TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG

Ausarbeitung eines Seminarvortrags

Ein Gruppensignaturschema mit gleichzeitiger Verschlüsselung und seine Sicherheit

cand. inform. Gerrit Stöhr

15. Januar 2007

Institut für Theoretische Informatik
Prof. Dr. Dietmar Wätjen

Inhaltsverzeichnis

1 Einleitung

Die Idee eines Gruppensignaturverfahrens wurde das erste Mal bei der Konferenz EUROCRYPT '91 von David Chaum und Eugene van Heyst vorgestellt [5]. Ein Gruppensignaturschema ist dabei eine Methode, die einem Mitglied einer Gruppe erlaubt, eine Nachricht im Namen der Gruppe anonym zu signieren. Als Beispiel könnte dieses Verfahren von einem Angestellten einer Firma benutzt werden, um eine Nachricht ananym im Namen der Firma zu signieren. Für den Empfänger ist es wichtig, dass die Nachricht von einem Mitarbeiter der Firma signiert wurde aber nicht speziell von welchem. Eine andere Anwendung wären Keycards, um den authorisierten Zugang zu Räumen zu gewährleisten, wobei es hierbei nicht zulässig ist die Bewegung einzelner Mitarbeiter zu verfolgen.
Essentiell für ein Gruppensignaturschema ist der Gruppenmanager. Er ist verantwortlich für das Hinzufügen der Teilnehmer zu der Gruppe, berechnet den Gruppenschlüssel sowie die privaten Teilnehmerschlüssel und kann im Zweifelsfall die Anonymität einer Signatur aufheben (Feststellen des Teilnehmers, der die Nachricht signiert hat).[1]
Aus der Beschreibung und dem Beispiel ergeben sich folgende Sicherheitsaspekte und Eigenschaften, die jedes Gruppensignaturschema erfüllen muss:

- **Korrektheit:** Eine gültige Signatur eines Teilnehmers einer Gruppe wird vom Empfänger immer korrekt verifiziert.
- **Fälschungssicherheit:** Nur Mitglieder einer Gruppe können eine gültige Signatur für diese Gruppe erzeugen. Außenstehende können nicht im Namen der Gruppe signieren.
- **Anonymität:** Bei gegebener Nachricht und Signatur ist das Herausfinden der Identität des Signierers unmöglich.
- **Querverbindungslosigkeit:** Zwei Nachrichten und zwei Signaturen geben keine Informationen zu dem Signierer und es kann nicht entschieden werden, ob die Signaturen von einer Person stammen.
- **Keine falsche Identität:** Auch wenn alle anderen Gruppenmitglieder und der Manager zusammenarbeiten, können sie keine gültige Signatur für ein nicht teilnehmendes Mitglied fälschen.

[1] Es existieren auch Verfahren bei denen die Funktion des Gruppenmanagers aufgeteilt ist auf einen Mitgliedsmanager, der die Teilnehmer verwaltet und einem Rückverfolgungsmanager, der die Identität eines Signierers feststellen kann.

- **Rückverfolgbarkeit:** Nur der Gruppenmanager ist in der Lage anhand einer Signatur den Signierer festzustellen, dabei ist es ihm unmöglich einen Teilnehmer fälschlicherweise zu beschuldigen.
- **Verschwörungssicherheit:** Eine Untergruppe an Personen kann keine gültige nicht rückverfolgbare Signatur erstellen. Der Gruppenmanager ist immer in der Lage einen Teilnehmer zu beschuldigen.

Die Geheimhaltung der signierten Nachricht kann erreichen werden, indem sie verschlüsselt wird. Um hier den Rechenaufwand und den Overhead der Nachricht gering zu halten, wird die Methode von Zheng [6] benutzt. Hierbei wird die Nachricht gleichzeitig signiert und verschlüsselt (signcrypt). Die Empfänger können sie entschlüsseln und verifizieren (unsigncrypt). Das folgende Bild zeigt den Ablauf der Designschritte von dem Verfahren von Kwak und Moon [2].

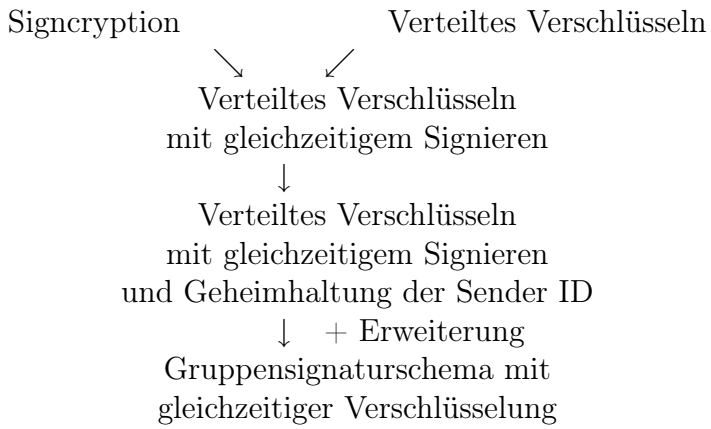

Tabelle 1.1: Übersicht der Designschritte

Sie entwickeln ihr Gruppensignaturschema mit gleichzeitiger Verschlüsselung ausgehend von dem Schema „Verteiltes Verschlüsseln mit gleichzeitigem Signieren“ und ändern es so ab, dass die Identität des Senders geheimgehalten wird. Dieses modifizierte Verfahren erweitern sie dann zu einem effizienten Gruppensignaturverfahren mit gleichzeitiger Verschlüsselung.
Der Rest der Arbeit gliedert sich wie folgt: In Kapitel 2 werden die grundlegenden Verfahren vorgestellt. Das dritte Kapitel enthält ein weiteres Gruppensignaturverfahren mit gleichzeitiger Verschlüsselung. In Kapitel 4 wird das Gruppensignaturverfahren mit gleichzeitiger Verschlüsselung von Kwak und Moon beschrieben. Auf die Effizienz und die Sicherheit des Verfahrens wird im fünften Kapitel eingegangen.

2 Grundlagen

In diesem Kapitel werden kurz die grundlegenden Verfahren vorgestellt, welche in den folgenden Kapiteln aufgebaut wird. Zu diesen gehören: Signcryption, verteiltes Ver- und Entschlüsseln sowie die Initialisierung einer Gruppe. Weitere Informationen zu den beschriebenen Verfahren sind in der angegebenen Literatur verfügbar [4],[5],[6].

2.1 Notationen

In dieser Arbeit steht p für eine große Primzahl, $\mathbb{G} \subseteq \mathbb{Z}_p^*$ für eine Untergruppe der multiplikativen Gruppe mit der Ordnung q für $q \mid (p-1)$, und $g \in \mathbb{G}$ für ein primitives Element. Desweiteren wird noch ein symmetrisches Verschlüsselungsverfahren E_k/D_k sowie eine parametrisierte stark kollisionsfreie Hashfunktion $hash_k(\,\cdot\,)$ benötigt. k ist hierbei jeweils der Schlüssel. Prinzipiell ist nicht vorgeschrieben, welches Verfahren genau benutzt werden soll.

2.2 Signierung mit Verschlüsselung

Die Standard Signierung mit gleichzeitger Verschlüsselung[1] basiert auf dem Digital Signature Standard (DSS) [3] mit einer kleinen Modifikation, die zu einer schnelleren Berechnung führt. Der modifizierte DSS wird bezeichnet als SDSS, von welchem es zwei Versionen (SDSS1 und SDSS2) gibt. Da jeweils der Sender und Empfänger beide das symmetrische Verschlüsselungssystem und die Hashfunktion benutzen, müssen sich beide vorher für je ein Verfahren entscheiden.

Der Ablauf ist wie folgt: Alice signiert und verschlüsselt gleichzeitig eine Nachricht und Bob entschlüsselt und verifiziert diese Nachricht. Dabei sind $x_a, y_a = g^{x_a} \bmod p$ und $x_b, y_b = g^{x_b} \bmod p$ jeweils die privaten und öffentlichen Schlüsselpaare von Alice und Bob. Der Schlüssel k wird in k_1 und k_2 aufgeteilt, so dass die Längen angemessen sind. Diese Methode ist öffentlich und Alice sowie Bob bekannt. Die Verschlüsselung der Nachricht erfolgt mit k_1. Der Schlüssel k_2 wird für die Hashfunktion benötigt. Die Signatur s erfolgt mit SDSS1 oder SDSS2.

Algorithmus 2.1 (Signatur-Verschlüsselung)
(1) Alice führt die folgenden Schritte aus:

[1] Alternative Verfahren, die für die Signierung mit Verschlüsselung benutzt werden können, wären das ElGamal Signaturverfahren und Schnorr's Signaturverfahren

(a) Sie wählt eine zufällige Zahl $z \in \mathbb{Z}_q$.
(b) Sie berechnet $k = y_b^z \bmod p$.
(c) Sie teilt k in k_1 und k_2 auf.
(d) Sie berechnet $r = hash_{k_2}(m)$.
$s = z(r + x_a)^{-1} \bmod p$ für SDSS1.
$s = z(1 + x_a \cdot r)^{-1} \bmod p$ für SDSS2.
$c = E_{k_1}(m)$
(e) Sie übermittelt (c, r, s) an Bob.
(2) Bob führt zur Verifizierung folgende Schritte aus:
(a) $k = (y_a \cdot g^r)^{s \cdot x_b} \bmod p$ für SDSS1.
$k = (y_a^r \cdot g)^{s \cdot x_b} \bmod p$ für SDSS2.
(b) Er teilt k in k_1 und k_2 auf.
(c) Er berechnet $m = D_{k_1}(c)$.
(d) Er akzeptiert die Signatur, falls $r = hash_{k_2}(m)$.

Beweis: Es reicht zu zeigen, dass Bob das gleiche k berechnet wie Alice, da die Korrektheit der Hashfunktion sowie des symmetrischen Verschlüsselungsverfahrens angenommen wird.
Für SDSS1: Wir erhalten

$$\begin{aligned} k &= (y_a \cdot g^r)^{s \cdot x_b} \bmod p = (y_a \cdot g^r)^{z(r+x_a)^{-1} \cdot x_b} \bmod p \\ &= (g^{x_a} \cdot g^r)^{z(r+x_a)^{-1} \cdot x_b} \bmod p = g^{(x_a+r) \cdot z(r+x_a)^{-1} \cdot x_b} \bmod p \\ &= y_b^z \bmod p, \end{aligned}$$

wobei $s = z(r + x_a)^{-1} \bmod p$ ist.
Für SDSS2: Wir erhalten

$$\begin{aligned} k &= (y_a^r \cdot g)^{s \cdot x_b} \bmod p = (y_a^r \cdot g)^{z(1+x_a \cdot r)^{-1} \cdot x_b} \bmod p \\ &= (g^{x_a \cdot r} \cdot g)^{z(1+x_a \cdot r)^{-1} \cdot x_b} \bmod p = g^{(x_a \cdot r+1) \cdot z(1+x_a \cdot r)^{-1} \cdot x_b} \bmod p \\ &= y_b^z \bmod p, \end{aligned}$$

wobei $s = z(1 + x_a \cdot r)^{-1} \bmod p$ ist. □

2.3 Verteiltes Verschlüsselungsverfahren

In diesem Abschnitt beschreiben wir das verteilte Verschlüsselungssystem. Wir betrachten eine Gruppe mit einer Anzahl von Mitgliedern und einem öffentlichen Schlüssel, der als öffentlicher Gruppenschlüssel bezeichnet wird. Jedes Mitglied der Gruppe besitzt einen privaten Schlüssel, der zu dem öffentlichen Gruppenschlüssel passt. Beim verteilten Verschlüsseln kann jeder Benutzer des Systems (innerhalb oder außerhalb der Gruppe) eine Nachricht mit dem öffentlichen Gruppenschlüssel verschlüsseln und den Gruppenmitgliedern schicken. Jedes Gruppenmitglied kann diese Nachricht nun mit seinem privaten Schlüssel entschlüsseln.

2.3.1 Initialisieren einer Gruppe

Jede Gruppe benötigt einen Gruppenmanager, der für die Erzeugung eines öffentlichen Schlüssels der Gruppe und für die Unterrichtung der Gruppenmitglieder verantwortlich ist. Um eine Gruppe mit n Mitgliedern zu konstruieren, wählt er einen Satz an Zahlen $\epsilon_i \in \mathbb{Z}_q$ für $i = 1, 2, ..., n$ und berechnet die Koeffizienten $\alpha_0, ..., \alpha_n \in \mathbb{Z}_q$ des folgenden Polynoms:

$$f(x) = \prod_{i=1}^{n}(x - \epsilon_i) = \sum_{i=0}^{n} \alpha_i x^i \tag{2.1}$$

Dann definieren wir $g \in \mathbb{G}$ und die Abbildung $g_i \leftarrow g^{\alpha_i} \bmod p$ für alle $i = 0, 1, ..., n$. Dies führt zu folgender Funktion:

$$F(\epsilon_l) = \prod_{i=0}^{n} g_i^{\epsilon_l^i} = 1 \bmod p \tag{2.2}$$

Hierbei ist ϵ_l ein Element aus $\{\epsilon_i\}$.
Beweis: Durch die Abbildung $g_i \leftarrow g^{\alpha_i} \bmod p$ lassen sich die g_i auflösen und das Produkt anschließend zu einer Summe umschreiben:

$$\prod_{i=0}^{n} g_i^{\epsilon_l^i} = \prod_{i=0}^{n} g^{\alpha_i \epsilon_l^i} = g^{\sum_{i=0}^{n} \alpha_i \epsilon_l^i}$$

Der Exponent entspricht der Gleichung 2.1 und wir können daher schreiben:

$$F(\epsilon_l) = g^{f(\epsilon_l)}$$

Da e_l ein Element aus $\{\epsilon_i\}$ ist, ist ein Faktor des Produktes aus Gleichung 2.1 null und daher ist auch $f(\epsilon_l) = 0$. Daraus folgt $F(\epsilon_l) = 1 \bmod p$. □

Würde das Polynom $\{g_i\}$ als öffentlicher Gruppenschlüssel benutzt werden und wären die ϵ_i mit $i = 1, 2, ..., n$ die privaten Schlüssel der Teilnehmer, könnte Oskar in diesem Fall ein weiteres $\epsilon_i^{'}$ dem Satz $\{\epsilon_i\}$ mit Hilfe des Polynoms $\{g_i\}$ hinzufügen.

Satz 2.1 $\{g_0, ..., g_n\}$ ist definiert wie zuvor. Mit der Wahl eines zufälligen ϵ_{n+1} kann ein Angreifer einen neuen Schlüssel $\{g^{\alpha_0^{'}}, ..., g^{\alpha_{n+1}^{'}}\}$ bestimmen, ohne die Kenntnis der ϵ_i mit $i = 1, ..., n$, so dass die Eigenschaft $\prod_{i=0}^{n+1} g_i^{\epsilon_l^i} = 1 \bmod p$ für alle ϵ_l mit $l = 1, ..., n+1$ erfüllt bleibt.
$\{\alpha_0^{'}, ..., \alpha_{n+1}^{'}\}$ sind die Koeffizienten des Polynoms $f(x) = \prod_{i=1}^{n}(x - \epsilon_i)(x - \epsilon_{n+1})$. Mit $\{g_0, ..., g_n\}$ kann der Angreifer die Koeffizien des Polynoms folgendermaßen berechnen:

$$\begin{aligned}
g^{\alpha_k^{'}} &= g^{\sum_{i_1 \neq ... \neq i_{n+1-k}}^{n} \epsilon_{i_1} .. \epsilon_{i_{n+1-k}}} \\
&= g^{\sum_{i_1 \neq ... \neq i_{n+1-k}, i_j \leq n}^{n} \epsilon_{i_1} ... \epsilon_{i_{n+1-k}}} g^{\sum_{i_1 \neq ... \neq i_{n-k}, i_j \leq n}^{n} \epsilon_{i_1} ... \epsilon_{i_{n-k}}} \\
&= g^{\alpha_{k-1}} g^{\epsilon_{n+1} \alpha_k} \\
&= g_{k-1}(g_k)^{\epsilon_{n+1}}, \qquad k = 0, 1, ..., n
\end{aligned}$$

Hierbei ist $g^{\alpha_{n+1}^{'}} = g$.

Daher wird die Methode aus [5] benutzt um die Sicherheit zu erhöhen, so dass sich Oskar nicht illegal in die Gruppe einschmuggeln kann: Für $\{\alpha_0, \alpha_1, ..., \alpha_n\}$ wird eine neue Menge $\{\alpha_0^{'}, \alpha_1^{'}, ..., \alpha_n^{'}\}$ mit $\alpha_0^{'} = \alpha_0, \alpha_n^{'} = \alpha_n, \alpha_1^{'} = ... = \alpha_{n-1}^{'} = \sum_{i=1}^{n-1} \alpha_i$ definiert. Daher wird $\beta_i \leftarrow g^{\alpha_i^{'}}$ und $A_l = \sum_{i=1,j=1,i\neq j}^{n-1} \alpha_j \epsilon_l^i$ umdefiniert, damit Gleichung 2.2 erfüllt bleibt.

$$F^{'}(\epsilon_l) = g^{-A_l} \prod_{i=0}^{n} \beta_i^{\epsilon_l^i} = g^{-A_l} g^{\sum_{i=0}^{n} \alpha_i^{'} \epsilon_l^i} = 1 \bmod p \tag{2.3}$$

Beweis:

$$g^{-A_l} g^{\sum_{i=0}^{n} \alpha_i^{'} \epsilon_l^i} = g^{-\sum_{i=1,j=1,i\neq j}^{n-1} \alpha_j \epsilon_l^i + \sum_{i=0}^{n} \alpha_i^{'} \epsilon_l^i}$$

Mit der Definition der $\{\alpha^{'}\}$ von oben ergibt sich die folgende Gleichung:

$$g^{\alpha_0 \epsilon_l^0 - \sum_{i=1,j=1,i\neq j}^{n-1} \alpha_j \epsilon_l^i + \sum_{i=1,j=1}^{n-1} \alpha_j \epsilon_l^i + \alpha_n \epsilon_l^n}$$

Die beiden Summen sind fast identisch. In der ersten Summe fehlen alle α_j mit $j = i$, in der zweiten Summe sind diese vorhanden. Daher kürzen sich alle α_j weg bis auf die, bei denen $j = i$ ist. Dies liefert die Gleichung:

$$g^{\sum_{i=0}^{n} \alpha_i \epsilon_l^i}$$

Die Summe entspricht der Gleichung 2.1, welche für ein gültiges ϵ_l null liefert. Daher ist $F^{'}(\epsilon_l) = g^{f(\epsilon_l)} = g^0 = 1 \bmod p$. □

Um einen öffentlichen Gruppenschlüssel zu erzeugen, wählt der Gruppenmanager ein zufälliges $\gamma \in \mathbb{Z}_q$ und berechnet sein Inverses $\gamma^{-1} \bmod q$. Der öffentliche Gruppenschlüssel wird definiert als ein $n+1$ Tupel $\{\beta_0, ..., \beta_{n+1}\} \leftarrow \{\beta_0, ..., \beta_n, g^{\gamma^{-1}}\}$. Für den privaten Schlüssel eines Mitglieds l der Gruppe berechnet der Manager $p_l \leftarrow -\gamma A_l \bmod q$ und sendet ϵ_l und p_l an ihn, der ϵ_l als seinen privaten Schlüssel benutzt. Der Wert γ und alle $\{\alpha_i\}$ werden vom Manager geheim gehalten.

Satz 2.2 Das Tupel des öffentlichen Schlüssels der Gruppe $\{\beta_i \mid i = 0, ..., n+1\}$ kann nicht illegal verändert werden, um ein Mitglied in die Gruppe aufzunehmen oder zu löschen.

Die Werte $\{g^{\alpha_i} \mid i = 0, ..., n\}$ sind nicht öffentlich bekannt, sondern nur in der Form $\left\{g^{\alpha_i^{'}} \mid i = 1, ..., n-1\right\}$, ebenso ist $\{A_l \mid l = 1, ..., n-1\}$ nicht öffentlich bekannt, auch nicht den Mitgliedern der Gruppe. Es ist unmöglich den öffentlichen Gruppenschlüssel so zu ändern, dass $F^{'}(x_l) = 0$ gilt.
Das beschriebene Gruppenverschlüsselungsverfahren hat die folgende Eigenschaft:

Satz 2.3 Ein Benutzer gilt als Mitglied einer Gruppe, wenn er beweisen kann, dass sein privater Schlüssel ϵ_l die Gleichung $F^{'}(\epsilon_l) = 1 \bmod p$ erfüllt.

2.3.2 Verteiltes Verschlüsseln und Entschlüsseln

Wir nehmen an, dass ein Mitglied Alice eine Nachricht m sicher an eine bestimmte Gruppe G schicken möchte. Sie wählt dazu einen zufälligen Schlüssel k, berechnet $w = hash(m)$, um anschließend den Text m zu verschlüsseln und den Chiffretext $c = (c_1, c_2)$ wie folgt zu erhalten:

$$\begin{aligned} c_1 &\leftarrow \{a_0, ..., a_{n+1}\} \leftarrow \left\{g^k\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\right\} \\ c_2 &= mg^k \bmod p. \end{aligned}$$

Bob ist ein Mitglied der Gruppe, an welche Alice die Nachricht geschickt hat. Sein privates Schlüsselpaar ist (ϵ_b, p_b). Zur Entschlüsselung führt Bob die folgende Rechnung durch:

$$c_1^{'} \leftarrow a_0(\prod_{i=1}^{n} a_i^{\epsilon_b^i})a_{n+1}^{p_b} = g^k(\prod_{i=0}^{n} g^{w\alpha_i\epsilon_b^i}) = g^k g^{wf(\epsilon_b)} = g^k \bmod p.$$

Nun kann Bob den Chiffretext $c = (c_1, c_2)$ entschlüsseln mit $m = c_2/c_1^{'}$. Jedes Mitglied der Gruppe, welches ein privates Schlüsselpaar zu G besitzt, kann den Chiffretext entschlüsseln um m zu erhalten. Hat er die Nachricht m entschlüsselt, kann er die Korrektheit der Verschlüsselung überprüfen, indem er selbst $c_1 \leftarrow \{g^k\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\}$ mit g^k und w berechnet.

Beweis: Es gilt hier zu zeigen, dass der Empfänger $c_1^{'} = g^k \bmod p$ berechnet, denn $m = \frac{c_2}{c_1^{'}} = \frac{mg^k}{g^k} = m \bmod p$.
Mit $a_0 = g^k\beta_0^w = g^k g^{w\alpha_0^{'}}, a_{n+1} = \beta_{n+1}^w = g^{\gamma^{-1}}$

$$a_0(\prod_{i=1}^{n} a_i^{\epsilon_b^i})a_{n+1}^{p_b} = g^k(\prod_{i=0}^{n} a_i^{\epsilon_b^i})g^{-\gamma\gamma^{-1}wA_b} = g^k((\prod_{i=0}^{n} \beta_i^{\epsilon_b^i})g^{-A_b})^w = g^k(F^{'}(\epsilon_b))^w$$

Da Bob einen gültigen privaten Schlüssel ϵ_b der Gruppe besitzt, liefert die Gleichung $F'(\epsilon_b) = 0$ und $c_1^{'} = g^k$. □

3 Verteiltes Signieren mit Verschlüsselung

In diesem Abschnitt wird das existierende Verteilte Signatur-Verschlüsselungssystem sowie seine Erweiterung zu einem Gruppensignaturschema mit gleichzeitiger Verschlüsselung vorgestellt [4]. Im Anschluß betrachten wir die Sicherheit und Effizienz.

3.1 Distributed Signcryption

Das verteilte Verschlüsselungsverfahren aus Kapitel 2.3.2 wird nun so erweitert, dass das gleichzeitige Signieren und Verschlüsseln möglich ist.
Algorithmus 3.1 (Signatur-Verschlüsselung)
Alice führt die folgenden Schritte aus, um Bob eine Nachricht (c_1, c_2, r, s) zu schicken.

(a) Sie wählt ein zufälliges $z \in \mathbb{Z}_q$ und berechnet $k = g^z \bmod p$.
(b) Sie teilt k in k_1 und k_2 auf.
(c) Sie berechnet $r = hash_{k_2}(m)$.
(d) Sie berechnet $s = z(kr + x_a)^{-1} \bmod q$.
(e) Sie berechnet $w = hash(m)$.

$$\begin{aligned} c_1 &= \{a_0, ..., a_n, a_{n+1}\} = \left\{g^{kr}\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\right\} \\ c_2 &= E_{k_1}(m) \end{aligned}$$

Bob, der ein Mitglied der Gruppe G ist, kann die Nachricht mit seinem privaten Schlüssel (ϵ_b, p_b) entschlüsseln und verifizieren. Dazu muss er den geheimen Schlüssel k berechnen.

$$k = (y_a a_0 (\prod_{i=1}^{n} a_i^{\epsilon_b^i}) a_{n+1}^{p_b})^s \bmod p = g^z \bmod p$$

Anschließend teilt Bob k in k_1 und k_2 auf. Er entschlüsselt die Nachricht $m = D_{k_1}(c_2)$ und verifiziert sie: $r? = hash_{k_2}(m)$.

Beweis: Es reicht zu zeigen, dass Bob das gleiche k berechnet wie Alice.

$$(y_a g^{rk} \prod_{i=0}^{n} g^{\alpha_i w \epsilon_b^i})^s \bmod p = (y_b g^{rk} g^{w \cdot \sum_{i=0}^{n} \alpha_i \epsilon_b^i})^s \bmod p = (y_b g^{rk} g^{w f(\epsilon_b)})^s \bmod p$$

Besitzt Bob einen gültigen Schlüssel (ϵ_b, p_b) der Gruppe, so ist $f(\epsilon_b) = 0$ und die Gleichung vereinfacht sich zu

$$(y_a g^{rk})^s \bmod p.$$

Ab hier ist der Beweis ähnlich dem in Kapitel 2.2 zu führen. Nach Einsetzen von $y_a = g^{x_a}$ und $s = z(kr + x_a)^{-1}$ ergibt sich

$$(g^{x_a} g^{rk})^{z(k+x_a)^{-1}} \bmod p = g^{(x_a+rk)\,\cdot\, z(kr+x_a)^{-1}} \bmod p = g^z \bmod p. \; \square$$

3.2 Erweiterung zu einem Gruppenverfahren

Das eben vorgestellte verteilte Verschlüsselungsverfahren kann zu einem Gruppensignaturverfahren erweitert werden, welches einem Mitglied einer Gruppe erlaubt, eine Nachricht im Namen seiner Gruppe zu signieren sowie gleichzeitig zu verschlüsseln und an eine Person in einer anderen Gruppe zu schicken. Wir betrachten zwei bestimmte Gruppen G_A und G_B, außerdem nehmen wir an, dass Alice, die zu der Gruppe G_A gehört, eine signierte und verschlüsselte Nachricht m an Bob schicken möchte, der zu der Gruppe G_B gehört. Alice besitzt ϵ_a, ein Teil ihres privaten Schlüssels der Gruppe G_A, welches $f(\epsilon_a) = 0 \bmod p$ erfüllt (definiert wie in 2.3.1). Bob besitzt ebenfalls einen privaten Schlüssel ϵ_b mit derselben Eigenschaft. Die öffentlichen Schlüssel der Gruppen G_A und G_B sind $\{\overline{\beta}_0, \overline{\beta}_1, ..., \overline{\beta}_{n+1}\}$ bzw. $\{\beta_0, \beta_1, ..., \beta_{n+1}\}$. Beide öffentlichen Schlüssel haben die gleiche Form wie schon bereits im Abschnitt 3.1 definiert.

Signatur-Verschlüsselung: Um eine Nachricht m zu signieren und gleichzeitig zu verschlüsseln, führt Alice die folgenden Schritte aus und hält dabei (z, k, w) geheim:

(a) Sie wählt ein zufälliges $z \in \mathbb{Z}_q$ und berechnet $k = g^z \bmod p$.
(b) Sie teilt k in k_1 und k_2 auf.
(c) Sie berechnet $w = hash(m)$.
(d) Sie berechnet die Signatur $u_j \leftarrow \overline{\beta}_j^w$ für $j = 1, ..., n$.
(e) Sie berechnet $r = hash_{k_2}(m)$.
(f) Sie berechnet $s_j = z(\epsilon_a^j - ru_j) \bmod q$ für $j = 1, ..., n$.
(g) Die signierte und verschlüsselte Nachricht lautet wie folgt:

$$\begin{aligned} c_1 &\leftarrow \{\alpha_0, ..., \alpha_n, \alpha_{n+1}\} \leftarrow \{g^{kr}\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\} \\ c_2 &\leftarrow \{\overline{\alpha_{00}}, u_1, ..., u_n, \overline{\alpha}_{n+1}\} \leftarrow \{g^{z-rk}\overline{\beta}_0^w, u_1, ..., u_n, \overline{\beta}_{n+1}^{wp_a}\} \\ c_3 &= E_{k_1}(m) \end{aligned}$$

Sie sendet das folgende Tupel $(c_a, c_2, c_3, r, s1)$ an die Gruppe G_B.

Bob, oder jemand anderes aus der Empfänger-Gruppe, kann die Nachricht entschlüsseln und die Gruppensignatur verifizieren. Dies basiert auf der Rekonstruktion

des Schlüssels k

$$\begin{aligned} k &= \left[a_0(\prod_{i=1}^{n} a_i^{\epsilon_b^i})a_{n+1}^{p_b}\right]\left[\overline{a}_0 \prod_{j=1}^{n} u^{ru_j}\overline{\beta}_j^{s_j})\overline{a}_{n+1}\right] \\ &= \left[g^{kr}\beta_0^w(\prod_{i=0}^{n}\beta_i^{w\epsilon_b^i})\beta_{n+1}^{p_b}\right]\left[g^{z-rk}\overline{\beta}_0^w(\prod_{j=1}^{n}\overline{\beta}_j^{w\epsilon_a^j})\overline{\beta}_{n+1}^{wp_a}\right] \\ &= \left[g^{rk}\prod_{i=0}^{n}\beta_i^{w\epsilon_b^i}\right]\left[g^{z-rk}\prod_{j=0}^{n}\overline{\beta}_j^{w\epsilon_a^j}\right] \\ &= g^z \bmod p \end{aligned}$$

und anschließendem Überprüfen von a_0 und $\overline{a}_0$ mit dem Schlüssel k.

3.3 Analyse

Distributed Signcryption Dieses Verfahren gewährleistet nicht die Geheimhaltung der Identität des Senders, denn der Empfänger der signierten und verschlüsselten Nachricht (a_1, c_2, r, s) benötigt den öffentlichen Schlüssel y_a des Senders. In diesem Fall müsste der Sender sein Zertifikat vorab der Empfänger-Gruppe übermitteln, oder alle Teilnehmer, welche die Nachricht entschlüsseln wollen, müssen sich den öffentlichen Schlüssel des Senders y_a besorgen. Daher ist dieses Verfahren nicht geeignet für Anwendungen, die eine Geheimhaltung der Identität des Senders erforden, wie zum Beispiel bei elektronischen Wahlen oder der Beitreten Methode bei Gruppensignaturverfahren.

Erweiterung zu einem Gruppensignaturverfahren Zu Beginn, wenn die Gruppe initialisiert wird, wählt und berechnet der Gruppenmanager für jeden seiner Teilnehmer ein privates Schlüsselpaar $\{(\epsilon_i, p_i)\}_{i=1}^{n}$. In diesem Fall würde ein erfolgreicher Angriff auf den Gruppenmanager sämtliche privaten Schlüssel der Teilnehmer kompromitieren. Als Beispiel wird der private Schlüssel (ϵ_l, p_l) durch einen Angriff auf den Gruppenmanager aufgedeckt. Der Angreifer kann sich nun als Teilnehmer l ausgeben und erfolgreich eine Nachricht im Namen dieser Gruppe signieren und verschlüsseln. Daher ist dieser Fall sehr gefährlich in Protokollen, bei denen Gruppen involviert sind. In einem anderen Fall könnte ein Gruppenmanager eine Signatur im Namen der Gruppe von einem bestimmten Teilnehmer seiner Gruppe fälschen, da er ja sämtliche privaten Schlüssel kennt. Außerdem muss der Empfänger einer Nachricht die Identität des Senders kennen um sie zu entschlüsseln und zu verifizieren.
Die Effizienzbetrachtung folgt im Abschnitt 5.1.

4 Gruppensignaturverfahren mit gleichzeitiger Verschlüsselung

In diesem Kapitel beschreiben wir das Design des Gruppensignaturschemas mit gleichzeitiger Verschlüsselung von Kwak und Moon [2]. Dazu erläutern wir erst die Veränderung am vorhandenen Verfahren, so dass die Identität des Senders geheim gehalten wird und stellen dann die Erweiterung zu dem Gruppensignaturschema mit gleichzeitiger Verschlüsselung vor.

4.1 Verteiltes Signatur-Verschlüsselungssystem mit Geheimhaltung der ID des Senders

Das in 2.3.2 beschriebene Verfahren wird so erweitert, dass eine gleichzeitige Signierung und Verschlüsselung möglich ist sowie geändert, damit die Identität des Senders geheim gehalten wird. Bevor bei diesem Verfahren Nachrichten geschickt werden können, muss eine Gruppe wie es in Abschnitt 2.3.1 beschrieben wurde initialisiert werden.

Signatur-Verschlüsselung Alice führt die folgenden Schritte aus, um Bob, der zu einer bestimmten Gruppe gehört, eine Nachricht (c_1, c_2) zu schicken. $Cert_a$ ist das Zertifikat von Alice mit ihrem öffentlichen Schlüssel und $x \parallel y$ ist die Konkatenation von x und y:

(a) Sie wählt ein zufälliges $z \in \mathbb{Z}_q$ und berechnet $k = g^z \bmod p$.
(b) Sie teilt k in k_1 und k_2 auf.
(c) Sie berechnet $r = hash_{k_2}(m)$
(d) Sie berechnet $s = z(r + x_a)^{-1} \bmod q$ für SDSS1
$s = z(1 + x_a \cdot r)^{-1} \bmod q$ für SDSS2
(e) Sie berechnet $w = hash(m)$.
(f) Die verschlüsselte Nachricht ist:

$$\begin{aligned} c_1 &\leftarrow \{a_0, ...a_n, a_{n+1}\} \leftarrow \left\{k\beta_0^w, \beta_1^w, ..., \beta_{n+1}^w\right\} \\ c_2 &= E_{k_1}(m \parallel r \parallel s \parallel Cert_a) \end{aligned}$$

Der Sitzungsschlüssel k wird hier nicht für die Berechnung von s sondern für c_1 in der Form $k\beta_0^w$ benutzt; ähnlich der ElGamal Verschlüsselung. Durch diese Methode kann der Empfänger den Sitzungsschlüssel k ohne die Kenntnis von r, s zu berechnen.

Daher können wir r, s und $Cert_a$ in c_2 verschlüsseln. Die Identität des Senders ist mit in dem verschlüsselten Text c_2 enthalten.

Verifikation-Entschlüsselung Bob, oder jemand anderes aus der Gruppe, kann die verschlüsselte Nachricht mit seinem privaten Schlüssel (ϵ_b, p_b) entschlüsseln und verifizieren. Dies basiert auf dem Berechnen des Schlüssels k. Er führt dazu die folgenden Schritte durch:

(a) Er berechnet den Schlüssel $k = a_0(\prod_{i=1}^{n} a_i^{\epsilon_b^i}) a_{n+1}^{p_b} = g^z \bmod p$.
(b) Er teilt k in k_1 und k_2 auf.
(c) Er entschlüsselt c_2: $D_{k_1}(c_2) = m \parallel r \parallel s \parallel Cert_a$.
(d) Er verifiziert $r? = hash_{k_2}(m)$.
(e) Er verifiziert $g^z? = (y_a \cdot g^r)^s$ für SDSS1
$g^z? = (g \cdot y_a^r)^s$ für SDSS2

Beweis: a_0 kann aufgelöst werden zu $k\beta_0^w$. Anschließend kann β_0 mit in das Produkt gezogen werden, da $\epsilon_b^0 = 1$ ist. Schließlich ergibt sich mit $p_b = \gamma A_b$ durch Ersetzen der a und β folgendes:

$$a_0(\prod_{i=1}^{n} a_i^{\epsilon_b^i}) a_{n+1}^{p_b} = k \cdot (\prod_{i=0}^{n} \beta_i^{w\epsilon_b^i}) \beta_{n+1}^{p_b} = k \cdot (\prod_{i=0}^{n} g^{w\alpha_i'\epsilon_b^i}) \cdot g^{-\gamma A_b w \gamma^{-1}} \bmod p$$

Nach dem Ausklammern von w ergibt sich die folgende Gleichung, wobei der Therm in den Klammern der Gleichung genau $F'(\epsilon_b)$ entspricht:

$$k((\prod_{i=0}^{n} \beta_i^{\epsilon_b^i}) g^{-A_b})^w = k \cdot (F'(\epsilon_b))^w \bmod p$$

Da Bob einen gültigen privaten Schlüssel zu der Gruppe besitzt, ist $F'(\epsilon_b) = 1$. Somit ist $k = g^z \bmod p$. □

Anstelle nur der Berechnung des Schlüssels k bei der Verifikation-Entschlüsselung wird auch die Signatur überprüft. Es ist offensichtlich, dass der Empfänger die Nachricht mit der oben genannten Methode entschlüsseln und verifizieren kann. Die einzige Möglichkeit zur Entschlüsselung ist der Besitz eines geheimen Schlüsselpaares (ϵ_b, p_b) dieser Gruppe, welcher $f(\epsilon_b) = 0 \bmod q$ liefert. Sie muss den öffentlichen Schlüssel der Gruppe benutzen, denn sonst kann kein Mitglied der Gruppe ihre Nachricht entschlüsseln und verifizieren. Alice muss ebenfalls ihren geheimen Schlüssel x_a benutzen um s zu berechnen, da ihr öffentlicher Schlüssel zur Verifikation benutzt wird. Falls dieser nicht x_a enthält, wird die zweite Verifikation von Bob fehlschlagen.

4.2 Erweiterung zu einem Gruppensignaturverfahren

Nun wird dieses eben beschriebene Schema zu einem effizienten Gruppensignatur-Verschlüsselungsschema erweitert. Es gibt fünf Methoden, die ein Gruppensignaturverfahren beherrschen muss: Initialisierung, Beitreten, Signatur-Verschlüsselung, Verifikation-Entschlüsselung und die Rückverfolgung. Durch diese Methoden ist es dann einem Teilnehmer möglich, anonym eine Nachricht im Namen seiner Gruppe zu signieren und an eine andere Person zu schicken.

Initialisierung Zur Erzeugung der Informationen für die Gruppe berechnet der Gruppenmanager die folgenden Werte:

- p, q und g sind dieselben Werte wie zuvor, $h \in \mathbb{G}$ wird neu generiert.
- Er wählt ein n_A als großen RSA Modulus mit zwei zufälligen Primzahlen gleicher Größe, sowie den RSA Signaturschlüssel d_A des Gruppenmanagers der Gruppe G_A. Anschließend berechnet er e_A, wobei die Gleichung $e_A \cdot d_A \equiv 1 \mod \phi(n_A)$ erfüllt sein muss.

Der Gruppenmanager hält d_A als seinen Signaturschlüssel geheim und veröffentlicht (p, q, g, h, n_A, e_A) als Parameter der Gruppe.

Beitreten Jede Person die einer Gruppe betreten möchte, generiert ihren eigenen privaten Schlüssel ϵ_l und berechnet $\tau_l (= h^{\epsilon_l} \mod p)$ als Gruppenmitgliedsschlüssel. Anschließend übermittelt sie τ_l auf sicherem Wege an den Gruppenmanager und beweist ihm die Kenntnis des diskreten Logarithmus von τ zur Basis h. ϵ_l sollte geheim gehalten werden. Der Gruppenmanager braucht daher nicht die geheimen Schlüssel der Mitglieder zu generieren und speichern; dies kann die Sicherheit erhöhen. Für jeden Teilnehmer berechnet der Gruppenmanager $v_l (= \tau_l^{d_A} \mod n_A)$ als Mitgliedszertifikat. Die Benutzung der RSA Signatur ist einfach und effizient, um sicherzustellen ob, τ_l gültig ist oder nicht. Um die Gruppe zu initialisieren, berechnet der Gruppenmanager die Koeffizienten des folgenden Polynoms:

$$f(x) = \prod_{i=1}^{n} (x - \tau_i) = \sum_{i=0}^{n} \alpha_i x^i \tag{4.1}$$

Die Definition von $\{\alpha_i'\}$ und $\{\beta_i'\}$ für $i = 0, ..., n$ lautet wie in Abschnitt 2.3.1 und $A_l = \sum_{i=1, j=1, i \neq j}^{n-1} \alpha_j \tau_l^i$, dann besitzt die Gleichung 4.1 folgende Eigenschaft:

$$F'(\tau_l) = g^{-A_l} \prod_{i=0}^{n} \beta_i^{\tau_l^i} = g^{-A_l} g^{\sum_{i=0}^{n} \alpha_i' \tau_l^i} = g^{f(\tau_l)} = 1 \mod p \tag{4.2}$$

Zur Erzeugung eines öffentlichen Schlüssels der Gruppe wählt der Gruppenmanager eine zufällige Zahl γ und berechnet ihr Inverses γ^{-1} und $p_l = -\gamma A_l \bmod p$ für Mitglied l. Der öffentliche Schlüssel der Gruppe wird als ein $n+1$ Tupel definiert zu $\{\beta_0, ..., \beta_{n+1}\} \leftarrow \{\beta_0, ..., \beta_n, g^{\gamma^{-1}}\}$. Anschließend hält er γ, alle $\{\alpha_i\}$ sowie $\{\tau_l\}$ geheim und sendet v_l und p_l an das Gruppenmitglied l, welches dieses mit ϵ_l und tau_l als seinen privaten Schlüssel der Gruppe benutzt.
Die Initialisierung der Gruppe erfolgt also genau wie in Abschnitt 2.3.1, nur benutzt der Gruppenmanager für seine Berechnungen nicht die ϵ_l sondern τ_l, da er ja von jedem Teilnehmer nur τ_l kennt. Für die Beweise verweisen wir hier daher auf den Abschnitt 2.3.1.

Signatur-Verschlüsselung Wir betrachten zwei bestimmte Gruppen G_A und G_B und nehmen an, dass Alice zu der Gruppe G_A gehört und Bob einer der Empfänger aus der Gruppe G_B ist. Um eine Nachricht m zu signieren und gleichzeitig zu verschlüsseln, führt Alice die folgenden Schritte unter Benutzung ihrer ϵ_a, τ_a und v_a aus:

(a) Sie wählt ein zufälliges z und $t \in \mathbb{Z}_q$ und berechnet $k = g^z \bmod p$.
(b) Sie teilt k in k_1 und k_2 auf.
(c) Sie berechnet $r = hash_{k_2}(m)$
(d) Sie berechnet $s = z(r + \epsilon_a \cdot t)^{-1} \bmod q$ für SDSS1
$s = z(1 + \epsilon_a \cdot r \cdot t)^{-1} \bmod q$ für SDSS2
(e) Sie berechnet $w = hash(m)$.
(f) Sie berechnet $\lambda_a = (t^{e_A} \cdot \tau_a \bmod n_A) \bmod q, \delta_a = g^{\epsilon_a t}$ und $\theta_a = t \cdot v_a \bmod n_A$.
(g) Die verschlüsselte Nachricht (c_1, c_2) lautet wie folgt:

$$\begin{aligned} c_1 &\leftarrow \{\alpha_0, ..., \alpha_{n+2}\} \leftarrow \{k\beta_0^{w\tau_a}, \beta_1^{w\tau_a}, ..., \beta_{n+1}^{w\tau_a}, g^{\lambda_a}\} \\ c_2 &= E_{k_1}(ID_{G_A} \parallel m \parallel r \parallel s \parallel \delta_a \parallel \theta_a) \end{aligned}$$

Hierbei ist ID_{G_A} die Identität der Gruppe G_A mit den öffentlichen Informationen des Gruppenmanagers (n_A, e_A). Die restlichen Notationen sind wie im vorigen Kapitel. Das τ_a bei der Berechnung von c_1 wird später für die Rückverfolgung, also der Feststellung der Identität des Signierers, benötigt.
Wir sehen, dass Alice für die Berechnung von s ihr ϵ_a und die Zufallszahl t benutzt. Bei dem normalen Signcryption-Verfahren hat Alice ihren privaten Schlüssel x_a benutzt. Für die Verifizierung benötigte Bob den öffentlichen Schlüssel des Senders, den es aber bei diesem Verfahren nicht gibt. Daher schickt Alice diese Information $\delta_a = g^{\epsilon_a t}$, die ja Bob zum Verifizieren benötigt, in verschlüsselter Form an Bob.

Verifikation-Entschlüsselung Bob oder jedes andere Mitglied der Gruppe G_B kann die Nachricht, unter Benutzung seiner (τ_b, p_b), basierend auf der Rekonstruktion des Sitzungsschlüssels k entschlüsseln und verifizieren. Dazu müssen folgende Schritte ausgeführt werden:

(a) $k \leftarrow a_0(\prod_{i=1}^{n} a_i^{\tau_b^i}) a_{n+1}^{p_b} = g^z \prod_{i=0}^{n} g^{w\tau_a\alpha_i\tau_b^i} = g^z g^{w\tau_a f(\tau_b)} = g^z \bmod p$.

(b) Er teilt k in k_1 und k_2 auf.
(c) Er entschlüsselt $D_{k_1}(c_2) = ID_{G_A} \parallel m \parallel r \parallel s \parallel \delta_a \parallel \theta_a$.
(d) Er berechnet $\lambda_a^{'} = (\theta_a^{e_A} \bmod n_A) \bmod q$.
(e) Er verifiziert $r? = hash_{k_2}(m)$.

$$g^z? = (\delta_a \cdot g^r)^s \text{ für SDSS1}$$
$$g^z? = (g \cdot \delta_a^r)^s \text{ für SDSS2}$$
$$a_{n+2}? = g^{\lambda_a^{'}}.$$

Mit der Überprüfung von g^z verifiziert Bob die korrekte Berechnung des Schlüssels k sowie mit a_{n+2} die Gruppenzugehörigkeit des Senders.

Beweis: Für den Beweis der Korrektheit des Verfahrens müssen 3 Schritte (Berechnung des Schlüssels, Verifizierung Schlüssel und Verifizierung Gruppenzugehörigkeit) nachgewiesen werden:
Berechnung des Schlüssels: Der Beweis hierfür ist ähnlich dem in Abschnitt 4.1 und wird daher hier nicht explizit vorgestellt. Die einzige Änderung ist die Benutzung von Alices τ_a bei der Verschlüsselung der einzelnen Komponenten des öffentlichen Schlüssels, diese lassen sich ebenso wie w ausklammern.
Verifizierung Schlüssel: Bob verifiziert den Schlüssel, indem er ihn selbst mit Hilfe von δ_a, r und s folgendermaßen berechnet und mit dem aus Schritt 1 Berechneten vergleicht:

$$g^z = (\delta_a \cdot g^r)^s = (g^{\epsilon_a t} \cdot g^r)^s = g^{(\epsilon_a t + r) z (r + \epsilon_a t)^{-1}} = g^z \bmod p$$

Verifizierung Gruppenzugehörigkeit: Bob vergleicht hierfür das letzte Element aus c_1 mit g exponentiert mit seinem berechnetem $\lambda^{'}$:

$$\begin{aligned} a_{n+2} &= g^{\lambda^{'}} \\ g^{\lambda_a} &= g^{\lambda_a^{'}} \end{aligned}$$

Wir sehen hier, dass es genügt λ_a und $\lambda_a^{'}$ auf Gleichheit mod q zu überprüfen. Dies führt zu folgender Gleichung:

$$\begin{aligned} t^{e_A} \cdot \tau_a \bmod n_A &= \theta_a^{e_A} \bmod n_A \\ t^{e_A} \cdot \tau_a \bmod n_A &= (t \cdot v_a)^{e_A} \bmod n_A \\ t^{e_A} \cdot \tau_a \bmod n_A &= t^{e_A} \cdot \tau_a \bmod n_A \end{aligned}$$

Führt Bob diese Schritte erfolgreich durch, so akzeptiert er die Nachricht und Signatur von Alice. □

Die einzige Möglichkeit die Nachricht zu entschlüsseln, ist der Besitz eines gültigen Gruppenmitgliesschlüssels τ_b mit zugehörigem p_b, welcher $f(\tau_b) = 0 \bmod q$ liefert. Sie musste den öffentlichen Schlüssel der Gruppe benutzen, ansonsten kann niemand

aus der Gruppe die Nachricht entschlüsseln. Alice musste ebenfalls ihren privaten Gruppenschlüssel ϵ_a und τ_a sowie das Mitgliedszertifikat v_a für die Verschlüsselung benutzen, denn nur gültige Schlüssel und Zertifikate werden bei der Verifikation akzeptiert.

Rückverfolgung Im Falle eines Streits kann Bob c_1 und $w(= hash(m))$ an den Gruppenmanager der Gruppe G_A übermitteln, nachdem er c_2 entschlüsselt und die Identität der Gruppe festgestellt hat. In diesem Fall kann der Gruppenmanager feststellen, dass Alice die Nachricht geschickt hat, indem er die Gleichung $\{a_i? = (\beta_i^w)^{\tau_l}\}_{i=1}^{n+1}$ für die τ_l aller Mitglieder in G_A überprüft. Nach dieser Prozedur kann der Gruppenmanager den Streit lösen. Aber dieses Verfahren benötigt sehr viel Rechenzeit in großen Gruppen.

5 Analyse und Sicherheit

In diesem Kapitel betrachten wir die Effizienz des in Kapitel 4 beschriebenen Verfahrens von Kwak und Moon und vergleichen dies mit dem vorhandenen Schema aus Kapitel 3. Anschließend erläutern wir einen möglichen Angriff [1] auf das Verfahren und untersuchen mit diesen Erkenntnissen die am Anfang geforderten Sicherheitsaspekte.

5.1 Effizienz des Verfahrens

Der Rechenaufwand wird bezüglich der Modulo-Rechnung, bestehend aus der modularen Exponentiation, modularen Multiplikationen und der Bildung des modular Inversen betrachtet. Da diese Rechnungen am meisten Zeit benötigen, werden die anderen vernachlässigt. In der Tabelle steht I_q für die Anzahl an Inversen modulo q, $M_q(M_p)(M_{n_A})$ für die Anzahl an Multiplikationen modulo $q(p)(n_A)$ und $E_q(E_p)(E_{n_A})$ für die Anzahl an Exponentiationen modulo $q(p)(n_A)$. In dem in Kapitel 4 beschriebenem Verfahren von Kwak et al wird eine einfache RSA Signatur benutzt um Angriffen auf die Identität vorzubeugen. Die untenstehende Tabelle vergleicht den Rechenaufwand der beiden Gruppensignaturverfahren mit gleichzeitiger Verschlüsselung.

Kosten		Signcryption		Unsigncryption	
		SDSS1	SDSS2	SDSS1	SDSS2
Vergleichs Schema	M_q	2n+3	2n+3	n	n
	M_p	1	1	3n+3	3n+3
	E_q	n-1	n-1	n-1	n-1
	E_p	n+7	n+7	3n+1	3n+1
Schema von Kwak und Moon	I_q	1	1	-	-
	M_q	n+4	n+5	-	-
	M_p	1	1	n+2	n+2
	E_q	-	-	n-1	n-1
	E_p	n+5	n+5	n+4	n+4
	M_{n_A}	2	2	-	-
	E_{n_A}	1	1	1	1

Tabelle 5.1: Vergleich der beiden Verfahren

Als Ergebnis des Vergleichs der Rechenaufwände lässt sich sagen, dass das Verfahren von Kwak et al aus Kapitel 4 effizienter ist als das bisherige. Besonders mit wachsender Gruppengröße n gewinnt dieser Effizienzvorteil sowohl beim Verschlüsseln wie auch beim Entschlüsseln an Bedeutung.

5.2 Sicherheit und möglicher Angriff

Korrektheit Die Korrektheit des Verfahrens wurde bereits im Kapitel 4.2 bei dem Beweis der korrekten Entschlüsselung und Verifizierung durch den Empfänger gezeigt.

Fälschen der Signatur Nach Aussage des Autors kann der private Schlüssel ϵ_a des Signierers nicht aufgedeckt werden, aufgrund des Problems, den diskreten Logarithmus zu lösen. Daher können nur Mitglieder der Gruppe die Nachricht entschlüsseln und verifizieren. Dies bedeutet nicht, dass ein Angreifer nicht auch andere Methoden verwenden kann um eine gültige verschlüsselte Signatur zu erzeugen. Wir wollen zeigen, dass eine Methode, eine gültige Gruppensignatur für eine gegebene Nachricht zu erzeugen, unter der Annahme, dass wir kein gültiges Zertifikat eines Mitgliedes kennen (ϵ_i, τ_i, vi), existiert.
Hierzu muss Oskar die folgenden Komponenten fälschen: Schlüsselkapselung, korrekte Signatur mit Verschlüsselung, Sitzungsschlüssel und die Gruppenzugehörigkeit.
Wir nehmen an, dass Oskar eine signierte und verschlüsselte Nachricht an Bob schicken möchte, der ein Empfänger der zugehörigen Gruppe G_B ist. Der konkrete Angriff sieht wie folgt aus:

(a) Oskar wählt zufällige t, t_1 und $z \in \mathbb{Z}_q$ und berechnet $k = g^z \bmod p$.
(b) Er teilt k in k_1 und k_2 auf.
(c) Er berechnet $r = hash_{k_2}(m)$.
(d) Er berechnet $s = z(r + t_1)^{-1} \bmod q$.
(e) Er berechnet $w = hash(m)$.
(f) Er wählt ein zufälliges $u \in \mathbb{Z}_{n_A}^*$ und berechnet $\theta_a = u \bmod n_A$.
(g) Er berechnet $\lambda_a = (\theta_a^{e_A} \bmod n_A) \bmod q = (u^{e_A} \bmod n_A) \bmod q$ und $\delta_a = g^{t_1}$.
(h) Er wählt ein zufälliges $t_2 \in \mathbb{Z}_q^*$.

Die gefälschte signierte und verschlüsselte Nachricht lautet wie folgt:

$$\begin{aligned} c_1 &= \{\alpha_0, ..., \alpha_{n+2}\} = \{k\beta_0^{wt_2}, \beta_1^{wt_2}, ..., \beta_{n+1}^{wt_2}, g^{\lambda_a}\} \\ c_2 &= E_{k_1}(ID_G \parallel m \parallel r \parallel s \parallel \gamma_a \parallel \theta_a) \end{aligned}$$

Dabei ist ID_G beliebig.

Anonymität Mit einer korrekt entschlüsselten Nachricht ist das Identifizieren des Signierers berechnungsmäßig schwierig für jeden Teilnehmer bis auf den Gruppenmanager. Die Informationen über den Signierer sind verschlüsselt in $\delta_a (= g^{\epsilon_a t})$ und

$\theta_a (= t \cdot v_a)$ enthalten, wobei hier t zufällig gewählt ist, also bei jeder Anwendung wechselt. Daher wird keine Information über die Identität des Senderes durch $ID_{G_A} \parallel m \parallel r \parallel s \parallel \delta_a \parallel \theta_a$ offenbart.

Querverbindungslosigkeit Die Entscheidung, ob zwei korrekt entschlüsselte Nachrichten $(ID_{G_A} \parallel m \parallel r \parallel s \parallel \delta_a \parallel \theta_a)$ und $(ID_{G_A} \parallel \hat{m} \parallel \hat{r} \parallel \hat{s} \parallel \hat{\delta}_a \parallel \hat{\theta}_a)$ von ein und dem selben Teilnehmer der Gruppe berechnet wurden, ist berechnungsmäßig schwer. Genau wie bei der Anonymität reduziert sich das Problem der Querverbindung auf die Entscheidung, ob δ_a und $\hat{\delta}_a$ oder θ_a und $\hat{\theta}_a$ von einer Person berechnet wurden. Dies basiert auf dem Problem des diskreten Logarithmus und dem Finden einer zufälligen Zahl, die bei jeder Anwendung wechselt.

Rückverfolgbarkeit In der oben beschriebenen Methode zum Fälschen der Signatur benutzt der Angreifer ein zufälliges t_2 zum Berechnen der Signatur. Daher kann der Gruppenmanager nicht den Signierer identifizieren, indem er $\{a_i = (\beta_i^w)^{\tau_l}\}_{i=1}^{n+1}$ für alle τ_l der Teilnehmer seiner Gruppe testet, da $\{a_i = (\beta_i^w)^{t_2}\}$ für $i = 1, ..., n+1$. Dies würde auch für alle Gruppen G_i zutreffen.

Keine falsche Identität Damit eine Person in unserem System eine falsche Identität vortäuschen kann, benötigt sie das τ_l jenes Teilnehmers. Die einziger Person in unserem System, die außerdem diesen Wert kennt, ist der Gruppenmanager. Daher ist es nur für ihn möglich, eine Nachricht in fremden Namen zu signieren und zu verschlüsseln.

Verschwörungssicherheit Durch den möglichen Angriff haben wir gesehen, dass der Angreifer nicht zurückverfolgt werden kann. Würden die Teilnehmer, die sich verschworen haben, ebenfalls diese Methode benutzen um ihre Nachricht zu signieren und zu verschlüsseln, könnte kein Teilnehmer zurückverfolgt werden.

Geheimhaltung Die Nachricht m sowie die von dem Sender berechneten Werte werden mit dem symmetrischen Verschlüsselungsverfahren unter Benutzung des Schlüssels k_2 verschlüsselt. Die Sicherheit der Geheimhaltung beruht also auf der Sicherheit des verwendeten symmetrischen Verschlüsselungssystems.

6 Abschlussbemerkungen

Wir haben gesehen, dass das Verfahren von Kwak und Moon effizienter ist als das bisherige Gruppensignaturverfahren. Dies bezieht sich sowohl auf die Signierung mit Verschlüsselung als auch auf die Verifikation und Entschlüsselung. Ein Unterschied zwischen den beiden Varianten SDSS1 und SDSS2 ist dagegen kaum messbar.

Bedeutender ist aber die Tatsache, dass der in Abschnitt 5.2 skizzierte Angriff auf das Verfahren geglückt ist. Daher sind die folgenden, zu Beginn dieser Arbeit formulierten Sicherheitsaspekte, nicht erfüllt: Fälschungssicherheit, Verschwörungssicherheit und die Rückverfolgbarkeit. Jede beliebige Person kann also eine Nachricht im Namen einer Gruppe signieren und verschlüsseln. Durch diese schwerwiegende Sicherheitsverletzung kann dieses Verfahren in der Praxis nicht benutzt werden.

Eine weitere negative Eigenschaft des Gruppensignaturschemas von Kwak und Moon ist, dass die Teilnehmer einer Gruppe diese nicht verlassen können oder neue Teilnehmer die Gruppe betreten können. Die Gruppe ist also nicht dynamisch. Ändert sich die Gruppengröße, so muss der Gruppenmanager den öffentlichen Schlüssel der Gruppe neu berechnen. Diese Eigenschaft wurde zwar nicht explizit gefordert, wäre aber wünschenswert.

Literatur

[1] H. Bao, Z. Cao und H Q.: On the Security of a Group Signcryption Scheme from Distributed Signcryption Scheme. In: *Cryptology and Network Security CANS 2005* (2005), S. LNCS Vol. 3810, Seiten 26–34 17

[2] S. Moon, D K.: Efficient Distributed Signcryption Scheme as Group Signcryption. In: *Applied Cryptography and Network Security ACNS '03* (2003), S. LNCS Vol. 2846, Seiten 403–417 2, 11

[3] Standards, National I. ; Technology: Digital Signature Standard. (1994) 3

[4] Y. Mu, V. V.: Distributed signcryption. In: *Advanced in Cryptology - INDOCRYPT '2000* (2000), S. LNCS Vol. 1977, Seiten 155–164 3, 8

[5] Y. Mu, V. Varadharajan und K. Q. N.: Delegated decryption. In: *Cryptography and Coding '99* (1999), S. LNCS Vol. 1746, Seiten 258–269 1, 3, 6

[6] Zheng, Y.: Signcryption and its application in efficient public key solutions. In: *Information Security Workshop - ISW '97* (1997), S. LNCS Vol. 1396, Seiten 291–312 2, 3