Technische Universität Chemnitz-Zwickau

Fakultät für Informatik

Theoretische Informatik und Informationssicherheit

Technischer Report

TR-95-07

Selbstzertifizierende öffentliche Schlüssel

Yun Ding · Patrick Horster

Markus Michels · Holger Petersen

April 1995

Straße der Nationen 62	Tel: 0371-531-1479
D-09111 Chemnitz	Fax: 0371-531-1810

Selbstzertifizierende öffentliche Schlüssel

Yun Ding · Patrick Horster · Markus Michels · Holger Petersen

Theoretische Informatik und Informationssicherheit
Technische Universität Chemnitz-Zwickau
Straße der Nationen 62, D-09111 Chemnitz
E-mail: {yun,pho,mmi,hpe}@informatik.tu-chemnitz.de

Zusammenfassung

In diesem Beitrag werden die Eigenschaften selbstzertifizierender Schlüssel im Vergleich zu zertifikatsbasierten Schlüsseln diskutiert. Danach erfolgt eine Übersicht über bekannte Protokolle zur Realisierung selbstzertifizierender öffentlicher Schlüssel. Hierbei werden drei verschiedene Ansätze aufgezeigt. Anschließend werden Erweiterungen des Konzepts der selbstzertifizierenden Schlüssel in Form von hierarchischen Zertifikaten, Mehrfachzertifikaten, benutzergesteuerter Fortschaltung, verdeckten und pseudonymen selbstzertifizierenden Schlüsseln betrachtet. Zuletzt werden wichtige Anwendungen selbstzertifizierender Schlüssel vorgestellt. Hierbei wird auf digitale Signaturen, Nachrichtenverschlüsselung, Authentifikationsverfahren und Protokolle zum authentischen Schlüsselaustausch eingegangen. Bei den Signaturen wird speziell das Problem der Nicht-Zurückweisbarkeit erörtert, das sich bei selbstzertifizierenden Schlüsseln in besonderer Weise darstellt.

1. Einleitung

In einem asymmetrischen Kryptoverfahren [9] benötigt jeder Benutzer geheime und öffentliche Schlüssel. Die öffentlichen Schlüssel können in einem öffentlichen Verzeichnis zur Verfügung gestellt werden, das von einem Trust Center verwaltet wird, so daß jeder Benutzer des Systems die öffentlichen Schlüssel der anderen Benutzer erhalten kann. Eine Gefahr besteht darin, daß ein Angreifer den öffentlichen Schlüssel eines Benutzer durch einen eigenen ersetzt und damit die Sicherheit des Systems nicht mehr gewährleistet ist. Um die Authentizität der öffentlichen Schlüssel zu sichern, müssen diese zertifiziert werden, so daß dieser Angriff verhindert werden kann. Die ausgestellten Zertifikate der öffentlichen Schlüssel lassen sich in drei Klassen unterteilen:

1. *Zertifikatsbasierte öffentliche Schlüssel:* Ein Trust Center unterschreibt den öffentlichen Schlüssel gemeinsam mit der Identität und einem Zeitstempel, der die Gültigkeitsdauer des Zertifikats angibt.

2. *Identitätsbasierte öffentliche Schlüssel:* Hier wird die Identität des Benutzers zur Berechnung des öffentlichen Schlüssels explizit benötigt. Dadurch wird der Zusammenhang zwischen der Identität und dem Schlüssel gewährleistet. Diese Klasse läßt sich weiterhin in drei Unterklassen unterteilen:

   a) *Interaktionsfreie öffentliche Schlüssel:* Die öffentlichen Schlüssel entsprechen den Identitäten der Benutzer. Jeder autorisierte Benutzer im System kennt die öffentlichen Schlüssel direkt, ohne mit den Benutzern zu kommunizieren, woher die Bezeichnung *interaktionsfrei* rührt. Bei Kryptosystemen auf der Basis des diskreten Logarithmusproblems haben diese Schlüssel einen gravierenden Nachteil: Zur Bildung des geheimen Schlüssels x_U muß eine Einwegfunktion mit Trapdoor verwendet werden, da gelten soll: $y_U := ID_U \equiv \alpha^{x_U} \pmod{p}$. Damit ist x_A der diskrete Logarithmus von ID_U zur Basis α. Diese Trapdoor läßt sich in Primmoduln nicht realisieren, existiert aber in zusammengesetzten Moduln, die keine zyklischen Gruppen bilden [25, 28].

   b) *Selbstzertifizierende öffentliche Schlüssel:* Die Identität jedes Benutzers wird vom Trust Center unterschrieben. Einer der Signaturparameter wird als geheimer Schlüssel des Benutzers verwendet. Der zugehörige öffentliche Schlüssel läßt sich aus der Verifikationsgleichung unter Verwendung öffentlich bekannter Parameter wie etwa den gemeinsamen Systemparametern und der Identität des Betreffenden von allen Benutzern berechnen. Die Authentizität des öffentlichen Schüssels wird nicht direkt verifiziert. Der autorisierte Benutzer (der Besitzer) kennt den zugehörigen geheimen Schlüssel und kann daher diesen sinnvoll einsetzen.

   c) *Secret Key Zertifikate:* Der öffentliche Schlüssel des Benutzers wird derart unterschrieben, daß diese Unterschrift von jedem perfekt simuliert werden kann. Die Authentizität des Schlüssels ergibt sich wie bei den selbstzertifizierenden öffentlichen Schlüsseln erst aus der Verwendung des Schlüssels [2]. Im Gegensatz zu den selbstzertifizierenden öffentlichen Schlüsseln wird keiner der Signaturparameter direkt als geheimer Schlüssel eingesetzt.

Durch die Verwendung identitätsbasierter Schlüssel müssen die öffentlichen Paramter nicht mehr zusätzlich authentisch sein, da sich ihre Authentizität aus der Verwendung in den Protokollen ergibt. Daher benötigt man für die Verteilung der öffentlichen Benutzerschlüssel keine authentischen öffentlichen Verzeichnisse mehr, wohl aber für die Verteilung der zugehörigen authentischen öffentlichen Schlüssel der Trust Center. Diese können ebenfalls selbstzertifizierend berechnet werden, sofern eine Hierarchie zwischen den Trust Centern besteht und sich diese in einer Baumstruktur anordnen lassen [4, 17]. Damit benötigt jeder Benutzer im Extremfall lediglich die Kenntnis des authentischen öffentlichen Schlüssels des Trust Centers, das sich an der Spitze des Baumes befindet. Die genaue Funktionsweise wird in Abschnitt 3.1 beschrieben.

Die Sicherheit der meisten Protokolle basiert auf dem diskreten Logarithmusproblem. Dieses besitzt zum einen den Vorteil sich in unterschiedlichen Strukturen effizient realisieren zu lassen, zum anderen benötigt man nur einen systemweiten gemeinsamen Primmodul p und nicht für jeden Benutzer einen eigenen Modul n. Dieses bringt besonders bei der hierarchischen Fortschaltung der Schlüssel gravierende Vorteile hinsichtlich der Effizienz bzw. Sicherheit.

In Kapitel 2 beschreiben wir die Realisierungen selbstzertifizierender öffentlicher Schlüssel, in Kapitel 3 folgen Erweiterungen des Konzepts und schließlich in Kapitel 4 die Anwendungen, bei denen sich selbstzertifizierende Schlüssel gewinnbringend einsetzen lassen. Es bezeichnet $x \in_R$ M die gleichverteilte Wahl einer Pseudozufallszahl x aus einer Menge M.

2. Realisierungen

Durch Anwendung der Ideen aus [1, 13] können für alle Benutzer identitätsbasierte Zertifikate der öffentlichen Schlüssel erzeugt werden. Diese wurden von Girault als selbstzertifizierende Schlüssel bezeichnet [12]. Nachfolgend werden verschiedene Ansätze zu ihrer Realisierung vorgestellt.

Im folgenden bezeichnen p und q große Primzahlen, mit $q|(p-1)$, und α eine Primitivwurzel in $\mathbb{Z}_p$, die von allen Benutzern gemeinsam verwendet werden. Diese Parameter werden bei der Initialisierung des Systems von einem Trust Center gewählt und veröffentlicht.

2.1 Das Basisprotokoll

Die Idee dieses Protokolls wurde 1989 unabhängig von Günther sowie von Bauspieß und Knobloch vorgestellt. Bei diesem Verfahren unterschreibt das Trust Center Z die Identität ID_A der Benutzerin Alice mit einer ElGamal Unterschrift [10]. Hierzu wählt sie eine Zufallszahl $k_A \in_R \mathbb{Z}^*_{p-1}$, berechnet $r_A := \alpha^{k_A} \pmod p$, und löst die Signaturgleichung $ID_A \equiv x_Z \cdot r_A + k_A \cdot x_A \pmod{p-1}$ nach dem Parameter x_A auf:

$$x_A := (ID_A - x_Z \cdot r_A) \cdot k_A^{-1} \pmod{p-1}. \tag{1}$$

Sie übermittelt die Signaturparameter (r_A, x_A) an Alice, die hieraus ihren öffentlichen Schlüssel

$$y_A := r_A^{x_A} \equiv r_A^{(ID_A - x_Z \cdot r_A) \cdot k_A^{-1}} \equiv \alpha^{ID_A - x_Z \cdot r_A} \equiv \alpha^{ID_A} y_Z^{-r_A} \pmod p$$

berechnet. Dieser läßt sich als Funktion der öffentlichen authentischen Parameter α, p, ID_A, y_Z und des Signaturparameters r_A berechnen, der nicht authentisch sein muß.

2.2 Verallgemeinerung mittels des Meta-Signaturverfahren

Der obige Ansatz läßt sich unter Verwendung des Meta-Signaturverfahrens verallgemeinern [18]. Diese Verallgemeinerung wird in der Literatur ausführlich dargestellt [22]. In den nachfolgenden Abschnitten wird bevorzugt eine Variante des Meta-Signaturverfahrens verwendet (Variante EG II.3), die kurz beschrieben werden soll [14].

Die Initialisierung erfolgt wie oben. Das Trust Center Z unterschreibt die Identität der Benutzerin Alice mittels folgender Signaturgleichung:

$$x_A := x_Z \cdot (ID_A + r_A) + k_A \pmod q \tag{2}$$

Alice berechnet ihren selbstzertifizierenden öffentlichen Schlüssel zu

$$y_A := y_Z^{ID_A + r_A} \cdot r_A \pmod p. \tag{3}$$

Sicherheit gegenüber dem Trust Center

Der Nachteil bei der Erzeugung der selbstzertifizierenden öffentlichen Schlüssel liegt darin, daß das Trust Center die geheimen Schlüssel aller Benutzer kennt und daher ihr volles

Vertrauen besitzen muß. Zur Vermeidung dieser Schwäche kann Alice's geheime Schlüssel x_A aus einer verdeckten oder blinden Unterschrift erzeugt werden [16, 20, 24], so daß er anschließend von ihr einmalig modifiziert werden kann, ohne daß das Trust Center seinen neuen Wert erfährt [23]. Ein Protokoll hierfür wird in Abschnitt 3.4 beschrieben.

2.3 Das Protokoll von Girault

Im Gegensatz zum bisher vorgestellten Ansatz basiert die Sicherheit der geheimen Schlüssel in diesem Protokoll [12] ebenfalls auf dem RSA-Problem [29] und nicht alleine auf dem diskreten Logarithmusproblem. Die Beschreibung des Protokolls wurde der in diesem Papier verwendeten Notation angepaßt. Dadurch ist die Bedeutunng der Parameter y_A und s_A im Vergleich zum Original vertauscht.

Initialisierung

Das Trust Center Z wählt einen Modul $n := p \cdot q$ mit zwei großen Primzahlen p, q sowie einen Generator α, der Primitivwurzel in $GF(p)$ und $GF(q)$ ist und damit eine maximale Untergruppe von $\mathbb{Z}_n^*$ erzeugt. Zusätzlich wählt es zwei Zahlen d und e, mit

$$d \cdot e \equiv 1 \pmod{(p-1)(q-1)},$$

wie beim RSA-Verfahren [29]. Die Werte n, α und e werden veröffentlicht.

Die Benutzerin Alice wählt eine Zufallszahl $x_A \in_R \mathbb{Z}_{\varphi(n)}$ und berechnet den Wert $y_A := \alpha^{x_A} \pmod{n}$. Dann beweist sie gegenüber Z mittels eines Zero-Knowledge-Beweises [31], daß sie den diskreten Logarithmus von y_A kennt. Z überprüft Alice's Identität und berechnet zu ID_A die Signatur

$$s_A := (y_A - ID_A)^d \pmod{n} \Rightarrow y_A \equiv s_A^e + ID_A \pmod{n}.$$

Die Signatur s_A teilt sie Alice mit. Damit läßt sich der öffentliche Schlüssel y_A als Funktion der öffentlichen Parameter ID_A, e und des nicht notwendigerweise authentischen Parameters s_A berechnen. Nachdem alle Benutzer registriert worden sind, kann der geheime Parameter d gelöscht werden. Das Problem bei diesem Protokoll ist, daß das Trust Center die geheimen Schlüssel aller Benutzer erfährt.

Verallgemeinerung

Wählt man eine Operation $op \in \{+, -, \cdot, /, \oplus\}$ und definiert die inverse Operation op^{-1} so, daß $((a \text{ op } b) \text{ op}^{-1}\ b) := a =: ((a \text{ op}^{-1}\ b) \text{ op } b)$ gilt, so kann man die Signatur schreiben als

$$s_A := (y_A \text{ op } ID_A)^d \pmod{n} \Rightarrow y_A \equiv s_A^e \text{ op}^{-1}\ ID_A \pmod{n}.$$

Damit ergeben sich vier weitere Protokollvarianten, von denen besonders die zu sich selbst inverse Operation $op = \oplus$ zu beachten ist, für die sich der geringste Aufwand ergibt.

3. Erweiterungen des Konzepts

Die nachfolgenden Erweiterungen können mittels der Protokolle auf der Basis des diskreten Logarithmusproblems, wie sie in den Abschnitten 2.1 und 2.2 vorgestellt wurden realisiert werden. Exemplarisch wird die Realisierung mittels einer Variante des Meta-Signaturverfahrens vorgestellt [18], wie sie in Abschnitt 2.2 beschrieben wurde.

3.1 Hierarchische Schlüssel

Nehmen in einem System viele Benutzer teil, so ist es möglich, daß nicht alle Benutzer ihren *Benutzerausweis*, d.h. die Unterschrift ihrer Identität vom gleichen Trust Center erhalten haben, sondern daß die Trust Center hierarchisch in einem Zertifizierungsbaum angeordnet sind (vgl. [4]). Die Überprüfung der Unterschrift muß in diesem Fall rekursiv über mehrere Trust Center zurückverfolgt werden, bis man ein Trust Center findet, dessen authentischer öffentlicher Schlüssel bereits bekannt ist.

Aus den Daten seines Benutzerausweises kann ein Benutzer eine ElGamal-ähnliche Unterschrift generieren und damit Benutzerausweise für andere Benutzer ausstellen. Diese besitzen die gleiche Struktur wie sein eigener Benutzerausweis. Daher läßt sich das Verfahren rekursiv fortsetzen.

Die Möglichkeit der hierarchischen Authentifizierung bietet sich bei allen ElGamal-ähnlichen Unterschriftsverfahren, d.h. es existiert eine Vielzahl von Varianten [22], die in den Stufen auch wechselweise verwendet werden können, sofern die jeweils verwendete Unterschriftsvariante kenntlich gemacht wird. Das Verfahren wird im folgenden exemplarisch an Hand der Variante EG II.3 vorgeführt.

Hierarchische Ausgabe von Benutzerausweisen

Zur hierarchischen Ausgabe von Benutzerausweisen wird ein übergeordnetes Trust Center Z_0 benötigt, dem alle Benutzer vertrauen. Im folgenden wird beschrieben, wie das Trust Center Z_i einen Benutzerausweis für das Trust Center Z_{i+1} (bzw. einen Benutzer B_{i+1}) erzeugt.

Initialisierung:
Das Trust Center Z_0 wählt Primzahlen p und q, einen Generator α und seinen geheimen Schlüssel x_0. Daraus berechnet es seinen öffentlichen Schlüssel $y_0 := \alpha^{x_0} \pmod p$, den es zusammen mit den Systemparametern p, q und α authentisch veröffentlicht.

Ausstellen der Benutzerausweise:
Das Trust Center Z_i besitzt seinen Benutzerausweis (ID_i, r_i, x_i). Der Wert x_i ist dabei sein geheimer Schlüssel, die Werte r_i und ID_i werden veröffentlicht. Möchte es einen Benutzerausweis für das Trust Center Z_{i+1} mit der Identität ID_{i+1} ausstellen, so wählt es eine Zufallszahl $k_{i+1} \in_R \mathbb{Z}^*_{p-1}$ und berechnet $r_{i+1} := \alpha^{k_{i+1}} \pmod p$. Es löst die Kongruenz

$$x_{i+1} := x_i \cdot (ID_{i+1} + r_{i+1}) + k_{i+1} \pmod q. \tag{4}$$

Das Tripel $(ID_{i+1}, r_{i+1}, x_{i+1})$ stellt den Benutzerausweis des Trust Centers Z_{i+1} dar. Es gilt folgender Zusammenhang für den öffentlichen Schlüssel:

$$y_{i+1} := \alpha^{x_{i+1}} \equiv y_i^{ID_{i+1}+r_{i+1}} \cdot r_{i+1} \pmod p.$$

Rekursive Berechnung des öffentlichen Schlüssels:
Der öffentliche Schlüssel y_i des Trust Centers Z_i läßt sich folgendermaßen rekursiv berechnen:

$$\begin{aligned} y_i &\equiv \alpha^{x_i} \equiv y_{i-1}^{r_i+ID_i} \cdot r_i \equiv (y_{i-2}^{r_{i-1}+ID_{i-1}} \cdot r_{i-1})^{r_i+ID_i} \cdot r_i \\ &\equiv (((\cdots(y_0^{r_1+ID_1} \cdot r_1)^{r_2+ID_2} \cdot r_2)\cdots)^{r_{i-1}+ID_{i-1}} \cdot r_{i-1})^{r_i+ID_i} \cdot r_i \\ &\equiv y_0^{\prod_{j=1}^{i} r_j+ID_j} \cdot \prod_{j=1}^{i} r_j^{\prod_{k=j+1}^{i} r_k+ID_k} \pmod p. \end{aligned}$$

Aus dieser Kongruenz können alle Instanzen (d.h. die Trust Center und alle Benutzer) den öffentlichen Schlüssel y_i authentisch berechnen, sofern sie die ersten $i-1$ öffentlichen

Parameter (ID_j, r_j) und die Parameter α, y_0 und p kennen. Vertrauen müssen sie dabei nur dem Schlüssel y_0 und der Primitivwurzel α entgegenbringen. Kennt eine Instanz bereits irgendeinen authentischen öffentlichen Schlüssel $y_j, j \in [1 : i-1]$, so vereinfacht sich die Berechnung für sie, da sie die Rekursion auf dieser Stufe abbrechen kann.

Sicherheitsanalyse:
Die Identitäten ID_i der Trust Center i (bzw. der Benutzer i) sind systemweit authentisch bekannt (z.B. durch Verwendung des Hashwertes der E-Mail Adresse der Instanz als ihre Identität). Daher kann ein Angreifer nur versuchen, die öffentlichen Parameter r_i zu fälschen. Hierdurch kann zwar der korrekte Schlüssel y_{i+1} nicht mehr berechnet werden, mit dem falschen Schlüssel $\tilde{y}_{i+1}$ kann der Angreifer jedoch keinen konstruktiven Angriff starten, da er den diskreten Logarithmus von $\tilde{y}_{i+1}$ nicht berechnen kann. Um diesen zu erfahren, müßte er den diskreten Logarithmus von y_0 berechnen können.

Sicherheit gegenüber übergeordneten Trust Centern:
Um zu vermeiden, daß das Trust Center Z_i den geheimen Schlüssel x_{i+1} des Trust Centers Z_{i+1} und damit auch die geheimen Parameter aller darunterliegenden Instanzen erfährt, kann z.B. eines der verdeckten ElGamal-Unterschriftsverfahren bei der Ausstellung der Benutzerausweise verwendet werden (vgl. Abschnitt 3.4). Dadurch behält jede Instanz ihren geheimen Schlüssel für sich und es ist kein Angriff von einem übergeordneten Trust Center möglich.

Verwendung RSA-basierter Schlüssel
RSA-basierte öffentliche Schlüssel, wie sie z.B. in Abschnitt 2.3 erwäht wurden, eignen sich zur Realisierung einer Schlüsselhierarchie nicht, da jeder Benutzer einen eigenen Modul n wählen müßte, dessen Faktorisierung nur er kennt. Da jedoch die Unterschrift jeweils vom vorherigen Trust Center geleistet wird, muß diese den geheimen Schlüssel d, die öffentlichen Schlüssel e und n und damit zwangsläufig die Faktorisierung von n kennen [7], womit es die komplette Kommunikation ihrer Nachfolgeinstanz abhören könnte. Verwenden alle Benutzer den gleichen Modul mit unterschiedlichen Schlüsselpaaren (d, e), so ist das Gesamtsystem total unsicher [7, 32].

3.2 Selbstzertifizierende Mehrfachzertifikate

Möchte man in einer Zertifizierungshierarchie neben der linearen Fortschaltung der Schlüssel entlang eines Zertifizierungspfades auch Querverbindungen zur Verkürzung des Pfades zulassen, so ist es erforderlich, daß Zertifikate eines Schlüssels von mehreren Trust Centern gleichzeitig ausgestellt werden können. Dieses ist bei zertifikatsbasierten Systemen einfach zu verwirklichen, da hier lediglich der öffentliche Schlüssel eines Benutzers mehrfach unterschrieben werden muß. Verwendet man das RSA-basierte Protokoll von Girault aus Abschnitt 2.3, so lassen sich Mehrfachzertifikate ebenfalls durch mehrfache Unterschriften realisieren. Diese lassen sich jedoch aus den oben genannten Gründen nicht in eine Hierarchie einbetten.

Bei den Protokollen auf der Basis des diskreten Logarithmusproblems wird der geheime Schlüssel als Signaturparameter erzeugt. Daher besteht eine Möglichkeit zur Realisierung von Mehrfachzertifikaten mit Hilfe digitaler Multisignaturen [21] für zwei Nachrichtenblöcke. Hierbei unterschreiben alle ausgewählten Trust Center einen geheimen Schlüssel gemeinsam. Kennt man mindestens einen öffentlichen Schlüssel dieser Instanzen authentisch, so ist die Authentizität des resultierenden öffentlichen Schlüssels gewährleistet. Auf eine detaillierte Beschreibung sei an dieser Stelle verzichtet.

3.3 Benutzergesteuerte Fortschaltung

Neben der hierarchischen Fortschaltung der Schlüssel zwischen den Benutzern, kann der Benutzer seinen eigenen geheimen Schlüssel ebenfalls benutzergesteuert fortschalten, um zu verschiedenen Zeitpunkten unterschiedliche Schlüssel verwenden zu können. Dieses Prinzip läßt sich für die interaktionsfreie Sitzungsschlüsselgenerierung, wie sie in Abschnitt 4.5 vorgestellt wird, einsetzen. Die nachfolgend vorgestellten Fortschaltungen unterscheiden sich bezüglich ihrer Sicherheit, die sie dem Benutzer im Falle der Offenlegung alter geheimer Schlüssel bieten. Eine Fortschaltung bezeichen wir dabei als *n-sicher*, sofern sie bis zur Offenlegung des $n-$ten geheimen Schlüssels Sicherheit für alle weiteren Schlüssel bietet.

Initialisierung:
Das Trust Center Z wählt Primzahlen p und q, einen Generator α und seinen geheimen Schlüssel x_Z. Daraus berechnet es seinen öffentlichen Schlüssel $y_Z := \alpha^{x_Z} \pmod p$. Mit seinem Schlüsselpaar zertifiziert es das Basisschlüsselpaar $x_{U,0}$, $y_{U,0} := \alpha^{x_{U,0}} \pmod p$ des Benutzers U.

Angenommen, die Benutzerin Alice möchte ihren geheimen Schlüssel nach einer Zeitdauer *period* fortschalten. Dann verwendet sie im Zeitintervall $I_t = [t \cdot period : (t+1) \cdot period]$ ihren geheimen Schlüssel $x_{A,t}$ und den öffentlichen Schlüssel $y_{A,t} := \alpha^{x_{A,t}} \pmod p$. Unterteilt man beispielsweise ein Jahr in Zeitintervalle von jeweils einer Woche, so ergeben sich 52 Intervalle, für die jeweils ein frisches Schlüsselpaar verwendet werden kann.

3.3.1 1-sichere Fortschaltung

Möchte Alice im Zeitintervall I_t kommunizieren, so wählt sie ihren zufälligen geheimen Schlüssel $x_{A,t} \in_R \mathbb{Z}_q$, berechnet $y_{A,t} := \alpha^{x_{A,t}} \pmod p$ und bildet die Unterschrift

$$-s_{A,t} := x_{A,t-1} \cdot (y_{A,t} + ID_A) - x_{A,t} \pmod q \tag{5}$$

unter Verwendung des alten geheimen Schlüssels $x_{A,t-1}$. Der Parameter $s_{A,t}$ und der neue öffentliche Schlüssel werden veröffentlicht. Die Authentizität des neuen öffentlichen Schlüssels ergibt sich durch Überprüfung der Gleichung

$$y_{A,t} \equiv y_{A,t-1}^{y_{A,t}+ID_A} \cdot \alpha^{s_{A,t}} \pmod p.$$

Die Fortschaltung ist lediglich 1-sicher, da aus der Kenntnis der Werte $s_{A,t}, y_{A,t}, ID_A$ und $x_{A,t}$ bzw. $x_{A,t-1}$ in Gleichung (5) $x_{A,t-1}$ bzw. $x_{A,t}$ berechnet werden kann. Diese Sicherheit läßt sich in Richtung von $x_{A,t}$ zu $x_{A,t-1}$ geringfügig erhöhen, indem man $x_{A,t}$ solange neu wählt, bis $\mathrm{ggT}(y_{A,t} + ID_A, q) \neq 1$ ist, was für $q = p - 1 = 2p_1, p_1 \in \mathbb{P}$, mit einer Wahrscheinlichkeit von ca. 0.5 der Fall ist. Damit kann $x_{A,t-1}$ nicht mehr eindeutig berechnet werden.

3.3.2 2-sichere Fortschaltung

Alice wählt eine Zufallszahl $k_A \in_R \mathbb{Z}_q^*$ und berechnet den Parameter $r_A := \alpha^{k_A} \pmod p$, den sie veröffentlicht. Möchte sie im Zeitintervall I_t kommunizieren, so berechnet sie ihren geheimen Schlüssel folgendermaßen:

$$\begin{aligned}
x_{A,t} &:= x_{A,t-1} \cdot (r_A + ID_A) + k_A \pmod q \\
&\equiv ((\cdots(x_{A,0} \cdot (r_A + ID_A) + k_A)\cdots) \cdot (r_A + ID_A) + k_A) \cdot (r_A + ID_A) + k_A \pmod q \\
&\equiv x_{A,0} \cdot (r_A + ID_A)^t + k_A \cdot \sum_{i=0}^{t-1} (r_A + ID_A)^i \pmod q \\
&\equiv x_{A,0} \cdot (r_A + ID_A)^t + k_A \cdot (1 - (r_A + ID_A)^t) \cdot (1 - (r_A + ID_A))^{-1} \pmod q
\end{aligned}$$

Aus dieser Kongruenz ist ersichtlich, daß alle geheimen Schlüssel von dem eingangs gewählten geheimen Schlüssel $x_{A,0}$ abhängen.

Berechnung von Alice's öffentlichem Schlüssel

Jeder Benutzer erhält den Parameter r_A, Alice's Identität ID_A und ihren öffentlichen Schlüssel $y_{A,0}$. Er kann den hierarchischen selbstzertifizierenden öffentlichen Schlüssel $y_{A,t}$ für das Zeitintervall I_t rekursiv berechnen zu

$$\begin{aligned}
y_{A,t} &:= y_{A,t-1}^{r_A+ID_A} \cdot r_A \pmod p \\
&\equiv (((\cdots(y_{A,0}^{r_A+ID_A} \cdot r_A)^{r_A+ID_A} \cdot r_A)\cdots)^{r_A+ID_A} \cdot r_A)^{r_A+ID_A} \cdot r_A \pmod p \\
&\equiv y_{A,0}^{(r_A+ID_A)^t} \cdot r_A^{\sum_{i=0}^{t-1}(r_A+ID_A)^i} \pmod p \\
&\equiv y_{A,0}^{(r_A+ID_A)^t} \cdot r_A^{(1-(r_A+ID_A)^t)\cdot(1-(r_A+ID_A))^{-1}} \pmod p.
\end{aligned}$$

Berechnet ein Benutzer den öffentlichen Schlüssel $y_{A,t}$ unter Verwendung des öffentlichen Schlüssels $y_{A,t-1}$ aus dem vorherigen Zeitintervall, so benötigt er hierfür nur eine Exponentiation. Die Fortschaltung ist lediglich 2-sicher, da aus der Kenntnis zweier aufeinanderfolgender geheimer Schlüssel bereits der geheime Fortschaltungsparameter k_A berechnet werden kann.

Breitenfortschaltung

Anstelle der rekursiven Fortschaltung der Schlüsselpaare, d.h. der Berechnung von $x_{A,t}$ aus $x_{A,t-1}$ und $y_{A,t}$ aus $y_{A,t-1}$, kann man auch alle Schlüssel aus dem Basisschlüsselpaar $(x_{A,0}, y_{A,0})$ fortschalten. Dadurch verringert sich der Berechnungsaufwand, sofern man einen Schlüssel explizit aus dem Basisschlüssel berechnen möchte. Die Fortschaltungsfunktion lautet folgendermaßen:

$$x_{A,t} := x_{A,0} \cdot (ID_A + r_A^t) + k_A \cdot t \pmod q. \tag{6}$$

Der zugehörige öffentliche Schlüssel kann von allen Benutzer mittels Gleichung (7) berechnet werden.

$$y_{A,t} := y_{A,0}^{ID_A+r_A^t} \cdot r_A^t \pmod p. \tag{7}$$

Aus der Kenntnis zweier geheimer Schlüssel kann ein Angreifer den Basisschlüssel $x_{A,0}$ sowie die geheime zufällige Zahl k_A berechnen, da die beiden zugehörigen Gleichungen vom Typ (6) nur zwei Unbekannte enthalten.

3.3.3 n-sichere Fortschaltung

Für eine n-sichere Fortschaltung muß Alice für jedes Zeitintervall ein neues Paar $(k_{A,t}, r_{A,t})$ berechnen und alle Werte $r_{A,t}$ zusammen mit dem öffentlichen Basisschlüssel bekanntgeben. Dadurch kann aus zwei aufeinanderfolgenden offengelegten geheimen Schlüsseln lediglich einer der Parameter k berechnet werden, der zufällig gewählt wurde und daher keine Rückschlüsse auf die anderen Parameter k zuläßt. Die Fortschaltungsfunktion ist identisch zur Fortschaltung bei den hierarchischen Schlüsseln (vgl. Abschnitt 3.1), bis auf die Tatsache, daß ID_j jedesmal gleich ID_A gewählt wird.

Breitenfortschaltung

Wie bereits bei der 2-sicheren Fortschaltung gesehen, können auch hier die Schlüsselpaare ausgehend von einem Basisschlüsselpaar fortgeschaltet werden. Hierdurch verringert sich der Aufwand bei einer expliziten Berechnung des Schlüsselpaares für jedes Zeitintervall. Die Berechnung des geheimen Schlüssels erfolgt mittels Gleichung (8), die Berechnung des öffentlichen Schlüssels mittels der Gleichung (9).

$$x_{A,t} := x_{A,0} \cdot (ID_A + r_{A,t}) + k_{A,t} \pmod q. \tag{8}$$

$$y_{A,t} := y_{A,0}^{ID_A+r_{A,t}} \cdot r_{A,t} \pmod p. \tag{9}$$

3.4 Verdeckte, selbstzertifizierende Schlüssel

Bei den selbstzertifizierenden Schlüsseln berechnet das Trust Center den geheimen Schlüssel der Benutzer. Dadurch sind diese nicht vor Angriffen des Trust Centers geschützt. Daher wurde das Konzept der Beglaubigungen (engl. testimonials) eingeführt, die dadurch charakterisiert sind, daß das Trust Center den zu beglaubigenden geheimen Schlüssel nicht erfährt [16]. Im folgenden wird das Protokoll für das ElGamal-Signaturverfahren vorgestellt [10]. Es läßt sich auf eine Reihe von Varianten des Meta-Signaturverfahrens übertragen [20].

Initialisierung:
Das Trust Center Z wählt Primzahlen p und q, einen Generator α und seinen geheimen Schlüssel x_Z. Daraus berechnet es seinen öffentlichen Schlüssel $y_Z := \alpha^{x_Z} \pmod p$, den es zusammen mit den Systemparametern p, q und α veröffentlicht.

Erzeugung des selbstzertifizierenden Schlüssels
Die Benutzerin Alice wählt einen Wert $h \in_R \mathbb{Z}^*_{p-1}$ und berechnet $\beta = \alpha^h \pmod p$. Diesen Wert sendet sie dem Trust Center, das ein $k \in_R \mathbb{Z}^*_q$ wählt und daraus $r := \beta^k \pmod p$ berechnet. Es löst die Kongruenz

$$x_Z \cdot r + \tilde{x}_A \cdot k \equiv ID_A \pmod q \Rightarrow \tilde{x}_A := k^{-1} \cdot (ID_A - x_Z \cdot r) \pmod q$$

für den Wert $\tilde{x}_A$ und übermittelt Alice das Tupel $(r, \tilde{x}_A)$. Alice verifiziert die Signatur und berechnet ihren (aufgedeckten) geheimen Schlüssel $x_A := \tilde{x}_A \cdot h^{-1} \pmod q$. Es ergibt sich das Protokoll 1.

Benutzerin Alice	Kanal	Trust Center
$h \in_R \mathbb{Z}^*_{p-1}$		
$\beta := \alpha^h \pmod p$	$\longrightarrow$	(ID_A, β)
		$k \in_R \mathbb{Z}^*_{p-1}$
		$r := \beta^k \pmod p$
$(r, \tilde{x}_A)$	$\longleftarrow$	$\tilde{x}_A := (ID_A - x_Z \cdot r) \cdot k^{-1} \pmod q$
$x_A := \tilde{x}_A \cdot h^{-1} \pmod q$		
Verifikation: $\alpha^{ID_A} \equiv y_Z^r \cdot r^{x_A} \pmod p$.		

Protokoll 1: Das Beglaubigungsschema

Hieraus kann jeder autorisierte Benutzer den selbstzertifizierenden öffentlichen Schlüssel y_A berechnen, wobei er als Primitivwurzel den Parameter r verwenden muß:

$$y_A := r^{x_A} \equiv \alpha^{ID_A} \cdot y_Z^{-r} \pmod p.$$

3.5 Schwach blinde, selbstzertifizierende Schlüssel

Schwach blinde, selbstzertifizierende öffentliche Schlüssel besitzen die Eigenschaft, zunächst gegenüber dem Trust Center sowohl den geheimen Schlüssel wie auch seine Identität und den Signaturparameter r zu verbergen. Diese Eigenschaft gilt allerdings nur solange, wie die Schlüssel nicht gegenüber dem Trust Center eingesetzt werden. Das Trust Center kann aufgrund der öffentlichen Parameter eine Relation zwischen den schwach blinden und ungeblindeten Parametern feststellen und damit die Blindung des geheimen Schlüssels aufdecken. Realisierungen schwach blinder, selbstzertifizierender Schlüssel ergeben sich durch Anwendung der schwach blinden Signaturverfahren in [19]. Die Benutzerin

Alice erscheint beim Trust Center, weist ihre Identität ID_A, z.B. durch ihren Ausweis aus, und erhält einen schwach blinden, öffentlichen Schlüssel zu ihrem Pseudonym PS_A. Für die von uns in Abschnitt 2.2 gewählte Variante ergibt sich das Protokoll 2.

Benutzerin Alice	Kanal	Trust Center
$a \in_R \mathbb{Z}_q^*$		$\tilde{k} \in_R \mathbb{Z}_q^*$
$\tilde{r}$	$\longleftarrow$	$\tilde{r} := \alpha^{\tilde{k}} \pmod p$
$r := \tilde{r}^a \pmod p$		
$\tilde{PS}_A := (PS_A + r) \cdot a^{-1} - \tilde{r} \pmod q$	$\longrightarrow$	$\tilde{PS}_A$
$\tilde{x}_A$	$\longleftarrow$	$\tilde{x}_A := x_Z \cdot (\tilde{PS}_A + \tilde{r}) + \tilde{k} \pmod q$
$x_A := \tilde{x}_A \cdot a \pmod q$		
Verifikation: $\alpha^{x_A} \equiv y_Z^{(PS_A+r)} \cdot r \pmod p$		

Protokoll 2: Schwach blinde, selbstzertifizierende Schlüssel

Die Signatur ist lediglich schwach blind, da das Trust Center bei Kenntnis von PS_A und r folgende Kongruenz überprüfen kann:

$$\alpha^{\tilde{x}_A} \equiv y_Z^{\tilde{PS}_A+\tilde{r}} \cdot r^{(\tilde{PS}_A+\tilde{r})\cdot(PS_A+r)^{-1}} \pmod p$$

3.6 Pseudonyme, selbstzertifizierende Schlüssel

Pseudonyme, selbstzertifizierende Schlüssel dienen zur anonymen Registrierung eines Benutzers in einem System, bei der neben der Identität auch der geheime Schlüssel gegenüber dem Trust Center verborgen bleibt. Dazu erscheint die Benutzerin Alice beim Trust Center, weist ihre Identität ID_A, z.B. durch ihren Ausweis aus, und erhält einen pseudonymen, öffentlichen Schlüssel. Das Trust Center unterschreibt ihr Pseudonym PS_A mittels einer Variante des Meta-blinden Signaturverfahrens [19]. Dabei besteht keine Relation zwischen den aufgedeckten und geblindeten Signaturparametern, woduch die dauerhafte Anonymität des öffentlichen Schlüssels gewährleistet ist.

Benutzerin Alice	Kanal	Trust Center
$a, b \in_R \mathbb{Z}_q^*$		$\tilde{k} \in_R \mathbb{Z}_q^*$
$\tilde{r}$	$\longleftarrow$	$\tilde{r} := \alpha^{\tilde{k}} \pmod p$
$r := \tilde{r} \cdot y_Z^{-a} \cdot \alpha^b \pmod p$		
$\tilde{PS}_A := a - PS_A - r - \tilde{r} \pmod q$	$\longrightarrow$	$\tilde{PS}_A$
$\tilde{x}_A$	$\longleftarrow$	$\tilde{x}_A := x_Z \cdot (\tilde{PS}_A + \tilde{r}) + \tilde{k} \pmod q$
$x_A := \tilde{x}_A + b \pmod q$		
Verifikation: $\alpha^{x_A} \equiv y_Z^{(PS_A+r)} \cdot r \pmod p$		

Protokoll 3: Pseudomyme, selbstzertifizierende Schlüssel

Bei den selbstzertifizierenden öffentlichen Schlüsseln auf der Basis des RSA-Verfahrens, wie sie in Abschnitt 2.3 vorgestellt wurden, kann eine Blindung unter Verwendung des blinden Signaturverfahrens von Chaum [5] erfolgen.

4. Anwendungen

Die folgenden Anwendungen können mittels der Protokolle auf der Basis des diskreten Logarithmusproblems, wie sie in den Abschnitten 2.1 und 2.2 vorgestellt wurden realisiert werden. Exemplarisch wird die Realisierung mittels einer Variante des Meta-Signaturverfahrens vorgestellt [18], die in Abschnitt 2.2 beschrieben wurde.

4.1 Digitale Unterschrift

Die Benutzerin Alice kann mittels ihres vom Trust Center erhaltenen Schlüsselpaares Nachrichten unterzeichnen. Dieses funktioniert im Prinzip wie bei den hierarchischen öffentlichen Schlüsseln, lediglich wird nicht die Identität des nachfolgenden Benutzers unterschrieben, sondern eine frei gewählte Nachricht m [13]. Es ergibt sich Protokoll 4.

Unterzeichner Alice	Kanal	Verifizierer Bob
$k \in_R \mathbb{Z}_{p-1}^*$		$y_A := y_Z^{ID_A + r_A} \cdot r_A \pmod p$
$r := \alpha^k \pmod p$		
$s := x_A \cdot (m + r) + k \pmod q$		
(m, r, s)	$\longrightarrow$	(m, r, s)
		Unterschrift akzeptiert, falls
		$\alpha^s \equiv y_A^{m+r} \cdot r \pmod p$

Protokoll 4: Digitale Unterschrift

Non-Repudiation der Unterschrift

Verwendet man zertifikatsbasierte öffentliche Schlüssel, so läßt sich für jeden Benutzer eindeutig feststellen, ob z.B. bei der Verifikation einer Unterschrift diese von der vorgegebenen Person geleistet wurde oder nicht. Dazu wird zunächst die Authentizität des öffentlichen Schlüssels dieser Person anhand des Zertifikats geprüft und anschließend festgestellt, ob die hiermit geleistete Unterschrift authentisch ist.

Bei selbstzertifizierenden Schlüsseln lassen sich diese beiden Schritte zunächst nicht voneinander trennen. Ist genau eine Unterschrift überprüfen zu lassen, deren Verifikation inkorrekt ist, so kann man nicht unterscheiden, ob dieses am selbstzertifizierenden öffentlichen Schlüssel liegt oder an der vorgelegten Unterschrift. Damit kann eine korrekte Unterschrift durch Verwendung eines falschen öffentlichen Schlüssels (fälschlicherweise) zurückgewiesen werden.

Um die Echtheit der Unterschrift durch eine dritte Partei überprüfen zu können, muß sich der Benutzer mittels des fraglichen (evtl. nicht authentischen) öffentlichen Schlüssels gegenüber dieser Partei authentifizieren. Ist ihm dieses möglich, so kann davon ausgegangen werden, daß in diesem Fall die fragliche Unterschrift falsch war.

Stimmt die Unterschrift hingegen, so ist die Authentizität beider Komponenten, d.h. des öffentlichen Schlüssels und der Unterschrift, gewährleistet. In diesem Fall kann der Unterzeichner die Unterschrift nicht ableugnen. Ebenso kann auf die Authentizität des öffentlichen Schlüssels geschlossen werden, falls mit ihm eine erfolgreiche Authentifikation durchgeführt wurde.

4.2 Verschlüsselung

Verschlüsselt die Benutzerin Alice eine Nachricht unter Verwendung des selbstzertifizierenden öffentlichen Schlüssels y_B des Empfängers Bob, so kann nur dieser die Nachricht entschlüsseln, da er hierzu den zugehörigen geheimen Schlüssel x_B kennt. Ein Angreifer, der den Parameter r_B des Empfängers Bob modifiziert, kann hieraus keinen Nutzen ziehen, da er den geheimen Schlüssel x_B nicht ermitteln kann. Exemplarisch soll die Nachrichtenverschlüsselung anhand des ElGamal Verschlüsselungsverfahrens [10] in Protokoll 5 demonstriert werden.

Benutzerin Alice	Kanal	Empfänger Bob
$y_B := y_Z^{ID_B + r_B} \cdot r_B \pmod p$		
$k_A \in_R \mathbb{Z}_{p-1}$		
$r_A := \alpha^{k_A} \pmod p$		
$K_{AB} := y_B^{k_A} \pmod p$		
$c_m := K_{AB} \cdot m \pmod p$	$\longrightarrow$	(r_A, c_m)
		$K_{AB} := r_A^{x_B} \pmod p$
		$m := K_{AB}^{-1} \cdot c_m \pmod p$

Protokoll 5: ElGamal Verschlüsselung

4.3 Authentifikation

In einem Authentifikationsverfahren beweist die Benutzerin Alice, die sich gegenüber dem Verifizierer Bob authentifiziert, daß sie den diskreten Logarithmus x_A ihres selbstzertifizierenden öffentlichen Schlüssels $y_A := \alpha^{x_A} \pmod p$ zur Basis α kennt. Dieser Beweis kann mittels eines „Challenge und Response“- Protokolls oder eines Zero-Knowledge-Protokolls geführt werden.

4.3.1 Challenge und Response Authentifikation

Die Authentifizierin Alice unterzeichnet eine vom Verifizierer Bob gewählte zufällige Nachricht m unter Verwendung ihres geheimen Schlüssels x_A. Hierzu kann das Unterschriftsverfahren aus Abschnitt 4.1 eingesetzt werden. Bob kann die Gültigkeit der Unterschrift mittels der Verifikationsgleichung überprüfen. Stimmt die Unterschrift, so konnte Alice sich erfolgreich authentifizieren.

Das Problem bei dieser Art der Authentifikation besteht in der Möglichkeit einer adaptiven Attacke, bei der Bob durch die geschickte Auswahl der zu signierenden Nachrichten eine Attacke auf Alice's geheimen Schlüssel versucht. Diese kann z.B. durch Verwendung einer One-Way Hashfunktion vermieden werden, indem die Challenge vor dem Unterzeichnen zusätzlich mit einer zufälligen Zahl des Beweisers multipliziert und gehasht wird. Es ergibt sich das Protokoll 6.

Beweiserin Alice	Kanal	Verifizierer Bob
$d, k \in_R \mathbb{Z}_{p-1}^*$		$y_A := y_Z^{ID_A + r_A} \cdot r_A \pmod p$
$r := \alpha^k \pmod p$		
c	$\longleftarrow$	$c \in_R \mathbb{Z}_p^*$
$m := h(c \cdot d)$		
$s := x_A \cdot (m + r) + k \pmod q$		
(d, r, s)	$\longrightarrow$	(d, r, s)
		Beweis akzeptiert, falls
		$\alpha^s \equiv y_A^{h(c \cdot d) + r} \cdot r \pmod p$

Protokoll 6: Challenge und Response Authentifikation

4.3.2 Zero-Knowledge Authentifikation

Es kann eine Variante des Zero-Knowledge-Proofs von Chaum, Evertse und van de Graaf verwendet werden [6, 31]. Nach erfolgreichem Beenden des Protokolls ist Bob von Alice's Authentizität überzeugt, ohne nützliche Seiteninformationen, wie z.B. eine Unterschrift, von ihr erhalten zu haben.

Es besteht das Problem der „Middleperson-Attacke“ [8], bei der sich der Verifizierer Bob parallel zum Beweis von Alice gegenüber Clara als Alice ausgibt, indem er ihre Protokolldaten an Clara weiterleitet. Diese Attacke kann durch eine der Gegenmaßnahmen in [26] vermieden werden. Hier verwenden wir eine Modifikation der ersten in [26] beschriebenen Gegenmaßnahme, in der wir die Identität des Empfängers anfügen.

Beweiserin Alice		Verifizierer Bob
$u_A \in_R \mathbb{Z}_q$		
$v_A := \alpha^{u_A} \pmod{p}$		
$w_A := h(v_A, ID_B)$	$\longrightarrow$	(ID_B, r_A, w_A)
		$y_A := y_Z^{ID_A + r_A} \cdot r_A \pmod{p}$
c_A	$\longleftarrow$	$c_A \in_R \mathbb{Z}_q$
$s_A := x_A c_A + u_A \pmod{q}$	$\longrightarrow$	s_A
		Akzeptiert, falls
		$w_A \equiv h(\alpha^{s_A} y_A^{-c_A} \pmod{p}, ID_B)$

Protokoll 7: Zero-Knowledge Authentifikationsverfahren

4.4 Authentischer Schlüsselaustausch

Um ein authentisches Schlüsselaustauschverfahren zu erhalten, können wir die Ansätze aus [1, 13] oder den SELANE-Protokollen [16] verwenden, die auf den selbstzertifizierenden öffentlichen Schlüsseln aufsetzen. Dabei ist es wichtig, daß die beiden erzeugten Teilschlüssel zusammenmultipliziert werden, da ihre Authentizität aufgrund einer möglichen „Middleperson-Attacke“ nur einseitig garantiert ist [26, 27].

1.Ansatz

Der erste Ansatz verbindet beidseitige Authentifikation mit dem Diffie-Hellman-Schlüsselaustausch [9], indem die authentischen Daten, die während der erfolgreichen Authentifikation ausgetauscht wurden, zur Berechnung des Sitzungsschlüssels benutzt werden. Dieser kann aus den öffentlichen Parametern w_A, w_B aus dem obigen Zero-Knowledge Authentifikationsverfahren und zwei zufällig gewählten geheimen Parametern $d_A, d_B \in_R \mathbb{Z}_q$ sowie den daraus berechneten öffentlichen Werten $e_A := \alpha^{d_A}$ und $e_B := \alpha^{d_B}$ folgendermaßen erzeugt werden:

$$K := e_B^{v_A} w_B^{d_A} \equiv \alpha^{v_A d_B + v_B d_A} \equiv \alpha^{v_A d_B + v_B d_A} \equiv w_A^{d_B} e_A^{v_B} \pmod{p}.$$

Der Schlüssel K hängt nicht direkt von den öffentlichen Schlüsseln der beiden Parteien ab. Daher ist die Sicherheit der anderen Schlüssel durch die Kompromittierung eines bereits verwendeten Sitzungsschlüssels nicht gefährdet.

2.Ansatz

Im zweiten Ansatz wird ein Diffie-Hellman Schlüssel berechnet, den nur die beiden Parteien kennen können. Dieser muß nach der Berechnung von beiden Seiten überprüft werden, damit beide Parteien überzeugt sind, daß sie den gleichen Schlüssel besitzen. Dieses kann beispielsweise durch Übermittlung einer ersten verschlüsselten Nachricht mit diesem Schlüssel und einer entsprechenden Antwort geschehen [17]. Protokoll 8 verdeutlicht den Ablauf.

Wurde die Übermittlung der Werte w_A und w_B nicht von einem Angreifer gestört, so sind die beiden Sitzungsschlüssel identisch, wie die folgende Kongruenz zeigt:

$$K_A := h(w_B^{x_A} y_B^{v_A}) \equiv h((\alpha^{v_B})^{x_A} (\alpha^{x_B})^{v_A}) \equiv h((\alpha^{v_A})^{x_B} (\alpha^{x_A})^{v_B}) \equiv h(w_A^{x_B} y_A^{v_B}) = K_B \pmod{p}.$$

Benutzerin Alice	Kanal	Benutzer Bob
$y_B := y_Z^{ID_B + r_B} \cdot r_B \pmod{p}$		$y_A := y_Z^{ID_A + r_A} \cdot r_A \pmod{p}$
$v_A \in_R \mathbb{Z}_q$		$v_B \in_R \mathbb{Z}_q$
$w_A := \alpha^{v_A} \pmod{p}$		$w_B := \alpha^{v_B} \pmod{p}$
w_B	$\longleftrightarrow$	w_A
$K_A := h(w_B^{x_A} y_B^{v_A} \pmod{p})$		$K_B := h(w_A^{x_B} y_A^{v_B} \pmod{p})$

Protokoll 8: Authentischer Schlüsselaustausch

Man beachte, daß beide Seiten unterschiedliche Varianten des Meta-Signaturverfahrens zur Berechnung ihrer selbstzertifizierenden öffentlichen Schlüssel verwenden können.

Verwendet man im letzten Protokollschritt keine Hashfunktion h, so läßt sich auf das Verfahren eine Dreiecksattacke anwenden [3], bei der unter bestimmten Voraussetzungen der Sitzungsschlüssel zweier Benutzer unter Verwendung abgehörter Information berechnet werden kann. Diese Attacke läßt sich durch den Einsatz der Hashfunktion h verhindern.

Weitere Ansätze

Aus den Authentifikationsprotokollen von Lim und Lee [27] lassen sich drei weitere Protokolle zum authentischen Schlüsselaustausch herleiten. Diese können als 3-Wege Protokolle zur beidseitigen Authentifikation mit integriertem Schlüsselaustausch realisiert werden.

4.5 Interaktionsfreier Sitzungsschlüssel

Bei der interaktionsfreien Berechnung eines Sitzungsschlüssels wird ein Diffie-Hellman Schlüsselaustausch auf der Grundlage der fortgeschalteten selbstzertifizierenden Schlüssel durchgeführt, wie sie in Abschnitt 3.3 beschrieben wurden. Da keine Interaktion erfolgt, eignet sich dieses Protokoll für asynchrone verschlüsselte Nachrichtenübermittlung, die z.B. bei elektronischer Mail (E-Mail) vorkommt. Anstelle der Fortschaltung nach einer bestimmten Zeitdauer *period* kann sie auch nach jeder Kommunikation erfolgen.

Initialisierung

Das Trust Center Z wählt Primzahlen p und q, einen Generator α und seinen geheimen Schlüssel x_Z. Daraus berechnet es seinen öffentlichen Schlüssel $y_Z := \alpha^{x_Z} \pmod{p}$, den es zusammen mit den Systemparametern p, q und α veröffentlicht.

Registrierung der Benutzer

Die Benutzerin Alice erhält vom Trust Center Z einen geheimen Basisschlüssel x_A, und den zugehörigen selbstzertifizierenden öffentlichen Schlüssel y_A zusammen mit dem Parameter R_A. Sie wählt einen geheimen Parameter k_A und berechnet daraus den öffentlichen Parameter $r_A := \alpha^{k_A} \pmod{p}$. Diesen veröffentlicht sie zusammen mit ihrem Parameter R_A.

Berechnung der öffentlichen Schlüssel durch die anderen Benutzer

Die Berechnung erfolgt mit Hilfe des Parameters r_A gemäß der Beschreibung der 2-sicheren Fortschaltung in Abschnitt 3.3. Sie kann entweder rekursiv erfolgen, sofern bereits öffentliche Schlüssel aus früheren Zeitintervallen bekannt sind, oder durch Verwendung der expliziten Formel in einem Schritt. Protokoll 9 verdeutlicht den Ablauf.

Als Sitzungsschlüssel ergibt sich $K_t := h(y_{B,t}^{x_{A,t}}) \equiv h(\alpha^{x_{A,t} \cdot x_{B,t}}) \equiv h(y_{A,t}^{x_{B,t}}) \equiv h(K_{B,t})$. Dieser besitzt amortisierende Sicherheit pro Zeitintervall, da man aus der Kenntnis eines

Benutzerin Alice		Benutzer Bob
$x_{A,t} := x_{A,t-1} \cdot (r_A + ID_A) + k_A$ $y_{B,t} := y_{B,t-1}^{r_B+ID_B} \cdot r_B \pmod p$ $K_{A,t} := y_{B,t}^{x_{A,t}} \pmod p$ $K_t := h(K_{A,t})$		$x_{B,t} := x_{B,t-1} \cdot (r_B + ID_B) + k_B$ $y_{A,t} := y_{A,t-1}^{r_A+ID_A} \cdot r_A \pmod p$ $K_{B,t} := y_{A,t}^{x_{B,t}} \pmod p$ $K_t := h(K_{B,t})$

Protokoll 9: Interaktionsfreie Berechnung des Sitzungsschlüssels

alten Sitzungsschlüssels $K_i, i \le t$, den aktuellen Sitzungsschlüssel nicht effizient berechnen kann. Hierzu wäre zuerst die Invertierung der Hashfunktion h erforderlich.

Angenommen, ein Angreifer kennt das Urbild des Sitzungsschlüssel $K_{A,t-1} := y_{B,t-1}^{x_{A,t-1}} \equiv \alpha^{x_{A,t-1} \cdot x_{B,t-1}} \pmod p$. Zur Berechnung des Sitzungsschlüssels $K_t := h(K_{A,t})$ hieraus müßte er folgenden Wert berechnen können:

$$\begin{aligned} K_{A,t} &:= y_{B,t}^{x_{A,t}} \equiv \alpha^{x_{A,t} \cdot x_{B,t}} \equiv \alpha^{(x_{A,t-1} \cdot (r_A+ID_A)+k_A)(x_{B,t-1} \cdot (r_B+ID_B)+k_B)} \\ &\equiv \alpha^{(x_{A,t-1} \cdot x_{B,t-1}) \cdot (ID_A+r_A)(ID_B+r_B)} \cdot \alpha^{x_{A,t-1} \cdot (ID_A+r_A) k_B} \cdot \alpha^{x_{B,t-1} \cdot (r_B+ID_B) k_A} \cdot \alpha^{k_A k_B} \\ &\equiv K_{A,t-1}^{(ID_A+r_A)(ID_B+r_B)} \cdot y_{A,t-1}^{(ID_A+r_A)k_B} \cdot r_A^{(ID_B+r_B) x_{B,t-1}} \cdot r_A^{k_B} \pmod p. \end{aligned}$$

Hierzu muß er neben dem Urbild $K_{A,t-1}$ des alten Sitzungsschlüssels K_{t-1} noch die geheimen Parameter k_B und $x_{B,t-1}$ bzw. k_A und $x_{A,t-1}$ kennen. Diese Werte kennen aber nur genau die beiden Benutzer Alice und Bob, die hiermit ihre Fortschaltung realisieren. Alternativ könnte er auch zwei alte geheime Parameter $x_{A,i}$ und $x_{A,j}$ kennen, aus denen er aufgrund der 2-sicheren Fortschaltung alle weiteren geheimen Schlüssel berechnen könnte. Damit wird selbst bei Kenntnis des Urbildes eines Sitzungsschlüssels die Sicherheit der nachfolgenden Sitzungsschlüssel nicht beeinträchtigt. Es ist ein offenens Problem, wie viele der $(t-1)^2$ möglichen alten Sitzungsschlüssel ein Angreifer kennen darf, damit das Protokoll weiterhin amortisierende Sicherheit bietet.

Alternativ kann auch die $n-$sichere Fortschaltung aus Abschnitt 3.3.3 benutzt werden, woraus eine erhöhter Sicherheit gegenüber dem Kompromittieren alter geheimer Schlüssel, bei größerem Speicher- und Berechnungsaufwand zur Berechnung der verwendeten öffentlichen und geheimen Schlüssel, ergibt.

4.6 Anwendungen interaktionsfreier Sitzungsschlüssel

Zum einen kann die Benutzerin Alice im Zeitintervall I_t direkt eine vertrauliche Kommunikation mit dem Benutzer Bob aufbauen, ohne zuvor mit ihm kommuniziert zu haben. Dabei kann sie ihren geheimen Schlüssel $x_{A,t}$ und Bob's öffentlichen Schlüssel $y_{A,t}$ verwenden. Diese Methode hat aber den Nachteil, daß Alice und Bob während des ganzen Zeitintervalls den gleichen Sitzungsschlüssel verwenden müssen.

Daher empfiehlt sich folgende Lösung: Alice verwendet einen beliebigen geheimen Schlüssel $x_{A,i}$ sowie einen beliebigen öffentlichen Schlüssel $y_{B,j}$ von Bob, wobei sie die Nummern i und j an ihre verschlüsselte Nachricht anfügt. Der Empfänger Bob berechnet zunächst aus den Nummern i und j Alice's öffentlichen Schlüssel $y_{A,i}$ und seinen geheimen Schlüssel $x_{B,j}$. Daraus kann er den Sitzungsschlüssel $K_{i,j}$ berechnen und mit diesem das Chiffrat entschlüsseln.

Durch dieses Protokoll wird das Paradigma aufgelöst, das zwischen der Aktualität des Sitzungsschlüssels auf der einen Seite und der Interaktion auf der anderen Seite bei allen bisher bekannten Protokollen bestand [3].

5. Zusammenfassung

In diesem Papier wurden unterschiedliche Realisierungen selbstzertifizierender Schlüssel aufgezeigt. Anschließend wurden zahlreiche Anwendungen dieser Schlüssel vorgestellt. Die Charakteristik der selbstzertifizierenden öffentlichen Schlüssel liegt in ihrer einfachen Verbreitbarkeit, da neben den Systemparametern die weiteren Parameter zu ihrer Erzeugung nicht authentisch bekannt sein müssen. Dadurch sind große öffentliche Verzeichnisse mit zertifizierten öffentlichen Schlüsseln nicht erforderlich und es werden lediglich kleinere Verzeichnisse, die die authentischen öffentlichen Schlüssel der Trust Center enthalten, benötigt.

Literatur

[1] F.Bauspieß, H.-J.Knobloch, „How to keep authenticity alive in a computer network“, Lecture Notes in Computer Science 434, Advances in Cryptology: Proc. Eurocrypt '89, Berlin: Springer Verlag, (1990), S. 38–46.

[2] S.Brands, „Secret-key certificates“, CWI Report CSR-9510, (1995), 10 Seiten.

[3] M.Burmester, „On the risk of opening distributed keys“, Lecture Notes in Computer Science 839, Advances in Cryptology: Proc. Crypto '94, Berlin: Springer Verlag, (1994), S. 308–317.

[4] CCITT, Recommendation X.509 : The Directory–Authentication Framework, Blue Book - Melbourne, (1988), Fascicle VIII.8: Data communication networks: directory, International Telecommunication Union, Genf, (1989), S. 48–81.

[5] D.Chaum, „Security without identification: Transaction systems to make big brother obsolete“, Commuications of the ACM, Vol. 28, (1985), S. 1030–1044.

[6] D.Chaum, J.H.Evertse, J.van de Graaf, „Demonstrating possession of a discrete logarithms and some generalizations“, Lecture Notes in Computer Science 304, Advances in Cryptology: Proc. Eurocrypt '87, Berlin: Springer Verlag, (1988), S. 127–141.

[7] J.M.DeLaurentis, „A further weakness in the common modulus protocol for the RSA cryptoalgorithm“, Cryptologia, Vol. 8, No. 3, (1984), S. 253–259.

[8] Y.Desmedt, C.Goutier, S.Bengio, „Special uses and abuses of the Fiat-Shamir passport protocol“, Advances in Cryptology: Proc. Crypto '87, Lecture Notes in Computer Science 293, Berlin: Springer Verlag, (1988), S. 21–39.

[9] W.Diffie, M.Hellmann,„New directions in cryptography“, IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November, (1976), S. 644–654.

[10] T.ElGamal, „A public key cryptosystem and a signature scheme based on discrete logarithms“, IEEE Transactions on Information Theory, Vol. IT-30, No. 4, Juli, (1985), S. 469–472.

[11] M.Girault, „An identity-based identification scheme based on discrete logarithms modulo a composite number“, Lecture Notes in Computer Science 473, Advances in Cryptology: Proc. Eurocrypt '90, Berlin: Springer Verlag, (1991), S. 481–486.

[12] M.Girault, „Self-Certified Public Keys“, Lecture Notes in Computer Science 547, Advances in Cryptology: Proc. Eurocrypt '91, Berlin: Springer Verlag, (1991), S. 490–497.

[13] C.G.Günther, „An identity based key exchange protocol“, Lecture Notes in Computer Science 434, Advances in Cryptology: Proc. Eurocrypt ’89, Berlin: Springer Verlag, (1990), S. 29–37.

[14] L.Harn, „New digital signature scheme based on discrete logarithm“, Electronics Letters, Vol. 30, No. 5, (1994), S. 396–398.

[15] L.Harn, S.Yang, „ID-Based Cryptographic Schemes for User Identification, Digital Signatures and Key Distribution“, IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, Juni, (1993), S. 757–760.

[16] P.Horster, H.-J.Knobloch, „Discrete Logarithm based protocols“, Lecture Notes in Computer Science 547, Advances in Cryptology: Proc. Eurocrypt ’91, Berlin: Springer Verlag, (1992), S. 399–408.

[17] P.Horster, B.Klein, H.-J.Knobloch, „Authentifikation als Basis sicherer Kommunikation“, DATASAFE ’91, VDE-Verlag, Berlin, (1991), S. 219–236.

[18] P.Horster, M.Michels, H.Petersen, „Meta-ElGamal signature schemes“, Proc. der 2. ACM Conference on Computer and Communications Security, Fairfax, Virginia, 2.-4. Nov., (1994), S. 96–107.

[19] P.Horster, M.Michels, H.Petersen, „Meta-Message recovery and Meta-blind signature schemes based on the discrete logarithm problem and their applications“, Lecture Notes in Computer Science 917, Advances in Cryptology: Proc. Asiacrypt ’94, Berlin: Springer Verlag, (1995), S. 224–237.

[20] P.Horster, M.Michels, H.Petersen, „Hidden signature schemes based on the discrete logarithm problem and related concepts“, Proc. Communications and Multimedia Security, Graz, 20.–21. September, (1995), 12 Seiten.

[21] P.Horster, M.Michels, H.Petersen, „Meta-Multisignature schemes based on the discrete logarithm problem“, Proc. IFIP/SEC ’95, Kapstadt, Südafrika, (1995), S. 128–142.

[22] P.Horster, M.Michels, H.Petersen, „Das Meta-ElGamal Signaturverfahren und seine Anwendungen“, Verläßliche IT-Systeme, Proc. der GI-Fachtagung VIS ’95, DuD Fachbeiträge 22, (1995), S. 207–228.

[23] P.Horster, H.Petersen, „Signatur und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems“, Interner Bericht 94-9, RWTH Aachen, ISSN 0935-3232, Überarbeitete Fassung, März, (1994), 102 Seiten.

[24] P.Horster, H.Petersen, „Classification of blind signature schemes and examples of hidden and weak blind signatures“, Vorgetragen auf der Rump Session der Eurocrypt ’94, Perugia, Italy, (1994), 6 Seiten.

[25] C.H.Lim, P.J.Lee, „Modified Maurer-Yacobi’s scheme and its applications“, Lecture Notes in Computer Science 718, Advances in Cryptology: Proc. Auscrypt ’92, Berlin: Springer Verlag, (1992), S. 308–326.

[26] C.H.Lim, P.J.Lee, „Algorithmic Measures for preventing middleperson attack in identification schemes“, Electronics Letters, Vol. 29, No. 14, (1993), S. 1281–1282.

[27] C.H.Lim, P.J.Lee, „Several practical protocols for authentication and key exchange“, Information Processing Letters, Vol. 53, (1995), S. 91–96.

[28] U.E.Maurer, Y.Yacobi, „A remark on a non-interactive Public-Key Distribution System“, Lecture Notes in Computer Science 658, Advances in Cryptology: Proc. Eurocrypt '92, Berlin: Springer Verlag, (1993), S. 458–460.

[29] R.L.Rivest, A.Shamir, L.Adleman, „A method for obtaining digital signatures and public-key cryptosystems“, Comm. of the ACM, Vol. 21, (1978), S. 120–126.

[30] R.Rueppel, P.C.can Oorschot, „Modern key agreement techniques“, Computer communcations, Vol. 17, No. 7, (1994), S. 458 – 465.

[31] C.P.Schnorr, „Efficient identification and signatures for smart cards“, Lecture Notes in Computer Science 435, Advances in Cryptology: Proc. Crypto '89, Berlin: Springer Verlag, (1990), S. 239–251.

[32] G.J. Simmons, „A 'weak' privacy protocol using RSA crypto algorithm“, Cryptologia, Vol. 7, No. 2, (1983), S. 180-182.

[33] Y.Yacobi, Z.Shmuely, „On key distribution systems“, Lecture Notes in Computer Science 435, Advances in Cryptology: Proc. Crypto '89, Berlin: Springer Verlag, (1990), S. 344–355.