de versleutelde stem (resultaat van stap 2.2) samen in een byte array met de volgende layout:

Lengte 3DES sleutel (32 bits, 4 bytes).	Versleutelde 3DES sleutel	Lengte versleutelde stem (32 bits, 4 bytes)	Versleutelde stem

De encrypted stem wordt vervolgens in de database opgeslagen. Dat gebeurt in random volgorde om reconstructie op basis van tijd onmogelijk te maken.

De combinatie van RSA en 3DES encryptie is gebruikelijk voor dit soort toepassingen: RSA is een rekenintensief algoritme, en omwille van de performance wordt alleen de 3DES sleutel met dit algoritme vercijferd. Deze oplossing combineert de voordelen van RSA (asymmetrische encryptie) met de snelheid van 3DES.

2.3 Implementatie

Voor alle encryptie wordt de IAIK toolkit 1.3gebruikt. De IAIK is een implementatie van de Java Cryptography Extensions (JCE) 1.3, de specificatie voor encryptie die is opgesteld door SUN. Voor een afzonderlijke applicatie die hertellen mogelijk maakt is het niet noodzakelijk om dezelfde toolkit te gebruiken; een andere toolkit die dezelfde Engine Class implementeert kan ook gebruikt worden.

Het is wel aan te bevelen om gebruik te maken van Java. De Sun documentatie (zie http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html#AppB) bevat de noodzakelijke details en verwijzingen naar de bron van de gebruikte algoritmes, maar gebruik van een implementatie in een andere taal brengt zeker risico's van incompatibiliteit met zich mee.

De sleutels die het stembureau gebruikt worden gegenereerd in een kleine stand-alone Java applicatie. De sleutel wordt vercijferd met een wachtwoord en als een byte array naar disk weggeschreven. Bijlage B.1 bevat een verkort code fragment van de applicatie waarmee dat gebeurt.

3Hertellen

3.1 Het export bestand

Hertellen gebeurt op basis van een export bestand dat gemaakt wordt door het stembureau. Dit is een XML bestand dat er als volgt uitziet:

```
<?xml version="1.0" encoding="UTF-8"?>
<REPORT>
      <GLOBAAL STEMBUREAU="DigitaalStembureau"
            VOORZITTER="Onbekend"
            STATE="Gesloten"
            VERKIEZING="Europese verkiezingen van 2004"
            PERIODE_START="10 Mei 2004 9:00"
            PERIODE_EIND="10 Juni 2004 17:00"
            CURTIME="12-januari-2004 14:36:34">
            <KIESKRINGEN>
                  <KIESKRING NUMMER="12" NAAM="'s-Gravenhage"/>
            </KIESKRINGEN>
      </GLOBAAL>
      <TABLE NAME="ENCRYPTEDESB">
            <ROW STEMNUMMER="483835783413760"
STEM="00000040382351488488F776A87E758485612CC10CF455AD448BE4E9BE187F42004C147571F0AAD
F8E96382657F468FE7E0E2B899D7C961CE8B3EDEFF2B873E207CA64A6000000507C29B0217296AA9CE529
106ECB7E36BFE4D0682D4D453BA8C54DBF1CA22EF57D050953913F31F43F6CFA40904E179A820D617699C
BF21A216AEA6CB4DA6E2D88ECEC45CD4660E9D0E05CFE52800288D8"/>
   </TABLE>
</REPORT>
```

De kop van het bestand bevat de naam van het stembureau, de naam van het lid van het stembureau dat de opdracht tot de export heeft gegeven, de huidige toestand van het KoA systeem (zal meestal “gesloten” zijn, maar kan ook zijn “stembus gelicht”, wat aangeeft dat er in het KoA systeem al een telling van de stemming heeft plaats gevonden), de officiële start- en eindtijd van de stemming, en de datum en tijd waarop de export heeft plaats gevonden. Dit wordt gevolgd door een opsomming van alle kieskringen die in het systeem zijn gedefinieerd; dit zal er bij het KoA experiment slechts één zijn, maar voor een algemene oplossing dient rekening te worden gehouden met meerdere kieskringen. Voor iedere kieskring wordt nummer en naam vermeld.

Naam van het stembureau, aanduiding van de verkiezing, stemperiode en de kieskring zijn van belang voor de hertelling omdat deze gegevens op het rapport met het resultaat van de telling dienen voor te komen. De overige gegevens zijn alleen van informatief van aard.

Vervolgens bevat het bestand de inhoud van de versleutelde stembus; de naam van de tabel in de KoA database is ENCRYPTEDESB. Voor ieder record uit deze tabel wordt een ROW element geschreven met als attributen het stemnummer en de versleutelde stem. Het stemnummer (de naam is ongelukkig gekozen) is een random getal dat binnen de KoA applicatie gebruikt wordt om de stemmen op te sorteren voor ze worden ontcijferd; het heeft verder geen inhoudelijke betekenis.

Het byte array met de stem wordt hexadecimaal weergegeven, dus met twee tekens per byte. De stem uit het voorbeeld dient dus als volgt te worden geïnterpreteerd:

00 00 00 40	De lengte van de 3DES sleutel. Na conversie naar een 32 bits integer representeert dit de waarde 64.
38 23 51 48 84 88 F7 76 A8 7E 75 84 85 61 2C C1 0C F4 55 AD 44 8B E4 E9 BE 18 7F 42 00 4C 14 75 71 F0 AA DF 8E 96 38 26 57 F4 68 FE 7E 0E 2B 89 9D 7C 96 1C E8 B3 ED EF F2 B8 73 E2 07 CA 64 A6	De 3DES sleutel, vercijferd met RSA. 128 tekens die na conversie 64 bytes beslaan.
00 00 00 50	De lengte van de versleutelde salt+stem. Na conversie naar een 32 bits integer representeert dit de waarde 80.
7C 29 B0 21 72 96 AA 9C E5 29 10 6E CB 7E 36 BF E4 D0 68 2D 4D 45 3B A8 C5 4D BF 1C A2 2E F5 7D 05 09 53 91 3F 31 F4 3F 6C FA 40 90 4E 17 9A 82 0D 61 76 99 CB F2 1A 21 6A EA 6C B4 DA 6E 2D 88 EC EC 45 CD 46 60 E9 D0 E0 5C FE 52 80 02 88 D8	De stem, vercijferd met de 3DES sleutel. 160 tekens die na conversie 80 bytes opleveren.

Tabel 2: Interpretatie versleutelde stem

3.2 Het ontcijferen van de stemmen

Voor het ontcijferen is behalve de export zelf ook de geheime sleutel van het RSA sleutelpaar van het stembureau nodig.

Op iedere versleutelde stem dient het volgende algoritme te worden toegepast (het spiegelbeeld van het algoritme beschreven in § 2.2)

1. Converteer de hexadecimale string terug naar een byte array (dat dus de helft van de lengte van de hexadecimale string heeft).
2. Bepaal de lengte van de 3DES sleutel (bytes 1-4 van het byte) en licht met behulp van dit aantal de 3DES sleutel uit het byte array.
3. Ontsleutel de 3DES sleutel met behulp van de geheime RSA sleutel.
4. Bepaal de lengte van het “salt” element en de stem (de 4 bytes na de 3DES sleutel), en licht dit element uit het tekstblok. Controleer de totale lengte.
5. Ontsleutel de stem met behulp van de 3DES sleutel verkregen in stap 3.2.
6. Negeer de eerste 5 bytes van de ontsleutelde stem.
7. Splits de rest van de tekstbuffer op de puntkomma’s. Dit levert de stemcode en de aangehaakte gegevens beschreven in Tabel 1 op.

De code die stappen 1-6 van dit algoritme implementeert binnen de KoA applicatie is te vinden in § B.2.

Voor een goede verwerking van de stemmen is het wenselijk dat de tel applicatie ook de beschikking heeft op de export van de kandidatenlijsten zoals beschreven in hoofdstuk 2. Indien dat het geval is kunnen de stemmen als volgt worden verwerkt:

1. Zoek de kandidaatcode op in het overzicht van kandidatenlijsten.
2. Verifieer dat de naam van de kandidaat, voorletters, lijstnaam en nummer, en de positie op de lijst correct zijn;

3. Verifieer dat de kieskring voorkomt in de kop van het export bestand en overeenkomt met de kieskring van de kandidaat (kandidatenlijsten kunnen per kieskring verschillend zijn).
4. Verhoog de teller van het aantal stemmen voor de gevonden kandidaat.

Als dit proces is voltooid kan een rapport worden opgemaakt met het resultaat van de telling. Een voorbeeld van een dergelijk rapport is te vinden in bijlage A.

3.3 Controleren van de vingerafdruk

De KoA applicatie maakt op vooraf bepaalde momenten een vingerafdruk ("digest") van bepaalde gegevens in de database (zie Functioneel ontwerp, § 8.2). Hiermee kan het stembureau controleren of er geen ongeoorloofde aanpassingen van de gegevens zijn geweest. De vingerafdrukken worden in hexadecimale vorm opgeslagen in de auditlog en kunnen door het stembureau worden afgedrukt. Eén van de vingerafdrukken heeft betrekking op de versleutelde stembus op het moment van sluiten van de stemming.

Door de vingerafdruk in de hertel applicatie opnieuw te bepalen kan worden gecontroleerd of de inhoud van het ontvangen bestand overeenstemt met de inhoud van de stembus op moment van sluiten. Vingerafdrukken worden gemaakt met behulp van de MessageDigest class uit de Java Security API 1.3.

De vingerafdruk dient te worden bepaald over de binaire representatie van de versleutelde stemmen (het resultaat van stap 3.2 in § 3.2). Het algoritme daarvoor is als volgt:

1. Creëer een MessageDigest object met MD5 als algoritme.
2. Sorteer de binaire stemmen op stemnummer (zie § 3.1).
3. Doe achtereenvolgens een update() van het MessageDigest object met alle binaire stemmen.
4. Bereken de waarde van de MessageDigest met de digest() methode van de MessageDigest klasse.
5. Converteer de resulterende waarde byte voor byte naar een hexadecimale waarde (waarbij indien nodig een voorloop nul wordt geplaatst, dus de waarde 0xF wordt weergegeven als "0F").

De KoA implementatie van dit algoritme is te vinden in § B.3.

Bijlage A Voorbeeld tellingen rapport

Onderstaand voorbeeld is een voorbeeld van een rapport gemaakt de module uit het KoA systeem. Merk op dat de gegevens over het aantal kiesgerechtigden niet uit de stembus komen maar uit het Kiesregister, en dus niet beschikbaar zullen zijn in de tel applicatie.

Resultaat van de stemming

Kieskring: 12 - 's-Gravenhage

Stembureau: DigitaalStembureau

Stemperiode: 10 Mei 2004 9:00 - 10 Juni 2004 17:00

Verkiezing: Europese verkiezingen van 2004

Starttijd stemming: 12-January-2004 12:50

Eindtijd stemming: 12-January-2004 14:30

Aantal kiesgerechtigden: 999

Aantal kiesgerechtigden dat heeft gestemd: 2

Aantal kiesgerechtigden dat niet heeft gestemd: 997

Lijstnummer: 1

Lijstnaam: Europese Kleurenpartij (EKP)
Aantal stemmen: 0

1. W.F. Aźúùr	0
2. Ĉ. de Parelgrijs	0
3. Ý.M. Blauŵ	0
4. G.M.H. Kersen-Rood	0
5. L. Crème	0
6. B.C. Helblaûw	0
7. K. Kardinaalrood	0
8. Ô.L. van den Čariben Diepblaüw tot Pimpelpaars	0
9. G.M. Zwart	0
10. W.R.Î. Mokka-Brũin	0
11. D. Violet	0
12. Ş.H.Ú. Goudǵeel	0
13. D. Groen	0
14. E.H. Ċhocobrŭin	0
15. P.J. Ĺila	0
16. Ò.M. Zonneĝeel	0
17. G.M. Zwart	0

18. Ë. van Vuűrroos	0
19. P.Ç. Fućhsia	0
20. Š. van den Citroĕnğeel	0
21. A. Grasgroen	0
22. G.K.L. Rose	0
23. R.T. Tůrqūoise	0
24. F. Geel	0
25. D.D. Dennengroen	0
26. B.W.V. de Fel Oranje	0
27. A.Ŝ. Śienna	0
28. B. Hemėlsblaɥw	0
29. M. Ólijf	0
30. G. Ävondġrijs	0

Lijstnummer: 2

Lijstnaam: Planten voor het Volk (P.v.h.V)
Aantal stemmen: 0

1. G. Roōs	0
2. C.J. van Chŕÿsant	0
3. V. Ťulp	0
4. S.A. Lelie	0
5. H. Conifeeř	0
6. A.M. Dopĥeíde	0
7. J.G. Vlijtig Liesje	0
8. F. Ânjeɽ	0
9. D.K. Gerbera	0
10. Ŋ.M. Freśia	0
11. T.H.K.L. den Aronskelk	0
12. W.J. barones van Rubber-van den Boom	0
13. L.H. Örchidee	0
14. N. Iris	0
15. Z.G.L. Guldenroede	0
16. W.K.L.H. Fućhsia	0
17. B. de Lamŝoor	0
18. R.N. Hęrtshooì	0

19. J.K.L. Ašter	0
20. G.F. van der Zonnēbloem	0
21. D. Ãmaryllis	0
22. D.Ţ. Narciş	0
23. P. Halskruid	0
24. M.M. van Ģladiool	0
25. T.Y.K. Begonia	0
26. Ì.M. Ĥortensïa	0
27. H. Kaaps Viooltje	0
28. J.C.Ų. Kerstster	0
29. F. Yuçĉa	0
30. R.R. Ġeranium	0
31. C. Viooĺtje	0

Lijstnummer: 3

Lijstnaam: EUROPESE WEERMANNEN
Aantal stemmen: 1

1. E. Ŵolk	0
2. C.G. de Sneeuw	0
3. H.J.K.L. van der Kou	0
4. G.F. Hągel	0
5. L.G. van der Zonneschijn	0
6. N. Bui	0
7. D.S. Ŗégen	1
8. R. Hitte	0

Lijstnummer: 4

Lijstnaam: Anonieme Politici
Aantal stemmen: 0

1. M.W.P. Vilt	0
2. Ŷ.M. vån Zijden	0
3. M.L. Velours	0
4. G.A. Ļinnen	0
5. C. de Leer	0
6. R. Spijķer	0

7. L.J. Sātijn	0
8. A. van der Katoenen	0
9. W. Suede	0

Lijstnummer: 5

Lijstnaam: Europese Dierenalliantie (EDA)
Aantal stemmen: 0

1. K.L. de Ōlifănt	0
2. W. Ŏrka	0
3. L.R. Tijger	0
4. Ě.Ė. Leguaań	0
5. E.G. van der Ēgel	0
6. J.D.Ğ.Ř. van den Stokstaartje	0
7. Ő.P.D. Ĵacht-Luipaard	0
8. M. Gaželle	0
9. B.V. IJs Bèer	0
10. Z.A. Zebrã	0
11. B. Koniĵn	0
12. C.Õ.P. Koala	0
13. W.P. Leeuw	0
14. U. Walvĩs	0
15. L.J.K.M. Ăardvarkeñ	0
16. R. Hond	0
17. E.Ê. van Muis	0
18. A.Ÿ. Ans	0
19. W.Ń. Kĩkker	0
20. G.J. Edel-Hêrt	0
21. C. de Paard	0
22. Ž. Okàpi	0
23. B.O. Źeeschįldpad	0

Lijstnummer: 6

Lijstnaam: VKP (gecombineerd met lijst 7)
Aantal stemmen: 1

1. C. Mieriks-van Wortel	0

2. T.L.Ù. Âô¿emarijn	0
3. È.T. Basilièum	0
4. H.Ā de Marjòleiņ	1
5. P.L. da Salie	0
6. A. Oregaòó	0
7. A.S.M. Dille	0
8. Á.V. Bonekruid	0
9. R.P. Laurier	0
10. S. van der Vlierbessen	0
11. S. Tijm	0
12. H.P. Selderij	0
13. G. Citröen-Melisse	0
14. K.J.Ê. Peþerselie	0
15. K. Koriander	0
16. F.Ĩ.D. Bies Lôok	0

Lijstnummer: 7

Lijstnaam: Lijst Smaak (gecombineerd met lijst 6)
Aantal stemmen: 0

1. M. Pittig	0
2. J.H.Ĩ. ¯oet	0
3. B.Ò.V.D. de Zout	0
4. S. Flauw	0
5. F.J. van der Zuur	0
6. T.P. le Demi-Seċ	0
7. R.¥. Bitter	0

Lijstnummer: 8

Lijstnaam: Partij van de Sport (PVDS)
Aantal stemmen: 0

1. A Boogschieten	0
2. Ý.T. de Hàndbal	0
3. K.L. Ĝolf	0
4. B.M. Åtletiek	0
5. R.T. Bowling	0

6. S. Judõ	0
7. V.N. Tennis	0
8. É. Karten	0
9. F.Æ. Voetbal	0
10. Z.S.G.Í. van Zwëmmen	0
11. H. Sjõelen	0
12. L.W. van der Ïárts	0

Lijstnummer: 9

Lijstnaam:
Aantal stemmen: 0

1. A Blaat	0

Aantal blanco stemmen

1. Blanco	0

Bijlage B Code voorbeelden

Deze bijlage bevat een aantal fragmenten uit de KoA code die de encryptie algoritmen kunnen verduidelijken. In de code zijn ter wille van de leesbaarheid hier en daar wat details weggelaten die met afhandeling van fouten binnen de KoA applicatie te maken hebben.

B.1 Het genereren van het sleutelpaar: de GenerateKeyPair klasse

Deze klasse implementeert het aanmaken, opslaan en teruglezen van een RSA sleutelpaar.

```
/** ------------------------------------------------------------------------
 *
 *   com.logicacmg.koa.security.GenerateKeyPair.java
 *
 * ------------------------------------------------------------------------
 *
 *  (c) 2003  Ministerie van Binnenlandse Zaken en Koninkrijkrelaties
 *
 */

package com.logicacmg.koa.security;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.GeneralSecurityException;
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;
```

…

```
/**
 * main application for the keypair generation
 *
 *
 */
public class GenerateKeyPair {

        public static final int PRIVATE_KEY = Cipher.PRIVATE_KEY;
        public static final int PUBLIC_KEY = Cipher.PUBLIC_KEY;
```

Gebruikte algoritmes: het algoritme voor het sleutelpaar is RSA met een sleutel lengte van 512. In de KOA applicatie is de sleutellengte een configureerbare parameter (property). Het algoritme voor de opslag van de sleutel in een bestand is password based encryption met gebruik van MD5 en DES.

```
        private static final String KEYPAIR_GENERATOR_ALGORITHM = "RSA";
```

```
        private int KEYPAIR_KEY_LENGTH = 512;
        private static final String KEY_ENCRIPTION_AKGORITHM = "PBEWithMD5AndDES";
        private static final byte[] SALT = { (byte)0x19, (byte)0x36, (byte)0x78, (byte)0x99,
                                             (byte)0x52, (byte)0x3e, (byte)0xea, (byte)0xf2 };

        // the wrapped key pair
        private KeyPair keyPair;

        /**
         * Zero argument constructor for generation a key pair
         * There will be a public key and a private key available
         *
         * @throws GeneralSecurityException
     *     This exception will be thrown when there is a problem
     *     while generating a key pair
         */
        public GenerateKeyPair() throws GeneralSecurityException
        {
```

…

```
                KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance(KEYPAIR_GENERATOR_ALGORITHM);
                keyPairGen.initialize(KEYPAIR_KEY_LENGTH);
                keyPair = keyPairGen.generateKeyPair();
        }
```

Het ontcijferen van een sleutelpaar. Omdat van een sleutelpaar altijd of de privé sleutel wordt gebruikt (tijdens de stemopneming) of de publieke sleutel (tijdens het stemmen) wordt altijd maar één van de twee ontcijferd en ter beschikking gesteld.

```
        /**
         * Constructor for decryption of a key.
         * Only the public of private key of the <code>keyType</code> will be available the other will return null
         *
         * @param password    The password used for decryption
         * @param criptKey              A stream with the encrypted key
         * @param keyType     The type of the key public (</code PUBLIC_KEY>) or private (</code PRIVATE_KEY>)
         * @throws GeneralSecurityException This exception will be thrown when there is a problem with decryption
         */
        public GenerateKeyPair(String password, InputStream cryptKey, int keyType) throws GeneralSecurityException,
IOException
        {
                PBEParameterSpec paramSpec = new PBEParameterSpec(SALT, 20);
                PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray());
                SecretKeyFactory kf = SecretKeyFactory.getInstance (KEY_ENCRIPTION_AKGORITHM);
                SecretKey passwordKey = kf.generateSecret(keySpec);

                Cipher cipher = Cipher.getInstance(KEY_ENCRIPTION_AKGORITHM);
                cipher.init(Cipher.UNWRAP_MODE, passwordKey, paramSpec);

                byte[] dummy = new byte[128];
                int length;
                ByteArrayOutputStream byteArray = new ByteArrayOutputStream();
                while ((length = cryptKey.read(dummy)) != -1) {
                        byteArray.write(dummy, 0, length);
                }

                if (keyType == PRIVATE_KEY) {
                        Key unwrappedKey = cipher.unwrap(byteArray.toByteArray(), KEYPAIR_GENERATOR_ALGORITHM,
keyType);
                        keyPair = new KeyPair(null, (PrivateKey) unwrappedKey);
                }
                else if (keyType == PUBLIC_KEY) {
                        Key unwrappedKey = cipher.unwrap(byteArray.toByteArray(), KEYPAIR_GENERATOR_ALGORITHM,
keyType);
                        keyPair = new KeyPair((PublicKey) unwrappedKey, null);
                }
                else {
                        throw new InvalidKeyException("criptKey does not represent a wrapped key of type keyType");
                }
```

```
	}
```

Het ophalen van de privé sleutel uit het sleutelpaar:

```
	/**
	 * Returns a private key if available otherwise it returns null
	 *
	 * @return PrivateKey private key
	 */
	public PrivateKey getPrivateKey()
	{
		return keyPair.getPrivate();
	}
```

Het ophalen van de publieke sleutel uit het sleutelpaar:

```
	/**
	 * Returns a public key if available otherwise it returns null
	 *
	 * @return PublicKey public key
	 */
	public PublicKey getPublicKey()
	{
		return keyPair.getPublic();
	}
```

Het vercijferen van een publieke sleutel met een wachtwoord:

```
	/**
	 * Encrypt the public key with a password and returns the encypted key with the outputstream
	 *
	 * @param password	The password used for encryption
	 * @param output		The encrypted key
	 * @throws IOException		This exception will be thrown when there is a problem with the stream
	 * @throws GeneralSecurityException
	 *					This exception will be thrown when there is a problem with the decription
	 */
	public void getPublicKeyEncrypt(String password, OutputStream output) throws GeneralSecurityException, IOException
	{
		if (keyPair.getPublic() != null)
		{
			getKeyEncrypt(password, output, keyPair.getPublic());
		}
		else {
			throw new GeneralSecurityException ("This key does not excist");
		}
	}
```

Het vercijferen van een privé sleutel met een wachtwoord:

```
	/**
	 * Encrypt the private key with a password and returns the encypted key with the outputstream
	 *
	 * @param password	The password used for encryption
	 * @param output		The encrypted key
	 * @throws GeneralSecurityException
	 *					This exception will be thrown when there is a problem getting the encrypted
private key
	 */
	public void getPrivateKeyEncrypt (String password, OutputStream output) throws GeneralSecurityException,
IOException
	{
		if (keyPair.getPrivate() != null) {
			getKeyEncrypt(password, output, keyPair.getPrivate());
		}
		else {
			throw new GeneralSecurityException ("This key does not excist");
		}
	}
```

Deze methode implementeert het eigenlijke vercijferen van de privé sleutel of de publieke sleutel van een paar met een wachtwoord en de opslag ervan:

```
	/**
```

```
         * Encrypt a private key or public key with a password and returns the encypted key with the outputstream
         *
         * @param password    The password used for encryption
         * @param output                The encrypted key
         * @param key           The public or private key
         * @throws GeneralSecurityException This exception will be thrown when there is a getting the key
         */
        private void getKeyEncrypt(String password, OutputStream output, Key key) throws GeneralSecurityException,
IOException
        {
                // create a secrate key whit the password
                PBEParameterSpec paramSpec = new PBEParameterSpec(SALT, 20);
                PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray());
                SecretKeyFactory kf = SecretKeyFactory.getInstance (KEY_ENCRIPTION_AKGORITHM);
                SecretKey passwordKey = kf.generateSecret(keySpec);

                // encript the key
                Cipher c = Cipher.getInstance(KEY_ENCRIPTION_AKGORITHM);
                c.init(Cipher.WRAP_MODE, passwordKey, paramSpec);
                byte[] wrappedKey = c.wrap(key);

                // write the key to the stream
                output.write(wrappedKey);
        }
}
```

B.2 Encryptie en decryptie van stemmen: de KOAEncryptionUtil klasse

Onderstaande code bevat een gedeelte van de KOAEncryptionUtil klasse, waarin de encryptie en decryptie van stemmen is ondergebracht. Code die betrekking heeft op foutafhandeling en debug code voor performance metingen is weggelaten.

```
/** ---------------------------------------------------------------------
 *
 *   com.logicacmg.koa.security.KOAEncryptionUtil.java
 *
 * ---------------------------------------------------------------------
 *
 *  (c) 2003  Ministerie van Binnenlandse Zaken en Koninkrijkrelaties
 *
 */

package com.logicacmg.koa.security;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
…
import java.security.Security;
import com.logicacmg.koa.exception.KOAException;
import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
```

```
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;

import com.logicacmg.koa.constants.ErrorConstants;

/**
 * Utility class that contains does the encryption actions
 * for the KOA System.
 *
 */
public class KOAEncryptionUtil {

	private final static String PUBLIC_KEY_CRYPT_ALGO = "RSA";
	private final static String SECRET_KEY_CRYPT_ALGO = "DESede/CBC/PKCS5Padding";
	private final static String SECRET_KEY_GENERATOR_ALGO = "DESede";
	private final static int SECRET_KEY_LENGTH = 168;

	private static final byte[] SALT = { (byte) 0x19, (byte)0x36, (byte)0x78, (byte)0x99,
	                                     (byte)0x52, (byte)0x3e, (byte)0xea, (byte)0xf2 };

	private static KeyGenerator xKeyGen;

	private final static String encryptionKey = "KOAKey";

	static {
		java.security.Security.addProvider (new iaik.security.provider.IAIK());
	}
```

…

```
	/**
	 * encrypts the String with the public key and returns the encrtypted data
	 * as a byte array
	 *
	 * @param xRSAKey    the public key used te crypt the data
	 * @param sData      the data that is cryped
	 * @return byte[]    the cryped data
	 */
	public static byte[] encrypt(PublicKey xRSAKey, String sData) throws KOAException{
		try {
			// generate triple des key
			if (xKeyGen      == null) {
				xKeyGen = KeyGenerator.getInstance(SECRET_KEY_GENERATOR_ALGO);
				xKeyGen.init(SECRET_KEY_LENGTH);
			}
			SecretKey xDesKey = xKeyGen.generateKey();

			// generete salt + data
			String sDataToCript = RandomGenerator.getInstance().getSalt() + sData;

			// cript key with rsa
			Cipher xCipherKey = Cipher.getInstance(PUBLIC_KEY_CRYPT_ALGO);
			xCipherKey.init(Cipher.WRAP_MODE, xRSAKey);
			byte[] baCriptKey = xCipherKey.wrap(xDesKey);

			//cript salt + tekst
			Cipher xCipherData = Cipher.getInstance(SECRET_KEY_CRYPT_ALGO);
			IvParameterSpec xParamSpec = new IvParameterSpec(SALT);
			xCipherData.init(Cipher.ENCRYPT_MODE, xDesKey, xParamSpec);
			byte[] baCriptData = xCipherData.doFinal(sDataToCript.getBytes());

			// stream triple des key + cript tekst
			ByteArrayOutputStream xByteArrayOutput = new ByteArrayOutputStream();
			DataOutputStream xDataOutput = new DataOutputStream(xByteArrayOutput);
			// write length of key
			xDataOutput.writeInt(baCriptKey.length);
			// write key
			xDataOutput.write(baCriptKey, 0, baCriptKey.length);
```

```
                        // write length of key
                        xDataOutput.writeInt(baCriptData.length);
                        // write cript data
                        xDataOutput.write(baCriptData, 0, baCriptData.length);
                        // close streams
                        xDataOutput.close();
                        xByteArrayOutput.close();
                        // return byte array (length criptkey + criptkey + length cript data + cript data)
                        return xByteArrayOutput.toByteArray();
                }
                catch (NoSuchAlgorithmException nsae) {
```

(code die een aantal fouten uit de encryptie bibliotheek opvangt en omzet naar KoA fouten weggelaten).

```
                }
        }
```

Het ontcijferen van de stem:

```
        /**
         * decrypts the encryped byte aray with the private key and returns decrtypted
         * data as a String
         *
         * @param xRSAKey the private key used te decrypt the data
         * @param encryptedBytes the encryped bytearray
         * @return String the decryped data
         */
        public static String decrypt(PrivateKey xRSAKey, byte[] encryptedBytes) throws KOAException {

                try {
```

(debug code weggelaten)

```
                        // create stream from encrypted bytes
                        ByteArrayInputStream xByteArrayInput = new ByteArrayInputStream(encryptedBytes);
                        DataInputStream xDataInput = new DataInputStream(xByteArrayInput);

                        // decrypt key with rsa
                        int iKeyLenght = xDataInput.readInt();
                        byte[] baCriptKey = new byte[iKeyLenght];
                        xDataInput.read(baCriptKey);
                        Cipher xCipherKey = Cipher.getInstance(PUBLIC_KEY_CRYPT_ALGO);
```

(debug code weggelaten)

```
                        xCipherKey.init(Cipher.UNWRAP_MODE, xRSAKey);
                        SecretKey xDesKey =
                                (SecretKey)xCipherKey.unwrap(baCriptKey, SECRET_KEY_GENERATOR_ALGO,
Cipher.SECRET_KEY);
```

(debug code weggelaten)

```
                        // decript salt + data
                        Cipher xCipherData = Cipher.getInstance(SECRET_KEY_CRYPT_ALGO);
                        IvParameterSpec xParamSpec = new IvParameterSpec(SALT);
                        xCipherData.init(Cipher.DECRYPT_MODE, xDesKey, xParamSpec);
```

(debug code weggelaten)

```
                        int iDataLenght = xDataInput.readInt();
                        byte[] baCriptData = new byte[iDataLenght];
                        xDataInput.read(baCriptData);
                        byte[] baDecryptedData = xCipherData.doFinal(baCriptData);
```

(debug code weggelaten)

```
                        // remove salt from decript tekst.
                        return new String(baDecryptedData).substring(RandomGenerator.SALT_LENGTE);
                }
```

```
            catch (NoSuchAlgorithmException nsae) {
        (code voor afhandelen van foutmeldingen uit cryptobibliotheek weggelaten)
        }
}
```

B.3 Het maken van een vingerafdruk

Vingerafdrukken worden in de KoA applicatie gemaakt met de MessageDigest klasse uit de Java Security API[1]. Het gebruikte algoritme is MD5.

Onderstaand de methode die gebruikt wordt om een vingerafdruk te genereren. De methode maakt deel uit van de KOAEncryptionUtil klasse die is besproken in § B.2. De methode is algemeen van opzet: bij de aanroep wordt opgegeven om welke databasetabel het gaat en welke kolommen moeten worden gebruikt. Voor de vercijferde stemmen is er slechts één kolom van toepassing.

```
/**
         * Creates the fingerprint of the provided database table. Special method for tables containing blobs
         *
         * @param datasourceName The datasource to get the fingerprint for
         * @param schemaName The schema name to get the fingerprint for
         * @param tableName The table name to get the fingerprint for
         * @param columns The columns to use in the creation of the fingerprint
         * @param sortKey the key to sort the rows on
         *
         * @return byte [] containing the fingerprint
         *
         * @throws KOAException when something goes wrong during the creation of the fingerprint.
         *
         */
        public static byte[] getBLOBFingerPrint(String datasourceName, String schemaName, String tableName,
                                                String[] columns, String sortKey) throws KOAException {
```

Initialiseer de fingerprint:

```
            DBUtils db = new DBUtils(datasourceName);
            Connection conn = db.getConnection();
            Statement stmt = null;
            BLOBResultSet rSet = null;
            FingerPrint fPrint = new FingerPrint(encryptionKey);

            try {
```

Haal de gegevens op:

```
                stmt = conn.createStatement();
                String cols;
                if (columns.length > 0) {
                    rSet = new BLOBResultSet(datasourceName, schemaName, tableName, sortKey, columns);
```

…

Werk de fingerprint bij voor alle rijen:

```
                while (rSet.next()) {
                    // System.out.println("rst.next");
                    for (int i=1; i <= columns.length; ++i) {
                        byte[] tmp = null;
```

…

[1] Zie http://java.sun.com/j2se/1.4.2/docs/ guide/security/CryptoSpec.html#MessageDigest

```
                                        /* rows can be fetched as a byte[] */
                                        tmp = rSet.getBytes(columns[i-1]);

                                        if (tmp != null) {
                                                fPrint.update(tmp);
                                        } else {
                                                // No value
                                        }
                                }
                        }


                } else {
```

…

```
                }
        }
        catch (Exception e) {
```

… error handling

```
        }
        finally
        {
                rSet.close();
        }
```

Converteer naar een hexadecimale string:

```
        byte[] bDigest = null;
        bDigest = fPrint.getDigest();
        StringBuffer sb = new StringBuffer(2*bDigest.length);

        for (int i=0; i<bDigest.length; ++i) {
                int k = bDigest[i] & 0xFF;
                if(k<0x10) sb.append('0');
                sb.append(Integer.toHexString(k));
        }
```

…

```
        return sb.toString().getBytes();
}
```

Bijlage C Document Type Definitions (DTD's)

Deze bijlage bevat de Document Type Definitions (DTD's) voor de bestanden met kandidaat-gegevens en voor de export van de vercijferde stemmen, zoals die uitgewisseld worden tussen de opdrachtgever en de stemdienst. De DTD's voor kandidaat-gegevens zijn ontleend aan het Technisch Ontwerp KoA 1.3.

C.1 DTD Inlezen kandidaat-gegevens

Met dit bestand levert de opdrachtgever de kandidaat-gegevens aan de stemdienst. Indien de aanwezigheid van een blanco stem gewenst is, dient deze aanwezig te zijn in de in te lezen gegevens als een kandidaat met achternaam [Blanco].

Bijvoorbeeld als:

```
<kieslijst nummer="99" groepering="[Blanco]">
  <positie nummer="1" achternaam="[Blanco]" voorletters="-" roepnaam="-" geslacht="M" woonplaats="-" />
</kieslijst>
```

Hier volgt de DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT import (metadata, kieskring+)>
<!ATTLIST import
        action CDATA #FIXED "replace"
        contenttype CDATA #FIXED "kieslijst"
>
<!ELEMENT metadata (requestreference, creationtime, kieskringcount, districtcount, kieslijstcount, positiecount)>
<!ELEMENT requestreference (#PCDATA)>
<!ELEMENT creationtime (#PCDATA)>
<!ELEMENT kieskringcount (#PCDATA)>
<!ELEMENT districtcount (#PCDATA)>
<!ELEMENT kieslijstcount (#PCDATA)>
<!ELEMENT positiecount (#PCDATA)>
<!ELEMENT kieskring (district+, kieslijst+)>
<!ATTLIST kieskring
        nummer CDATA #REQUIRED
        naam CDATA #REQUIRED
>
<!ELEMENT district EMPTY>
<!ATTLIST district
        nummer CDATA #REQUIRED
        naam CDATA #REQUIRED
>
<!ELEMENT kieslijst (positie+)>
<!ATTLIST kieslijst
        nummer CDATA #REQUIRED
        groepering CDATA #REQUIRED
>
<!ELEMENT positie EMPTY>
<!ATTLIST positie
        nummer CDATA #REQUIRED
        achternaam CDATA #REQUIRED
        voorletters CDATA #REQUIRED
        roepnaam CDATA #REQUIRED
        geslacht (M | V) #REQUIRED
        woonplaats CDATA #REQUIRED
```

```
>
```

C.2 DTD Retour bestand kandidaat-gegevens

Het bestand dat de stemdienst terugstuurt is grotendeels identiek aan het bestand waarin de gegevens zijn aangeleverd aan de stemdienst. Het bevat als extra gegeven een referentienummer voor het antwoord en de toegekende kandidaatcodes. Ook in dit bestand kan dus weer een speciale lijst [Blanco] met één kandidaat [Blanco] voorkomen.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT result (metadata, kieskring+)>
<!ATTLIST result
        action CDATA #FIXED "replace"
        contenttype CDATA #FIXED "kieslijst"
>
<!ELEMENT metadata (requestreference, responsereference, creationtime,
                                        kieskringcount, districtcount, kieslijstcount, positiecount)>
<!ELEMENT requestreference (#PCDATA)>
<!ELEMENT responsereference (#PCDATA)>
<!ELEMENT creationtime (#PCDATA)>
<!ELEMENT kieskringcount (#PCDATA)>
<!ELEMENT districtcount (#PCDATA)>
<!ELEMENT kieslijstcount (#PCDATA)>
<!ELEMENT positiecount (#PCDATA)>
<!ELEMENT kieskring (district+, kieslijst+)>
<!ATTLIST kieskring
        nummer CDATA #REQUIRED
        naam CDATA #REQUIRED
>
<!ELEMENT district EMPTY>
<!ATTLIST district
        nummer CDATA #REQUIRED
        naam CDATA #REQUIRED
>
<!ELEMENT kieslijst (positie+)>
<!ATTLIST kieslijst
        nummer CDATA #REQUIRED
        groepering CDATA #REQUIRED
>
<!ELEMENT positie (code+)>
<!ATTLIST positie
        nummer CDATA #REQUIRED
        achternaam CDATA #REQUIRED
        voorletters CDATA #REQUIRED
        roepnaam CDATA #REQUIRED
        geslacht (M | V) #REQUIRED
        woonplaats CDATA #REQUIRED
>
<!ELEMENT code (#PCDATA)>
```

C.3 DTD Export versleutelde stemmen

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT report  (globaal, table)>
<!ELEMENT globaal (kieskringen+)>
<!ATTLIST globaal
```

```
          stembureau CDATA #REQUIRED
          voorzitter     CDATA #REQUIRED
          state          CDATA #REQUIRED
          verkiezing     CDATA #REQUIRED
          periode_start CDATA #REQUIRED
          periode_eind CDATA #REQUIRED
          curtime        CDATA #REQUIRED
>
<!ELEMENT kieskringen (kieskring+)>
<!ELEMENT kieskring EMPTY>
<!ATTLIST kieskring
          nummer CDATA #REQUIRED
          naam     CDATA #REQUIRED
>
<!ELEMENT table (row+)>
<!ATTLIST table
          name CDATA #FIXED "ENCRYPTEDESB"
>
<!ELEMENT row EMPTY>
<!ATTLIST row
          stemnummer CDATA #REQUIRED
          stem           CDATA #REQUIRED
>
```