(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 699 18 818 T2** 2005.08.25

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 066 699 B1**
(21) Deutsches Aktenzeichen: **699 18 818.0**
(86) PCT-Aktenzeichen: **PCT/CA99/00244**
(96) Europäisches Aktenzeichen: **99 908 723.2**
(87) PCT-Veröffentlichungs-Nr.: **WO 99/049612**
(86) PCT-Anmeldetag: **23.03.1999**
(87) Veröffentlichungstag der PCT-Anmeldung: **30.09.1999**
(97) Erstveröffentlichung durch das EPA: **10.01.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **21.07.2004**
(47) Veröffentlichungstag im Patentblatt: **25.08.2005**

(51) Int Cl.[7]: **H04L 9/08**
H04L 9/32

(30) Unionspriorität:

2232936	**23.03.1998**	**CA**
2235359	**20.04.1998**	**CA**

(84) Benannte Vertragsstaaten:
CH, DE, FR, GB, LI

(73) Patentinhaber:
Certicom Corp., Mississauga, Ontario, CA

(72) Erfinder:
QU, Minghua, Mississauga, CA; VANSTONE, A., Scott, Waterloo, CA

(74) Vertreter:
Flaccus, R., Dipl.-Chem. Dr.rer.nat., Pat.-Anw., 50389 Wesseling

(54) Bezeichnung: **Verfahren zur Erzeugung eines öffentlichen Schlüssels in einem sicheren digitalen Kommunikationssystem und implizites Zertifikat**

Beschreibung

[0001] Die Erfindung betrifft Schlüsselverteilungsschemata für den Transfer und die Authentifizierung von Verschlüsselungsschlüsseln

HINTERGRUND DER ERFINDUNG

[0002] Mit der Diffie-Hellman-Schlüsselübereinstimmung ("Diffie-Hellman key agreement") wurde die erste praktikable Lösung für das Schlüsselverteilungsproblem in kryptographischen Systemen zur Verfügung gestellt. Das Schlüsselübereinstimmungsprotokoll ermöglichte es, zwei Parteien, die zuvor noch keinen Kontakt miteinander gehabt oder kein Schlüsselmaterial ausgetauscht hatten, ein gemeinsames Geheimnis durch den Austausch von Nachrichten über einen offenen (ungeschützten) Kanal zu etablieren. Hierbei beruht die Sicherheit auf der Nicht-Zurückverfolgbarkeit des Diffie-Hellman-Problems und dem damit zusammenhängenden Problem der Berechnung diskreter Logarithmen.

[0003] Mit dem Aufkommen des Internets und ähnlicher Einrichtungen gewinnen das Erfordernis der Verteilung von öffentlichen Schlüsseln ("public keys") in großem Maßstab sowie Zertifikate für öffentliche Schlüssel ("public key"-Zertifikate) zunehmend an Bedeutung. Public-Key-Zertifikate sind ein Vehikel, mit dem öffentliche Schlüssel gespeichert, verteilt oder zugestellt werden können, und zwar über ungesicherte Medien, ohne dass dabei die Gefahr einer unentdeckbaren Manipulation besteht. Ziel hierbei ist es, den öffentlichen Schlüssel einer Partei anderen in einer Weise zur Verfügung zu stellen, dass dessen Authentizität und Gültigkeit ("validity") überprüft werden können.

[0004] Ein Public-Key-Zertifikat ist eine Datenstruktur, die aus einem Datenteil und einem Signaturteil besteht. Der Datenteil enthält Klartextdaten, welche, als Minimum, einen öffentlichen Schlüssel und einen die mit diesem Schlüssel zu assoziierende Partei identifizierenden Datenstring enthält. Der Signaturteil besteht aus einer digitalen Signatur einer Zertifizierungsstelle (CA, "certificate authority") über dem Datenteil, hierdurch wird die Identität der Entitäten an den spezifizierten öffentlichen Schlüssel gebunden. Die CA ist eine vertrauenswürdige dritte Partei, deren Signatur auf dem Zertifikat für die Authentizität des an die betreffende Entität gebundenen öffentlichen Schlüssels bürgt.

[0005] Identitätsbasierte Systeme ("ID-based systems") sind gewöhnlichen Public-Key-Systemen insofern ähnlich als sie eine geheime Transformation und eine öffentliche Transformation beinhalten, jedoch haben die Parteien, anders als vorher, keine expliziten öffentlichen Schlüssel. Stattdessen wird der öffentliche Schlüssel durch die öffentlich zugängliche Identitätsinformation (z.B. Name oder Netzadresse) einer Partei ersetzt. Als Identitätsinformation kann hierbei jede öffentlich zugängliche Information dienen, durch welche die Partei eindeutig identifiziert wird und die unleugbar mit dieser Partei assoziiert werden kann.

[0006] Eine alternative Vorgehensweise bei der Verteilung öffentlicher Schlüssel verwendet implizit zertifizierte öffentliche Schlüssel. Zwar existieren hier explizite öffentliche Schlüssel der Benutzer, doch müssen diese rekonstruiert werden anstatt dass sie, wie bei zertifikatsbasierten Systemen, durch Public-Key-Zertifikate transportiert werden. Somit können implizit zertifizierte öffentliche Schlüssel als ein alternatives Mittel zur Verteilung öffentlicher Schlüssel (z.B. Diffie-Hellman-Schlüssel) Verwendung finden.

[0007] Ein Beispiel für einen implizit zertifizierten Public-Key-Mechanismus ist in dem schweizerischen Patent CH 678134 beschrieben und als Günthersches Verfahren mit implizit zertifiziertem (ID-basierten) Public-Key bekannt. Bei diesem Verfahren:

1. wählt ein vertrauenswürdiger Server T eine geeignete, feste, öffentliche Primzahl p und einen Generator α von Z_p^*. T wählt eine zufällige ganze Zahl t, mit $1 \le t \le p-2$ und $\gcd(t,p-1) = 1$, als seinen geheimen Schlüssel und veröffentlicht seinen öffentlichen Schlüssel $u = \alpha^t \bmod p$, zusammen mit α, p.
2. T weist jeder Partei A einen eindeutigen Namen oder einen identifizierenden Datenstring I_A und eine zufällige ganze Zahl k_A mit $\gcd(k_A, p-1)=1$ zu. Dann berechnet T $P_A = \alpha^{K_A} \bmod p$. P_A sind A's öffentliche Schlüsselrekonstruktionsdaten, die es anderen Parteien ermöglichen, $(P_A)^a$, siehe unten, zu berechnen.
3. Unter Verwendung einer geeigneten Hash-Funktion h löst T die nachfolgende Gleichung für a:

$$H(I_A) \equiv t.P_A + k_A\, a \pmod{p-1}$$

4. T schickt das Paar $(r,s)=(P_A,a)$, wobei es sich um T's ElGama1-Signatur auf I_A handelt, gesichert an A (a ist A's geheimer Schlüssel für die Diffie-Hellman-Schlüsselübereinstimmung).
5. Jede andere Partei ist dann in der Lage, den Diffie-Hellman Public-Key P_A^a der Partei A vollständig aus öffentlich zugänglichen Informationen (α, I_A, u, P_A, p) durch Berechnen von

$$P_A^a \equiv \alpha^{h(I_A)} u^{-P_A} \bmod p$$

zu rekonstruieren.

[0008] Somit ist bei Problemen der Berechnung des diskreten Logarithmus zur Signierung eines Zertifikates eine einzige Potenzierungsoperation erforderlich, die Rekonstruktion des ID-basierten implizit-verifizierbaren öffentlichen Schlüssels jedoch erfordert zwei Potenzierungen. Bekanntermaßen ist die Potenzierung in der Gruppe Z_P^* sowie ihre analoge skalare Multiplikation eines Punktes in $E(F_q)$ rechnerisch aufwendig. So ist z.B. ein RSA-Schema im Vergleich zu Systemen auf Basis elliptischer Kurven extrem langsam. Trotz der enormen Überlegenheit von EC-Systemen gegenüber Systemen des RSA-Typs stellt der Rechenaufwand insbesondere für Recheneinrichtungen, die über begrenzte Rechenleistung verfügen, wie z.B. "Smart Cards", "Pager" und dergleichen, jedoch immer noch ein Problem dar.

ZUSAMMENFASSUNG DER ERFINDUNG

[0009] Das Ziel der vorliegenden Erfindung besteht in der Bereitstellung eines effizienten Schemas mit ID-basiertem implizitem Zertifikat, welches gegenüber bestehenden Schemata eine verbesserte Rechengeschwindigkeit aufweist. Der Einfachheit halber werden hier die Schemata über Z_p beschrieben, jedoch können diese Schemata ebenfalls in auf elliptischen Kurven basierenden Kryptosystemen implementiert werden.

[0010] Erfindungsgemäß wird ein Verfahren zur Erzeugung eines identitätsbasierten öffentlichen Schlüssels in einem sicheren digitalen Kommunikationssystem, das wenigstens eine vertrauenswürdige Entität CA und Teilnehmerentitäten A umfasst, bereitgestellt, welches folgende Schritte umfasst:

a) für jede Entität A wählt die vertrauenswürdige Entität CA eine eindeutige Identität I_A, durch welche die Entität A gekennzeichnet wird;

b) Erzeugung öffentlicher Daten γ_A zur Rekonstruktion des öffentlichen Schlüssels der Entität A durch mathematisches Kombinieren eines Generators der vertrauenswürdigen Partei CA mit einem geheimen Wert der Entität A, so dass das Paar (I_A, γ_A) als das implizite Zertifikat von A dient;

c) Kombinieren der impliziten Zertifikatsinformation (I_A, γ_A) gemäß einer mathematischen Funktion $F(\gamma_A, I_A)$, um eine Entitätsinformation f abzuleiten;

d) Erzeugen eines geheimen Schlüssels a der Entität A durch Signieren der Entitätsinformation f und

Übertragen des geheimen Schlüssels a an die Entität A, wobei der öffentliche Schlüssel der Entität A aus der öffentlichen Information, dem Generator γ_A und der Identität I_A relativ effizient rekonstruiert werden kann.

[0011] Gemäß einem anderen Aspekt der Erfindung wird ein Public-Key-Zertifikat bereitgestellt, das eine Mehrzahl öffentlicher Schlüssel unterschiedlicher Bit-Stärken umfasst und bei dem einer der öffentlichen Schlüssel ein implizit zertifizierter öffentlicher Schlüssel ist.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0012] Im Folgenden werden, lediglich beispielhaft, Ausführungen der vorliegenden Erfindung unter Bezug auf die begleitenden Zeichnungen beschrieben; die Zeichnungen zeigen:

[0013] Fig. 1: eine schematische Darstellung einer ersten Systemkonfiguration gemäß einer Ausführungsform der vorliegenden Erfindung; und

[0014] Fig. 2: eine schematische Darstellung einer zweiten Systemkonfiguration gemäß einer Ausführungsform der vorliegenden Erfindung.

DETAILLIERTE BESCHREIBUNG EINER BEVORZUGTEN AUSFÜHRUNGSFORM

[0015] Das Folgende bezieht sich auf Fig. 1, in der ein System mit implizit zertifizierten öffentlichen Schlüsseln dargestellt ist, allgemein mit **10** gekennzeichnet. Dieses System **10** umfasst eine vertrauenswürdige dritte Partei CA und wenigstens ein Paar aus einem ersten und einem zweiten Korrespondenten A bzw. B. Die Korrespondenten A und B tauschen über einen Kommunikationskanal Informationen aus und sowohl A als auch B weisen eine Kryptographieeinheit zur Ausführung visueller Finde-/Prüf-Operationen und Verschlüsselungs-/Entschlüsselungsoperationen auf.

[0016] Im Folgenden wird wiederum auf Fig. 1 Bezug genommen: Die vertrauenswürdige Partei CA wählt

eine geeignete Primzahl p mit p=tq+1, wobei q eine große Primzahl ist, und einen Generator α der Ordnung q. Die CA wählt eine zufällige ganze Zahl c, mit $1 \le c \le q-1$, als ihren geheimen Schlüssel, dann berechnet sie den öffentlichen Schlüssel $\beta=\alpha^c \bmod p$ und veröffentlicht (β, α, p, q).

Schema 1:

1. Für jede Partei A wählt die CA eine(n) eindeutige(n), unterscheidbare(n) Namen ("distinguished name") oder Identität I_A (z.B. Name, Adresse, Telefonnummer) sowie eine zufällige ganze Zahl c_A mit $1 \le c_A \le q-1$. Dann berechnet CA $\gamma_A=\alpha^{CA} \bmod p$. ($\gamma_A$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von A). Das Paar (I_A, γ_A) dient als das implizite Zertifikat von A)
2. CA wählt eine Funktion $f=F(I_A, \gamma_A)$. Zum Beispiel $F(\gamma_A, I_A)=\gamma_A+h(I_A)$ oder $F(\gamma_A,I_A)=h(\gamma_A+I_A)$, in denen h eine sichere Hash-Funktion ist, und löst die nachfolgende Gleichung für a, welches den geheimen Schlüssel der Partei A darstellt. Falls a=0, so wählt CA ein anders c_A und löst die Gleichung neu.

$$1 = cf + c_A a \pmod q$$

3. CA sendet das Tripel (γ_A, a, I_A), welches CA's Signatur auf I_A darstellt, auf sicherem Wege an A. Somit ist
α ist der geheime Schlüssel von A;
γ_A ist der Generator von A; und
γ_A^a ($=\alpha^{cAa}$) ist der öffentliche Schlüssel von A.
A veröffentlicht (α, I_A, β, γ_A, p, q) im frei zugänglichen Bereich ("Public Domain").
4. Jedermann kann den (ID-basierten) implizit verifizierbaren öffentlichen Schlüssel der Partei A aus der Public Domain durch Berechnen von

$$\gamma_A^a = \alpha\beta^{-f} \pmod p$$

erhalten, d.h. der öffentliche Schlüssel wird aus der obigen Gleichung hergeleitet, was lediglich eine einzige Potenzierungsoperation erfordert.

[0017] Zwar kann der öffentliche Schlüssel der Partei A von jedermann aus öffentlichen Daten rekonstruiert werden, jedoch bedeutet dies nicht, dass der rekonstruierte öffentliche Schlüssel γ_A^a zertifiziert ist. Dieses Schema ist effektiver, wenn es mit einem Applikationsprotokoll kombiniert wird, welches zeigt, dass der Partei A der entsprechende geheime Schlüssel vollständig bekannt ist, so z.B. mit dem MQV-Schlüsselübereinstimmungsschema oder einem beliebigen Signaturschema und insbesondere mit einem KCDSA (Digitaler Signaturalgorithmus auf Basis des koreanischen Zertifikats). Generell kann dieses Schema mit implizitem Zertifikat mit jedem Schema verwendet werden, bei dem eine Verifizierung des Zertifikats erforderlich ist. Dies kann unter Bezugnahme auf das Signaturschema mit digitalem Signaturalgorithmus (DSA) gezeigt werden.

[0018] Angenommen Alice hat den geheimen Schlüssel α, den Generator γ_A und veröffentlicht ($\alpha,I_A,\beta,\gamma_A,p,q$) in der Public Domain. Alice will nun eine Nachricht M unter Verwendung von DSA unterzeichnen.

[0019] Alice geht wie folgt vor:
1. Sie wählt ein zufälliges k, berechnet $r=\gamma_A^k \pmod p$;
2. sie berechnet e=sha-1(M);
3. sie berechnet $s=k^{-1}(e + ar) \pmod p$.
4. Die Signatur auf M ist (r,s).

[0020] Der Verifizierer geht wie folgt vor:
1. Er beschafft sich Alices öffentliche Daten (α, I_A, β, γ_A, p, q) und rekonstruiert den öffentlichen Schlüssel

$$\delta_A=\gamma_A^a=\alpha\beta^{-f} \pmod p;$$

2. berechnet e=sha-1(M);
3. berechnet $u_1 = es^{-1} \pmod q$ und $u_2 = rs^{-1} \pmod q$;
4. berechnet $r' = \gamma_A^{u_1}\delta_A^{u_2} \bmod p$;
5. wenn r=r', so ist die Signatur verifiziert. Gleichzeitig ist Alices (ID-basierter) öffentlicher Schlüssel implizit verifiziert.

[0021] Das Paar (I_A,γ_A) dient als Alices Zertifikat. Die Rekonstruktion des öffentlichen Schlüssels dient als implizite Verifizierung, sofern das Ergebnis des Applikationsprotokolls eine gültige Verifizierung ist. Man erinnere, dass lediglich eine einzige Potenzierungsoperation erforderlich ist, um den öffentlichen Schlüssel zu erhalten.

[0022] Bei einer alternativen Ausführungsform lässt sich das Schema durch entsprechendes Modifizieren der Signaturgleichung zu den meisten ElGama1-Signaturschemata verallgemeinern. Im Folgenden einige Beispiele:

Schema 2:

[0023] CA verwendet die Signaturgleichung $1=ca+c_A f \pmod q$. CA schickt das Tripel (γ_A, a, I_A) gesichert an A, a ist hierbei der geheime Schlüssel von A, β ist der Generator von A, und β^a ist der öffentliche Schlüssel von A. A veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain. Jedermann kann den (ID-based) implizit zertifizierten öffentlichen Schlüssel der Partei A durch Berechnen von

$$\beta^a = \alpha\gamma_A^{-f} \pmod p$$

aus der Public Domain erhalten.

[0024] Bei diesem Schema hat jeder Benutzer den gleichen Generator β, der den öffentlichen Schlüssel der Partei CA darstellt.

Schema 3:

[0025] CA verwendet die Signaturgleichung $a = cf + {}_{CA} \pmod q$. CA schickt das Tripel (γ_A, a, I_A) gesichert an A; a ist hierbei der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \pmod p$$

[0026] Bei diesem Schema hat jeder Benutzer, einschließlich CA, den gleichen Generator α.

Schema 4:

[0027] CA verwendet die Signaturgleichung $a \equiv {}_{CA} f + c \pmod q$. CA schickt das Tripel (γ_A, a, I_A) gesichert an A; a ist hierbei der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \gamma_A^f \beta \pmod p$$

[0028] Bei diesem Schema hat jeder Benutzer, einschließlich CA, den gleichen Generator α.

[0029] In den obigen Schemata hat die Partei A nicht die Freiheit, ihren eigenen geheimen Schlüssel zu wählen. Bei den nachfolgenden Schemata, dargestellt in **Fig. 2**, haben sowohl CA als auch der Benutzer die Kontrolle über den geheimen Schlüssel des Benutzers, jedoch kennt nur der Benutzer seinen geheimen Schlüssel.

Schema 5':

[0030] A wählt zunächst eine zufällige ganze Zahl k und berechnet α^k, dann schickt A α^k an CA. Der CA berechnet $\gamma_A = \alpha^{kC_A} \bmod p$ und löst die folgende Signaturgleichung für k_A

$$1 = cf + c_A k_A \pmod q..$$

[0031] Dann berechnet CA $\gamma_A^1 = \alpha^{C_A} \bmod p$ und schickt das Tripel (γ_A^1, k_A, I_A) an A. A berechnet $a=k_A k^{-1} \pmod q$ und $\gamma_A = (\gamma_A^1)^k \pmod p$. a ist hierbei der geheime Schlüssel von A, γ_A ist der Generator von A und γ_A^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\gamma_A^a = \alpha\beta^{-f} \pmod p$$

Schema 6:

1. A wählt zufällig eine ganze Zahl k und berechnet β^k, dann sendet A β^k an CA.
2. CA wählt eine zufällige ganze Zahl c_A, berechnet $\gamma_A = \beta^k \alpha^{c_A} \pmod p$ und $f=F(\gamma_A, I_A)$, löst die Signaturglei-

chung für k_A (wenn k_A=0, wähle ein anderes c_A)

$$1 = ck_A + c_A f \pmod{q}.$$

CA berechnet $\gamma_A^1 = \beta^{c_A c^{-1}} \pmod{p}$ und sendet das Tripel (γ_A^1, k_A, I_A) an A.
Anmerkung: (γ_A^1, k_A, I_A) kann über einen öffentlichen Kanal gesendet werden.
3. A berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \alpha^k \pmod{p}$, f=F($\gamma_A$,$I_A$) und a=$k_A$-kf (mod q) (wenn a=0,1, zurück zu Schritt 1). A prüft dann, ob $\beta^a = \alpha\gamma_A^f$. Nun ist a der geheime Schlüssel von A, β ist der Generator von A und β^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta^a = \alpha\gamma_A^{-f} \pmod{p}$$

Schema 7:

[0032] A wählt zunächst zufällig eine ganze Zahl k und berechnet α^k, welches A an CA schickt. Nun berechnet CA $\gamma_A=\alpha^k\alpha^{CA}$ (mod p) und löst die Signaturgleichung für k_A

$$k_A \equiv cf + c_A \pmod{q}$$

[0033] Dann berechnet CA $\gamma_A^1 = (\alpha^k)^{CA}$ (mod p) und schickt das Tripel (γ_A^1, k_A, I_A) an A. A berechnet $\gamma_A = (\gamma_A^1)^{k-1}$ α^k (mod p). Dann ist a = k_A + k(mod q) der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \pmod{p}$$

Schema 8:

1. A wählt zufällig eine ganze Zahl k, berechnet α^k und sendet α^k an CA.
2. CA wählt eine zufällige ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod{p}$ und f=F(γ_A, I_A), berechnet k_A (wenn k_A=0, wähle ein anderes c_A)

$$k_A \equiv c_A f + c \pmod{q}.$$

Dann berechnet CA $CA\ \gamma_A^1 = (\alpha^k)^{c_A} \pmod{p}$ und schickt das Tripel (γ_A^1, k_A, I_A) an A.
Anmerkung: (γ_A^1, k_A, I_A) kann über einen öffentlichen Kanal geschickt werden.
3. A berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \alpha^k \pmod{p}$, f = F($\gamma_A$,$I_A$) und a = k_A + kf (mod q) (wenn a=0,1, geht A zurück zu Schritt 1). A prüft dann, ob $\alpha^a = \gamma_A^f\beta$; a ist nun der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \gamma_A^f \beta \pmod{p}$$

[0034] Bei den obigen Schemata 5-8 kann jeder eine Teilinformation über den geheimen Schlüssel α des Nutzers A erhalten, da k_A über einen öffentlichen Kanal gesendet wird. Um diese Information zu verbergen und um die Berechnung der obige Schemata zu beschleunigen, wird DES-Verschlüsselung eingeführt, so dass sich die nachfolgenden Schemata 9-12 durch Modifizieren der Schemata 5-8 ergeben. Die Vorteile von Schemata 9-12 liegen darin, dass der Benutzer K leicht berechnen kann, da β fest ist.

Schema 9:

1. A wählt zufällig eine ganze Zahl k und berechnet α_k, dann schickt A α_k an CA.
2. CA wählt zufällig eine ganze Zahl C_A, berechnet $\gamma_A = \alpha^{kc_A} \pmod{p}$ und f = F(γA,β,I_A), A löst die Signaturgleichung für k_A (wenn k_A = 0, wähle ein anderes C_A).

$$1 = cf + c_A k_A \pmod{q}$$

Als nächstes berechnet CA K = $(a^k)^c$(mod p) und $\overline{k}_A$ = DES_K (k_A), dann schickt A das Tripel (γ_A,$\overline{k}_A$,I_A) an A. γ_A
3. A berechnet K = β^k (mod p), k_A = DES_k ($\overline{k}_A$), und a = $k_A k^{-1}$ (mod q) (wenn a=1, zurück zu Schritt 1). A prüft

dann, ob $\gamma_A^a = \alpha\beta^{-f}$. A ist nun der geheime Schlüssel von A, γ_A ist der Generator von A und γ_A^a ist der öffentliche Schlüssel von A. A veröffentlicht $(\alpha, I_A, \beta, \gamma_A, p, q)$ in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\gamma_A^a = \alpha\beta^{-f} \pmod p$$

Schema 10:

1. A wählt eine zufällige ganze Zahl k und berechnet β^k, dann schickt A β^k an CA.
2. CA wählt eine zufällige ganze Zahl C_A, berechnet $\gamma_A = \beta^k \alpha^{C_A} \pmod p$ und $f = F(\gamma_A, \beta, I_A)$, löst die Signaturgleichung für k_A (wenn $k_A = 0$, wähle ein anderes C_A)

$$1 = ck_A + c_A f \pmod q$$

Als nächstes berechnet CA $K = (\beta^k)^{c_a c^{-1}} = \alpha^{kc_A} \pmod p$ und $\overline{k}_A = DES_K(k_A)$, dann sendet CA das Tripel $(\gamma_A\ \overline{k}_A, I_A)$ an A.
Anmerkung: $(\gamma_A\ \overline{k}_A, I_A)$ kann über einen öffentlichen Kanal geschickt werden.
3. A berechnet $K = (\gamma_A / \beta^k)^k = \alpha^{kc_A} \pmod p, k_A = DES_k(\overline{k}_A), f = F(\gamma_A, \beta, I_A)$ und berechnet $\alpha = k_A - kf \pmod q$ (wenn a=0,1, zurück zu Schritt 1). A prüft dann, ob $\beta^a = \alpha\gamma_A^{-f}$. Nun ist a der geheime Schlüssel von A, β ist der Generator von A und β^a ist der öffentliche Schlüssel von A. A veröffentlicht $(\alpha, I_A, \beta, \gamma_A, p, q)$ in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta^a = \alpha\gamma_A^{-f} \pmod p$$

Schema 11

1. A wählt zufällig eine ganze Zahl k und berechnet a^k, dann schickt A a^k an CA.
2. CA wählt eine ganze Zahl C_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und $f = F(\gamma_A, \beta, I_A)$ berechnet k_A (wenn $k_A = 0$, wähle ein anderes C_A)

$$k_A = cf + c_A \pmod q.$$

Als nächstes berechnet CA $K = (\alpha^k)^c \pmod p$ und $\overline{k}_A = DES_K(k_A)$, und schickt das Tripel $(\gamma_A, \overline{k}_A, I_A)$ an A.
Anmerkung: $(\gamma_A, \overline{k}_A, I_A)$ kann über einen öffentlichen Kanal geschickt werden.
3. A berechnet $K = \beta^k \pmod p$, $k_A = DES_K(\overline{k}_A)$ und $\alpha = k_A + k \pmod q$ (wenn a=0,1, zurück zu Schritt 1). Dann prüft A, ob $\alpha^a = \beta^f\gamma_A$. Nun ist a der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht $(\alpha, I_A, \beta, \gamma_A, p, q)$ in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von $\alpha^a = \gamma_A^f \pmod p$.

Schema 12:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl C_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und $f = F(\gamma_A, \beta, I_A)$ berechnet k_A (wenn $k_A = 0$, wähle ein anders C_A) $k_A = c_A f + c \pmod q$.
Als nächstes berechnet CA $K = (\alpha^k)^c \pmod p$ und $\overline{k}_A = DES_k(k_A)$, dann schickt CA das Tripel $(\gamma_A, \overline{k}_A, I_A)$ an A.
Anmerkung: $(\gamma_A, \overline{k}_A, I_A)$ kann über einen öffentlichen Kanal geschickt werden.
3. A berechnet $K = \beta^k \pmod p$, $k_A = DES_k(\overline{k}_A)$, $f = F(\gamma_A, \beta, I_A)$ und $a = k_A + kf \pmod q$ (wenn a=0,1, zurück zu Schritt 1). Dann prüft A, ob $\alpha^a = \gamma_A^f\beta$. Nun ist a der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht $(\alpha, I_A, \beta, \gamma_A, p, q)$. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \gamma_A^f \beta \pmod p$$

[0035] Die Vorteile der Schemata 9-12 liegen darin, dass K vom Nutzer A leicht berechnet werden kann, da β fix ist, und dass k_A so verschlüsselt ist, dass niemand anderes k_A kennen kann.

[0036] Man bemerke, dass der Nutzen der Schemata 5-12 durch Hinzufügen eines Optionsparameters OP

zu der Funktion $F(\gamma_A,\beta,I_A)$ (d.h. $f = F(\gamma_A,\beta,I_A,OP)$ erhöht wird. Beispielsweise ist $OP = \alpha^{a_E}$, worin a_E der geheime Verschlüsselungsschlüssel des Nutzers A ist und α^{a_E} der öffentliche Verschlüsselungsschlüssel des Nutzers A. Auf Schema 15 folgt eine Modifizierung des Schemas 7. Schemata 5-12 können in gleicher Weise modifiziert werden. Auch die Schemata 1-4 können in dieser Weise modifiziert werden.

Schema 13

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, A schickt dann α^k an CA.
2. CA wählt eine zufällige ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A}$ und $f = F(\gamma_A,I_A,OP)$, berechnet k_A (wenn $k_A = 0$, wähle ein anderes c_A)

$$k_A \equiv cf + c_A \pmod{q}.$$

Als nächstes berechnet CA $K = H((\alpha^k)^c)$ und $\overline{k}_A = DES_K(k_A)$, dann schickt CA das Tripel $(f, \overline{k}_A, I_A)$ an A.
3. A berechnet $K = H(\beta^k)$, $k_A = DES_K(k_A)$ und $a = k_A + k \pmod{q}$ (wenn a=0,1, zurück zu Schritt 1.) Dann berechnet A $\gamma_A = \alpha^a \beta^{-f} \pmod{p}$ und prüft, ob $f = F(\gamma_A,I_A,OP)$. Nun ist a der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht $(\alpha,I_A,\beta,\gamma_A,p,q)$ in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \pmod{p}$$

[0037] Weiterhin kann die Bandbreite durch das nachfolgende Schema 14 vermindert werden.

Schema 14:

1. A wählt eine zufällige ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod{p}$ und setzt $\Diamond_A$ als die ersten 80 niedrigstwertigen Bits von γ_A. Dann berechnet CA $f = F(\Diamond_A,I_A,OP)$ und k_A (wenn ka=0, wähle ein anderes c_A)

$$k_A \equiv cf + c_A \pmod{q}.$$

Als nächstes berechnet CA $K = (\alpha^k)^c \pmod{p}$ und $\overline{k}_A = DES_K(k_A)$, dann sendet CA $(\Diamond_A, \overline{k}_A, I_A)$ an A.
Anmerkung: $(\Diamond_A, \overline{k}_A, I_A)$ kann über einen öffentlichen Kanal geschickt werden.
3. A berechnet $K = \beta^k \pmod{p}$, $k_A = DES_K(\overline{k}_A)$ und $ak_A + k \pmod{q}$ (wenn a=0,1, zurück zu Schritt 1). A berechnet dann $F(\Diamond_A,\beta,I_A)$, $\gamma_A = \alpha^a \beta^{-f} \pmod{p}$ und prüft, ob die ersten 80 niedrigstwertigen Bits von γ_A gleich $\Diamond_A$ sind. Nun ist a der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht $(\alpha,I_A,\beta,\gamma_A,p,q)$ in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \pmod{p}$$

[0038] Das Sicherheitsniveau 14 ist nicht so hoch wie das anderer, oben besprochener Schemata. Schema 14 hat eine 80-Bit-Sicherheit, die jedoch derzeit für praktische Anwendungen ausreicht. Die ersten 80 niedrigstwertigen Bits können auf die niedrigstwertigen Bits bis zur Hälfte der Bits von γ_A erweitert werden.

[0039] Das implizite Zertifikat kann zur Zertifizierung anderer nützlicher Information verwendet werden, und zwar durch Aufnahme der Information in den Optionsparameter OP. Beispielsweise $OP = \alpha^{a_E} || OP_2$, wobei a_E ein anderer geheimer Schlüssel des Benutzers A und α^{a_E} der korrespondierende öffentliche Schlüssel ist. Das folgende Schema 15 stellt eine Modifikation des Schemas 7 dar. Andere Schemata können in der gleichen Weise modifiziert werden.

Schema 15:

1. A wählt zufällig eine ganze Zahl a_E und berechnet α^{a_E}.
2. A wählt zufällig eine ganze Zahl k und berechnet α^k, A schickt dann α^k und α^{a_E} an CA.
3. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod{p}$ und $f = F(\gamma_A,\beta,I_A,\alpha^{a_E})$. (Beispielsweise $f = F(\gamma_A,\beta,I_A,\alpha^{a_E}) = h(\gamma_A \| \beta \| I_A \| \alpha^{a_E}))$, berechnet k_A (wenn $k_A = 0$, wähle ein anderes C_A)

$k_A = cf + cA \pmod q$

Dann berechnet $CA\ \gamma_A^1 = (\alpha^k)^{c_A} \pmod p$ und sendet das Tripel (γ_A^1, k_A, I_A) an A.
Anmerkung: (γ_A^1, k_A, I_A) kann über einen öffentlichen Kanal geschickt werden.
4. A berechnet a=k_A+k (mod q) (wenn a=0,1, zurück zu Schritt 1) und berechnet $\gamma_A = \gamma_A^{1)^{k^{-1}}} \alpha^k \pmod p$. Danach prüft A, ob $\alpha^a = \beta^f \gamma_A$. Nun ist a der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A, a_E ist der geheime Verschlüsselungsschlüssel von A und α^{a_E} ist der öffentliche Verschlüsselungsschlüssel von A. A veröffentlicht ((α, α^{a_E}, I_A, β, γ_A, p, q) in der Public Domain.
5. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$\alpha^a = \beta^f \gamma_A \pmod p$

Anmerkungen: (zu Schemata 13-15)

1. Die Identität I_A kann entweder von CA oder durch die Entität A gewählt werden.
2. CA sollte die Entität A authentifizieren. Dies kann mittels des in Anmerkung 2 zu Schema 11 beschriebenen Verfahrens erfolgen.
3. (f, $\bar{k}_A$, I_A) oder ($\Diamond_A$, k_A,I_A) oder (γ_A^1, k_A, I_A) kann über einen öffentlichen Kanal geschickt werden.

[0040] Bei unseren Schemata ist (α, γ_A) die Signatur von CA auf der ID I_A von A, es wurde davon ausgegangen, dass (α,γ_A) öffentlich bekannt ist. Jetzt ist a jedoch nur dem Benutzer A bekannt. Werden diese Schemata verwendet, sollte man also sicherstellen, dass in dem Applikationsprotokoll der Benutzer A seinen eigenen geheimen Schlüssel kennt. Anders ausgedrückt, muss das Applikationsprotokoll sicherstellen, dass A bei den Berechnungen seinen geheimen Schlüssel verwendet.

[0041] Die Sicherheit des neuen Schemas hängt von den Signaturgleichungen ab. Beispielsweise lautet die Signaturgleichung in Schema 1:

$$1 = cf + c_A a \pmod q. \qquad (1)$$

[0042] Im Folgenden wird gezeigt werden, dass bei einer gewissen Wahl der Einwegfunktion ($F(\gamma_A, I_A)$ das neue Schema 1 äquivalent ist mit DSA.

[0043] Angenommen CA verwendet die DSA-Signaturgleichung, um die Identität I_A der Partei A zu signieren. Zunächst wählt CA zufällig ein c_A und berechnet $\gamma_A = \alpha^{cA}$ mod p, dann verwendet CA eine sichere Hash-Funktion h, um $h(I_A)$ zu berechnen, schließlich löst CA die Gleichung für s.

$$h(I_A) \equiv c\gamma_A + c_A s \pmod q. \qquad (2)$$

[0044] Hierbei ist (γ_A,s) CAs Signatur auf I_A.

[0045] Durch Multiplikation der Gleichung (2) mit $h(I_A)^{-1}$ erhalten wir

$$1 \equiv c\gamma_A\, h(I_A)^{-1} + c_A s h(I_A)^{-1} \pmod q$$

[0046] Sei $F(\gamma_A,I_A) = \gamma_A h(I_A)^{-1}$ und ersetze in obiger Gleichung $sh(I_A)^{-1}$ durch a, so ergibt dies Gleichung (1). Gleichung (2) ist offensichtlich äquivalent mit Gleichung (1), wenn $F(\gamma_A, I_A) = \gamma_a h(I_A)^{-1}$. Das bedeutet, dass wenn jemand in der Lage ist, das die Signaturgleichung (1) verwendende Schema zu brechen, so kann er auch das Schema brechen, das die Signaturgleichung (2) verwendet und bei dem es sich um ein DSA-Schema handelt.

[0047] Heuristische Argumente legen nahe, dass unsere neuen Schemata für eine geeignete Auswahl von $F(\gamma_A,I_A)$, mit $F(\gamma_A,I_A) = \gamma_A h(I_A)$ oder $F(\gamma_A,I_A) = h(\gamma_A,I_A)$, sicher sind. Man bemerke, dass $F(\gamma_A,I_A)$ auch ein anderes Format aufweisen kann, wenn z.B. I_A klein ist, z.B. 20 Bits, q jedoch mehr als 180 Bits enthält, so kann $F(\gamma_A,I_A)=\gamma_A + I_A$ verwendet werden. Ein Nachteil der neuen Schemata besteht darin, dass alle Benutzer und die CA die gleiche Feldgröße verwenden. Allerdings arbeiten alle Schemata mit ID-basiertem implizit zertifiziertem Public-Key auf diese Weise, z.B. Girаults RSA-basiertes Diffie-Hellman-Schlüsselübereinstimmungsschema.

[0048] Ein weiterer Satz Schemata kann auch folgendermaßen beschrieben werden:
System-Setup: Eine vertrauenswürdige Partei CA wählt eine geeignete Primzahl p mit p=tq+1, wobei q eine große Primzahl ist, und einen Generator α der Ordnung q. Die Partei CA wählt eine zufällige ganze Zahl c, mit 1<c<q als ihren geheimen Schlüssel, berechnet den öffentlichen Schlüssel $\beta=\alpha^c$ mod p und veröffentlicht

(β,α,p,q). Dann wählt CA eine spezielle kryptographische Funktion f = $F(\gamma_A I_A, OP)(f:\{0,1\}^* \rightarrow \{1,2,...(q-1)\})$, so dass mit dieser Funktion das Signaturschema, das zur Erzeugung des impliziten Zertifikates verwendet wird, sicher ist, wobei OP für Optionsparameter steht, die den Benutzer betreffen (z.B. das Datum, oder β, der öffentliche Schlüssel der Partei CA). Beispielsweise sei h eine sichere Hash-Funktion, f kann eines der nachfolgend aufgeführten Formate aufweisen:

1. $F(\gamma_A I_A OP) = \gamma_A+\beta+h(I_A)$
2. $F(\gamma_A,I_A,OP) = h(\gamma_A \| \beta \| I_A)$
3. $F(\gamma_A,I_A,OP) = \gamma_A+\beta+I_A$ wobei I_A ein Muster aufweist (oder wenn I_A klein ist, z.B. 20 Bits, und q mehr als 180 Bits hat)
4. $F(\gamma_A I_A,OP) = \gamma_A+h(I_A)$
5. $F(\gamma_A,I_A,OP) = h(\gamma_A \| I_A)$
6. $F(\gamma_A,I_A,OP) = \gamma_A+I_A$ wobei I_A ein Muster aufweist (oder wenn I_A klein ist, z.B. 20 Bits, und q mehr als 180 Bits hat)
7. Die Parameter ein wenig zu verändern, um aus einem gegebenen sicheren Signaturschema ein sicheres Signaturschema zu erhalten, ist sehr leicht. So kann $F(\gamma_A,I_A,OP)$ jedes andere Format aufweisen, mit dem garantiert ist, dass das zur Erzeugung des impliziten Zertifikats verwendete Signaturschema sicher ist. Man bemerke, dass durch geeignete Wahl von $F(\gamma_A,I_A,OP)$ jedes bisher bekannte Elgamal-artige Signaturschema mit einem der hierin vorgeschlagenen 4 Schematafamilien äquivalent ist, wenn es als implizites Zertifikat-Schema nach Modifizierung verwendet wird. Jedoch besitzen die von uns vorgeschlagenen Schemata die größte Effizienz.

Anmerkung: In den nachfolgenden Schemata wird von dem obigen System-Setup ausgegangen.

Schema 1.a:

1. Für jede Entität A wählt CA einen eindeutigen unterscheidbaren Namen ("distinguished name") oder Identität I_A (z.B. Name, Adresse, Telefonnummer) sowie eine zufällige ganze Zahl c_A mit $1<c_A<q$. Dann berechnet $CA\ \gamma_A = \alpha^{c_A} \bmod p$. ($\gamma_A$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von A. (I_A, γ_A) dient als das implizite Zertifikat von A.)
2. CA berechnet $f = F(\gamma_A,I_A,OP)$ und löst die folgende Gleichung für a (wenn a = 0,1, c, $c_A^{-1}c$, so wählt CA ein anderes c_A und löst die Gleichung erneut).

$$1 = cf + c_A a \pmod q.$$

3. CA schickt das Tripel (γ_A, a, I_A), welches CAs Signatur auf I_A darstellt, gesichert an A. Hierbei ist a As geheimer Schlüssel, γ_A ist As Generator und $\gamma_A^a\ (= \alpha^{c_A a})$ ist As öffentlicher Schlüssel. A veröffentlicht ($\alpha,I_A,\beta,\gamma_A,p,q$) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\gamma_A^a\ (= \alpha\beta^{-f})\ (mod\ p)$$

Anmerkung:

1. In Schritt 1 kann die Identität I_A von der Entität A gewählt werden.
2. In Schritt 2 sei a=0,1 ausgeschlossen, da sonst jedermann mit Leichtigkeit den geheimen Schlüssel der Partei A ermitteln kann. Insbesondere wenn a=0, $c_A^{-1}c$, kann CAs geheimer Schlüssel c von jedermann aus $1=cf \pmod q$ berechnet werden.
3. Bei diesem Schema hat jeder Benutzer einen unterschiedlichen Systemgenerator γ_A.

Schema 1.b:

1. Für jede Entität A wählt CA eine(n) eindeutige(n) unterscheidbaren Namen oder Identität I_A (z.B. Name, Adresse, Telefonnummer) sowie eine zufällige ganze Zahl c_A mit $1<c_A<q$. Danach berechnet $CA\ \gamma_A = \alpha^{c_A} \bmod p$. ($\gamma_A$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels der Partei A. (I_A, γ_A) dient als As implizites Zertifikat.)
2. CA berechnet $f = F(\gamma_A,I_A, OP)$ und löst die folgende Gleichung für a(wenn a=0,1,c, so wählt CA ein anderes c_A und löst die Gleichung erneut).

$$1 \equiv ca + c_A f \pmod q$$

3. CA schickt das Tripel (γ_A, a, I_A), bei dem es sich um CAs Signatur auf I_A handelt, sicher an A. Hierbei ist a der geheime Schlüssel von A, β der Generator von A und β^a der öffentliche Schlüssel von A. A veröffent-

licht (α, I_A,β, γ_A, p, q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta^{a} = \alpha\gamma_{A}^{-f} \qquad (mod\ p)$$

Anmerkung:

1. In Schritt 1 kann die Identität I_A von der Entität A gewählt werden.
2. In Schritt 2 sei a=0,1 ausgeschlossen, da sonst jedermann mit Leichtigkeit den geheimen Schlüssel der Partei A ermitteln kann. Ist a=0, so ist CA nicht in dem Zertifikat involviert.
3. Bei diesem Schema hat jeder Benutzer den gleichen Systemgenerator β.

Schema 1.c:

1. Für jede Entität A wählt CA eine(n) eindeutige(n) unterscheidbaren Namen oder Identität I_A (z.B. Name, Adresse, Telefonnummer) sowie eine zufällige ganze Zahl c_A mit $1<c_A<q$. Danach berechnet $CA\ \gamma_A = \alpha^{c_A} \bmod p$. ($\gamma_A$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels der Partei A. (I_A, γ_A) dient als As implizites Zertifikat.)
2. CA berechnet f = F(γ_A,I_A, OP) und löst die folgende Gleichung für a (wenn a=0,1 oder c, so wählt CA ein anderes c_A und löst die Gleichung erneut).

$$a \equiv cf + c_A \qquad (mod\ q)$$

3. CA schickt das Tripel (γ_A, a, I_A), bei dem es sich um CAs Signatur auf I_A handelt, gesichert an A. Hierbei ist a der geheime Schlüssel von A, $_A$der Generator von A und α^a der öffentliche Schlüssel von A. A veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^{a} = \beta^{f}\gamma_A \qquad (mod\ p)$$

Anmerkung:

1. In Schritt 1 kann die Identität I_A von der Entität A gewählt werden.
2. In Schritt 2 sei a=0,1 ausgeschlossen, da sonst jedermann mit Leichtigkeit den geheimen Schlüssel der Partei A ermitteln kann.
3. Bei diesem Schema hat jeder Benutzer den gleichen Systemgenerator α.

Schema 1.d:

1. Für jede Entität A wählt CA eine(n) eindeutige(n) unterscheidbaren Namen oder Identität I_A (z.B. Name, Adresse, Telefonnummer) sowie eine zufällige ganze Zahl c_A mit $1<c_A<q$. Danach berechnet $CA\ \gamma_A = \alpha^{c_A} \bmod p$. ($\gamma_A$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels der Partei A. (I_A, γ_A) dient als As implizites Zertifikat.)
2. CA berechnet f=F(γ_A,I_A, OP) und löst die folgende Gleichung für a (wenn a=0,1 oder c, so wählt CA ein anderes c_A und löst die Gleichung erneut).

$$a \equiv c_A f + c \qquad (mod\ q)$$

3. CA schickt das Tripel (γ_A, a, I_A), bei dem es sich um CAs Signatur auf I_A handelt, gesichert an A. Hierbei ist a der geheime Schlüssel von A, α der Generator von A und α^a der öffentliche Schlüssel von A. A veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^{a} = \gamma_A^{f}\ \beta \qquad (mod\ p)$$

Anmerkung:

1. In Schritt 1 kann die Identität I_A von der Entität A gewählt werden.
2. In Schritt 2 sei a=0,1 ausgeschlossen, da sonst jedermann mit Leichtigkeit den geheimen Schlüssel der Partei A ermitteln kann.

3. Bei diesem Schema hat jeder Benutzer den gleichen Systemgenerator α.

[0049] Zwar kann jeder den öffentlichen Schlüssel der Partei A aus den öffentlichen Daten rekonstruieren, doch bedeutet dies nicht, dass der rekonstruierte öffentliche Schlüssel zertifiziert wurde. Um das Zertifikat explizit zu verifizieren, muss a bekannt sein. Sobald man a kennt, besteht das Verifizierungsverfahren darin, CAs Signatur auf I_A zu verifizieren. Wenn beispielsweise in Schema 1.a der Verifizierer $\alpha\beta^{-f}$ berechnet und ein Benutzer A γ_A^a unter Verwendung von a berechnet, so können diese das Zertifikat zusammen verifizieren. Der Verifizierer muss aber sicherstellen, dass der Benutzer A tatsächlich a kennt. Somit dient die Rekonstruktion des öffentlichen Schlüssels nur dann als implizite Verifizierung, wenn sie mit einem Applikationsprotokoll kombiniert ist, welches zeigt, dass der Benutzer A den entsprechenden geheimen Schlüssel vollständig kennt. Generell kann das auf implizitem Zertifikat basierende Schema mit jedem Public-Key-Schema verwendet werden, das die Subjekt-Entität und den öffentlichen Schlüssel authentifizieren muss.

[0050] Im Folgenden soll dies durch Verwendung des DSA-Signaturschemas als System mit implizit zertifiziertem öffentlichen Schlüssel und des Schemas 1.a als Schema mit implizitem Zertifikat demonstriert werden.

[0051] Angenommen, Alice hat den geheimen Schlüssel a, den Generator γ_A und veröffentlicht $(\alpha, I_A, \beta, \gamma_A, p, q)$ in der Public Domain. Alice möchte nun eine Nachricht M unter Verwendung eines DSA signieren.

[0052] Alice geht wie folgt vor:

1. sie wählt zufällig ein k, berechnet $r = \gamma_A^x \pmod p$.
2. sie berechnet e=sha-1(M).
3. sie berechnet $s=x^{-1}(e+ar) \pmod q$.
4. Die Signatur auf M ist (r,s).

[0053] Der Verifizierer geht wie folgt vor:

1. er beschafft sich Alices öffentliche Daten $(\alpha, I_A, \beta, \gamma_A, p, q)$, berechnet f und rekonstruiert den öffentlichen Schlüssel

$$\beta_A = \gamma^a{}_A = \alpha\beta^{-f} \pmod p$$

2. berechnet e=sha-1(M).
3. berechnet $u_1 = es^{-1} \pmod q$ und $u_2 = rs^{-1} \pmod q$
4. berechnet $r' = \gamma_A^{u_1}\delta_A^{u_2} \pmod p$
5. ist r=r', so ist die Signatur verifiziert. Gleichzeit ist Alices (ID-basierter) öffentlicher Schlüssel implizit verifiziert.

[0054] Das Paar (I_A, γ_A) dient als Alices Zertifikat. Für DSA wissen wir, dass es sehr schwer ist, Alices Signatur zu fälschen, ohne a zu kennen. Die Rekonstruktion des öffentlichen Schlüssels dient dann als implizite Verifizierung, falls das Applikationsprotokoll mit gültig endet. Man erinnere, dass lediglich eine Potenzierungsoperation erforderlich ist, um den öffentlichen Schlüssel zu erhalten. Aus diesem Grunde lässt sich sagen, dass die Verifizierung des impliziten Zertifikats eine Potenzierungsoperation erfordert.

[0055] Die nachfolgenden Schemata mit implizitem Zertifikat können durch Modifizieren der obigen Schemata, derart dass sowohl CA als auch die Entität die Kontrolle über den geheimen Schlüssel der Entität haben, jedoch nur die betreffende Entität ihren geheimen Schlüssel kennt, abgeleitet werden.

[0056] In diesem Abschnitt ist ein weiterer Systemparameter H(*) erforderlich, wobei H(*) eine Kryptofunktion ist, bei der es sich um eine sichere Hash-Funktion oder Einweg-Funktion oder eine Identity-Map handeln kann.

Schema 2.a:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^{kc_A} \pmod p$ und $f = F(\gamma_A, I_A, OP)$, löst die Signaturgleichung für k_A (wenn $k_A = 0$ oder c, so wählt CA ein anderes c_A)

$$1 = cf + c_A k_A \pmod q$$

Dann berechnet $C\ \gamma_A^1 = \alpha^{c_A} \pmod p$ und schickt das Tripel (γ_A^1, k_A, I_A) an A.
3. A berechnet $a=k_A k^{-1} \pmod q$ (ist a = 1, so geht A zurück zu Schritt 1) und berechnet $\gamma_A = (\gamma_A^1)^k \pmod p$. Dann prüft A, ob $\gamma_A^a = \alpha\beta^{-f}$. Hierbei ist a der geheime Schlüssel von A, γ_A der Generator von A, und γ_A^a ist der

öffentliche Schlüssel von A. A veröffentlicht ($\alpha, I_A, \beta, \gamma_A, p, q$) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\gamma_A^a = \alpha\beta^f \pmod p$$

Schema 2.b:

5. A wählt zufällig eine ganze Zahl k und berechnet β^k, dann schickt A β^k an CA.
6. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \beta^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A, OP), löst die Signaturgleichung für k_A (ist k_A = 0, c, so wählt CA ein anderes c_A)

$$1 = ck_A + c_A f \pmod q.$$

Dann berechnet CA $\gamma_A^1 = (\beta^k)^{c_A c^{-1}} \pmod p$ und schickt das Tripel (γ_A^1, k_A, I_A) an A.
7. A berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \beta^k \pmod p$, $f = F(\gamma_A, I_A, OP)$ und a = k_A – k_f (mod q) (wenn a =0,1, so geht A zurück zu Schritt 1). Dann prüft A, ob $\beta^a = \alpha\gamma_A^f$. Hierbei ist a der geheime Schlüssel von A, β ist der Generator von A, und β^a ist der öffentliche Schlüssel von A. A veröffentlicht ($\alpha, I_A, \beta, \gamma_A, p, q$) in der Public Domain.
8. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta^a = \alpha\gamma_A^{-f} \pmod p$$

Schema 2.c:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A,OP), berechnet k_A (wenn k_A = c, so wählt CA ein anderes c_A)

$$k_A \equiv cf + c_A \pmod p$$

Dann berechnet CA $\gamma_A^1 = (\alpha^k)^{c_A} \pmod p$ und schickt das Tripel (γ_A^1, k_A, I_A) an A.
3. A berechnet a=k_A + k (mod q) (wenn a=0,1, so geht A zurück zu Schritt 1) und berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \alpha^k \pmod p$. Danach prüft A, ob $\alpha^a = \beta^f \gamma_A$. Hierbei ist a der geheime Schlüssel von A, α ist der Generator von A, und α^a ist der öffentliche Schlüssel von A. A veröffentlicht ($\alpha, I_A, \beta, \gamma_A, p, q$) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \pmod p$$

Schema 2.d:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A, OP), berechnet k_A (wenn k_A = c_A, so wählt CA ein anderes c_A)

$$k_A \equiv c_A f + c \pmod q$$

Dann berechnet CA $\gamma_A^1 = (\alpha^k)^{c_A} \pmod p$ und schickt das Tripel (γ_A^1, k_A, I_A) an A.
3. A berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \alpha^k \pmod p$, $f = F(\gamma_A, I_A, OP)$, und a = k_A + k_f (mod q). (Wenn a = 0,1, so geht A zurück zu Schritt 1). Danach prüft A, ob $\alpha^a = \gamma_A^f \beta$ Hierbei ist a der geheime Schlüssel von A, α ist der Generator von A, und α^a ist der öffentliche Schlüssel von A. A veröffentlicht ($\alpha, I_A, \beta, \gamma_A, p, q$) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \gamma_A^f \beta \pmod p$$

Anmerkungen: (zu den Schemata 2.a, 2.b, 2.c, 2.d)

1. Die Identität I_A kann entweder von CA oder der Entität A gewählt werden.
2. CA sollte die Entität A authentifizieren. Dies kann durch Anwesenheit vor CA geschehen, oder über einen sicheren Kanal oder über die Stimme (z.B. durch Telefon), oder durch das folgende Verfahren:
Anstatt das Tripel (γ_A^1, k_A, I_A) an A zu senden, schickt CA in Schritt 2 zuerst γ_A^1 an A. A berechnet γ_A, setzt

K=H(γ_A), verschlüsselt die Authentifizierungsinformation A_{AI} von A (z.B. VISA-Information) durch DES (oder ein anderes auf einem symmetrischen Schlüssel basierendes System, "Symmetric-Key-System") und schickt $DES_K(A_{AI})$ an CA. CA entschlüsselt $DES_K(A_{AI})$, um (A_{AI}) zu erhalten. Nach Prüfung der Validität von (A_{AI}) schickt CA (k_A,I_A) an A.
3. (γ_A^1, k_A, I_A) kann über einen öffentlichen Kanal geschickt werden.

[0057] In den obigen Schemata 2.a-2.d, werden die Schemata mit implizitem Zertifikat durch die Subjekt-Entität und die CA beendet. Jedes Schema ist im Wesentlichen in zwei Teile aufgeteilt: der Schlüsselaustauschteil und der Signaturteil. Eine Funktion des Schlüsselaustauschteiles besteht in der Übertragung der Information des impliziten Zertifikats von CA an A über einen öffentlichen Kanal (hierauf wird in Abschnitt 6 näher eingegangen). Um die Berechnung der obigen Schemata zu beschleunigen, kann der Schlüsselaustauschteil modifiziert werden. Die nachfolgenden Schemata 3.a-3.d werden durch Modifizierung der Schemata 2.a-2.d erhalten. Der Vorteil von Schemata 3.a-3.d besteht darin, dass der Benutzer A, daß fix ist, in der Lage ist, K zu berechnen, bevor er von CA eine Antwort erhält. Hier handelt es sich um eine vorteilhafte Eigenschaft, insbesondere für den Online-Fall.

Schema 3.a:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^{kc_A} \pmod p$ und f = F(γ_A,I_A, OP), löst die Signaturgleichung für k_A (wenn k_A = 0, so wählt CA ein anders c_A)

$$1 = cf + c_A k_A \pmod q$$

Als nächstes berechnet CA K = H($(\alpha^k)^c$) und $\overline{k}_A = DES_K(k_A)$, dann schickt CA das Tripel (γ_A,$\overline{k}_A$,I_A) an A.
3. A berechnet K = H(β^k), $k_A = DES_K(k_A)$ und $a=k_A k^{-1}$ (mod q). (Wenn a=1, so geht A zurück zu Schritt 1.) Dann prüft A, ob $\gamma_A^a = a\beta^{-f}$. Hierbei ist a der geheime Schlüssel von A, γ_A ist der Generator von A, und γ_A^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\gamma_A^a = \alpha\beta^{-f} \pmod p$$

Schema 3.b:

1. A wählt zufällig eine ganze Zahl k und berechnet β^k, dann schickt A β^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \beta^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A,OP), löst die Signaturgleichung für k_A (wenn k_A = 0, so wählt CA ein anderes c_A)

$$1 = ck_A + c_A f \pmod q.$$

Als nächstes berechnet CA $K = H((\beta^k)^{c_A c^{-1}}) = H(\alpha^{kc_A})$ und $\overline{k}_A = DES_K(k_A)$ und schickt dann das Tripel (γ_A, $\overline{k}_A$, I_A) an A.
3. A berechnet $K = H((\gamma_A \mid \beta^k)^k) = H(\alpha^{kc_A})$, $k_A = DES_K(\overline{k}_A)$, $f = F(\gamma_A, I_A, OP)$ und berechnet a = k_A – kf (mod q) (wenn a = 0,1, so geht A zurück zu Schritt 1). Dann prüft A, ob $\beta^a = \alpha\gamma_A^{-f}$
Hierbei ist a der geheime Schlüssel von A, β ist der Generator von A, und β^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta^a = \alpha\gamma_A^{-f} \pmod p$$

Anmerkung: (zu Schema 3.b)

1. Die Identität I_A kann entweder von CA oder der Entität A gewählt werden.
2. CA sollte die Entität A authentifizieren. Dies kann durch Anwesenheit vor CA geschehen, oder über einen sicheren Kanal oder über die Stimme (z.B. durch Telefon), oder durch das folgende Verfahren:
Anstatt das Tripel (γ_A, $\overline{k}_A$, I_A) an A zu senden, schickt CA in Schritt 2 zuerst γ_A an A. A berechnet $K = H((\gamma_A \mid \beta^k)^k) = H(\alpha^{kc_A})$, verschlüsselt die Authentifizierungsinformation A_{AI} von A (z.B. VISA-Information) durch DES (oder ein anderes mit einem symmetrischen Schlüssel arbeitendes System, "Symmetric-Key-System") und schickt $DES_K(A_{AI})$ an CA. CA entschlüsselt $DES_K(A_{AI})$, um A_{AI} zu erhalten. Nach Prüfung der Validität von A_{AI} schickt CA(k_A, I_A) an A.

3. (γ_A,$\overline{k}_A$,I_A) kann über einen öffentlichen Kanal geschickt werden.

Schema 3.c:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A, OP), berechnet k_A (wenn k_A = 0, so wählt CA ein anderes c_A)

$$k_A \equiv cf + c_A \quad (mod\ p)$$

Als nächstes berechnet CA K = H((α^k)^c) und $\overline{k}_A = DES_K(k_A)$, dann schickt CA das Tripel (γ_A,$\overline{k}_A$,I_A) an A.
3. A berechnet K = H(β^k), $k_A = DES_K(k_A)$ und a=k_A+k (mod q) (wenn a=0,1, zurück zu Schritt 1.) Dann prüft A, ob $\alpha^a = \beta^f \gamma_A$ Hierbei ist a der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_A,β,γ_A,p,q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \quad (mod\ p)$$

Schema 3.d:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A, OP), berechnet k_A (wenn k_A = 0, so wählt CA ein anderes c_A)

$$k_A \equiv c_A f + c \quad (mod\ q)$$

Als nächstes berechnet CA K = H((α^k)^c) und $\overline{k}_A = DES_K(k_A)$, dann schickt CA das Tripel (γ_A,$\overline{k}_A$,I_A) an A.
3. A berechnet K = H(β^k), $k_A = DES_K(k_A)$, f = F(γ_A,I_A, OP) und a=k_A+kf (mod q) (wenn a=0,1, zurück zu Schritt 1.) Dann prüft A, ob $\alpha^a = \gamma_A^f \beta$. Hierbei ist α der geheime Schlüssel von A, α ist der Generator von A und α^a ist der öffentliche Schlüssel von A. A veröffentlicht (α,I_Aβ,γ_A,p,q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \gamma_A^f \beta \quad (mod\ p)$$

Anmerkungen: (zu Schemata 3.a, 3.c, 2.d)

1. Die Identität I_A kann entweder von CA oder der Entität A gewählt werden.
2. CA sollte die Entität A authentifizieren. Dies kann entweder durch Anwesenheit vor CA geschehen, oder über einen sicheren Kanal oder über die Stimme (z.B. durch Telefon), oder durch das folgende Verfahren: In Schritt 1 berechnet A α^k und K=H(β^k) und schickt dann α^k und $DES_K(A_{AI})$ an CA. CA berechnet K = H((α^k)^c) und entschlüsselt $DES_K(A_{A1})$, um (A_{A1}) zu erhalten. Nach Prüfung der Validität von A_{A1} fährt CA mit Schritt 2 fort.
3. (γ_A,k_A,I_A) kann über einen öffentlichen Kanal geschickt werden.

[0058] Die Vorteile der Schemata 3.a, 3.c und 3.d liegen darin, dass der Benutzer A problemlos in der Lage ist, K zu berechnen, da β fix ist, und dass k_A so verschlüsselt ist, dass andere keine Kenntnis von k_A haben können. Tatsächlich wird durch die öffentliche Bekanntheit von k_A die Sicherheit des Zertifikatsschemas nicht vermindert. Zweck der Verschlüsselung von k_A ist es, sicherzustellen, dass k der Entität bekannt ist. Somit kann bei den Schemata 3.a-3.d der DES-Verschlüsselungsteil entfallen und $\overline{k}_A$ durch k_A ersetzt werden, vorausgesetzt, dass das Zertifikatsschema das in Anmerkung 2 beschriebene Verfahren anwendet.

[0059] Um in den obigen Schemata an Übertragungsbandbreite einzusparen, können die Schemata durch Senden von f = F(γ_A,I_A, OP) anstelle von γ_A modifiziert werden. (Man bemerke, dass die Größe von γ_A im Allgemeinen größer ist als 160 Bits, während f lediglich 160 Bits enthält.) Das nachfolgende Schema 4.c ist eine Modifizierung von Schema 3.c.

Schema 4.c:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} \pmod p$ und f = F(γ_A,I_A, OP), berechnet k_A

(wenn $k_A = 0$, so wählt CA ein anderes c_A)

$$k_A \equiv cf + c_A \quad (mod\ q)$$

Als nächstes berechnet CA K = $H((\alpha^k)^c)$ und $\overline{k}_A = DES_K(k_A)$ und schickt dann das Tripel (f, $\overline{k}_A$, I_A) an A.
3. A berechnet K = $H(\beta^k)$, $k_A = DES_K(k_A)$ und a = k_A + k (mod q) (wenn a = 0,1, so geht A zurück zu Schritt 1.) Dann berechnet A $\gamma_A = \alpha^a\beta^{-f}$ (mod p) und prüft, ob f = $F(\gamma_A, I_A, OP)$. Hierbei ist a der geheime Schlüssel von A, α der Generator von A und α^a der öffentliche Schlüssel von A. A veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \quad (mod\ p)$$

[0060] Des Weiteren kann die Bandbreite durch das folgende Schema 5.c vermindert werden:

Schema 5.c:

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA.
2. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} (\mathrm{mod}\ p)$ und setzt $\Diamond_A$ als die ersten 80 niedrigstwertigen Bits von γ_A. Dann berechnet CA f = $F(\Diamond_A, I_A, OP)$ und k_A (wenn $k_A = 0$, so wählt CA ein anderes c_A)

$$k_A \equiv cf + c_A \quad (mod\ q)$$

Als nächstes berechnet CA K = $(\alpha^k)^c$ (mod p) und $\overline{k}_A = DES_K(k_A)$ und schickt das Tripel ($\Diamond_A$,$\overline{k}_A$,I_A) an A.
Anmerkung: ($\Diamond_A$,$\overline{k}_A$,I_A) kann über einen öffentlichen Kanal geschickt werden.
3. A berechnet K = β^k (mod p), $k_A = DES_K(\overline{k}_A)$ und a = k_A + k (mod q) (wenn a = 0,1, so geht A zurück zu Schritt 1.) Dann berechnet A f = $F(\Diamond_A, \beta, I_A)$, $\gamma_A = \alpha^a\beta^{-f}$ (mod p), $\gamma_A = \alpha^a\beta^{-f}$ (mod p) und prüft, ob die ersten 80 niedrigstwertigen Bits von γ_A gleich $\Diamond_A$ sind. Hierbei ist a der geheime Schlüssel von A, α der Generator von A und α^a der öffentliche Schlüssel von A. A veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain.
4. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\alpha^a = \beta^f \gamma_A \quad (mod\ p)$$

[0061] Das Sicherheitsniveau 5.c ist nicht so hoch wie das anderer, oben besprochener Schemata. Schema 5.c hat nur eine 80-Bit-Sicherheit, die jedoch derzeit für praktische Anwendungen ausreicht. Die ersten 80 niedrigstwertigen Bits können auf die niedrigstwertigen Bits bis zur Hälfte der Bits von γ_A erweitert werden.

[0062] Das implizite Zertifikat kann zur Zertifizierung anderer nützlicher Informationen verwendet werden, und zwar durch Aufnahme der Information in den Optionsparameter OP. Beispielsweise $OP = \alpha^{a_E} \| OP_2$, wobei a_E ein anderer geheimer Schlüssel des Benutzers A und α^{a_E} der korrespondierende öffentliche Schlüssel ist. Das folgende Schema 6.c stellt eine Modifikation des Schemas 2.c dar. Andere Schemata können in der gleichen Weise modifiziert werden.

Schema 6.c:

1. A wählt zufällig eine ganze Zahl a_E und berechnet α^{a_E}.
2. A wählt zufällig eine ganze Zahl k und berechnet a^k, dann schickt A α^k und α^{a_E} an CA.
3. CA wählt zufällig eine ganze Zahl c_A, berechnet $\gamma_A = \alpha^k \alpha^{c_A} (\mathrm{mod}\ p)$ und $f = F(\gamma_A, I_A, \alpha^{a_E}, OP_2)$, (zum Beispiel $F(\gamma_A, I_A, \alpha^{a_E}, OP_2) = h(\gamma_A \| I_A \| \alpha^{a_E})$), berechnet k_A (wenn $k_A = 0$, so wählt CA ein anderes c_A)

$$k_A \equiv cf + c_A \quad (mod\ q)$$

Dann berechnet CA $\gamma_A^1 = (\alpha^k)^{c_A} (\mathrm{mod}\ p)$ und schickt das Tripel (γ_A^1, k_A, I_A) an A.
4. A berechnet a=k_A+k (mod q) (wenn a = 0,1, so geht A zurück zu Schritt 1) und berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \alpha^k (\mathrm{mod}\ q)$. Dann prüft A, ob $\alpha^a = \beta^f\gamma_A$. Hierbei ist a der geheime Signierschlüssel von A, α der Generator von A und α^a der öffentliche Signierschlüssel von A. α_E ist der geheime Verschlüsselungsschlüssel von A α^{a_E} und der öffentliche Verschlüsselungsschlüssel von A. A veröffentlicht (α, α^{a_E}, I_A, β, γ_A, p, q) in der Public Domain.
5. Jeder kann den (ID-basierten) implizit zertifizierten öffentlichen Schlüssel der Partei A aus der Public Do-

main erhalten, und zwar durch Berechnen von

$$\alpha^{a} = \beta^{f} \gamma_{A} \quad (\mathrm{mod}\ p)$$

Anmerkung: (zu den Schemata 4.c, 5.e, 6.c)

1. Die Identität I_A kann von CA oder durch die Entität A gewählt werden.
2. CA sollte die Entität A authentifizieren. Dies kann durch das in der Anmerkung 2 zu Schema 3.c beschriebene Verfahren erfolgen.
($f, \bar{k}_A, I_A$) oder ($\Diamond A, \bar{k}_A, I_A$) oder ($\gamma_A^1, k_A, I_A$) können über einen öffentlichen Kanal geschickt werden.

CA-Verkettungsschema

[0063] zur Implementierung einer CA-Verkettungsstruktur, d.h. CA1 authentifiziert CA2, CA2 authentifiziert CA3 und CA3 authentifiziert den Benutzer A. In diesem Abschnitt wird ein Beispiel mit 3 CAs in der CA-Kette beschrieben. Wir verwenden das Grundschema 3', um dieses Beispiel zu demonstrieren.

System-Setup:

[0064] Die Partei CA1, die das höchste Vertrauen genießt, wählt eine geeignete Primzahl p mit p=tq +1, wobei q eine große Primzahl ist, und einen Generator α der Ordnung q. CA1 wählt eine zufällige ganze Zahl c_1, mit $1 \le c_1 \le q-1$ als ihren geheimen Schlüssel, dann berechnet CA1 den öffentlichen Schlüssel $\beta_1 = \alpha^{c_1} \bmod p$ und veröffentlicht (β_1, α, p, q).

Phase 1. CA2 beantragt einen implizit zertifizierten öffentlichen bei CA1

1. CA2 wählt zufällig eine ganze Zahl k_2 und berechnet α^{k_2}, dann schickt CA2 α^{k_2} an CA1.
2. CA1 wählt eine(n) eindeutige(n) unterscheidbare(n) Namen oder Identität I_{CA2} und eine zufällige ganze Zahl c_{CA2}, mit $1 \le c_{CA2} \le q-1$. Dann berechnet CA1 $\gamma_{CA2} = \alpha^{k_2} \alpha^{c_{CA2}} \pmod p$ (γ_{CA2} sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von CA2).
3. CA1 wählt eine Funktion $f_1 = F(\gamma_{CA2}, I_{CA2})$ und berechnet k_{CA2} (wenn $k_{CA2}=0$, so wählt CA1 in Schritt 2 ein anderes c_{CA2} und führt eine neue Berechnung von k_{CA2} durch).

$$k_{CA2} \equiv c_1 f_1 + c_{CA2} \quad (mod\ q)$$

4. CA1 berechnet $\gamma_{CA2}^1 = (\alpha^{k_2})^{c_{CA2}} (\mathrm{mod}\ p)$ und schickt das Tripel (γ_{CA2}^1, k_{CA2}, I_{CA2}) an CA2.
5. CA2 berechnet $\gamma_{CA2} = (\gamma_{CA2}^1)^{k_2^{-1}} \alpha^{k_2} (\mathrm{mod}\ p)$. Somit ist $c_2 = k_{CA2} + k_2 \pmod q$ der geheime Schlüssel von CA2, α ist der Generator von CA2 und $\beta_2 = \alpha^{c_2}$ ist der öffentliche Schlüssel von CA2. CA2 veröffentlicht (α, I_{CA2}, β_1, β_2, γ_{CA2}, p, q) in der Public Domain.
Anmerkung: Vertraut ein Benutzer CA2, so kann er β_2 direkt verwenden.
6. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei CA2 aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta_2 = \alpha^{c_2} = \beta_1^{f_1} \gamma_{CA2} (\mathrm{mod}\ p)$$

Phase 2. CA3 fordert bei CA2 einen implizit zertifizierten Schlüssel an

1. CA3 wählt zufällig eine ganze Zahl k_3 und berechnet α^{k_3}, und schickt α^{k_3} dann an CA2.
2. CA2 wählt eine(n) eindeutige(n) unterscheidbare(n) Namen oder Identität I_{CA3} und eine zufällige ganze Zahl c_{CA3} mit $1 \le c_{CA3} \le q-1$. Dann berechnet CA2 $\gamma_{CA3} = \alpha^{k_3} \alpha^{c_{CA3}} (\mathrm{mod}\ p)$. ($\gamma_{CA3}$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von CA3.)
3. CA2 wählt eine Funktion $f_2 = F(\gamma_{CA3}, I_{CA3})$ und berechnet k_{CA3} (wenn $k_{CA3}=0$, so wählt CA2 in Schritt 2 ein anderes c_{CA3} und führt eine neue Berechnung von k_{CA3} durch).

$$k_{CA3} \equiv c_2 f_2 + c_{CA3} \quad (mod\ q)$$

4. CA2 berechnet $\gamma_{CA3}^1 = (\alpha^{k_3})^{c_{CA3}} (\mathrm{mod}\ p)$ und schickt das Tripel (γ_{CA3}^1, k_{CA3}, I_{CA3}) an CA3.
5. CA3 berechnet $\gamma_{CA3} = (\gamma_{CA3}^1)^{k_3^{-1}} \alpha^{k_3} (\mathrm{mod}\ p)$. Somit ist $c_3 = k_{CA3} + k_3 \pmod q$ CA3s geheimer Schlüssel, α ist CA3s Generator und $\beta_3 = \alpha^{c_3}$ ist CA3s öffentlicher Schlüssel. CA3 veröffentlicht (α, I_{CA3},β_2, β_3, γ_{CA3}, p, q) in der Public Domain.
Anmerkung: Vertraut eine Entität CA3, so kann sie β3 direkt verwenden.
6. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei CA3 aus der Public

Domain erhalten, und zwar durch Berechnen von

$$\beta_3 = \alpha^{c_3} = \beta_2^{f_2} \gamma_{CA3} \pmod{p}$$

Phase 3. Der Benutzer A beantragt bei CA3 einen implizit zertifizierten öffentlichen Schlüssel.

1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA3.
2. CA3 wählt einen eindeutigen unterscheidbaren Namen oder Identität I_A und eine zufällige ganze Zahl c_A mit $1 \le c_A \le q-1$. Dann berechnet $CA3\ \gamma_A = \alpha^k \alpha^{c_A} \pmod{p}$. ($\gamma_A$ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von A.)
3. CA3 wählt eine sorgfältig ausgewählte Funktion $f_3=F(\gamma_A, I_A)$ und berechnet k_A (wenn $k_A=0$, so wählt CA3 in Schritt 2 ein anderes c_A und führt eine neue Berechnung von k_A durch).

$$k_A \equiv c_3 f_3 + c_A \quad (mod\ q)$$

4. CA3 berechnet $\gamma_A^1 = (\alpha^k)^{c_A} \pmod{p}$ und schickt das Tripel (γ_A^1,k_A,I_A) an A.
5. A berechnet $\gamma_A = (\gamma_A^1)^{k^{-1}} \alpha^k \pmod{p}$. Somit ist $a = k_A + k \pmod{q}$ As geheimer Schlüssel, α ist As Generator und $\beta_A = \alpha^a$ ist As öffentlicher Schlüssel. A veröffentlicht (α, I_A, β_3, β_A, γ_A, p, q) in der Public Domain.
Anmerkung: Vertraut ein Benutzer A, so kann er β_A direkt verwenden.
6. Jeder kann den (ID-basierten) implizit verifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von

$$\beta_A = \alpha^a = \beta_3^{f_3} \gamma_A \pmod{p}$$

Phase 4. Die Signatur des Benutzers A und Verifizierung

[0065] Zur Signierung einer Nachricht M geht der Benutzer A wie folgt vor:
1. A wählt zufällig x, berechnet $r=\alpha^x \pmod{p}$.
2. A berechnet $e=f_A= F(r, M)$, wobei F eine feste Funktion ist.
3. A berechnet $s=ae+x \pmod{q}$
4. Die Signatur auf M ist (r,s).

[0066] Der Verifizierer geht wie folgt vor:
1. Er holt sich die öffentlichen Daten von CA1, CA2, CA3 und die des Anwenders A

$$(\alpha, I_{CA2}, I_{CA3}, I_A, \beta_1, \beta_2, \beta_3, \beta_A, \gamma_{CA2}, \gamma_{CA3}, \gamma_A, p, q)$$

2. er rekonstruiert As öffentlichen Schlüssel

$$\beta_A = \beta^{f_1 f_2 f_3} \gamma_{CA}^{f_2 f_3} \gamma_{CA}^{f_3} \gamma_A \quad (mod\ p)$$

3. berechnet $e=f_A=F(r, M)$
4. berechnet $r' = \alpha^s \beta_A^{-e} \pmod{p}$
5. wenn r=r', so ist die die Signatur verifiziert. Gleichzeitig ist der (ID-basierte) öffentliche Schlüssel von CA2, CA3 und des Benutzers A implizit verifiziert.

[0067] Die Rekonstruktion des öffentlichen Schlüssels des Benutzers A erfordert nur 3 Potenzierungsoperationen mit bekannter Basis sowie 3 Multiplikationsoperationen. Ist die Signatur gültig, so ist der (ID-basierte) öffentliche Schlüssel von CA2, von CA3 und des Benutzers A implizit verifiziert.

Anmerkungen:

1. Vertraut der Verifizierer der Partei A, so ist As öffentlicher Schlüssel β_A
2. Vertraut der Verifizierer der Partei CA3, so ist der öffentliche Rekonstruktionsschlüssel $\beta_A = \beta_3^{f_3} \gamma_A \pmod{p}$
3. Vertraut der Verifizierer der Partei CA2, so ist der öffentliche Rekonstruktionsschlüssel $\beta_A = \beta_2^{f_2 f_3} \gamma_{CA3}^{f_3} \gamma_A \pmod{p}$

Mitzeichnungsschema ("Co-signing Scheme")

[0068] Im Folgenden wird ein Schema beschrieben, dass es einer Mehrzahl von CAs erlaubt, EIN implizites Zertifikat zu signieren. Dies wird anhand des Falles erläutert, in dem drei CAs, ein Zertifikat mit Verwendung

des Grundschemas 3' mitzeichnen.

System-Setup:

[0069] CA1, CA2 und CA3 sollen gemeinsame Systemparameter aufweisen: (1) die Primzahl p mit p =tq+1, wobei q eine große Primzahl ist; (2) ein Generator α der Ordnung q; (3) eine sorgfältig ausgewählte Funktion $f = F(\gamma, (I_{A1} + I_{A2} + I_{A3}))$ Die Partei CA1 wählt ein zufällige ganze Zahl c_1, mit $1 \le c_1 \le q-1$ als ihren geheimen Schlüssel, berechnet dann den öffentlichen Schlüssel $\beta_1 = \alpha^{c_1} \bmod p$ und veröffentlicht (β_1,α, p, q). CA2 wählt eine zufällige Zahl c_2, mit $1 \le c_2 \le q-1$ als ihren geheimen Schlüssel, berechnet dann den öffentlichen Schlüssel $\beta_2 = \alpha^{c_2}$ und veröffentlicht (β_2,α, p, q). CA3 wählt eine zufällige ganze Zahl c_3, $1 \le c_3 \le q-1$ als ihren geheimen Schlüssel, berechnet dann den öffentlichen Schlüssel $\beta_3 = \alpha^{c_3} \bmod p$ und veröffentlicht (β_3,α, p, q).

[0070] Schritt 1. A wählt zufällig eine ganze Zahl k und berechnet α^k, dann schickt A α^k an CA1, CA2 und CA3.

[0071] Schritt 2. Die CAs tauschen Information aus und berechnen implizite Zertifikate.

Phase 1.

1. CA1 wählt eine(n) eindeutige(n) unterscheidbare(n) Namen oder Identität I_{A1} und eine zufällige ganze Zahl c_{A1} mit $1 \le c_{A1} \le q-1$, berechnet $\alpha^{c_{A1}}$ und schickt $(\alpha^{c_{A1}}, I_{A1})$ an CA2 und CA3.
2. CA2 wählt eine(n) eindeutige(n) unterscheidbare(n) Namen oder Identität I_{A2} und eine zufällige ganze Zahl c_{A2} mit $1 \le c_{A2} \le q-1$, berechnet $(\alpha^{c_{A2}}, I_{A2})$ und schickt $\alpha^{c_{A2}}$ an CA1 und CA3.
3. CA3 wählt eine(n) eindeutige(n) unterscheidbare(n) Namen oder Identität I_{A3} und eine zufällige ganze Zahl c_{A3} mit $1 \le c_{A3} \le q-1$, berechnet $(\alpha^{c_{A3}}, I_{A3})$ und schickt $\alpha^{c_{A3}}$ an CA1 und CA2.

Phase 2.

1. CA1 berechnet $\gamma = \alpha^k \alpha^{c_{A1}} \alpha^{c_{A2}} \alpha^{c_{A3}} \pmod p$. (γ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von A), berechnet $f=F(\gamma,(I_{A1}+I_{A2}+I_{A3}))$ und berechnet k_{A1} (wenn $k_{A1} = 0$, so geht C_{A1} zurück zu Phase 1)

$$k_{A1} \equiv c_1 f + c_{A1} \pmod q$$

CA1 berechnet $\gamma_{A1}^1 = (\alpha^k)^{c_{A1}} \pmod p$ und schickt das Tripel (γ_{A1}^1, k_{A1}, I_{A1}) an A.
2. CA2 berechnet $\gamma = \alpha^k \alpha^{c_{A1}} \alpha^{c_{A2}} \alpha^{c_{A3}} \pmod p$. (γ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von A), berechnet $f=F(\gamma,(I_{A1}+I_{A2}+I_{A3}))$ und berechnet k_{A2} (wenn $k_{A2} = 0$, so geht CA2 zurück zu Phase 1)

$$k_{A2} \equiv c_2 f + c_{A2} \pmod q$$

CA2 berechnet $\gamma_{A2}^1 = (\alpha^k)^{c_{A2}} \pmod p$ und schickt das Tripel (γ_{A2}^1, k_{A2}, I_{A2}) an A.
3. CA3 berechnet $\gamma = \alpha^k \alpha^{c_{A1}} \alpha^{c_{A2}} \alpha^{c_{A3}} \pmod p$ γ sind die öffentlichen Daten zur Rekonstruktion des öffentlichen Schlüssels von A), berechnet $f = F(\gamma, (I_{A1}+I_{A2}+I_{A3}))$ und berechnet k_{A3} (wenn $k_{A3} = 0$, so geht CA3 zurück zu Phase 1).

$$k_{A3} \equiv c_3 f + c_{A3} \pmod q$$

CA3 berechnet $\gamma_{A3}^1 = (\alpha^k)^{c_{A3}}$ und schickt das Tripel (γ_{A3}^1, k_{A3}, I_{A3}) an A.

[0072] Schritt 3A berechnet implizit co-zertifizierte geheime Schlüssel und die Information zur Public-Key-Rekonstruktion

1. A berechnet $a=k_{A1} + k_{A2} + k_{A3}+k \pmod q$. (Wenn a 0 oder 1 ist, geht A zurück zu Schritt 1.)
2. A berechnet $\gamma = (\gamma_{A1}^1 \gamma_{A2}^1 \gamma_{A3}^1)^{k^{-1}} \alpha^k \pmod p, f = F(\gamma, (I_{A1} + I_{A2} + I_{A3}))$. Dann verifiziert A, ob

$$\alpha^a = (\beta_1 \beta_2 \beta_3)^f \gamma \pmod p$$

3. Hierbei ist a der implizit co-zertifizierte geheime Schlüssel von A, α ist der Generator von A, $I_A = I_{A1}+I_{A2}+I_{A3}$ ist die gemeinsame ID von A und $(\beta_1\beta_2\beta_3)^f\gamma$ ist der implizit co-zertifizierte öffentliche Schlüssel von A.
4. A veröffentlicht ($\alpha, I_{A1}, I_{A2}, I_{A3}, \beta_1, \beta_2, \beta_3, \gamma, p, q$) in der Public Domain.
5. Jeder kann den (ID-basierten) implizit mitzertifizierten öffentlichen Schlüssel der Partei A aus der Public Domain erhalten, und zwar durch Berechnen von $(\beta_1,\beta_2,\beta_3)^f\gamma \pmod p$

Anwendungen

[0073] Die nachfolgenden Beispiele werden mit Bezug auf Schema 3 (oder Schema 7') als CAs Signaturgleichung beschrieben, da bei diesem Schema jeder den gleichen Generator hat. Jeder Benutzer kann eine andere CA haben, solange die CAs die gleichen Systemparameter (p,q,d) verwenden und jeder Benutzer die gleiche Generierung hat.

Setup:

CA1: Systemparameter (α, β_1, p,q,d)
Alice hat einen geheimen Schlüssel a und einen Generator α und veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain.

CA2: Systemparameter (α, β_2, p, q)
Bob hat einen geheimen Schlüssel b und einen Generator α und veröffentlicht (α, I_A, β, γ_A, p, q) in der Public Domain.

[0074] Um zu zeigen, wie das neue Schema zu verwenden ist, wird das MTI/C0-Schlüsselübereinstimmungsprotokoll angewandt.

[0075] Angenommen Alice und Bob möchten einen Schlüsselaustausch durchführen. Das MTI/C0-Protokoll

1. Alice rekonstruiert Bobs öffentlichen Schlüssel $\alpha^b = \beta^{F(\gamma_B, I_B)}\gamma_B$, wählt zufällig eine ganze Zahl x und berechnet $(a^b)^x$, dann schickt Alice $(\alpha^b)^x$ an Bob.
2. Bob rekonstruiert Alices öffentlichen Schlüssel $\alpha^a = \beta^{F(\gamma_A, I_A)}\gamma_A$, wählt zufällig eine ganze Zahl y und berechnet $(\alpha^a)^y$, dann schickt $(\alpha^a)^y$ an Alice.
3. Alice berechnet den gemeinsamen Schlüssel $K_A = (\alpha^{ay})^{xa^{-1}} = \alpha^{xy}$
4. Bob berechnet den gemeinsamen Schlüssel $K_B = (\alpha^{bx})^{yb^{-1}} = \alpha^{xy}$

[0076] Es handelt sich hierbei um ein Two-Pass-Protokoll. Bei dem erfindungsgemäßen Schema mit implizitem Zertifikat führt jede Partei nur drei Potenzierungsoperationen aus, um den gemeinsamen Schlüssel zu erhalten, während sie gleichzeitig eine Verifizierung der Authentifizierungsschlüssel-Übereinstimmung und des impliziten öffentlichen Schlüssels durchführen.

[0077] Die nachfolgenden Beispiele sind Beispiele für Schemata mit kombinierter Signatur und Verschlüsselung ("Signcryption Schemes"). Als CAs Signaturgleichung wird hier das Schema 3 (oder Schema 7) verwendet, da bei diesem Schema jeder den gleichen Generator hat. Für das danach folgende Schema wird Schema 13 als CAs Signaturgleichung verwendet. Bei allen in diesem Abschnitt aufgeführten Schemata kann jeder Benutzer eine andere CA haben, solange die CAs die gleichen Systemparameter (p,q,α) verwenden und jeder Benutzer den gleichen Generator hat.

Setup:

CA1: Systemparameter (α, β_1, p,q)
Alice: geheimer Schlüssel a, Generator α, und (α I_A, β_1, γ_A, p, q) in der Public Domain.
CA2: Systemparameter (α, β_2, p, q)
Bob: geheimer Schlüssel b, Generator α, und (α, I_B, β_2, γ_B, p, q) in der Public Domain.

[0078] Bob möchte Alice eine signierte und verschlüsselte Nachricht M schicken.

Signcryption-Protokoll (Protokoll mit kombinierter Signierung und Verschlüsselung) 1:

[0079] Angenommen, Bob möchte Alice eine signierte und verschlüsselte Nachricht M schicken: Bob geht wie folgt vor:

1. rekonstruiert Alices öffentlichen Schlüssel $\alpha^a = \beta^{F(\gamma_A, I_A)}\gamma_A \bmod p$
2. wählt zufällig eine ganze Zahl x und berechnet einen Schlüssel r = $(\alpha^a)^x$(mod p)
3. berechnet $C=DES_r(M)$
4. berechnet e = hash(C||I_A)
5. berechnet s=be+x(mod q)
6. schickt das Paar (C,s) an Alice, somit ist C die verschlüsselte Nachricht und s die Signatur.

[0080] Um die Nachricht wiederherzustellen, geht Alice wie folgt vor:

1. berechnet e = hash(C||I_A)
2. rekonstruiert Bobs öffentlichen Schlüssel $\alpha^b = \beta^{F(\gamma_B, I_B)}\gamma_B \bmod p$
3. berechnet $\alpha^{as}(\alpha^b)^{-ac}$ (mod p), was r ist
4. entschlüsselt die Nachricht M=DES_r(C)
5. prüft auf Redundanz

[0081] Somit führt Bob lediglich zwei und Alice drei Potenzierungsoperationen aus. Sowohl Alice als auch Bob sind jedoch bezüglich ihrer jeweiligen Authentifizierung sicher. Es sei darauf hingewiesen, dass bei diesem Schema die Nachricht M eine gewisse Redundanz oder ein gewisses Muster aufweisen muss.

Signcryption-Protokoll 2 (allgemeiner Fall):

Setup:

CA1: Systemparameter (α, β_1, p, q)
Alice: geheimer Schlüssel a, Generator α, und (α, I_A, β_1, γ_A, p, q) in der Public Domain.
CA2: Systemparameter (α, β_2, p, q)
Bob: geheimer Schlüssel b, Generator α, und (α, I_A, β_2, γ_B, p, q) in der Public Domain.

[0082] Anmerkung: Dieses Setup gilt für ein implizites Zertifikat. Für Schemata mit gewöhnlichem Zertifikat ist es lediglich erforderlich, dass Alice und Bob den gleichen Generator haben.

[0083] Zum kombinierten Signieren und Verschlüsseln einer Nachricht an Alice geht Bob wie folgt vor:
1. holt sich Alices öffentlichen Schlüssel α^a (im Falle eines Schemas mit implizitem Zertifikat rekonstruiert er Alices öffentlichen Schlüssel $\alpha^a = \beta_1^{F(\gamma_A, \beta_1, I_A)}\gamma_A \pmod p$
2. wählt zufällig eine ganze Zahl x und berechnet r = $(\alpha^a)^x$ (mod p)
3. berechnet C=DES_r(M)
4. berechnet e = hash(C||α^a)
5. berechnet s=be+x (mod q)
6. schickt (C,s) an Alice. C ist die verschlüsselte Nachricht und s die Signatur.

[0084] Um die Nachricht wiederherzustellen, geht Alice wie folgt vor:
1. berechnet e = hash(C||α^a)
2. holt sich Bobs öffentlichen Schlüssel α^b (im Falle eines Schemas mit implizitem Zertifikat rekonstruiert sie Bobs öffentlichen Schlüssel $\alpha^b = \beta_2^{F(\gamma_B, \beta_2, I_B)}\gamma_B \pmod p)$
3. berechnet α^{as} $(\alpha^b)^{-ae}$ (mod p), was r ist
4. entschlüsselt die Nachricht M=DES_r(C)

Anmerkung:

1. Wenn es sich bei dem Zertifikatsschema nicht um das hier beschriebene implizite Zertifikat handelt, so sollte Alices und Bobs öffentlicher Schlüssel verifiziert werden.
2. Die Nachricht M muss eine gewisse Redundanz oder ein gewisses Muster aufweisen.
3. Jeder, der einen Wert r kennt, ist in der Lage, jede Nachricht von Bob an Alice zu entschlüsseln, da der Wert α^{ab} offen gelegt wird.
4. Generell sollte ein Optionsparameter in den Hash e aufgenommen werden, d.h. e = hash(C||α^a||OP). Beispielsweise OP = α^b oder OP = $\alpha^b||\beta_i||\beta_2$

[0085] Die obigen Signcryption-Schemata haben den Nachteil, dass dann, wenn der Signierer seinen geheimen Signierschlüssel verlieren sollte, alle von dem Signierer gleichzeitig signiert und verschlüsselten Nachrichten dem Publikum offen gelegt werden. Um Nachrichten nach der Verschlüsselung zu schützen, wird hiermit ein neues Signcryption-Schema vorgeschlagen. Bei diesem neuen Schema verfügt jeder Benutzer über zwei Schlüsselpaare, davon ist ein Paar für den Signaturschlüssel und ein Paar ist der Verschlüsselungsschlüssel. Das neue Schema kann mit jedem Zertifikatsschema verwendet werden, jedoch ist es effizienter, wenn es mit dem erfindungsgemäßen Zertifikatsschema verwendet wird.

Signcryption-Protokoll 3 (allgemeiner Fall):

Setup:

Alice: geheimer Signierschlüssel a und geheimer Verschlüsselungsschlüssel a_E, Generator α, und (α, α^{a_E}, I_A, β_1, γ_A, p, q) in der Public Domain
CA2: Systemparameter (α, β_2, p, q)
Bob: geheimer Signierschlüssel b und geheimer Verschlüsselungsschlüssel b_E, Generator α, und (α, α^{b_E}, I_B, β_2, γ_B, p, q) in der Public Domain

[0086] Anmerkung: Dieses Setup gilt für ein implizites Zertifikat, welches das Schema 6.c verwendet. Für Systeme, die auf Schemata mit gewöhnlichem Zertifikat basieren, ist es lediglich erforderlich, dass Alice und Bob den gleichen Generator haben

[0087] Zur Signierverschlüsselung einer Nachricht an Alice geht Bob wie folgt vor:
1. Er beschafft sich Alices öffentlichen Signaturschlüssel α^a und den öffentlichen Verschlüsselungsschlüssel α^{a_E} im Falle eines Schemas mit implizitem Zertifikat rekonstruiert er Alices öffentlichen Signaturschlüssel $\alpha^a = \beta_1^{F(\gamma_A,\beta_1,I_A,\alpha^{a_E})}\gamma_A \pmod p$
2. wählt zufällig eine ganze Zahl x und berechnet $r = (\alpha^a \alpha^{a_E})^x \pmod p$
3. berechnet $C=DES_r(M)$
4. berechnet $e = hash(C \| \alpha^a \| \alpha^{a_E} \| \alpha^b \| \alpha^{b_E} \| OP.)$
5. berechnet $s = be + x + b_E \pmod q$
6. schickt (C,s) an Alice. C ist die verschlüsselte Nachricht und s die Signatur.

[0088] Um die Nachricht wiederherzustellen, geht Alice wie folgt vor:
1. Sie berechnet $e = hash(C \| \alpha^a \| \alpha^{a_E} \| \alpha^b \| \alpha^{b_E} \| OP)$
2. beschafft sich Bobs öffentlichen Signaturschlüssel α^b und den öffentlichen Verschlüsselungsschlüssel (im Falle eines Schemas mit implizitem Zertifikat rekonstruiert Alice Bobs öffentlichen Signaturschlüssel $\alpha^b = \beta_2^{F(\gamma_B,\beta_2,I_B,\alpha^{b_E})}\gamma_B \pmod p$).
3. berechnet $\alpha^{(a+a_E)s}(\alpha^b)^{-(a+a_E)e}\ \alpha^{-(a+a_E)b_E} \pmod p$, was gleich r ist
4. entschlüsselt die Nachricht $M=DES_r(C)$

Anmerkung:

1. Es ist denkbar, dass der geheime Schlüssel des Empfängers Alice $a+a_E$ lautet. Dies bedeutet, dass der Empfänger nur einen geheimen Schlüssel anstelle von zwei geheimen Schlüsseln benötigt. Der Sender Bob benötigt hingegen zwei geheime Schlüssel. Im Falle eines normalen Zertifikats benötigt der Empfänger nur einen einzigen privaten Schlüssel.
2. Wenn das Zertifikatsschema nicht das in dieser Anmeldung beschriebene implizite Zertifikat ist, so sollten Alices und Bobs öffentliche Schlüssel verifiziert werden.
3. Die Nachricht M muss eine gewisse Redundanz oder ein Muster aufweisen.
4. Der Parameter OP im Hash $e = hash(C \| \alpha^a \| \alpha^{a_E} \| \alpha^b \| \alpha^{b_E} \| OP)$ kann leer sein oder OP = $\beta_1 || \beta_2$
5. Die Kenntnis eines einzigen Wertes r enthüllt keine Information der späteren Nachrichten.
6. Bei einem Schema mit implizitem Zertifikat führt Bob lediglich 2 Potenzierungsoperationen durch und Alice 4 Potenzierungsoperationen. Alice und Bob sind sich jedoch beide sicher, dass es sich bei dem jeweils anderen um eine authentifizierte Partei handelt.
7. Kennt jemand Alices geheimen Schlüssel $a+a_E$ oder verliert Bob beide geheimen Schlüssel, so kann die Nachricht nach der Verschlüsselung nicht geschützt werden.

[0089] Bei normalen Signaturen besteht ein Problem darin, dass der Signierer die Signierung leugnet. Dies wird als Nichtanerkennung (Repudiation) bezeichnet. Die obigen Protokolle 1 und 2 weisen ein Nachweisbarkeitsmerkmal (Non-repudiation) auf, vorausgesetzt, dass dem Richter vertraut wird. Dies bedeutet, dass der Signierer nicht leugnen kann, dass er die signierte und verschlüsselte Nachricht signiert hat. Protokoll 3 hat auch dann ein Nachweisbarkeitsmerkmal, wenn der Richter kein Vertrauen genießt. Das nachfolgende Protokoll zeigt, wie ein Richter einen Fall entscheidet, in dem Bob die Signatur leugnen will.

Non-Repudiation-Protokoll (Protokoll mit Nachweisbarkeit):

1. Alice schickt (C,s) an den Richter
2. Der Richter berechnet $e = hash(C \| \alpha^a \| \ \alpha^{a_E} \| \alpha^b \| \alpha^{b_E} \| OP)\ und\ \alpha^x = \alpha^s(\alpha^b)^{-e}\alpha^{-b_E}$ (Anmerkung: Alices und Bobs zwei Paare

von öffentlichen Schlüsseln sollten verifiziert werden. Im Falle eines Schemas mit implizitem Zertifikat sollten die öffentlichen Schlüssel mit den öffentlichen Rekonstruktionsdaten rekonstruiert werden.)
3. Der Richter wählt zufällig zwei ganze Zahlen r_1 und r_2, berechnet $L = (\alpha^x)^{r_1}\alpha^{r_2}$ und schickt L an Alice.
4. Alice berechnet L^{a+a_E} und schickt dies zurück an den Richter
5. Der Richter berechnet $r = (L^{(a+a_E)}(\alpha^a\alpha^{a_E})^{-r_2})^{r_1^{-1}}$ und stellt die Nachricht durch $M=DES_r(C)$ her
6. Hat M das richtige Format, so muss (C,s) von Bob signiert und verschlüsselt werden.
7. Nachdem der Richter eine Entscheidung gefällt hat, schickt er die Werte $(\alpha^x, r_1, r_2, L, L^{a+a_E}, r)$ an Alice und Bob, um seine Entscheidung zu stützen.

[0090] Bei den beiden anderen Signcryption-Protokollen sind die Non-Repudiation-Protokolle ähnlich unter der Voraussetzung, dass der Richter volles Vertrauen genießt.

[0091] Schließlich ist ersichtlich, dass das vorliegende Schema, wenn es kombiniert wird mit einem Applikationsprotokoll, für das der geheime Schlüssel des Anwenders in der Berechnung direkt verwendet werden muss, einen implizit zertifizierten ID-basierten öffentlichen Schlüssel des Benutzers bereitstellt. Diese Schemata können auch von einem Schlüsselauthentifizierungszenter (KAC) verwendet werden, um implizit zertifizierte öffentliche Schlüssel an die Nutzer zu verteilen.

[0092] Eine weitere Anwendung implizit zertifizierter öffentlicher Schlüssel besteht darin, dass die Bitstärke der Zertifizierungsstelle genauso groß ist, wie die öffentlichen Schlüssel des Benutzers oder der Entität, welche zertifiziert werden. "Bitstärke" bedeutet die relativen Schlüsselgrößen und die Rechenleistung der betreffenden Entitäten.

[0093] Eine Herangehensweise an dieses Problem besteht darin, die implizit zertifizierten öffentlichen Schlüssel in traditionellere Zertifikatsstrukturen, z.B. wie spezifiziert in X.509-Zertifikaten, einzubetten, bei denen die Signatur auf dem Zertifikat eine größere Bitstärke aufweist als der implizit zertifizierte öffentliche Schlüssel.

[0094] Die CA hat also den öffentlichen Schlüssel des Nutzers auf zwei verschiedenen Sicherheitsniveaus zertifiziert. Jede andere Entität, die einen öffentlichen Schlüssel abruft, kann entscheiden, welches Sicherheitsniveau sie akzeptieren möchte. Bei einigen Anwendungen kann es sein, dass zur Erbringung der erforderlichen Leistung lediglich das durch den impliziten Wert bereitgestellte niedrigere Sicherheitsniveau nötig ist.

[0095] Zwar wurde die Erfindung in Verbindung mit spezifischen Ausführungsformen und spezifischen Anwendungen beschrieben, doch sind für den Fachmann auch verschiedene abgewandelte Ausführungsformen ersichtlich, ohne dass diese vom Erfindungsgedanken, wie er in den beigefügten Ansprüchen angegeben ist, abweichen. Beispielsweise wird in der obigen Beschreibung bevorzugter Ausführungsformen multiplikative Notation angewendet, jedoch kann das Verfahren der vorliegenden Erfindung ebenso gut unter Verwendung der additiven Notation beschrieben werden. Es ist z.B. allgemein bekannt, dass der im ECDSA enthaltene elliptische-Kurve-Algorithmus zu dem DSA äquivalent ist und dass das elliptische-Kurve-Analogon ein Äquivalent eines Diskreter-log-Algorithmus ist, der üblicherweise in einer Gruppe F_p^*, der multiplikativen Gruppe der ganzen Zahlen modulo eine Primzahl, beschrieben wird. Es besteht eine Korrespondenz zwischen den Elementen und Operationen der Guppe F_p^* und der elliptische-Kurve-Gruppe $E(F_q)$. Weiterhin ist diese Signatur-Verfahrensweise ebenso gut auf Funktionen anwendbar, die in einem über F_p und F_{2^n} definierten Feld ausgeführt werden. Es wird außerdem darauf hingewiesen, dass das oben beschriebene DSA-Signaturschema ein spezifisches Beispiel des verallgemeinerten ElGamal-Signaturschemas darstellt, dass den Fachleuten bekannt ist; daher sind die vorliegenden Verfahrensweisen auf dieses anwendbar.

Patentansprüche

1. Verfahren zur Erzeugung eines öffentlichen Schlüssels in einem sicheren digitalen Kommunikationssystem (**10**) mit wenigstens einer, das Vertrauen genießenden Entität CA und mit Teilnehmerentitäten A, wobei das Verfahren die folgenden Schritte umfasst:
A) für jede Entität A wählt die das Vertrauen genießende Entität CA eine eindeutige Identität I_A, durch welche die Entität A gekennzeichnet ist;
B) die das Vertrauen genießende Entität CA erzeugt öffentliche Daten γ_A zur Rekonstruktion des öffentlichen Schlüssels einer Entität A durch mathematische Kombination öffentlicher Werte, die aus jeweiligen geheimen Werten der das Vertrauen genießenden Entität CA und der Entität A erhalten wurden, um ein Paar (I_A, γ_A) zu erhalten, das als das implizite Zertifikat von A dient;
C) Kombinieren der impliziten Zertifikatsinformation (I_A, γ_A) gemäß einer mathematischen Funktion $F(I_A, \gamma_A)$, um eine Entitätsinformation f abzuleiten;

D) Erzeugen eines Wertes k_A durch Verknüpfen der besagten Entitätsinformation f mit geheimen Werten der das Vertrauen genießenden Entität CA,
Übertragen des besagten Wertes k_A an die Entität A, um es A zu ermöglichen, einen geheimen Schlüssel aus besagtem Wert k_A, dem geheimen Wert der Entität A und dem impliziten Zertifikat zu erzeugen, wobei der öffentliche Schlüssel der Entität A aus öffentlicher Information, den besagten öffentlichen Daten γ_A zur Rekonstruktion des öffentlichen Schlüssels und der besagten Identität I_A rekonstruiert werden kann.

2. Verfahren nach Anspruch 1, bei dem die mathematische Funktion F eine sichere Hash-Funktion ist.

3. Verfahren nach Anspruch 1, bei dem der geheime Wert der Entität A bei der Entität A zur Verfügung gestellt wird und der daraus erhaltene, entsprechende öffentliche Wert bei der das Vertrauen genießenden Entität CA zur Verfügung gestellt wird.

4. Verfahren nach Anspruch 1, bei dem die mathematische Kombination in Schritt (b) eine Multiplikation ist.

5. Verfahren nach Anspruch 1, bei dem die geheimen Werte der das Vertrauen genießenden Entität CA einen geheimen Schlüssel und eine ganze Zahl enthalten.

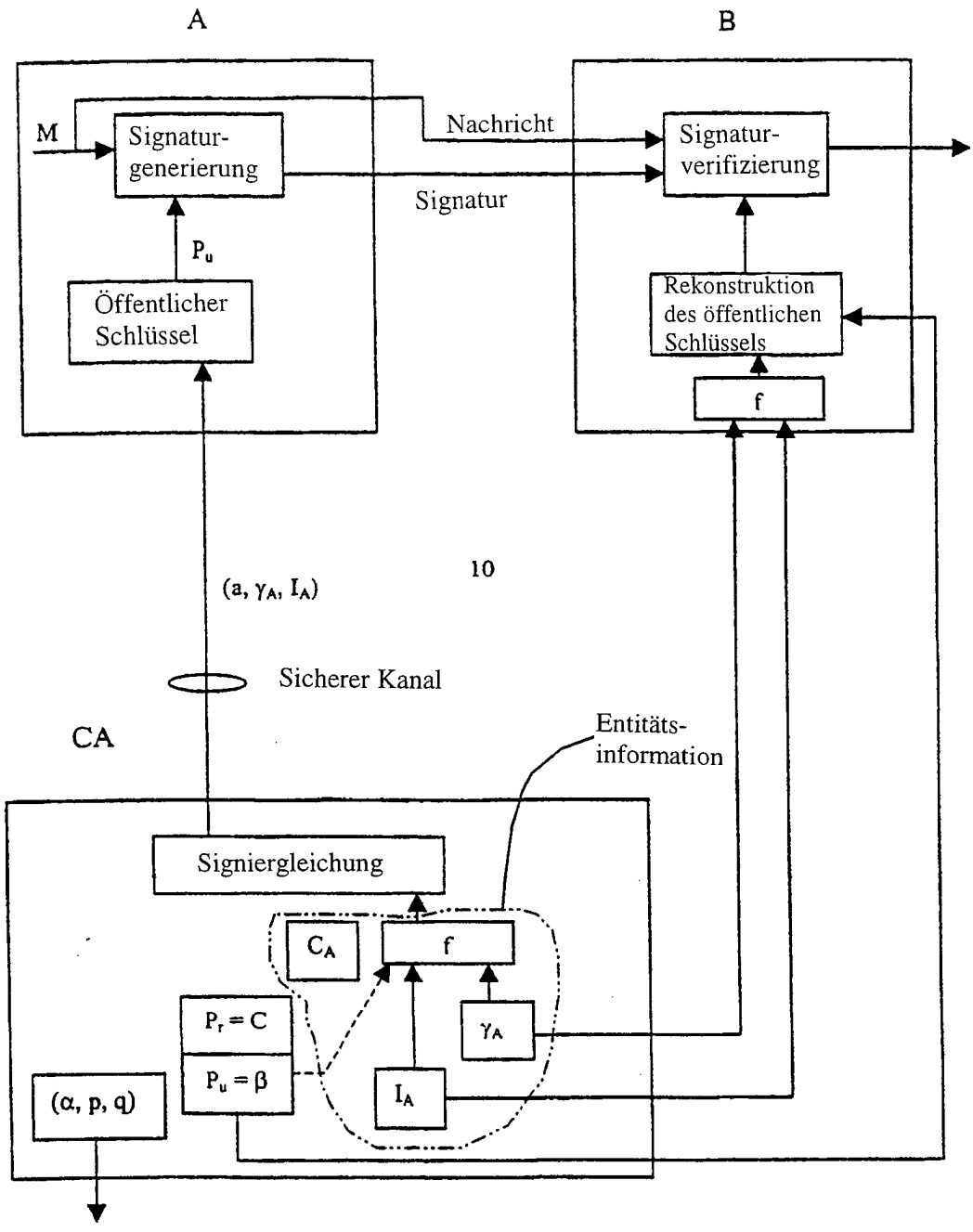
6. Verfahren nach Anspruch 5, bei dem einer der in Schritt (b) verwendeten öffentlichen Werte dem besagten geheimen Schlüssel der das Vertrauen genießenden Entität CA entspricht.

7. Verfahren nach Anspruch 5 oder 6, bei dem der Wert k_A durch Multiplizieren der besagten Entitätsinformation f der Entität A mit der besagten ganzen Zahl und Hinzuaddieren des besagten geheimen Schlüssels der das Vertrauen genießenden Entität CA berechnet wird.

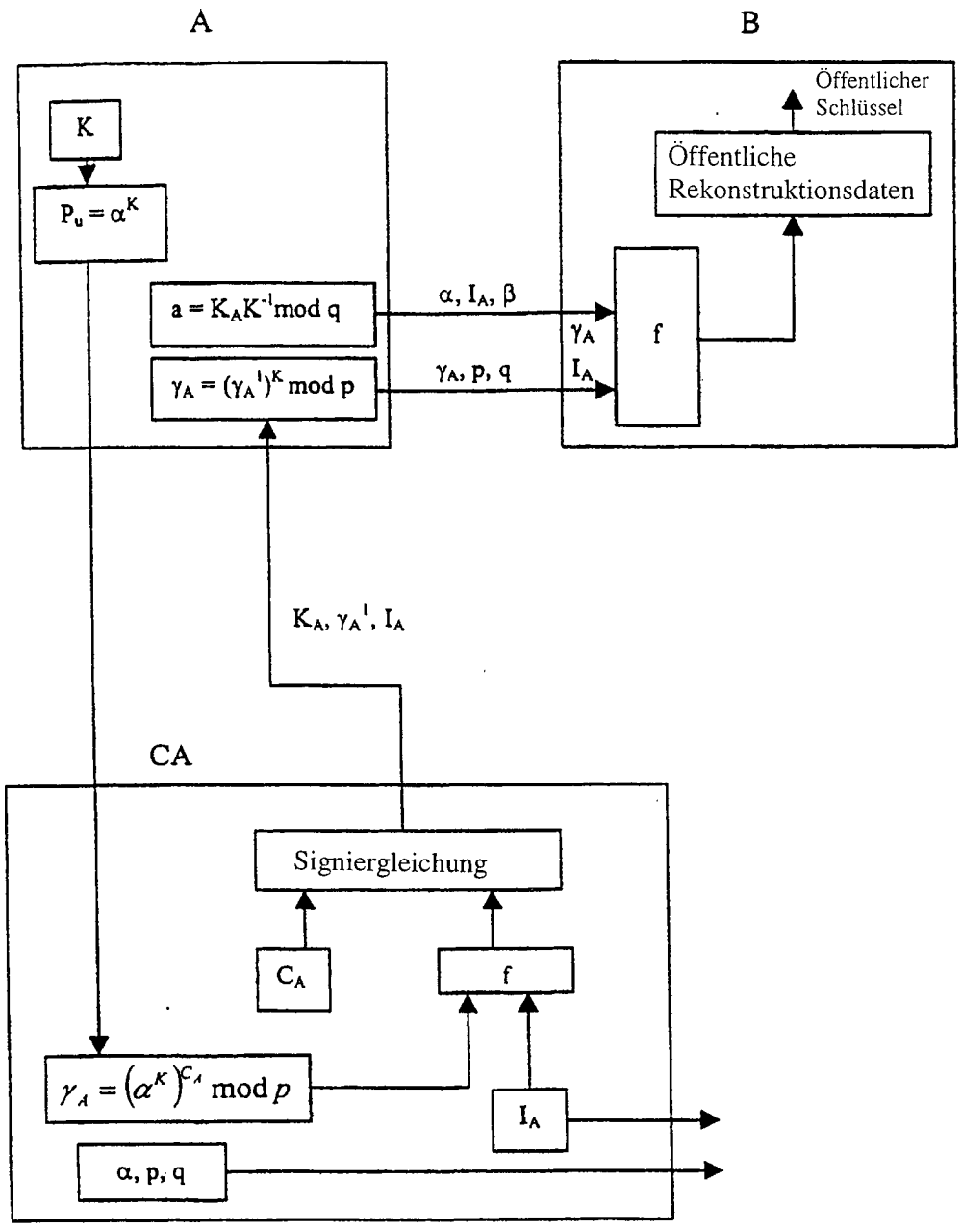
8. Ein von einer das Vertrauen genießenden Entität CA erzeugtes Zertifikat, das zur Erzeugung eines öffentlichen Schlüssels durch eine Teilnehmerentität A in einem sicheren digitalen Kommunikationssystem (**10**) dient, welcher durch Kombinieren der Zertifikatsinformation gemäß einer mathematischen Funktion erzeugt wird, um eine Entitätsinformation f abzuleiten, und Erzeugen eines Wertes K_A durch Verknüpfen dieser Entitätsinformation f mit geheimen Werten der besagten CA, wobei das Zertifikat eine eindeutige Identität I_A umfasst, welche die besagte Entität A kennzeichnet, sowie öffentliche Daten zur Rekonstruktion des öffentlichen Schlüssels der besagten Entität A, welches durch mathematisches Kombinieren öffentlicher Werte erzeugt wird, die aus jeweiligen geheimen Werten der besagten das Vertrauen genießenden Partei CA und der Entität A erhalten werden.

Es folgen 2 Blatt Zeichnungen

Anhängende Zeichnungen



Figur 1


A
B
K
$P_u = \alpha^K$
$a = K_A K^{-1} \bmod q$
$\gamma_A = (\gamma_A{}^1)^K \bmod p$
α, I_A, β
γ_A, p, q
γ_A
I_A
f
Öffentlicher Schlüssel
Öffentliche Rekonstruktionsdaten
K_A, $\gamma_A{}^1$, I_A
CA
Signiergleichung
C_A
f
$\gamma_A = (\alpha^K)^{C_A} \bmod p$
I_A
α, p, q

Figur 2