Bachelorarbeit

Analyse & Evaluation eines identitätsbasierten Signcryption Verfahrens

Gennadij Liske
E-Mail: utyf@mail.uni-paderborn.de

Paderborn den 28.09.2009

vorgelegt bei
Prof. Dr. Johannes Blömer
Prof. Dr. Friedhelm Meyer auf der Heide

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Paderborn, den 28.09.2009

Gennadij Liske

Inhaltsverzeichnis

1 Einleitung

Vertraulichkeit und Verbindlichkeit sind zwei wichtige Ziele der modernen Kryptographie, die mittels asymmetrischer Verfahren erreicht werden können. Heute brauchen viele Anwendungen in der Praxis sowohl das eine als auch das andere. Derzeitige Vorgehensweise in solchen Fällen ist es erst die Nachricht zu signieren und dann zu verschlüsseln. Schon 1997 stellte Zheng [Zhe97] die Frage, ob effizientere Verfahren entwickelt werden können, wenn man beide Ziele zusammen angeht und nicht getrennt betrachtet. Er nannte solche Verfahren Signcryptions.

Identitätsbasierte Kryptographie ist eine von Shamir im Jahr 1984 [Sha84] vorgeschlagene Form der asymmetrischen Kryptographie, bei der eine beliebige eindeutige Identität des Benutzers als öffentlicher Schlüssel eingesetzt werden kann. Dadurch entfällt die Notwendigkeit den öffentlichen Schlüssel zu zertifizieren. Seit 2001, als Boneh und Franklin das erste voll funktionierende und beweisbar sichere identitätsbasierte Verschlüsselungsverfahren basierend auf Abbildungsgruppen [BF01] vorgestellt haben, wurden viele neuartige identitätsbasierte Verfahren, unter anderem auch Signcryptions, entwickelt.

Ende 2004 stellten Baretto, Libert, McCullagh und Quisquater [BLMQ05] eine neue identitätsbasierte Signatur (IBS[1]) und ein darauf aufbauendes identitätsbasiertes Signcryptionverfahren (IBSC[2]) vor und zeigten deren Sicherheit im Zufallsorakel-Modell. Weiterhin verglichen die Autoren die Effizienz der Verfahren mit den anderen bis dahin bekannten identitätsbasierten Signatur- und Signcryptionverfahren, die bilineare Abbildungsgruppen benutzen, wobei die neuen Verfahren effizienter als alle anderen waren. Im Anhang wurde ein weiteres Signaturverfahren allerdings ohne Darlegung des Sicherheitsbeweises vorgeschlagen.

Ziel dieser Bachelorarbeit ist es, diese drei Verfahren vorzustellen sowie deren Effizienz und Sicherheit zu analysieren. Dabei hat sich während der Bearbeitung der Arbeit herausgestellt, dass die Sicherheitsbeweise in der Originalarbeit an vielen Stellen sehr unpräzise bis inkorrekt durchgeführt wurden. Aus diesem Grund wurden die Beweise vervollständigt und teilweise sehr stark verändert.

Weiterhin wird in Hinsicht auf die Anwendung in der Praxis untersucht, inwiefern das Signcryptionverfahren sich so anpassen lässt, dass ohne größere Änderungen an

[1] „Identity-based signature"
[2] „Identity-based signcryption"

dem Verfahren selbst dieses als reines identitätsbasiertes Verschlüsselungsverfahren benutzt werden kann.

Die Bachelorarbeit ist wie folgt strukturiert: Zuerst werden im Kapitel 2 alle für diese Arbeit notwendigen Definitionen gegeben und erläutert. Im Kapitel 3 werden dann die beiden Signaturverfahren vorgestellt und die Sicherheitsbeweise für diese analysiert. Im Kapitel 4 wird das Signcryptionverfahren vorgestellt und dessen Sicherheitsbeweise analysiert. Im Kapitel 5 wird dann das Sakai-Kasahara-Verschlüsselungsverfahren [SK03] vorgestellt, denn durch einige wenige Anpassungen des Signcryptionverfahren lässt sich dieses als Sakai-Kasahara-Verschlüsselungsverfahren anwenden.

2 Grundlagen

In diesem Kapitel wird zunächst in Abschnitt 2.1 die Notation vorgestellt und grundlegende Definitionen angegeben, die allen folgenden Teilen zugrunde liegt. In Abschnitt 2.2 werden dann für die Arbeit notwendige kryptographische Primitive erläutert. In Abschnitt 2.3 wird über die beweisbare Sicherheit diskutiert und ein für diese Arbeit besonders wichtiges Lemma vorgestellt. In den letzten beiden Abschnitten werden die Sicherheitsdefinitionen für die in den nächsten Kapiteln vorgestellte Verfahren angegeben.

2.1 Notationen und grundlegende Definitionen

In diesem Abschnitt werden zuerst Vereinbarungen für Notationen getroffen sowie grundlegende Definitionen angegeben, die in der Arbeit an vielen Stellen genutzt werden.

Auswahl eines Elementes aus einer Menge

An vielen Stellen in der Arbeit muss ein Element x zufällig, gleichverteilt aus einer Menge A ausgewählt werden. In diesem Fall schreiben wir:

$$x \xleftarrow{R} A$$

R steht dabei für „Random".

Definitionen/Zuweisungen

Definition eines Elementes bzw. Zuweisung eines Wertes werden mit dem Symbol $:=$ dargestellt. Wenn z.B. x als Summe von zwei Elementen y und z definiert wird, schreiben wir:

$$x := y + z$$

Probabilistische Algorithmen

Die meisten in der Arbeit vorgestellten Algorithmen sind probabilistisch. Jeder derartige Algorithmus ist prinzipiell ein deterministischer Algorithmus mit einer zusätzlichen Eingabe ρ von Zufallsbits, die bei der Ausführung benutzt werden. Wenn man also einen probabilistischen Algorithmus zweimal mit der gleichen Eingabe und den gleichen Zufallsbits ausführt, verhält sich dieser wie ein deterministischer Algorithmus und liefert die gleiche Ausgabe bei beiden Ausführungen. An den Stellen, wo wir auf solche Ausführung des Algorithmus A mit der Eingabe x und Zufallsbits ρ hinweisen wollen, schreiben wir:

$$A\left(x;\rho\right)$$

Ausführung von Algorithmen

Für die Ausführung eines deterministischen Algorithmus A mit der Eingabe x, bei der die Ausgabe y produziert wird, schreiben wir:

$$y := A\left(x\right)$$

Bei einem probabilistischen Algorithmus B schreiben wir dementsprechend entweder

$$y \overset{R}{\leftarrow} B\left(x\right)$$

wenn die Zufallsbits ρ zufällig, gleichverteilt gewählt werden oder

$$y := B\left(x;\rho\right)$$

wenn wir diesen mit bestimmten Zufallsbits ρ ausführen.

Fehlersymbol

Algorithmen, die ihre Arbeit abbrechen, weil z.B. die Verifizierung fehlschlägt, geben das Fehlersymbol $\perp$ zurück.

Vernachlässigbare Funktionen

Bei den Sicherheitsbeweisen fordert man, dass die Wahrscheinlichkeit, dass ein Angreifer das Verfahren bricht, sehr klein ist. Formal braucht man dafür die Definition der vernachlässigbaren Funktionen.

Definition 1. Funktion *negl* heißt *vernachlässigbar*, wenn für jede Konstante $c \in \mathbb{N}$ ein $N \in \mathbb{N}$ existiert, so dass für alle $k > N$ gilt:

$$negl\left(k\right) < k^{-c}$$

Beispiel. Die Funktion $\frac{1}{2^k}$ ist vernachlässigbar. Weiterhin sind die Funktionen $negl(k) + negl(k)$ und insbesondere $negl(k) \cdot poly(k)$ für eine zeitpolynomielle (in k) Funktion $poly(k)$ vernachlässigbar. Wir werden das in dieser Arbeit ohne weitere Anmerkungen verwenden.

Bilineare Abbildungsgruppen

Alle Verfahren, die in dieser Arbeit vorgestellt werden, basieren auf bilinearen Abbildungsgruppen, die im Bereich der identitätsbasierten Kryptographie in zahlreichen Verfahren verwendet werden. Die folgende Definition ist äquivalent zu der Definition in der Originalarbeit [BLMQ05].

Definition 2. Seien $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ zyklische Gruppen der gleichen Primordnung p und seien $P \in \mathbb{G}_1$ und $Q \in \mathbb{G}_2$ Generatoren der jeweiligen Gruppen. $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ heißen *bilineare Abbildungsgruppen*, wenn eine bilineare Abbildung $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ mit folgenden Eigenschaften existiert:

1. e ist bilinear, d.h. es gilt:

$$\forall A_1, A_2 \in \mathbb{G}_1, B \in \mathbb{G}_2 : e(A_1 \cdot A_2, B) = e(A_1, B) \cdot e(A_2, B)$$

$$\forall A \in \mathbb{G}_1, B_1, B_2 \in \mathbb{G}_2 : e(A, B_1 \cdot B_2) = e(A, B_1) \cdot e(A, B_2)$$

2. e ist nicht entartet, d.h. $Bild(e) \neq \{1\}$
3. e ist effizient berechenbar
4. Es existiert ein öffentlich bekannter und effizient berechenbarer Isomorphismus $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, so dass $\psi(Q) = P$ gilt.

In der Literatur wird Bilinearität oft über die folgende Eigenschaft definiert:

$$\forall A \in \mathbb{G}_1, B \in \mathbb{G}_2, x, y \in \mathbb{Z} : e(A^x, C^y) = e(A, C)^{x \cdot y}$$

Man kann zeigen, dass diese Eigenschaft für zyklische Gruppen mit Primordnung tatsächlich zu denen in Punkt 1 angegebenen Eigenschaften äquivalent ist.

Wichtig ist auch, dass bei den Gruppen mit Primordnung alle Elemente außer dem 1-Element Generatoren sind und auf Grund der zweiten Eigenschaft auch das Element $e(P, Q) \in \mathbb{G}_T$ ein Generator ist, wenn $P \in \mathbb{G}_1$ und $Q \in \mathbb{G}_2$ Generatoren sind. Darauf wird in der Arbeit nicht immer explizit hingewiesen.

In dieser Arbeit schreiben wir im Unterschied zu der Originalarbeit alle Gruppen multiplikativ, in der Notation von [BB04], um unnötige Missverständnisse an einigen Stellen zu vermeiden.

Kryptographische Sicherheitsannahmen

Die Sicherheitsbeweise, die in dieser Arbeit vorgestellten werden, basieren auf Annahmen, dass bestimmte Probleme in bilinearen Abbildungsgruppen schwer zu lösen sind. An dieser Stelle werden diese Probleme und dazugehörige Annahmen vorgestellt.

Ein bekanntes Problem bei zyklischen Gruppen ist das Problem des diskreten Logarithmus, bei dem für ein gegebenes Generator $G \in \mathbb{G}$ und ein weiteres Element $\widetilde{G} \in \mathbb{G}$ der Exponent $\alpha \in \mathbb{Z}_{|\mathbb{G}|}$ bestimmt werden muss, so dass $G^{\alpha} = \widetilde{G}$ gilt. Das Problem kann auch auf bilineare Abbildungsgruppen übertragen werden. Wir benutzen allerdings noch strengere Annahmen. Diese entsprechen den Definitionen in der Originalarbeit [BLMQ05]:

Definition 3. Seien $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ bilineare Abbildungsgruppen mit Generatoren $P \in \mathbb{G}_1$ und $Q \in \mathbb{G}_2$, so dass $\psi(Q) = P$ gilt. Das *q-strenge Diffie-Hellman-Problem* (***q*-SDH-Problem**) in den Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ besteht darin, bei einem gegebenen $(q+2)$-Tupel $\left(P, Q, Q^{\alpha}, Q^{\alpha^2}, ..., Q^{\alpha^q}\right)$ mit $\alpha \in \mathbb{Z}_p^*$ ein Paar $\left(c, P^{\frac{1}{c+\alpha}}\right)$ für ein beliebiges $c \in \mathbb{Z}_p^*$ zu berechnen.

Definition 4. Seien $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ bilineare Abbildungsgruppen mit Generatoren $P \in \mathbb{G}_1$ und $Q \in \mathbb{G}_2$, so dass $\psi(Q) = P$. Das *q-bilineare Diffie-Hellman-Invertierungsproblem* (***q*-BDHI-Problem**) in den Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ besteht darin, bei einem gegebenen $(q+2)$-Tupel $\left(P, Q, Q^{\alpha}, Q^{\alpha^2}, ..., Q^{\alpha^q}\right)$ mit $\alpha \in \mathbb{Z}_p^*$ das Element $e(P,Q)^{1/\alpha}$ zu berechnen.

Offensichtlich muss die Berechnung des diskreten Logarithmus in den Gruppen schwer sein, denn sonst könnte man α bestimmen und somit das gesuchte Paar $\left(c, P^{\frac{1}{c+\alpha}}\right)$ für jedes $c \neq -\alpha$ bzw. das Element $e(P,Q)^{1/\alpha}$ berechnen.

Um die entsprechenden Annahmen zu definieren, benötigen wir noch eine Definition des Parametergenerators, der für beide Probleme gleich ist.

Definition 5. Ein probabilistischer Algorithmus *Gen* heißt Parametergenerator, wenn bei Eingabe eines Sicherheitsparameters 1^k dieser in polynomieller Zeit (in k) eine Primzahl p mit $p \geq 2^k$, bilineare Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ mit der Ordnung p, die dazugehörigen Generatoren $P \in \mathbb{G}_1$ und $Q \in \mathbb{G}_2$ mit $\psi(Q) = P$ sowie die Funktionen e und ψ erzeugt.[1]

Definition 6. Sei *Gen* ein Parametergenerator. Wir definieren den Vorteil $Adv_{Gen,A}^{SDH}(k)$ eines Angreifers A beim Lösen des q-SDH-Problems als

$$Pr\left[A\left(\begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T), \\ P, Q, Q^{\alpha}, Q^{\alpha^2}, ..., Q^{\alpha^q}\end{array}\right) = \left(c, P^{\frac{1}{c+\alpha}}\right) \middle| \begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \xleftarrow{R} Gen\left(1^k\right) \\ \alpha \xleftarrow{R} \mathbb{Z}_p^*\end{array}\right]$$

[1] 1^k bedeutet, dass die Eingabe k unär kodiert ist.

und sagen, dass der Angreifer A einen Vorteil $\varepsilon(k)$ beim Lösen des q-SDH-Problems hat, wenn für genügend große k gilt:

$$Adv^{SDH}_{Gen,A}(k) \geq \varepsilon(k)$$

Wir sagen, dass ein Parametergenerator Gen der q-SDH-Annahme genügt, wenn für alle zeitpolynomielle (in k) probabilistische Algorithmen A der Vorteil $Adv^{SDH}_{Gen,A}(k)$ von A eine vernachlässigbare Funktion ist. Wenn Gen dieser Annahme genügt, sagen wir, dass das q-SDH-Problem hart in den von Gen generierten Abbildungsgruppen ist.

Definition 7. Sei Gen ein Parametergenerator. Wir definieren den Vorteil $Adv^{BDHI}_{Gen,A}(k)$ eines Angreifers A beim Lösen des q-BDHI-Problems als

$$Pr\left[A\left(\begin{array}{c}(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T),\\ P,Q,Q^{\alpha},Q^{\alpha^2},...,Q^{\alpha^q}\end{array}\right)=e(P,Q)^{1/\alpha}\;\middle|\;\begin{array}{c}(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_T)\overset{R}{\leftarrow}Gen\left(1^k\right)\\ \alpha\overset{R}{\leftarrow}\mathbb{Z}_p^*\end{array}\right]$$

und sagen, dass der Angreifer A einen Vorteil $\varepsilon(k)$ beim Lösen des q-BDHI-Problems hat, wenn für genügend große k gilt:

$$Adv^{BDHI}_{Gen,A}(k) \geq \varepsilon(k)$$

Wir sagen, dass ein Parametergenerator Gen der q-BDHI-Annahme genügt, wenn für alle zeitpolynomielle (in k) probabilistische Algorithmen A der Vorteil $Adv^{BDHI}_{Gen,A}(k)$ von A eine vernachlässigbare Funktion ist. Wenn Gen dieser Annahme genügt, sagen wir, dass das q-BDHI-Problem hart in den von Gen generierten Abbildungsgruppen ist.

In dieser Arbeit werden wir keinen Parametergenerator angeben, sondern gehen davon aus, dass dieser existiert und betrachten nur die jeweiligen Probleme. Ein Beispiel für einen derartigen Parametergenerator ist in [BF01] angegeben. Immer, wenn wir bilineare Abbildungsgruppen erzeugen werden, gehen wir davon aus, dass dabei ein solcher Parametergenerator benutzt wird.

2.2 Kryptographische Primitive

In diesem Abschnitt wird erst darauf eingegangen, welche Ziele in der modernen Kryptographie verfolgt werden und mit welchen Methoden diese erreicht werden können. Dann werden die drei großen Paradigma der modernen Kryptographie vorgestellt: symmetrische, asymmetrische und identitätsbasierte Verfahren. An dieser Stelle wird aber nicht über die Vor- und Nachteile diskutiert, sondern viel mehr die Prinzipien erklärt und auf die Unterschiede hingewiesen. Am Ende des Abschnitts werden zwei Typen von identitätsbasierten Verfahren, nämlich Signaturen und Signcryptions, die in diese Arbeit betrachtet werden, vorgestellt und formal definiert.

Ziele und Methoden der modernen Kryptographie

Kryptographische Verfahren werden nicht nur dafür eingesetzt, um eine Nachricht zu verschlüsseln, also die Vertraulichkeit der Informationen zu gewährleisten. Viel mehr existieren mehrere Ziele, die durch Kryptographie erreicht werden können. Für verschiedene Ziele werden verschiedene Methoden und Verfahren eingesetzt. An dieser Stelle werden die wichtigsten vorgestellt.

- **Vertraulichkeit** bedeutet, dass bestimmte Informationen nur für bestimmte Personen vorgesehen sind. D.h. dass diese von einem unbefugten Zugriff geschützt werden sollen. Dieses Ziel kann durch Verschlüsselungen erreicht werden.
- **Datenintegrität** bedeutet, dass Daten von Manipulationen geschützt werden müssen. Um dieses Ziel zu erreichen werden Hashfunktionen und MACs[2] eingesetzt.
- **Authentifizierung von Daten** bedeutet, dass die Identität des Senders vom Empfänger überprüft werden kann. Das Ziel erreicht man mit verschiedenen Authentifizierungprotokollen, MACs oder digitalen Signaturen.
- **Verbindlichkeit** bedeutet, dass der Absender von Daten deren Absenden nicht leugnen kann. D.h., dass nur der Sender und kein anderer eine Nachricht gesendet haben konnte, was bei Authentifizierungsmethoden nicht der Fall sein muss. Weiterhin kann nicht nur der Empfänger, sondern jeder überprüfen, ob die Nachricht vom Sender gesendet wurde. Dieses Ziel erreicht man mit digitalen Signaturen.

Natürlich können mehrere Ziele gleichzeitig erwünscht sein. Z.B. kann durch richtiges Einsetzen von Verschlüsselungen und MACs sowohl die Vertraulichkeit als auch die Datenintegrität erreicht werden. Manche Anwendungen in der Praxis benötigen alle vier Eigenschaften. Die Verbindlichkeit kann dabei nur mit Hilfe von asymmetrischen Verfahren erreicht werden. Derzeitige Vorgehensweise in solchen Fällen ist es erst die Nachricht zu signieren und dann zu verschlüsseln. Schon 1997 stellte Zheng [Zhe97] die Frage, ob effizientere Verfahren entwickelt werden können, wenn man beide Ziele zusammen angeht und nicht getrennt betrachtet. Er nannte solche Verfahren Signcryptions.

Nun wenden wir uns den drei großen kryptographischen Primitiven zu. Diese unterscheiden sich vor allem daran welche Geheimnisse welche Benutzer kennen. Wir stellen diese am Beispiel von Verschlüsselungsverfahren vor.

Symmetrische Kryptographie[3]

Die ältesten kryptographischen Verfahren sind die symmetrischen Verfahren. D.h. dass beide Kommunikationspartnern den geheimen Schlüssel kennen. Das Prinzip

[2] *„Message authentication code"*
[3] In der englischen Literatur auch *private-key cryptography* genannt.

wird im folgenden Diagramm am Beispiel von symmetrischen Verschlüsselung erläutert:

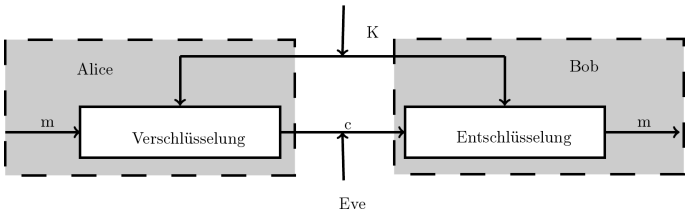

Abbildung 2.1: Symmetrische Verschlüsselung

Alice und Bob wollen miteinander über einen unsicheren Kanal kommunizieren. Dafür tauschen sie erst einen Schlüssel K über einen sicheren Kanal aus (das kann z.B. vorab geschehen). Alice verschlüsselt dann die Nachricht m mit dem Schlüssel K und schickt die Verschlüsselung c an Bob. Bob entschlüsselt die Nachricht mit dem Schlüssel K. Ein Angreifer Eve kann die Verschlüsselung möglicherweise lesen, darf aber nichts über die Nachricht m lernen.

Für viele moderne Applikationen reichen symmetrische Verfahren nicht aus. Das Problem liegt gerade darin, dass der Schlüssel erst auf einem sicheren Wege ausgetauscht werden muss. Wenn Alice und Bob räumlich weit voneinander entfernt sind, ist ein Treffen keine praktikable Lösung. Andererseits kann durch symmetrische Verfahren die Verbindlichkeit nicht erreicht werden, denn falls mehr als eine Person den geheimen Schlüssel kennt, kann man nicht überprüfen von wem genau eine Nachricht stammt.

Asymmetrische Kryptographie[4]

Die asymmetrische Kryptographie basiert auf einer anderen Idee. Bei diesen Verfahren besitzt Bob zwei Schlüssel, einen geheimen, den nur er kennt, und einen öffentlichen. Das Prinzip wird am Beispiel von Verschlüsselungen im nächsten Diagramm erläutert:

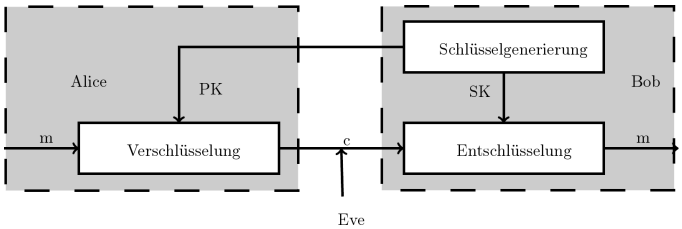

Abbildung 2.2: Asymmetrische Verschlüsselung

[4] In der englischen Literatur auch *public-key cryptography* genannt.

Bob berechnet ein Paar von öffentlichen und geheimen Schlüsseln (PK, SK).[5] Der Schlüssel PK wird öffentlich gemacht. Alice verschlüsselt die Nachricht m mit PK. Die verschlüsselte Nachricht kann nun über einen unsicheren Kanal an Bob geschickt werden. Bob entschlüsselt die Nachricht mit Hilfe seines geheimen Schlüssels SK. Der Angreifer Eve kann auch im Besitz von PK sein, darf aber trotzdem nichts aus der Verschlüsselung c über die Nachricht m lernen.

Aus dem Diagramm wird deutlich, dass bei asymmetrischen Verschlüsselungsverfahren nur Bob die Nachricht entschlüsseln kann, während bei symmetrischen Verfahren beide Kommunikationspartner sowohl die Ver- als auch die Entschlüsselungen berechnen können.

Nun stellt sich die Frage wie Alice in Besitz von PK kommt. Eine einfache Übertragung über einen unsicheren Kanal funktioniert nicht, da der Schlüssel manipuliert werden kann. Eine Lösung für dieses Problem ist der Aufbau von PKI (Public Key Infrastruktur). Vereinfacht dargestellt ist das eine vertrauenswürdige Stelle, der sowohl Alice als auch Bob vertrauen. Bob legt seinen öffentlichen Schlüssel bei dieser vertrauenswürdigen Stelle ab. Der Schlüssel wird von dieser zertifiziert und kann dann an andere Benutzer verteilt werden. Alice kann den Schlüssel von der vertrauenswürdigen Stelle abholen und dessen Echtheit überprüfen.

Ein weiteres wichtiges Ziel, das man nur mit asymmetrischen Verfahren erreichen kann ist die Verbindlichkeit. Alice will dabei eine Nachricht so verschicken, dass überprüft werden kann, ob diese wirklich von ihr stammt. Diese Überprüfung soll von jedem durchgeführt werden können und nicht nur vom Empfänger. Dabei benutzt Alice ihren geheimen Schlüssel, um zu einer Nachricht eine Signatur zu berechnen. Jeder andere kann dann mit Hilfe des öffentlichen Schlüssels von Alice überprüfen, ob die Signatur zu der Nachricht passt. Wenn keiner außer Alice eine solche Signatur berechnen kann, ist die Verbindlichkeit gegeben. Wir betrachten Signaturverfahren ausführlicher bei den identitätsbasierten Verfahren.

Mit asymmetrischen Verfahren kann man also alle vorgestellte Ziele erreichen. Allerdings sind diese Verfahren im Vergleich zu symmetrischen Verfahren ineffizient. Deswegen werden in der Praxis sowohl die einen, als auch die anderen Verfahren eingesetzt und zwar so, dass diese sich gegenseitig ergänzen. Z.B. kann man mit Hilfe von asymmetrischen Verfahren einen geheimen Schlüssel über einen unsicheren Kanal austauschen und dann ein symmetrisches Verschlüsselungsverfahren mit dem ausgetauschten Schlüssel benutzen.

Identitätsbasierte Kryptographie

Die Notwendigkeit der Zertifizierung der Schlüssel und der Aufbau von PKI kann man aber auch umgehen. Im Jahr 1984 stellte Shamir eine neue Form der asymmetrischen Kryptographie vor [Sha84], bei der jede beliebige Identität ID des Be-

[5] PK steht für *public key* und SK für *secret key*.

nutzers (z.B. seine E-Mail-Adresse oder sein Personalausweisnummer) als öffentlicher Schlüssel benutzt werden kann. Aber auch bei identitätsbasierten Verfahren ist eine vertrauenswürdige Stelle notwendig. Diese bekommt aber eine ganz andere Bedeutung. Die vertrauenswürdige Stelle baut das ganze System auf, indem es den Setup-Algorithmus ausführt und die öffentlichen Parameter *params* sowie den Masterschlüssel MK bestimmt. Die vertrauenswürdige Stelle kann als einzige die geheimen Schlüssel zu jeder Identität mit Hilfe von MK berechnen.

Weiterhin muss jeder Benutzer nur einmal die öffentlichen Parameter und seinen eigenen geheimen Schlüssel von der vertrauenswürdigen Stelle abholen, während bei gewöhnlichen asymmetrischen Verfahren man für jeden Benutzer, mit dem man kommunizieren will, seinen öffentlichen Schlüssel abholen muss.

Das Diagramm macht die Anwendung von solchen identitätsbasierten Verfahren deutlich:

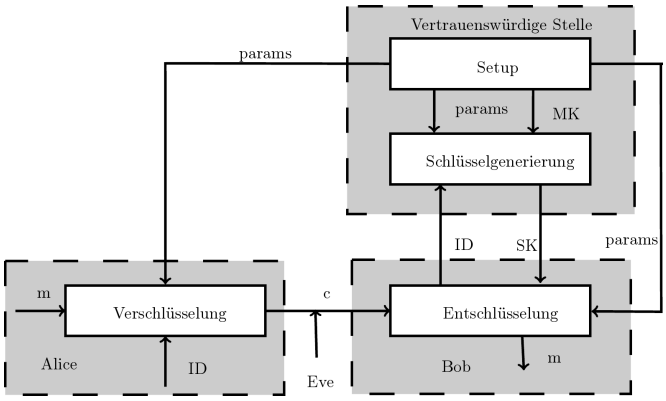

Abbildung 2.3: Identitätsbasierte Verschlüsselung

Alice kennt die Identität ID von Bob und kann eine Nachricht m mit dieser Identität als öffentlicher Schlüssel verschlüsseln. Bob entschlüsselt die Nachricht mit seinem geheimen Schlüssel SK. Der Angreifer kann zwar die Identität von Bob und die öffentlichen Parameter kennen, darf aber trotzdem nichts über die Nachricht m aus der Verschlüsselung c erfahren.

In der oben genannten Arbeit von Shamir stellte er das erste identitätsbasierte Signaturverfahren vor. Es dauerte aber noch mehr als 15 Jahren, bis 2001 Boneh und Franklin das erste voll funktionierende und beweisbar sichere identitätsbasierte Verschlüsselungsverfahren [BF01] vorgestellt haben. Somit gehören die identitätsbasierte Verfahren zu den neuartigen Verfahren der modernen Kryptographie.

Identitätsbasierte Signaturen

In den vorherigen Abschnitten wurden die Prinzipien von symmetrischen, asymmetrischen und identitätsbasierten Verfahren auf der Grundlage von Verschlüsselungen

erklärt. Nun wollen wir diese Prinzipien auf identitätsbasierte Signaturen und Signcryptions übertragen und dabei auch formale Definitionen für diese Verfahren angeben.

Betrachten wir zuerst identitätsbasierte Signaturen. Das Prinzip ist im folgenden Diagramm erklärt, wobei wir die vertrauenswürdige Stelle der Übersicht halber vereinfacht darstellen.

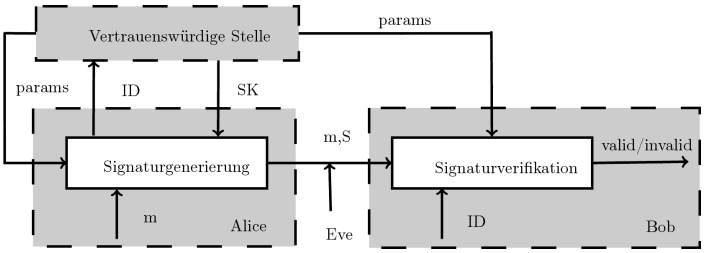

Abbildung 2.4: Identitätsbasierte Signatur

Alice will eine Nachricht m an Bob schicken, die nicht vertraulich ist. Es muss aber sicher gestellt werden, dass jeder überprüfen kann, ob die Nachricht von Alice geschickt wurde. Mit Hilfe ihres geheimen Schlüssels erstellt Alice eine Signatur S für die Nachricht m und sendet beide an Bob. Bob kann nun mit Hilfe der Identität von Alice überprüfen, ob die Nachricht wirklich von Alice stammt und nicht gefälscht wurde.

Aus dem Diagramm wird deutlich, dass Eve die Nachricht lesen kann. Die Vertraulichkeit der Nachricht wollten wir aber auch nicht erreichen. Wenn Eve aber versucht die Nachricht oder die Signatur zu verfälschen, sollte die Verifikation nicht mehr erfolgreich ablaufen. Dadurch erreichen wir insbesondere Verbindlichkeit, denn keiner außer Alice kann die Nachricht mit ihrem geheimen Schlüssel signieren. Wenn also die Verifikation einer Signatur mit der Identität von Alice erfolgreich ist, ist sichergestellt, dass diese und somit die Nachricht auch von Alice stammen.

Formal definiert man identitätsbasierte Signaturen wie folgt:

Definition 8. *Identitätsbasierte Signatur (IBS)* ist ein Tupel von vier zeitpolynomiellen Algorithmen $\Pi = (\text{Setup}, \text{Extract}, \text{Sign}, \text{Verify})$:

- **Setup** - ist ein probabilistischer Algorithmus, der von der vertrauenswürdigen Stelle ausgeführt wird und bei Eingabe des Sicherheitsparameters 1^k den geheimen Masterschlüssel s sowie die öffentlichen Systemparameter *params* erzeugt.

  $$(s, params) \xleftarrow{R} Setup\left(1^k\right)$$

  Die öffentlichen Systemparameter dienen implizit als Eingabe für alle anderen Algorithmen.

- **Extract** - ist ein (probabilistischer) Algorithmus, der bei Eingabe des Masterschlüssels s und einer Identität ID den entsprechenden geheimen Schlüssel S_{ID} berechnet.
$$S_{ID} \stackrel{R}{\leftarrow} Extract\,(ID, s)$$

- **Sign** - ist ein probabilistischer Algorithmus, der bei Eingabe des geheimen Schlüssel S_{ID} und einer Nachricht m die Signatur σ berechnet.
$$\sigma \stackrel{R}{\leftarrow} Sign\,(m, S_{ID})$$

- **Verify** - ist ein deterministischer Algorithmus, der bei Eingabe einer Identität ID des Senders und einer Nachricht m die Gültigkeit der entsprechenden Signatur σ überprüft. Der Algorithmus gibt ein Bit b aus. $b = 1$ bedeutet, dass die Signatur gültig ist und $b = 0$, dass sie ungültig ist.
$$b := Verify\,(m, \sigma, ID)$$

Das Verfahren ist korrekt, wenn für alle für das Verfahren gültige Werte von ID und m gilt:
$$Verify\,(m, Sign\,(m, S_{ID})\,, ID) = 1$$

Dies muss für beliebige von Setup und Extract zufällig gewählte Parameter und Schlüssels sowie für beliebige von dem Sign-Algorithmus erzeugte Signaturen gelten.

Identitätsbasierte Signcryptions

Nun wollen wir zusätzlich die Vertraulichkeit der Nachricht garantieren. Wie oben schon erwähnt, ist es möglich die Nachricht erst zu signieren und dann zu verschlüsseln. Mit Signcryptions versucht man aber dadurch, dass beides in einem Schritt gemacht wird, einen Effizienzgewinn zu erzielen. Im Vergleich zu den identitätsbasierten Signaturen, wo nur die Identität des Senders bei der Erstellung der Signatur benutzt wird, müssen bei Signcryptions sowohl die Identität des Senders als auch die Identität des Empfängers in die Berechnung der Signcryption eingehen. Das Prinzip wird im nächsten Diagramm deutlich gemacht, wobei wir die vertrauenswürdige Stelle der Übersicht halber vereinfacht darstellen:

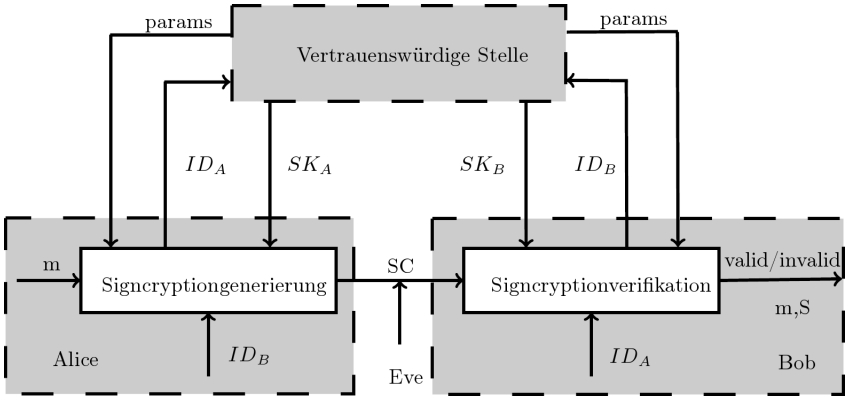

Abbildung 2.5: Identitätsbasierte Signcryption

Sowohl Alice als auch Bob brauchen deren jeweiligen geheimen Schlüssel SK_A bzw. SK_B. Alice kann nun mit Hilfe von SK_A und ID_B die Signcryption SC für die Nachricht m berechnen und diese an Bob schicken. Bob verifiziert die Signcryption mit Hilfe von SK_B und ID_A und falls die Verifikation erfolgreich war, wird die Nachricht m mit der dazugehörigen Signatur S berechnet. Eve kennt sowohl die beiden Identitäten als auch die Signcryption, darf aber nichts über die Nachricht m lernen und kann die Signcryption auch nicht verändern ohne, dass die Verifikation bei Bob fehlschlägt.

Man beachte, dass sowohl die Erzeugung als auch die Überprüfung der Signcryption zwei Algorithmen sind, die nicht ohne weiteres in Signatur bzw. Verschlüsselung getrennt werden können. Das wird aus der formalen Definition deutlich:

Definition 9. *Eine identitätsbasierte Signcryption (IBSC)* ist ein Tupel von fünf zeitpolynomiellen Algorithmen Π = (Setup, Extract, Sign/Encrypt, Decrypt/Verify, Verify):

- **Setup** - ist ein probabilistischer Algorithmus, der von der vertrauenswürdigen Stelle ausgeführt wird und bei Eingabe des Sicherheitsparameters 1^k den geheimen Masterschlüssel s sowie die öffentliche Systemparameter *params* erzeugt.

  $$(s, params) \xleftarrow{R} Setup\left(1^k\right)$$

  Die öffentlichen Systemparameter dienen implizit als Eingabe für alle anderen Algorithmen.

- **Extract** - ist ein (probabilistischer) Algorithmus, der bei Eingabe des Masterschlüssels s und einer Identität ID den entsprechenden geheimen Schlüssel S_{ID} berechnet.

  $$S_{ID} \xleftarrow{R} Extract\,(ID, s)$$

- **Sign/Encrypt** - ist ein probabilistischer Algorithmus, der bei Eingabe des geheimen Schlüssel S_{ID_S}, der Identität des Empfängers ID_R und einer Nachricht m die Signcryption ς berechnet.

$$\varsigma \stackrel{R}{\leftarrow} Sign/Encrypt\,(m, S_{ID_S}, ID_R)$$

- **Decrypt/Verify** - ist ein deterministischer Algorithmus, der bei Eingabe des geheimen Schlüssels S_{ID_R}, der Identität des Senders ID_S sowie der Signcryption ς die entsprechende Nachricht m mit der dazugehörigen Signatur σ berechnen und verifizieren kann. Falls die Verifizierung fehlgeschlagen ist, liefert der Algorithmus das Fehlersymbol $\perp$ zurück.

$$(m, \sigma) := Decrypt/Verify\,(\varsigma, ID_S, S_{ID_R})$$

- **Verify** - ist ein deterministischer Algorithmus, der bei Eingabe einer Identität ID_S des Senders und einer Nachricht m die Gültigkeit der entsprechenden Signatur σ überprüft. Der Algorithmus gibt ein Bit b aus. $b = 1$ bedeutet, dass die Signatur gültig ist und $b = 0$, dass sie ungültig ist.

$$b := Verify\,(m, \sigma, ID)$$

Das Verfahren ist korrekt, wenn für alle für das Verfahren gültige Werte von ID und m gilt:

$$(\widehat{m}, \sigma) := Decrypt/Verify\,(Sign/Encrypt\,(m, S_{ID_S}, ID_R), ID_S, S_{ID_R})$$

mit

$$\widehat{m} = m$$

sowie

$$Verify\,(\widehat{m}, \sigma, ID_S) = 1$$

Dies muss für beliebige von Setup- und Extract-Algorithmen gewählte Parameter und Schlüssel sowie für beliebige von dem Sign/Encrypt-Algorithmus erzeugte Signcryptions gelten.

2.3 Beweisbare Sicherheit

Moderne Kryptographie der letzten 30-40 Jahren ist durch strenge Sicherheitsbeweise gekennzeichnet. Es reicht nicht aus ein Verfahren zu entwickeln, das korrekt und scheinbar schwer zu brechen ist. Viel mehr erwartet man, dass durch einen Sicherheitsbeweis gezeigt wird, dass das Verfahren von keinem Angreifer mit bestimmter Leistung gebrochen werden kann. Warum braucht man aber solche Beweise?

Wenn man versucht ein Verfahren zu brechen und zu keinem Erfolg kommt, bedeutet dies noch nicht, dass ein anderer das nicht schaffen kann. Ein erfolgreicher Angriff auf ein eingesetztes Verfahren kann aber gravierende Folgen haben, wenn dieses z.B. von Banken eingesetzt wird.

Die nächste Frage, die sich stellt, ist, ob man perfekte Sicherheit erreichen kann. Das heißt informell, dass kein Angreifer, auch wenn er unbegrenzte Leistung zur Verfügung hat, das Verfahren brechen kann. Erstaunlicherweise lautet die Antwort auf diese Frage im Falle von symmetrischen Verschlüsselungsverfahren JA und sogar mit sehr einfachen Algorithmen. Allerdings sind diese nicht effizient und kaum praktikabel in den modernen Anwendungen. Die Sicherheitsbeweise für tatsächlich praktikable Verfahren basieren deswegen auf verschiedenen kryptographischen Annahmen. Ein bekanntes Beispiel für solche Annahmen ist das Faktorisierungsproblem, auf dem das RSA-Algorithmus basiert.

Viele Sicherheitsbeweise basieren somit auf Reduktionen, bei denen ein wohl bekanntes schwieriges Problems auf einen Angreifer für das Verfahren reduziert wird. Dabei konstruiert man aus einer Instanz des Problems die Eingabe für den Angreifer und simuliert ihn dann auf solche Art und Weise, dass aus seiner Ausgabe die Lösung für das Problem berechnet werden kann. Man zeigt damit, dass das Problem gelöst werden kann, wenn ein Angreifer existiert, der das Verfahren kontinuierlich brechen kann. Man fordert also, dass die Erfolgswahrscheinlichkeit $\varepsilon(k)$ eines Angreifers vernachlässigbar ist. Das heißt also, dass diese für jede Konstante c schneller als k^{-c} gegen 0 geht. Würde man das nicht für jedes c fordern, sondern nur für $c < n$, würde man einen Angreifer mit Erfolgswahrscheinlichkeit $\frac{1}{k^n}$ zulassen. Wenn man dann diesen Angreifer k^n mal ausführt, bricht man das System im Mittel einmal und dieser neue Angreifer ist immer noch zeitpolynomiell.

Um einen Sicherheitsbeweis führen zu können, müssen erst präzise Definitionen der Sicherheit eingeführt werden. Da man ganz verschiedene Ziele in der Kryptographie verfolgt, sind auch die entsprechenden Definitionen unterschiedlich. Aber auch für die gleichen Ziele können verschiedene Definitionen angegeben werden, die mehr oder weniger streng sind, denn nicht jede Anwendung braucht die strengst mögliche Sicherheit bei der mit Effizienzeinbüßen gerechnet werden muss. Somit muss man in jedem Fall anhand der verfolgten Zielen entscheiden welche Definition die passende ist.

Eine weit verbreitete Form der Sicherheitsdefinitionen ist eine Art Spiel, das in mehreren Phasen abläuft und bei der der Herausforderer und der Angreifer miteinander kommunizieren. Die Sicherheit eines Verfahrens hängt davon ab, mit welcher Wahrscheinlichkeit der Angreifer das Spiel gewinnt. In dieser Arbeit werden genau solche Definitionen benutzt.

Zufallsorakel-Modell

Die strengen Sicherheitsbeweise sind in der modernen Kryptographie der letzten 30-40 Jahren zur Normalität geworden. Die wissenschaftliche Vorgehensweise bei der Entwicklung von kryptographischen Verfahren ist schon längst auch in der Praxis angekommen. An vielen Stellen hat sich aber herausgestellt, dass beweisbare Sicherheit oft mit großen Effizienzverlusten verbunden ist. Auch wenn die Behauptung, dass so entwickelte Verfahren immer ineffizient sind, sicherlich falsch ist, sucht man in einigen Bereichen nach alternativen Wegen. Ein populäres Beispiel für eine solche alternative Methode ist das Zufallsorakel-Modell.[6]

Im Zufallsorakel-Modell unterstellt man die Existenz einer öffentlich zugänglichen Funktion H, deren Wert an der Stelle x des Wertebereiches nur durch eine Anfrage an das Orakel ausgewertet werden kann. Das Orakel kann man sich als eine Art „black box" vorstellen, die bei Eingabe x den Wert $H(x)$ zurück liefert. Der Wert $H(x)$ ist vollkommen unabhängig von allen anderen Werten von H. Somit wird dieser als zufälliges, gleichverteiltes Element aus der Menge aller möglichen Werte angesehen. Weiterhin ist das Orakel konsistent, d.h. dass bei mehrfachen Anfragen des Funktionswertes an einer beliebigen Stelle x das Orakel immer den gleichen Wert liefert.

Die Entwicklung von kryptographischen Verfahren im Zufallsorakel-Modell wird in zwei Schritten durchgeführt:

- Erst entwickelt man ein Verfahren und zeigt dessen Sicherheit unter der Annahme, dass ein Zufallsorakel H existiert.
- Bei der Implementierung in der realen Welt, in der kein Zufallsorakel existiert, ersetzt man dieses durch eine kryptographische Hashfunktion $\widehat{H}$.

Implizit wird dabei angenommen, dass die im zweiten Schritt benutzte Hashfunktion das Zufallsorakel gut genug approximiert, so dass der Sicherheitsbeweis auch für die reale Implementierung gilt. Die Problematik des Zufallsorakel-Modells liegt daran, dass es keinen Beweis dafür gibt, dass diese Annahme gilt. Man kann sogar Verfahren konstruieren, die beweisbar sicher im Zufallsorakel-Modell sind und unabhängig von der im zweiten Schritt benutzten Hashfunktion $\widehat{H}$ unsicher im üblichen Modell sind (solche Konstruktionen haben allerdings wenig mit den realen Verfahren zu tun). Weiterhin ist auch unklar welche Eigenschaften gute Hashfunktionen haben müssen. Man sollte also Sicherheitsbeweise im Zufallsorakel-Modell als eine Art Indiz dafür sehen, dass ein Verfahren keine inhärente Fehler aufweist.

An dieser Stelle soll nicht darüber diskutiert werden, ob die Anwendung des Zufallsorakel-Modells akzeptabel ist. Es gibt sowohl Gründe dafür, als auch dagegen und die Diskussionen darüber kann man in der Literatur nachlesen (z.B. [KL07] Kapitel 13). Man beachte aber, dass dieses Modell eine breite Verwendung in der Kryptographie gefunden hat und dass ein im üblichen Modell beweisbar sicheres und

[6] „Random oracle model" im Englischen

vergleichbar effizientes Verfahren einem, dessen Sicherheit im Zufallsorakel-Modell bewiesen wurde, bevorzugt werden sollte.

Laut [KL07] gibt es auch Beispiele dafür, dass für einige Verfahren, für die zuerst nur Sicherheitsbeweise im Zufallsorakel-Modell bekannt waren, später auch Sicherheitsbeweise im üblichen Modell durchgeführt wurden.

Das allgemeine Forking-Lemma

In diesem Abschnitt wird ein sehr nützliches Lemma, das so genannte Forking-Lemma, ohne Beweis vorgestellt und dessen Anwendung erörtert. Das Lemma wird sowohl bei den Sicherheitsbeweisen für die IBS-Verfahren, als auch für die IBSC-Verfahren eingesetzt. Aber zuerst soll die Idee des Lemmas erklärt werden.

Wenn man bei Sicherheitsbeweisen eine Reduktion eines kryptographischen Problems auf einen Angreifer führt, genügt ein einziger erfolgreicher Angriff in manchen Fällen nicht, um das vorgegebene Problem zu lösen und man braucht zwei solche Angriffe, bei denen die Ausgaben des Angreifers in einer bestimmten Relation zueinander stehen. Die Idee für solche Fälle besteht nun darin, dass man den Angreifer mehrmals simuliert und dabei seine Eingaben und Ausführung so anpasst, dass er mit bestimmter Wahrscheinlichkeit die erforderlichen Ausgaben produziert. Auf dieser Idee basiert das sogenannte allgemeine Forking-Lemma, das von M. Bellare und G.Neven in [BN06] vorgestellt wurde. Das Lemma ist von eigener Interesse, da die Notwendigkeit von zwei voneinander abhängigen Angriffen keine Seltenheit ist. Allein in dieser Arbeit wird das Forking-Lemma in den Beweisen von drei Verfahren angewendet. Aus diesem Grund wird dieses hier weitgehend erörtert.

Nun soll die Struktur des Lemmas erläutert werden. Um die Anwendung plausibel zu machen und die Aussage des Lemmas von den technischen Einzelheiten zu trennen, geben wir erst die Definitionen der einzelnen im Forking-Lemma benutzten Algorithmen und formulieren dann das Lemma selbst.

Sei A der gegebene Angreifer für das Verfahren und B der übergreifende Algorithmus, der mit Hilfe von A eine Lösung für ein kryptographisches Problem berechnet. B benutzt seinerseits mehreren Algorithmen:

- IG - *Input-Generator-Algorithmus* ist ein probabilistischer Algorithmus und wird als erster ausgeführt. IG berechnet die Eingabe für den Angreifer sowie alle für dessen Simulation notwendigen Elemente.

$$X \stackrel{R}{\leftarrow} IG$$

- S_A - *Simulationsalgorithmus* ist ein probabilistischer Algorithmus, der als Eingabe X sowie q weitere Parameter $h_1, ..., h_q$ aus einer Menge H nimmt und den Angreifer A simuliert. Die Werte $h_1, ..., h_q$ werden bei Simulation der Anfragen von A auf eine geeignete Art verwendet.

Als Erfolg von S_A definieren wir den Fall, dass A einen erfolgreichen Angriff durchführt und eine Ausgabe σ, die in Verbindung mit einem Wert h_J steht, ausgibt. In diesem Fall gibt S_A das Paar (J, σ) aus mit $J \in \{1, ...q\}$. Für die Ausführung von S_A schreiben wir:

$$(J, \sigma) \overset{R}{\leftarrow} S_A\left(X, h_1, ..., h_q\right)$$

Bei Misserfolg gibt S_A eine Fehlerkombination $(0, \varepsilon)$ aus. Wir definieren die Erfolgswahrscheinlichkeit acc von S_A als:

$$acc := Pr\left[J \neq 0 \,\middle|\, (J, \sigma) \overset{R}{\leftarrow} S_A\left(X, h_1, ..., h_q\right)\right]$$

- F - *Forking-Algorithmus* nimmt als Eingabe X und simuliert zweimal den Algorithmus S_A wie folgt:
  - wähle Zufallsbits ρ für S_A zufällig, gleichverteilt
  - wähle $h_1, ..., h_q \overset{R}{\leftarrow} H$
  - simuliere S_A das erste Mal:

$$(I, \sigma) := S_A\left(x, h_1, ..., h_q; \rho\right)$$

  - wenn $I = 0$ gilt, dann gebe $\perp$ aus
  - wähle $\widetilde{h_I}, ..., \widetilde{h_q} \overset{R}{\leftarrow} H$
  - simuliere S_A zum zweiten Mal mit der angepassten Eingabe:

$$\left(\widetilde{I}, \widetilde{\sigma}\right) := S_A\left(x, h_1, ..., h_{I-1}, \widetilde{h_I}, ..., \widetilde{h_q}; \rho\right)$$

  - wenn $I = \widetilde{I}$ und $h_I \neq \widetilde{h_I}$ gilt, dann gebe $\left((\sigma, h_I), \left(\widetilde{\sigma}, \widetilde{h_I}\right)\right)$ aus, sonst gebe $\perp$ aus.

Zusammenfassend können wir also für die Ausführung von F schreiben:

$$\left((\sigma, h_I), \left(\widetilde{\sigma}, \widetilde{h_I}\right)\right) \overset{R}{\leftarrow} F\left(X\right)$$

Wir definieren die Erfolgswahrscheinlichkeit frk von F als:

$$frk := Pr\left[Y \neq \perp \,\middle|\, Y \overset{R}{\leftarrow} F\left(X\right)\right]$$

Lemma 10. *[Allgemeines Forking-Lemma] Gegeben sei $q \in \mathbb{N}$ mit $q \geq 1$ und eine Menge H mit $|H| = h \geq 2$. Sei IG - der Input-Generator-Algorithmus, S_A der Simulationsalgorithmus mit der Erfolgswahrscheinlichkeit acc und F der Forking-Algorithmus mit Erfolgswahrscheinlichkeit frk. Dann gilt*

$$frk \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{h} \right)$$

und alternativ

$$acc \leq \frac{q}{h} + \sqrt{q \cdot frk}$$

Man merke, dass das Lemma allgemein formuliert ist und dass keine Annahmen über den Angreifer A gemacht werden. Die Erfolgswahrscheinlichkeit von A wird letztendlich in *acc* enthalten sein.

Somit erhält man folgendes Anwendungsschema für das Forking-Lemma:

- Gebe den IG-Algorithmus an, der aus einer Instanz des Problems P alle für die Simulation des Angreifers benötigten Elemente X berechnet.
- Gebe den Algorithmus S_A an, der bei Eingabe von $h_1, ..., h_q$ und mit Hilfe von X den Angreifer simuliert und seine Ausgabe σ mit einem der Werte h_J in Verbindung setzt.
- Wende das Forking-Lemma an und berechne aus den Paaren $\left((\sigma, h), \left(\widetilde{\sigma}, \widetilde{h} \right) \right)$ mit $h \neq \widetilde{h}$ die Lösung für das vorgegebene kryptographische Problem.

An dieser Stelle muss noch darauf hingewiesen werden, dass in der Originalarbeit bei den Sicherheitsbeweisen für IBS und IBSC ein anderes Forking-Lemma angewandt wurde, dass in [PS00] für Standardsignaturen definiert wurde. Allerdings ist nicht klar, dass das Lemma ohne weitere Anpassungen für IBS und insbesondere für IBSC gilt, da die entsprechenden Angriffe nicht identisch zu dem im Forking-Lemma betrachteten Angriff auf Standard-Signaturen sind. Aus diesem Grund habe ich mich entschieden die Beweise anzupassen und das vorgestellte allgemeine Forking-Lemma zu benutzten. Dieses macht, wie wir schon gesehen haben, keine Annahmen über das Verfahren und über den Angreifer und kann somit für verschiedene Verfahren angewandt werden.

2.4 Sicherheitsdefinitionen für IBS

In diesem Abschnitt wollen wir nun die Sicherheitsdefinitionen für IBS-Verfahren definieren.

Man unterscheidet die Angriffe auf Signaturverfahren einerseits nach dem Ergebnis, das ein Angreifer versucht zu erreichen und andererseits danach welche Mittel er zur Verfügung hat.

Die Angriffe werden je nach Ergebnis wie folgt unterteilt:

- *total break* - der Angreifer ermittelt den geheimen Schlüssel und bricht somit das komplette System
- *universal forgery* - der Angreifer kann jede Nachricht signieren
- *existential forgery* - der Angreifer kann ein gültiges Nachricht-Signatur Paar berechnen.

In dieser Arbeit betrachten wir nur die existentiellen Angriffe. Diese Art von Angriffen ist auch die schwächste, denn aus der Existenz eines der ersten beiden Typen von Angreifern folgt unmittelbar die Existenz des existentiellen Angreifers. Wir wollen also, dass der Angreifer keine gültige Signatur für eine beliebige Nachricht seiner Wahl berechnen kann. Man überlege sich, dass das eine sehr strenge Forderung ist.

Die Angreifer, die wir in dieser Arbeit betrachten, sind zeitpolynomielle Angreifer, die bestimmte Informationen zur Verfügung haben. Bei Standardsignaturen unterscheidet man zwischen folgenden Angreifern:

- *key-only attack* - der Angreifer kennt nur den öffentlichen Schlüssel vom Sender und kann somit den Verifizierungsalgorithmus ausführen.
- *known message attack* - der Angreifer kennt mehrere Nachricht/Signatur-Paare, die mit dem Schlüssel des Senders erzeugt wurden.
- *chosen message attack* - der Angreifer darf eine Liste von Nachrichten angeben, die vom Sender signiert werden. Der Angreifer bekommt den Zugriff auf diese Signaturen.

Der letzte Angreifer hat dabei die meisten Informationen zur Verfügung, denn er darf sogar die Nachrichten bestimmen, für die er eine Signatur bekommt. Wir werden in dieser Arbeit genau solche Angreifer betrachten, denn in der Praxis hat sich herausgestellt, dass gerade diese Angriffe die realen Angriffen am besten simulieren.

Nun wollen wir zwei Angreifer für IBS-Verfahren definieren, die sich daran unterscheiden, ob der Angreifer eine beliebige oder eine vorgegebene Identität angreifen muss. Im Vergleich zu den vorgestellten Angriffen auf Standardsignaturen bekommt der Angreifer noch die Möglichkeit Schlüssel für Identitäten seiner Wahl anzufragen, denn in der Praxis könnte er im Besitz von mehreren Schlüsseln sein.

Formale Definitionen geben wir mit Hilfe von entsprechenden Spielen. Diese modellieren sowohl die Fähigkeiten der Angreifer als auch deren Ziele.

Definition 11. Das *IBS-Sign-forge* $_{A,\Pi}(k)$-Spiel läuft wie folgt ab:

- **Setup:** Der Herausforderer C führt den Setup-Algorithmus aus:

$$(s, params) \xleftarrow{R} Setup\left(1^k\right)$$

- **Phase 1:** Der Angreifer A bekommt als Eingabe die öffentlichen Parameter

params und kann nun adaptiv[7] folgende Anfragen stellen, die von C beantwortet werden:

- *Extract*: Bei Eingabe einer beliebige Identitäten ID berechnet C den entsprechenden geheimen Schlüssel

$$S_{ID} \overset{R}{\leftarrow} Extract(ID, s)$$

- *Sign*: C berechnet bei Eingabe einer beliebigen Identitäten ID und einer beliebigen Nachrichten m die entsprechende Signatur

$$\sigma \overset{R}{\leftarrow} Sign\,(m, S_{ID})$$

- **Guess:** Der Angreifer A produziert ein Tupel (m, σ, ID^*).

Die Ausgabe des Experiments definieren wir als 1 genau dann, wenn der Angreifer keine Extract-Anfrage für ID^* sowie keine Sign-Anfrage für (m, ID^*) gestellt hat und es gilt:

$$Veryfy\,(m, \sigma, ID^*) = 1$$

Der Vorteil des Angreifers definieren wir als seine Erfolgswahrscheinlichkeit:

$$Adv_{A,\Pi}\,(k) = Pr\,[\text{IBS-Sign-forge}_{A,\Pi}\,(k) = 1]$$

Die nächste Definition entspricht der Definition 2 in [BLMQ05].

Definition 12. Ein IBS-Verfahren Π heißt *existentiell fälschungssicher* gegen einen **adaptive-chosen-message-and-ID-**Angreifer (CMIDA), wenn für jeden probabilistischen zeitpolynomiellen Angreifer A eine vernachlässigbare Funktion *negl* existiert, so dass für den Vorteil des Angreifers im IBS-Sign-forge$_{A,\Pi}\,(k)$-Spiel gilt:

$$Adv_{A,\Pi}\,(k) \leq negl\,(k)$$

Das *Given-ID-IBS-Sign-forge* $_{A,\Pi}\,(k)$-Spiel unterscheidet sich von dem *IBS-Sign-forge* $_{A,\Pi}\,(k)$-Spiel nur daran, dass in der Setup-Phase der Herausforderer zufällig gleichverteilt eine Identität ID^* wählt, für die der Angreifer dann eine Fälschung berechnen muss. Die entsprechende Sicherheitsdefinition sieht dann wie folgt aus:

Definition 13. Ein IBS-Verfahren Π = (Setup, Extract, Sign, Verify) heißt *existentiell fälschungssicher* gegen einen **adaptive-chosen-message-and-given-ID-**Angreifer (CMA), wenn für jeden probabilistischen zeitpolynomiellen Angreifer A eine vernachlässigbare Funktion *negl* existiert, so dass für den Vorteil des Angreifers im given-ID-IBS-Sign-forge$_{A,\Pi}\,(k)$-Spiel gilt:

$$Adv_{A,\Pi}\,(k) \leq negl\,(k)$$

[7] Adaptiv bedeutet, dass der Angreifer die Anfragen in Abhängigkeit von schon gestellten Anfragen stellen kann.

2.5 Sicherheitsdefinitionen für IBSC

Bei den Signcryptions wollen wir im Vergleich zu den Signaturverfahren nicht nur die Verbindlichkeit, sondern auch die Vertraulichkeit einer Nachricht erreichen. Deswegen brauchen wir zwei Definitionen, die die Sicherheitskonzepte dafür modellieren. Die Definition zum Erreichen von Verbindlichkeit wird der Definitionen bei IBS-Verfahren ähneln. Die Definition für die Vertraulichkeit muss aber erst genauer erläutert werden.

Eine weit verbreitete und anerkannte Definition der Sicherheit bei den Verschlüsselungen heißt Ununterscheidbarkeit von Chiffretexten[8]. Diese Definition besagt, dass kein Angreifer in der Lage sein soll zwei Verschlüsselungen von zwei verschiedenen Nachrichten zu unterscheiden. Das ist eine sehr strenge Definition, denn der Angreifer muss eine Nachricht nicht komplett entschlüsseln. Wenn er z.B. kontinuierlich einen Bit bestimmen kann, in dem sich die beiden Nachrichten unterscheiden, dann hat er laut dieser Definition das Verfahren gebrochen.

Die Fähigkeiten der zeitpolynomiellen Angreifer, die wir betrachten werden, sind sehr hoch. Diese werden vor allem dadurch definiert, welche Anfragen der Angreifer stellen darf. Wir betrachten adaptive-chosen-ciphertext-Angreifer, der adaptiv fast beliebige Signcryptions seiner Wahl entschlüsseln lassen darf. Bei identitätsbasierten Signcryptionverfahren hat der Angreifer nach wie vor die Möglichkeit die geheimen Schlüssel für Identitäten seiner Wahl nachzufragen sowie für beliebige Nachrichten seiner Wahl die Signcryption berechnen zu lassen.

Die nächste Definition des *IND-IBSC-CCA* $_{A,\Pi}(k)$-Spiels entspricht der in der Definition 3 in [BLMQ05] vorgestellten Spiels. Ein kleiner Unterschied zu dieser Definition liegt daran, dass wir in dem *IND-IBSC-CCA* $_{A,\Pi}(k)$-Spiel nur eine Decrypt/Verify-Anfrage für genau die Werte aus der Challenge-Phase verbieten. In der Originaldefinition darf der Angreifer dagegen keine Decrypt/Verify-Anfragen für die ihm vorgegebene Signcryption und Identität des Empfängers stellen. Die Anpassung entspricht der Intuition, dass die Decrypt/Verify-Anfragen für eine gültige Signcryption mit einer anderen Identität des Senders keinen Vorteil dem Angreifer bringen soll. Andererseits entspricht auch der Beweis in der Originalarbeit der hier angegebenen Definition.

Definition 14. Das *IND-IBSC-CCA* $_{A,\Pi}(k)$-Spiel läuft wie folgt ab:

- **Setup:** Der Herausforderer C führt das Setup-Algorithmus mit Eingabe 1^k aus, um den Masterschlüssel und die öffentliche Parameter zu berechnen:

$$(s, params) \leftarrow Setup\left(1^k\right)$$

- **Phase 1:** Der Angreifer A bekommt als Eingabe die öffentlichen Parameter *params* und kann nun adaptiv folgende Anfragen stellen, die von C beantwortet werden:

[8] „Indistinguishability" im Englischen

– *Extract:* Bei Eingabe einer beliebige Identität ID berechnet C mit Hilfe des Masterschlüssels den entsprechenden geheimen Schlüssel:

$$S_{ID} \overset{R}{\leftarrow} Extract\,(ID, s)$$

– *Sign/Encrypt*: C berechnet bei Eingabe von zwei beliebigen Identitäten ID_S (Identität des Senders) und ID_R (Identität des Empfängers) sowie einer beliebigen Nachrichten m die entsprechende Signcryption:

$$\varsigma \overset{R}{\leftarrow} Sign/Encrypt\,(m, S_{ID_S}, ID_R)$$

– *Decrypt/Verify*: C berechnet bei Eingabe von zwei beliebigen Identitäten ID_S und ID_R sowie einer Signcryption ς die entsprechende entschlüsselte Nachricht m und die Signatur σ. Falls die Gültigkeitsüberprüfung der Signatur für ID_S erfolgreich war, liefert C diese gültige Signatur zusammen mit der Nachricht zurück, sonst wird das Fehlersymbol $\perp$ zurückgeliefert:

$$(m, \sigma) := Decrypt/Verify\,(\varsigma, ID_S, S_{ID_R})$$

- **Challenge:** Der Angreifer A gibt zwei gültige Nachrichten $m_0, m_1 \in M$, sowie zwei Identitäten $\widehat{ID_S}$ und $\widehat{ID_R}$ aus. Dabei darf er vorher nicht den geheimen Schlüssel für $\widehat{ID_R}$ angefragt haben.[9] C wählt zufällig ein Bit $b \overset{R}{\leftarrow} \{0,1\}$ und liefert dem Angreifer die entsprechende Signcryption:

$$\widehat{\varsigma} \overset{R}{\leftarrow} Sign/Encrypt\left(m_b, \widehat{ID_S}, \widehat{ID_R}\right)$$

- **Phase 2:** Nun darf A wieder die gleichen Anfragen stellen, wie in Phase 1, mit der Einschränkung, dass er die Anfragen $Extract\left(\widehat{ID_R}\right)$ und $Decrypt/Verify\left(\widehat{\varsigma}, \widehat{ID_S}, \widehat{ID_R}\right)$ nicht stellen darf.
- **Guess:** A gibt seine Vermutung $b' \in \{0,1\}$ dafür, welche der beidem Nachrichten Verschlüsselt wurde, an und gewinnt wenn gilt:

$$b = b'$$

In diesem Fall definieren wir das Ergebnis des Spiels als 1.

Den Vorteil des Angreifers definieren wir als:

$$Adv_{A,\Pi}\,(k) = \left| Pr\left[\text{IND-IBSC-CCA}_{A,\Pi}\,(k) = 1\right] - \frac{1}{2} \right|$$

[9] Man beachte, dass der Angreifer den geheimen Schlüssel für $\widehat{ID_S}$ anfragen darf.

Definition 15. Ein identitätsbasiertes Signcryptionverfahren Π heißt *IND-IBSC-CCA-sicher* wenn für jeden probabilistischen zeitpolynomiellen Angreifer A eine vernachlässigbare Funktion *negl* existiert, so dass für den Vorteil des Angreifers im *IND-IBSC-CCA* $_{A,\Pi}(k)$-Spiel gilt:

$$Adv_{A,\Pi}(k) \leq negl(k)$$

Nun wollen wir die Definition zum Erreichen der Verbindlichkeit angeben. Dafür definieren wir wieder erst ein Spiel. Die Definition des *IBSC-Sign-forge* $_{A,\Pi}(k)$-Spiels entspricht der in der Definition 4 in [BLMQ05] vorgestellten Spiels. Allerdings mussten an dieser Stelle noch größere Änderungen vorgenommen werden, als bei der vorherigen Definition. Und zwar liefert die in der Originalarbeit angegebene Definition nicht die Verbindlichkeit für das Verfahren. Es muss nämlich gefordert werden, dass der Angreifer nicht nur keine gültige Signcryption sondern auch keine gültige Signatur berechnen kann, denn das eine impliziert nicht das andere.

Durch diese Anpassung wird auch sofort deutlich, warum der Angreifer in diesem Spiel keine Sign/Encrypt-Anfragen für die Nachricht und die Identität des Senders, für die er später eine gültige Signcryption oder Signatur berechnet, stellen darf. Aus der so berechneten Signcryption könnte der Angreifer nämlich eine gültige Signatur berechnen, wenn er den Schlüssel für die Identität des Empfängers anfragt.

Definition 16. Das *IBSC-Sign-forge* $_{A,\Pi}(k)$-Spiel läuft wie folgt ab:

- **Setup:** Der Herausforderer C führt Setup-Algorithmus mit Eingabe 1^k aus, um den geheimen Schlüssel und die öffentliche Parameter zu berechnen:

$$(s, params) \leftarrow Setup\left(1^k\right)$$

- **Phase 1:** Der Angreifer A bekommt als Eingabe die öffentlichen Parameter *params* und kann nun adaptiv folgende Anfragen stellen, die von C beantwortet werden:

  - *Extract*: Bei Eingabe einer beliebige Identität ID berechnet C den entsprechenden geheimen Schlüssel:

$$S_{ID} \overset{R}{\leftarrow} Extract(ID, s)$$

  - *Sign/Encrypt*: C berechnet bei Eingabe von zwei beliebigen Identitäten ID_S (Identität des Senders) und ID_R (Identität des Empfängers) sowie einer beliebigen Nachrichten m die entsprechende Signcryption:

$$\varsigma \overset{R}{\leftarrow} Sign/Encrypt(m, S_{ID_S}, ID_R)$$

– *Decrypt/Verify*: C berechnet bei Eingabe von zwei beliebigen Identitäten ID_S (Identität des Senders) und ID_R (Identität des Empfängers) sowie einer Signcryption ς die entsprechende Nachricht m und die Signatur σ. Falls die Gültigkeitsüberprüfung der Signatur für ID_S erfolgreich war, liefert C diese gültige Signatur zusammen mit der Nachricht zurück, sonst wird das Fehlersymbol $\perp$ zurückgeliefert:

$$(m, \sigma) = Decrypt/Verify\,(\varsigma, ID_S, S_{ID_R})$$

- **Guess:** Der Angreifer A gibt eine Signcryption $\left(\widehat{\varsigma}, \widehat{ID_S}, \widehat{ID_R}\right)$ oder eine Signatur $\left(m, \sigma, \widehat{ID_S}\right)$ aus und gewinnt das Spiel, wenn im Falle einer vorgegebenen Signcryption gilt:

$$\left(m', \sigma'\right) := Decrypt/Verify\left(\widehat{\varsigma}, \widehat{ID_S}, \widehat{ID_R}\right)$$

und

$$Verify\left(m', \sigma', \widehat{ID_S}\right) = 1$$

bzw. im Falle einer vorgegebenen Signatur gilt:

$$Verify\left(m, \sigma, \widehat{ID_S}\right) = 1$$

Dabei darf der Angreifer die Extract-Anfrage für $\widehat{ID_S}$ sowie die Sign/Encrypt-Anfragen von $\left(m', \widehat{ID_S}, ID_R\right)$ bzw. von $\left(m, \widehat{ID_S}, ID_R\right)$ für eine beliebige Identität ID_R nicht stellen. Das Ergebnis des Spiels definieren wir im Falle des Erfolges von A als 1.

Den Vorteil des Angreifers definieren wir als:

$$Adv_{A,\Pi}\,(k) = Pr\,[\text{IBSC-Sign-forge}_{A,\Pi}\,(k) = 1]$$

Definition 17. Ein identitätsbasiertes Signcryptionverfahren heißt *existentiell fälschungssicher* gegen einen **adaptive-chosen-message-and-ID-**Angreifer (CMI-DA), wenn für jeden probabilistischen zeitpolynomiellen Angreifer A eine vernachlässigbare Funktion *negl* existiert, so dass für den Vorteil des Angreifers im *IBSC-Sign-forge* $_{A,\Pi}\,(k)$-Spiel gilt:

$$Adv_{A,\Pi}\,(k) \leq negl\,(k)$$

Das *Given-ID-IBSC-Sign-forge* $_{A,\Pi}\,(k)$-Spiel unterscheidet sich vom *IBSC-Sign-forge* $_{A,\Pi}\,(k)$-Spiel nur daran, dass in der Setup-Phase der Herausforderer zufällig gleichverteilt eine Identität des Senders ID_S^* wählt, für die der Angreifer dann eine Signcryption oder eine Signatur berechnen muss. Die entsprechende Sicherheitsdefinition sieht dann wie folgt aus:

Definition 18. Ein IBS-Verfahren Π = (Setup, Extract, Sign, Verify) heißt *existentiell fälschungssicher* gegen einen **adaptive-chosen-message-and-given-ID**-Angreifer (CMA), wenn für jeden probabilistischen zeitpolynomiellen Angreifer A eine vernachlässigbare Funktion *negl* existiert, so dass für den Vorteil des Angreifers im *Given-ID-IBSC-Sign-forge* $_{A,\Pi}(k)$-Spiel gilt:

$$Adv_{A,\Pi}(k) \leq negl(k)$$

3 Identitätsbasierte Signaturen (IBS)

In diesem Kapitel werden die beiden Signaturverfahren präsentiert und deren Sicherheit geprüft. In Abschnitt 3.1 werden die beiden IBS-Verfahren formal definiert und deren Korrektheit bewiesen. In Abschnitt 3.2 werden dann die Sicherheitsbeweise für die IBS-Verfahren analysiert. Die Beweise werden für beide Verfahren gleichzeitig geführt, da zwischen diesen nur kleine Unterschiede gibt.

3.1 Die IBS-Verfahren

In der Originalarbeit wurden zwei IBS-Verfahren vorgeschlagen. Das eine, auf dem das Signcryptionverfahren basiert, wurde in Abschnitt 3 samt dem Sicherheitsbeweis vorgestellt (im Folgenden BLMQ-IBS) und das zweite, das so genannte Kurosawa-Heng-IBS-Verfahren (im Folgenden KH-IBS), wurde in Anhang E beschrieben. Die beiden Verfahren unterscheiden sich nur im Sign-Algorithmus. Die anderen drei Algorithmen sind identisch. In dieser Arbeit wird die Sicherheit des zweiten Verfahrens auch bewiesen. Da die beiden Verfahren aber sehr ähnlich sind und die Beweise sich kaum unterscheiden würden, ist es sinnvoll sie zusammen zu führen und an den entsprechenden Stellen auf die Unterschiede einzugehen.

Im Vergleich zur Definition in der Originalarbeit wurden einige Erweiterungen vorgenommen, die die Arbeitsweise des Verfahrens verdeutlichen sollen. Z.B. wird ein „öffentlicher Schlüssel" definiert, der als solches in identitätsbasierten Verfahren nicht existiert, aber die gleiche Funktionsweise hat (dieser wurde an vielen Stellen in der Originalarbeit als Q_{ID} bezeichnet und verwendet). Dieser kann direkt aus der Identität jedes Benutzers berechnet werden. Das Verfahren selbst wurde nicht geändert.

Definition 19. Das BLMQ-IBS-Verfahren $\Pi = (\text{Setup}, \text{Extract}, \text{Sign}, \text{Verify})$ ist definiert als:

- **Setup:** Gegeben sei der Sicherheitsparameter 1^k. Die vertrauenswürdige Stelle wählt bilineare Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ von Primordnung $p > 2^k$ mit der dazugehörigen bilinearen Funktion $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ und dem dazugehörigen Isomorphismus $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. In der zweiten Gruppe wird ein Generator gewählt und die anderen beiden Generatoren werden aus diesem berechnet:
$$Q \xleftarrow{R} \mathbb{G}_2;\ P := \psi(Q) \in \mathbb{G}_1;\ G := e(P, Q) \in \mathbb{G}_T$$

Der Masterschlüssel wird zufällig, gleichverteilt gewählt:

$$s \xleftarrow{R} \mathbb{Z}_p^*$$

Der systemweite öffentliche Schlüssel wird aus dem Masterschlüssel berechnet:

$$Q_{pub} := Q^s \in \mathbb{G}_2$$

Weiterhin werden zwei Hashfunktionen $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ und $H_2 : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_p^*$ zur Berechnung der Hashwerte der Identitäten bzw. der Nachrichten benutzt. Die öffentlichen Parameter sind also:

$$params := \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, G, e, \psi, Q_{pub}, H_1, H_2\}$$

- **Keygen:** Für eine gegebene Identität ID berechne den geheimen Schlüssel als:
$$S_{ID} := P^{\frac{1}{H_1(ID)+s}} \in \mathbb{G}_1$$
An dieser Stelle soll darauf hingewiesen werden, dass für eine $\widetilde{ID}$ mit $H_1\left(\widetilde{ID}\right) = -s$ kein geheimer Schlüssel $S_{\widetilde{ID}}$ berechnet werden kann. Das kann mit einer vernachlässigbaren Wahrscheinlichkeit $\varepsilon = \frac{1}{p-1} < \frac{1}{2^k-1}$ auftreten und würde den kompletten Bruch des Systems bedeuten. Somit betrachten wir diesen Fall von vornherein nicht. Unter anderem bedeutet das, dass wir $H_1(ID) + s$ als ein gleichverteiltes Element aus $\mathbb{Z}_p^*$ betrachten.
Der dazugehörige öffentliche Schlüssel kann folgendermaßen aus ID berechnet werden:
$$P_{ID} := Q^{H_1(ID)} \cdot Q_{pub} = Q^{H_1(ID)+s} \in \mathbb{G}_2$$
Man beachte, dass somit aus der Bilinearität von e direkt folgt:
$$e(S_{ID}, P_{ID}) = e(P, Q)^{\frac{H_1(ID)+s}{H_1(ID)+s}} = G \tag{3.1}$$

- **Sign:** Um eine Nachricht $m \in \{0,1\}^*$ zu signieren:
  1. Wähle $x \xleftarrow{R} \mathbb{Z}_p^*$ und berechne $R := G^x \in \mathbb{G}_T$.
  2. Setze $h := H_2(m, R) \in \mathbb{Z}_p^*$.
  3. Berechne $S := {S_{ID}}^{x+h} \in \mathbb{G}_1$.
  4. Die Signatur für m lautet:
$$\sigma := (h, S) \in \mathbb{Z}_p^* \times \mathbb{G}_1$$

- **Verify**: Die Signatur $\sigma = (h, S)$ für eine Nachricht m wird akzeptiert, falls gilt:
$$h \stackrel{?}{=} H_2\left(m, e(S, P_{ID}) \cdot G^{-h}\right)$$

Beweis. [Korrektheit] Da $h = H_2(m, R)$ reicht es zu zeigen, dass gilt:

$$R = e(S, P_{ID}) \cdot G^{-h}$$

Forme den rechten Teil um:

$$e(S, P_{ID}) \cdot G^{-h} = e\left({S_{ID}}^{x+h}, P_{ID}\right) \cdot G^{-h} \overset{3.1}{=} G^{x+h-h} = R$$

Somit ist das Verfahren korrekt. □

Definition 20. Das KH-IBS-Verfahren $\Pi = (\text{Setup}, \text{Extract}, \text{Sign}, \text{Verify})$ ist definiert als:

- Setup - wie in der BLMQ-IBS
- Extract - wie in der BLMQ-IBS
- **Sign:** Um eine Nachricht $m \in \{0,1\}^*$ zu signieren:
  1. Wähle $x \overset{R}{\leftarrow} \mathbb{Z}_p^*$ und berechne $R := e(P, P_{ID})^x \in \mathbb{G}_T$.
  2. Setze $h := H_2(m, R) \in \mathbb{Z}_p^*$.
  3. Berechne $S := P^x \cdot {S_{ID}}^h \in \mathbb{G}_1$.
  4. Die Signatur für m lautet:

$$\sigma := (h, S) \in \mathbb{Z}_p^* \times \mathbb{G}_1$$

- **Verify** - wie in der BLMQ-IBS

Beweis. [Korrektheit] Da $h = H_2(m, R)$ reicht es zu zeigen, dass gilt:

$$R = e(S, P_{ID}) \cdot G^{-h}$$

Forme den rechten Teil um:

$$e(S, P_{ID}) \cdot G^{-h} = e\left(P^x \cdot {S_{ID}}^h, P_{ID}\right) \cdot G^{-h} =$$

$$e(P, P_{ID})^x \cdot e\left({S_{ID}}^h, P_{ID}\right) \cdot G^{-h} \overset{3.1}{=} e(P, P_{ID})^x \cdot G^{h-h} = R$$

Somit ist das Verfahren korrekt. □

Man merke, dass das KH-IBS-Verfahren nicht so effizient ist, wie das BLMQ-IBS-Verfahren. Der Unterschied zwischen den beiden Verfahren liegt in dem 1. und dem 3. Schritt des Sign-Algorithmus. Im BLMQ-IBS-Verfahren wird im ersten Schritt eine Potenz von G berechnet, während im KH-IBS-Verfahren eine Potenz von $e(P, P_{ID})$. Dieses Element kann aber der Signierer vorher berechnen und somit die Berechnung der Paarung an dieser Stelle sparen. Somit ist der Aufwand für den ersten Schritt äquivalent zum BLMQ-IBS-Verfahren. Im dritten Schritt wird aber im BLMQ-IBS-Verfahren nur eine Potenz in der Gruppe $\mathbb{G}_1$ berechnet, während im KH-IBS-Verfahren zwei Potenzen sowie eine Multiplikation in $\mathbb{G}_1$ benötigt werden.

3.2 Die Sicherheitsanalyse

Der Sicherheitsbeweis für die IBS-Verfahren wurde an einigen Stellen im Vergleich zur Originalarbeit geändert und ergänzt. Die Idee des Beweises ist aber grundsätzlich beibehalten worden. Man führt eine Reduktion des q-SDH-Problems auf den **adaptive-chosen-message-and-ID-Angreifer (CMIDA)** für die IBS:

$$\text{q-SDH} \leq \text{CMIDA}$$

Diese Reduktion wird in zwei Schritten durchgeführt. Im ersten Schritt reduzieren wir den CMIDA Angreifer auf den **adaptive-chosen-message-Angreifer (CMA)** (Lemma 21):

$$\text{CMA} \leq \text{CMIDA}$$

Der Beweis dafür wurde in [CC03] vorgestellt und wird an dieser Stelle angegeben, da dieser auf einer Idee basiert, die auch für andere Beweise von Interesse sein wird.

Im zweiten Schritt (Lemma 22) zeigen wir dann die Reduktion:

$$\text{q-SDH} \leq \text{CMA}$$

Die größte Änderung an diesem Beweis liegt daran, dass das allgemeine Forking-Lemma angewendet wurde. Weiterhin wurde die Simulation etwas angepasst.

Lemma 21. *[CMA $\leq$ CMIDA] Wenn es ein CMIDA Angreifer F_0 auf IBS mit Laufzeit t_0 und Erfolgswahrscheinlichkeit ε_0 existiert, der q_{H_1} Anfragen an die Zufallsorakel H_1 stellt, dann existiert ein CMA Angreifer F_1 auf IBS, der in Laufzeit $t_1 \leq c \cdot t_0$ (wobei c eine geeignete Konstante ist) und mit Erfolgswahrscheinlichkeit $\varepsilon_1 \geq \varepsilon_0/q_{H_1}$ eine gültige Fälschung für eine gegebene Identität ID^* berechnet. Weiterhin macht der Angreifer F_1 die gleiche Anzahl aller Anfragen, wie F_0.*

Die Idee der Reduktion ist die folgende. Sei ID_{out} die Identität, für die der Angreifer F_0 die Fälschung berechnet. Ohne Beschränkung der Allgemeinheit (weiter nur O.B.d.A) gehen wir davon aus, dass der Angreifer F_0 den Hashwert $H_1(ID_{out})$ nachfragt (siehe die Begründung weiter hinten). Also rät F_1 die Stelle $d \in \{1, ..., q_{H_1}\}$, an der F_0 diesen Hashwert anfragt. Alle Anfragen, die ID_d betreffen, ersetzt F_1 durch entsprechende Anfragen mit seiner eigenen Identität ID^*, die er angreifen muss. Im Zufallsorakel-Modell kann F_0 diese Anpassung nicht erkennen, da alle Hashwerte der Identitäten unabhängig voneinander sind. Falls also F_1 die richtige Stelle erraten hat, also $ID_{out} = ID_d$ gilt, dann gibt F_0 tatsächlich eine gültige Signatur für ID^* mit seiner Erfolgswahrscheinlichkeit ε_0 aus, während er vermutet, dass er eine für ID_d berechnet hat.

Beweis. Der formale Beweis sieht wie folgt aus:

Zusatzannahmen o.B.d.A.

1. F_0 macht für jede ID maximal eine Anfrage des Hashwertes $H_1(ID)$. Das ist eine rein technische Annahme, die die Analyse erleichtert. Dem Angreifer bringen zwei gleiche Anfragen keinen Vorteil. Aus jedem Angreifer F_0, der mehrmals gleiche Anfragen stellt kann man leicht einen Angreifer $\hat{F}_0$ konstruieren, der die entsprechenden Werte speichert und dann weiter verwendet.
2. Vor jeder Extract- bzw. Sign-Anfrage erfolgt eine Anfrage des Hashwertes H_1 mit der entsprechenden Identität. Auch das ist eine rein technische Annahme, die die Beschreibung der Simulation erleichtert. Auch hier lässt sich ein Angreifer $\hat{F}_0$ angeben, der sich entsprechend der Annahme verhält und sonst die gleichen Berechnungen macht, wie F_0.
3. Der Angreifer fragt den Hashwert der Identität ID_{out}, die er angreift, irgendwann nach. Davon können wir ausgehen, da man aus jedem Angreifer einen anderen konstruieren kann, der das gleiche tut, wie der ursprüngliche und im letzten Schritt den Hashwert für ID_{out} anfragt.

Konstruktion

Der Algorithmus F_1 bei Eingabe ID^* darf seinerseits Anfragen an die Orakel $\widetilde{H_1}, \widetilde{H_2}, \widetilde{Extract}$ und $\widetilde{Sign}$ stellen. F_1 arbeitet wie folgt:

1. Wähle zufällig, gleichverteilt $d \xleftarrow{R} \{1, ..., q_{H_1}\}$. Bezeichne als ID_i die Identität, die bei der i-ten Anfrage von F_0 an das Zufallsorakel H_1 als Eingabe dient (wir vermuten also, dass $ID_{out} = ID_d$ gelten wird).
2. Simuliere F_0 und beantworte seine Anfragen wie folgt (Bestimmung von ID_i ist bei allen Anfragen möglich auf Grund von der Zusatzannahme 2):

   a) $H_1(ID_i) := \begin{cases} \widetilde{H_1}(ID_i) & \text{falls } i \neq d \\ \widetilde{H_1}(ID^*) & \text{falls } i = d \end{cases}$

   b) $H_2(m, R) := \widetilde{H_2}(m, R)$

   c) $Extract(ID_i) := \begin{cases} \widetilde{Extract}(ID_i) & \text{falls } i \neq d \\ \widetilde{Extract}(ID^*) & \text{falls } i = d \end{cases}$

   d) $Sign(ID_i, m) := \begin{cases} \widetilde{Sign}(ID_i, m) & \text{falls } i \neq d \\ \widetilde{Sign}(ID^*, m) & \text{falls } i = d \end{cases}$

Sei $(ID_{out}, m_{out}, \sigma_{out})$ die Ausgabe von F_0. Gebe $(ID^*, m_{out}, \sigma_{out})$ aus, falls $ID_{out} = ID_d$ und sonst gebe $\perp$ aus.

Korrektheit

Definiere folgende Ereignisse:

- E_1 - F_0 berechnet eine gültige Signatur.
- E_2 - das Ereignis, dass $ID_{out} = ID_d$.

Erst muss darauf hingewiesen werden, dass die H_1-Anfragen konsistent auf Grund von der Zusatzannahme 1 sind.

Falls F_1 bei seiner Ausführung die Anfragen $\widetilde{Extract}(ID^*)$ und $\widetilde{Sign}(ID^*, m_{out})$ macht, verliert er nach Definition das Spiel. Ebenso verliert F_0, wenn er die Anfragen $Extract(ID_{out})$ oder $Sign(ID_{out}, m_{out})$ stellt. Es ist leicht zu erkennen, dass wenn F_1 eine Fälschung $(ID^*, m_{out}, \sigma_{out})$ ausgibt (also die Stelle d richtig rät) und dabei die verbotenen Anfragen gestellt hat, dann hat auch F_0 die entsprechenden verbotenen Anfragen gestellt. Somit tritt dieser Fall nicht auf, wenn F_0 erfolgreich bei seinem Angriff ist. Insgesamt gilt, da F_1 die Anfragen mit ID_d durch ID^* ersetzt, dass falls F_0 eine Fälschung für ID_d berechnet hat, berechnet er tatsächlich diese für ID^*. Somit ist die Erfolgswahrscheinlichkeit für F_1:

$$\varepsilon_1 = Pr(E_1 \wedge E_3)$$

Analyse der Erfolgswahrscheinlichkeit

Da H_1 ein Zufallsorakel ist, hängen die Hashwerte nicht voneinander ab und der Angreifer F_0 sieht trotz der Modifizierung von F_1 die gleichen Wahrscheinlichkeitsverteilungen. Wir ignorieren den Fall, dass eine Identität $ID_j = ID^*$ mit $j \neq d$ existiert. Die Erfolgswahrscheinlichkeit von F_0 bleibt also unverändert:

$$Pr(E_1) \geq \varepsilon_0$$

An dieser Stelle gehen wir davon aus, dass eine Identität nur über eine Hashfunktion in alle Berechnungen eingeht, was für alle in der Arbeit betrachteten Verfahren der Fall ist.

F_1 rät nach Konstruktion richtig die Stelle, an der der Angreifer den Hashwert für die Identität anfragt, die er angreifen will, mit Wahrscheinlichkeit $1/q_{H_1}$.[1] Die Wahl von d ist dabei unabhängig vom Verhalten von F_0. Also gilt:

$$Pr(E_2|E_1) = \frac{1}{q_{H_1}}$$

Nun berechne die Erfolgswahrscheinlichkeit von F_1:

$$\varepsilon_1 = Pr(E_1 \wedge E_3) = Pr(E_2|E_1) \cdot Pr(E_1) \geq \frac{\varepsilon_0}{q_{H_1}}$$

[1] Hier verweisen wir auf die Zusatzannahme 1

Laufzeit

F_1 muss nur einen konstanten Zusatzaufwand betreiben. Die Laufzeit kann sich also nur um einen konstanten Faktor gegenüber der Laufzeit von F_0 verschlechtern. Insgesamt folgt also die Behauptung des Lemmas.

□

Der erste Reduktionsschritt ist für beide IBS-Verfahren identisch, da man an dieser Stelle keine besondere Annahmen über den Sign-Algorithmus gemacht hat.

Jetzt können wir den einfacheren adaptive-chosen-message-Angriff betrachten und reduzieren das q-SDH-Problem auf diesen Angriff.

Lemma 22. *[q-SDH≤CMA] Sei A ein CMA Angreifer auf einen der vorgestellten IBS-Verfahren, der q_{H_i} Anfragen an die Zufallsorakel H_i ($i = 1, 2$), q_E Extract-Anfragen sowie q_S Sign-Anfragen stellt. Angenommen A produziert innerhalb der Zeit t_1 eine Fälschung mit Erfolgswahrscheinlichkeit ε_1. Dann existiert ein Algorithmus B, der das q-SDH-Problem (für $q = q_{H_1} + q_E + q_S$) in Zeit $t_2 \leq 2 \cdot t_1 + \mathcal{O}(q_S \cdot \tau_p) + \mathcal{O}\left(q^2 \cdot \tau_{exp_{\mathbb{G}_2}}\right)$ und mit Erfolgswahrscheinlichkeit $\varepsilon_2 \geq \frac{\varepsilon_1^2}{q_{H_2}+q_S} - \frac{q_S \cdot \varepsilon_1^2 + \varepsilon_1}{2^{k-1}}$ löst. Dabei bezeichnet $\tau_{exp_{\mathbb{G}_2}}$ die Kosten für die Berechnung der Potenzen in $\mathbb{G}_2$ und τ_p die Kosten für die Berechnung der Paarung.*

Beweis. Gegeben sind bilineare Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Der Algorithmus B nimmt als Eingabe ein $(q+2)$-Tupel $\left(P, Q, Q^\alpha, Q^{\alpha^2}, ..., Q^{\alpha^q}\right)$ für ein $\alpha \in \mathbb{Z}_p^*$ und versucht ein Paar $\left(c, P^{\frac{1}{c+\alpha}}\right)$ für ein beliebiges $c \in \mathbb{Z}_p^*$ zu finden.

Der Beweis basiert auf dem allgemeinen Forking-Lemma und wird in mehreren Schritten durchgeführt:

- Der unbekannte Parameter α wird dabei implizit zum Masterschlüssel von IBS.
- Erst geben wir den Algorithmus IG an und zeigen wie aus der Instanz des q-SDH-Problems, ohne den Exponenten α zu kennen, die öffentlichen Parameter *params* von IBS berechnet werden können. Weiterhin berechnet man q Paare von der Form $\left(w, \widehat{P}^{\frac{1}{w+\alpha}}\right)$ (mit einem Generator $\widehat{P} \in \mathbb{G}_1$), die für die Beantwortung von *Extract*- und *Sign*-Anfragen benötigt werden. Diese werden als Hashwert einer Identität bzw. der geheime Schlüssel für diese Identität ausgegeben.
- Im zweiten Schritt geben wir den Simulationsalgorithmus S_A an und zeigen, wie der Angreifer simuliert werden kann. Dafür müssen wir seine Signaturanfragen für die vorgegebene Identität simulieren ohne den geheimen Schlüssel zu kennen. Für alle anderen Identitäten benutzen wir einfach die berechneten

Paare $\left(w, \widehat{P}^{\frac{1}{w+\alpha}}\right)$. Außerdem müssen wir zeigen, dass die Verteilungen über den Tripel (R, h, S) in IBS und von der Simulation identisch sind.

- Der nächste Schritt ist die Anwendung des allgemeinen Forking-Lemmas. Dadurch erhalten wir zwei Tupel (m, R, h_1, S_1) und (m, R, h_2, S_2) mit $h_1 \neq h_2$ und können somit das Element $\widehat{P}^{\frac{1}{w_0+\alpha}} = \left(S_1 \cdot S_2^{-1}\right)^{\frac{1}{h_1-h_2}}$ und daraus das Element $P^{\frac{1}{w_0+\alpha}}$ berechnen, wobei wir das Element w_0 kennen. Somit lösen wir also das q-SDH-Problem.
- Am Ende zeigen wir wie die Laufzeit und die Erfolgswahrscheinlichkeit zustande kommen.

Zusatzannahmen o.B.d.A.

1. Sei ID^* die vorgegebene Identität. Der Angreifer fragt den Wert $H_1(ID^*)$. In diesem Fall können wir diese Annahme machen, da ID^* vorgegeben wird und somit man aus jedem Angreifer A einen Angreifer $\widetilde{A}$ konstruieren kann, der diesen Wert sofort anfragt.
2. Für jede Identität, für die der Angreifer eine Anfrage stellt, macht er erst die H_1-Anfrage. Alle diese IDs sind weiterhin unterschiedlich und somit gibt es höchstens $q = q_{H_1} + q_E + q_S$ solche H_1-Anfragen.

Input Generator Algorithmus IG:

Algorithmus IG bei Eingabe $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T), \left(P, Q, Q^{\alpha}, Q^{\alpha^2}, ..., Q^{\alpha^q}\right)$:

1. Wähle eine zufällige Identität ID^*. Für jede Identität wähle einen zufälligen Wert, der als Hashwert für diese Identität ausgegeben wird. Wähle also

$$w_0, w_1, w_2, ..., w_{q-1} \xleftarrow{R} \mathbb{Z}_p^*$$

und multipliziere $f(z) := \prod_{i=1}^{q-1}(z + w_i)$ aus, um die Koeffizienten $c_0, ..., c_{q-1} \in \mathbb{Z}_p^*$, die in weiteren Schritten benutzt werden zu berechnen:

$$f(z) := (z + w_1) \cdot (z + w_2) \cdot ... \cdot (z + w_{q-1}) = \sum_{i=0}^{q-1} c_i \cdot z^i$$

2. Bestimme nun die öffentlichen Parameter von IBS. Berechne die Generatoren $\widehat{Q} \in \mathbb{G}_2$, $\widehat{P} \in \mathbb{G}_1$, $\widehat{G} \in \mathbb{G}_T$ als

$$\widehat{Q} := \prod_{i=0}^{q-1} \left(Q^{\alpha^i}\right)^{c_i} = Q^{\sum_{i=0}^{q-1}(c_i \cdot \alpha^i)} = Q^{f(\alpha)}$$

$$\widehat{P} := \psi\left(\widehat{Q}\right) = P^{f(\alpha)}$$

$$\widehat{G} := e\left(\widehat{P}, \widehat{Q}\right)$$

Den öffentlichen Schlüssel $\widehat{Q_{pub}} \in \mathbb{G}_2$ berechne als

$$\widehat{Q_{pub}} := \prod_{i=0}^{q-1} \left(Q^{\alpha^{i+1}}\right)^{c_i} = Q^{\sum_{i=0}^{q-1}(c_i \cdot \alpha^{i+1})} = \widehat{Q}^{\alpha}$$

Somit kann $\widehat{Q_{pub}}$ berechnet werden, obwohl α nicht bekannt ist.

3. Für alle i, $1 \leq i \leq q-1$, bestimme die Koeffizienten $d_{i0}, ..., d_{i(q-2)} \in \mathbb{Z}_p^*$ durch Polynomdivision:

$$f_i(z) := \frac{f(z)}{(z + w_i)} = \sum_{j=0}^{q-2} d_{ij} z^i$$

Berechne mit Hilfe von diesen Koeffizienten weiterhin für alle w_i außer w_0 die entsprechenden geheimen Schlüssel S_i und für w_0 den öffentlichen Schlüssel P_0:

$$S_i := \psi\left[\prod_{j=0}^{q-2} \left(Q^{\alpha^j}\right)^{d_{ij}}\right] = \psi\left[Q^{\sum_{j=0}^{q-2} d_{ij} \cdot \alpha^j}\right] = \psi\left[Q^{f_i(\alpha)}\right] = P^{\frac{f(\alpha)}{\alpha + w_i}} = \widehat{P}^{\frac{1}{\alpha + w_i}}$$

$$P_0 := \widehat{Q_{pub}} \cdot \widehat{Q}^{w_0} = \widehat{Q}^{\alpha + w_0}$$

(Mit einer vernachlässigbaren Wahrscheinlichkeit tritt der Fall auf, dass für einen Wert w_i gilt: $w_i = -\alpha$, was dazu führen wird, dass diese Berechnung nicht funktioniert. In diesem Fall können wir das q-SDH-Problem mit Hilfe von α direkt lösen. Also ignorieren wir diesen Fall, da er mit einer vernachlässigbaren Wahrscheinlichkeit von $(q-1)/2^k$ auftreten kann)

Die Ausgabe von IG ist nun

$$X := (ID^*, params, (w_0, P_0), (w_1, S_1), ..., (w_{q-1}, S_{q-1}))$$

mit

$$params = \left(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \widehat{P}, \widehat{Q}, \widehat{G}, \psi, e, \widehat{Q_{pub}}\right)$$

wobei e und ψ durch die im q-SDH-Problem vorgegebenen Abbildungsgruppen gegeben sind.

Simulationsalgorithmus S_A:

S_A bei Eingabe $\left(X, h_1, ...h_{q_{H_2}+q_S}\right)$ mit $h_i \in \mathbb{Z}_p^*$:

S_A initialisiert zwei Zähler $ctr_1 = 1$ sowie $ctr_2 = 1$ und startet A mit der Eingabe der Identität ID^* und der öffentlichen Parameter $params$.

Phase 1: Die Anfragen von A können nun folgendermaßen simuliert werden:

- $H_1(ID)$: Falls $ID = ID^*$ beantworte die Anfrage mit w_0, sonst beantworte die Anfrage mit w_{ctr_1}. Speichere das Tupel (ID, S_{ctr_1}) in der Liste L_1 und inkrementiere ctr_1.
- $H_2(m, R)$: Suche in der Liste L_2 nach einem Wert (m, R, h, I) und gebe h zurück, falls dieser gefunden wurde. Sonst beantworte die Anfrage mit h_{ctr_2} und speichere in der Liste L_2 die Werte (m, R, h_{ctr_2}, ctr_2). Inkrementiere ctr_2.
- $Extract(ID)$: Suche in der Liste L_1 nach dem Eintrag für ID und beantworte die Anfrage mit dem entsprechenden Wert S_i. Bei der Anfrage $Extract(ID^*)$ gebe $(0, \varepsilon)$ aus, da in diesem Fall der Angreifer direkt verloren hat. Weiterhin finden wir nach der Zusatzannahme 2 immer einen Eintrag in der Liste.
- $Sign(m, ID)$:
  - für $ID \neq ID^*$ suche in der Liste L_1 nach dem Eintrag für ID und berechne mit Hilfe des entsprechenden geheimen Schlüssels S_i die Signatur wie im Verfahren definiert. Um während dieser Berechnung den Wert $h = H_2(m, R)$ zu bestimmen, stelle eine Anfrage an den H_2-Orakel.
  - für $ID = ID^*$ kennen wir nur den öffentlichen Schlüssel P_0 und müssen eine Simulation angeben. Setze $h = h_{ctr_2}$, wähle:

    $$S \xleftarrow{R} \mathbb{G}_1$$

    und berechne

    $$R := e(S, P_0) \cdot \widehat{G}^{-h}$$

    Falls $R = 1$ gilt, wähle S neu (die Wahrscheinlichkeit dafür ist vernachlässigbar $1/2^k$ und dieser Fall wird weiter nicht betrachtet). Gebe die Signatur $\sigma = (h, S)$ zurück. Speichere in der Liste L_2 die Werte $(m, R, h, -1)$. Inkrementiere ctr_2.
    Durch diese Anfrage definieren wir den Wert $H_2(m, R)$ als h. Dadurch kann es zu Kollisionen kommen, falls dieser Wert bereits definiert wurde. S_A bricht bei diesem Ereignis ab und gibt $(0, \varepsilon)$ aus.

Guess: Sei also $(m_{out}, h_{out}, S_{out})$ die Fälschung von A:

- Suche in der Liste L_2 nach Einträgen (m', R', h', J') mit $m_{out} = m'$. Falls ein solcher Eintrag mit $J' = -1$ existiert, gebe $(0, \varepsilon)$ aus, da in diesem Fall der Angreifer eine ungültige Anfrage $Sign(m_{out}, ID^*)$ gestellt hat.

- Berechne
$$R_{out} := e\left(S_{out}, P_0\right) \cdot \widehat{G}^{-h_{out}}$$
und suche unter den gefundenen Einträgen einen mit $h_{out} = h'$ und $R_{out} = R'$. Falls kein Eintrag gefunden wurde, gebe $(0, \varepsilon)$ aus - der Angreifer hat den H_2-Wert nicht angefragt.
- Für gegebene h_{out}, S_{out} und P_0 existiert genau ein Wert R_{out}, bei dem die Signatur gültig ist. Das folgt aus der Gleichung $R = e\left(S, P_{ID}\right) \cdot \widehat{G}^{-h}$, die nach Konstruktion des Verfahrens gelten muss (siehe Korrektheit des Verfahrens). Es kann also maximal ein Eintrag in der Liste gefunden werden und zwar genau dann, wenn die entsprechende Anfrage $H_2\left(m_{out}, R_{out}\right)$ gestellt wurde. Gebe $(J', (m_{out}, R_{out}, h_{out}, S_{out}))$ aus.

Wir müssen noch zeigen, dass die Wahrscheinlichkeitsverteilungen der beiden Sign-Algorithmen und der Simulation für ein beliebiges Schlüsselpaar (S_0, P_0) gleich sind. Das gilt nach dem Lemma 23 am Ende des Beweises. Alle anderen Anfragen entsprechen den Berechnungen in den beiden IBS-Verfahren.

Somit kann der Angreifer A die Simulation nicht von den ursprünglichen Verfahren unterscheiden, solange keine Kollisionen bei der Simulation auftreten und hat in diesem Fall die gleiche Erfolgswahrscheinlichkeit.

Anwendung des Forking-Lemmas und Berechnung einer Lösung für das q-SDH-Problems:

Jetzt können wir das Forking-Lemma anwenden. Betrachten wir nur den Fall, dass der Forking-Algorithmus erfolgreich ist, also dass S_A bei beiden Ausführungen erfolgreich die Fälschungen $(J_1, (m_1, R_1, h_1, S_1))$ und $(J_2, (m_2, R_2, h_2, S_2))$ ausgibt und dabei $J := J_1 = J_2$ und $h_1 \neq h_2$ gilt.

Dadurch, dass bei beiden (deterministischen) Ausführungen von S_A die Eingabe X und alle h-Werte bis zum Wert h_J identisch sind und diese von S_A in der gleichen Reihenfolge benutzt werden, folgt, dass auch A sich bis zur Anfrage, bei der h_J benutzt wurde, deterministisch verhält. Daraus folgt, dass die entsprechenden Werte m und R bei der J-ten H_2-Anfrage bei beiden Ausführungen gleich sind, da diese vor der Anfrage bestimmt werden (das gilt sowohl für direkte H_2-Anfrage als auch für *Sign*-Algorithmus des Originalverfahrens). Also gilt $m := m_1 = m_2$ und $R := R_1 = R_2$.

Durch die Anwendung des allgemeinen Forking-Lemmas erhalten wir also zwei gültige Signaturen (m, R, h_1, S_1) und (m, R, h_2, S_2) mit $h_1 \neq h_2$. Jetzt zeigen wir, wie daraus eine Lösung für das vorgegebene q-SDH-Problem in zwei Schritten berechnet werden kann. Diese Berechnung entspricht der Berechnung aus der Originalarbeit.

- Erst zeigen wir, dass wir den geheimen Schlüssel $S_0 = \widehat{P}^{\frac{1}{(w_0+\alpha)}}$ als $\left(S_1 \cdot S_2^{-1}\right)^{\frac{1}{h_1-h_2}}$ berechnen können.

- Dann zeigen wir, wie wir aus S_0 das Element $P^{\frac{1}{w_0+\alpha}}$ berechnen können.

Da bei beiden Signaturen die Werte R gleich sind, folgt:

$$e\left(S_1, P_0\right) \cdot \widehat{G}^{-h_1} = e\left(S_2, P_0\right) \cdot \widehat{G}^{-h_2}$$

Daraus folgt:

$$e\left(S_1, P_0\right) \cdot \widehat{G}^{-h_1} = e\left(S_2, P_0\right) \cdot \widehat{G}^{-h_2} \Big| \cdot \widehat{G}^{h_1}$$

$$e\left(S_1, P_0\right) = e\left(S_2, P_0\right) \cdot \widehat{G}^{h_1-h_2} \Big| \cdot e\left(S_2, P_0\right)^{-1} = e\left(S_2^{-1}, P_0\right)$$

$$e\left(S_1, P_0\right) \cdot e\left(S_2^{-1}, P_0\right) = \widehat{G}^{h_1-h_2} \Longleftrightarrow$$

$$e\left(S_1 \cdot S_2^{-1}, P_0\right) = \widehat{G}^{h_1-h_2} \Big| \hat{} \frac{1}{h_1-h_2}$$

$$e\left(S_1 \cdot S_2^{-1}, P_0\right)^{\frac{1}{h_1-h_2}} = \widehat{G} \Longleftrightarrow$$

$$e\left(\left(S_1 \cdot S_2^{-1}\right)^{\frac{1}{h_1-h_2}}, P_0\right) = \widehat{G}$$

Setze nun $P_0 = \widehat{Q}^{w_0+\alpha}$ und $\widehat{G} = e\left(\widehat{P}, \widehat{Q}\right)$ ein. Es gilt also:

$$e\left(\left(S_1 \cdot S_2^{-1}\right)^{\frac{1}{h_1-h_2}}, \widehat{Q}^{w_0+\alpha}\right) = e\left(\widehat{P}, \widehat{Q}\right) \Longleftrightarrow$$

$$e\left(\left(S_1 \cdot S_2^{-1}\right)^{\frac{w_0+\alpha}{h_1-h_2}}, \widehat{Q}\right) = e\left(\widehat{P}, \widehat{Q}\right) \Longleftrightarrow$$

$$\left(S_1 \cdot S_2^{-1}\right)^{\frac{w_0+\alpha}{h_1-h_2}} = \widehat{P} \Big| \cdot \frac{1}{w_0+\alpha}$$

$$\left(S_1 \cdot S_2^{-1}\right)^{\frac{1}{h_1-h_2}} = \widehat{P}^{\frac{1}{w_0+\alpha}}$$

Somit gilt:

$$S_0 = P^{\frac{f(\alpha)}{(w_0+\alpha)}} = \left(S_1 \cdot S_2^{-1}\right)^{\frac{1}{h_1-h_2}}$$

Mit Hilfe von S_0 können wir nun $P^{\frac{1}{w_0+\alpha}}$ wie folgt bestimmen:

Rechne $\frac{f(z)}{z+w_0}$ mit Polynomdivision aus, um die Koeffizienten $\gamma, \gamma_0, ..., \gamma_{q-2} \in \mathbb{Z}_p^*$ zu bestimmen:

$$\frac{f(z)}{z+w_0} = \frac{\sum_{i=0}^{q-1} c_i \cdot z^i}{z+w_0} = \sum_{i=0}^{q-2} \gamma_i \cdot z^i + \frac{\gamma}{z+w_0}$$

Wir ignorieren den Fall, dass $w_0 = w_i$ für ein i gilt. Somit ist $\gamma \neq 0$.

Für S_0 bekommen wir also den folgenden Ausdruck:

$$S_0 = P^{\frac{f(\alpha)}{(w_0+\alpha)}} = P^{\sum_{i=0}^{q-2}(\gamma_i \cdot \alpha^i)} \cdot P^{\frac{\gamma}{\alpha+w_0}}$$

Den ersten Teil können wir berechnen als:

$$T := \psi\left[\prod_{i=0}^{q-2}\left(Q^{\alpha^i}\right)^{\gamma_i}\right] = \prod_{i=0}^{q-2} P^{\gamma_i \cdot \alpha^i} = P^{\sum_{i=0}^{q-2}(\gamma_i \cdot \alpha^i)}$$

Es gilt also:

$$S_0 = T \cdot P^{\frac{\gamma}{\alpha+w_0}} \;\Big|\cdot T^{-1}$$

$$S_0 \cdot T^{-1} = P^{\frac{\gamma}{w_0+\alpha}} \;\Big|\, \hat{}\, \frac{1}{\gamma}$$

$$P^{\frac{1}{w_0+\alpha}} = \left(S_0 \cdot T^{-1}\right)^{\frac{1}{\gamma}}$$

B gibt nun $\left(w_0, P^{\frac{1}{w_0+\alpha}}\right)$ als Lösung des q-SDH-Problems aus.

Erfolgswahrscheinlichkeit

Bestimme erst die Erfolgswahrscheinlichkeit *acc* vom Simulationsalgorithmus S_A. Definiere folgende Ereignisse:

- E_1 - A berechnet eine gültige Signatur (m, h, S) mit dem dazugehörigen Wert $R = e(S, P_0) \cdot \widehat{G}^{-h}$.
- E_2 - A fragt den Hashwert $H_2(m, R)$ während seiner Berechnung nach oder dieser Wert wurde durch eine Signaturanfrage festgelegt. Wir nehmen an, dass der Angreifer in beiden Fällen den Wert bestimmen kann (wir gehen also von dem für den Angreifer besten Fall aus).
- E_3 - Während der Simulation kommt es zu keinen Kollisionen.

Wir haben gezeigt, dass sich die Wahrscheinlichkeitsverteilung der Simulation nicht von den Wahrscheinlichkeitsverteilungen in den vorgestellten IBS-Verfahren unterscheidet. Also bleibt die Erfolgswahrscheinlichkeit von A unverändert:

$$Pr\left(E_1\right) \geq \varepsilon_1$$

Weiterhin ist die Wahrscheinlichkeit, dass der Angreifer eine richtige Signatur ausgibt ohne den Hashwert $H_2\left(m, R\right)$ nachzufragen höchstens gleich der Wahrscheinlichkeit, dass er diesen richtig rät:

$$Pr\left(E_1|\overline{E_2}\right) \leq \frac{1}{2^k - 1}$$

Nun berechne die Wahrscheinlichkeit $Pr\left(E_1 \wedge E_2\right)$, dass sowohl E_1 als auch E_2 gelten. Dafür machen wir eine Abschätzung:

$$\varepsilon_1 \leq Pr\left(E_1\right) = Pr\left(E_1|E_2\right) \cdot Pr\left(E_2\right) + Pr\left(E_1|\overline{E_2}\right) \cdot Pr\left(\overline{E_2}\right) \leq$$

$$Pr\left(E_1 \wedge E_2\right) + \frac{1}{2^k - 1} \cdot 1$$

und somit gilt

$$Pr\left(E_1 \wedge E_2\right) \geq \varepsilon_1 - \frac{1}{2^k - 1}$$

Betrachte nun die Wahrscheinlichkeit für $\overline{E_3}$. Da insgesamt $q_{H_2} + q_S$ Hashwerte definiert werden, ist die Kollisionswahrscheinlichkeit bei einer Signaturanfrage höchstens $\left(q_{H_2} + q_S\right) / \left(2^k - 1\right)$. Also gilt bei q_S Signaturanfragen:

$$Pr\left(\overline{E_3}\right) \leq \frac{q_S \cdot \left(q_{H_2} + q_S\right)}{2^k - 1}$$

Somit bekommen wir, da E_3 unabhängig von den anderen beiden Ereignissen ist:

$$Pr\left(E_1 \wedge E_2 \wedge E_3\right) = Pr\left(E_1 \wedge E_2\right) \cdot Pr\left(E_3\right) \geq$$

$$\left(\varepsilon_1 - \frac{1}{2^k - 1}\right) \cdot \left(1 - \frac{q_S \cdot \left(q_{H_2} + q_S\right)}{2^k - 1}\right) =$$

Den Term $\frac{1}{2^k - 1}$ und die Eins im Nenner können wir vernachlässigen. Somit erhalten wir:

$$acc = Pr\left(E_1 \wedge E_2 \wedge E_3\right) \geq \varepsilon_1 - \frac{\varepsilon_1 \cdot q_S \cdot \left(q_{H_2} + q_S\right)}{2^k}$$

Da beim Erfolg des Forking-Algorithmus immer korrekte Lösung berechnet wird, folgt, dass $\varepsilon_2 = fork$. Nach dem Forking-Lemma ergibt sich dann für $q = q_{H_2} + q_S$ und $|H| = 2^k - 1$ (Abschätzung durch 2^k)

$$\varepsilon_2 \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{|H|}\right) \geq \left(\varepsilon_1 - \frac{\varepsilon_1 \cdot q_S \cdot \left(q_{H_2} + q_S\right)}{2^k}\right) \cdot \left(\frac{\varepsilon_1}{q_{H_2} + q_S} - \frac{\varepsilon_1 \cdot q_S + 1}{2^k}\right)$$

Also in der Größenordnung von

$$\varepsilon_2 \geq \frac{\varepsilon_1^2}{q_{H_2} + q_S} + \frac{\varepsilon_1^2 \cdot q_S^2 \cdot \left(q_{H_2} + q_S\right)}{2^{2 \cdot k}} - \frac{q_S \cdot \varepsilon_1^2 + \varepsilon_1}{2^{k-1}} > \frac{\varepsilon_1^2}{q_{H_2} + q_S} - \frac{q_S \cdot \varepsilon_1^2 + \varepsilon_1}{2^{k-1}}$$

Laufzeit

Betrachte die Laufzeiten der angegebenen Algorithmen. Es ist zu beachten, dass alle Berechnungen in der Gruppe $\mathbb{Z}_p^*$ viel effizienter gemacht werden können, als die Berechnungen in den anderen betrachteten Gruppen. Die Berechnung der Paarungen ist weiterhin viel aufwändiger als die Berechnung von Potenzen und Multiplikationen. Es gilt also:

$$\tau_{\mathbb{Z}_p^*} \ll \tau_{mult} < \tau_{exp} \ll \tau_p$$

wobei $\tau_{\mathbb{Z}_p^*}$ für die Laufzeit der Operationen in der Gruppe $\mathbb{Z}_p^*$ steht, τ_{mult}, τ_{exp} und τ_p dementsprechend für die Laufzeit der Berechnung von Multiplikationen, Potenzen und Paarungen in bilinearen Abbildungsgruppen.

Aus diesem Grund werden die Berechnungen in $\mathbb{Z}_p^*$ am Ende gar nicht in die Laufzeit der Algorithmen eingehen und die Laufzeit für die Berechnung von Multiplikationen und Potenzen können in vielen Fällen vernachlässigt werden.

- IG berechnet außer Operationen in $\mathbb{Z}_p^*$ eine Paarung in Schritt 2, die wir allerdings ignorieren. Weiterhin benötigt IG für die Berechnung von $\widehat{Q}$ und $\widehat{Q}_{pub}$ insgesamt $\mathcal{O}\left[q \cdot \left(\tau_{exp_{\mathbb{G}_2}} + \tau_{mult_{\mathbb{G}_2}}\right)\right]$ Zeit. Die Laufzeit von IG wird aber durch den dritten Schritt bestimmt, denn für die Berechnung der geheimen Schlüssel braucht man $\mathcal{O}\left[q^2 \cdot \left(\tau_{exp_{\mathbb{G}_2}} + \tau_{mult_{\mathbb{G}_2}}\right)\right]$ Zeit. Da die Berechnung von Potenzen aufwändiger ist, bekommen wir insgesamt die Laufzeit von:

$$\mathcal{O}\left(q^2 \cdot \tau_{exp_{\mathbb{G}_2}}\right)$$

- Die Laufzeit von S_A wird durch die Laufzeit t_1 des Angreifers A und die *Sign*-Anfragen bestimmt. Die Berechnung der Signatur im BLMQ-IBS-Verfahren benötigt $\tau_{exp_{\mathbb{G}_2}}$ Zeit für die Berechnung von R und $\tau_{exp_{\mathbb{G}_1}}$ Zeit für die Berechnung von S. In dem KH-IBS-verfahren braucht die Berechnung von S insgesamt $2 \cdot \tau_{exp_{\mathbb{G}_1}} + \tau_{mult_{\mathbb{G}_1}}$ Zeit. Falls eine *Sign*-Anfrage simuliert werden muss, braucht man dafür allerdings $\tau_p + \tau_{exp_{\mathbb{G}_T}} + \tau_{mult_{\mathbb{G}_T}}$ Zeit. Da dieser Fall der Worst-Case ist und die Berechnung der Paarungen zeitaufwändiger ist, braucht S_A insgesamt $t_1 + \mathcal{O}\left(q_S \cdot \tau_p\right)$ Zeit.
- Der Forking-Algorithmus simuliert nur zwei mal S_A und benötigt sonst nur konstanten Aufwand. Also insgesamt:

$$2 \cdot t_1 + \mathcal{O}\left(q_S \cdot \tau_p\right)$$

- Die Berechnung der Lösung des q-SDH-Problems aus den Fälschungen braucht $2 \cdot \tau_{exp_{\mathbb{G}_1}} + \tau_{mult_{\mathbb{G}_1}}$ für die Berechnung von S_0. Die Berechnung von T benötigt $\mathcal{O}\left[q \cdot \left(\tau_{exp_{\mathbb{G}_2}} + \tau_{mult_{\mathbb{G}_2}}\right)\right]$ Zeit. Der letzte Schritt braucht dann wiederum $2 \cdot \tau_{exp_{\mathbb{G}_1}} + \tau_{mult_{\mathbb{G}_1}}$ Zeit. Also insgesamt

$$\mathcal{O}\left[q \cdot \left(\tau_{exp_{\mathbb{G}_2}}\right)\right]$$

Die Laufzeit wird also durch die Berechnungen von IG und die Simulation der Signaturanfragen bestimmt:

$$t_2 = 2 \cdot t_1 + \mathcal{O}\left(q_S \cdot \tau_p\right) + \mathcal{O}\left(q^2 \cdot \tau_{exp_{\mathbb{G}_2}}\right)$$

□

Analyse der Verteilungen

In diesem Abschnitt zeigen wir, dass die Verteilungen, die von den beiden Sign-Algorithmen erzeugt werden, bei einem beliebigen aber festen Masterschlüssel $s = \alpha$, Schlüsselpaar (S_0, P_0) und dem dazugehörigen Hashwert w_0 der Identität, mit der Verteilung der Simulation übereinstimmen.

Lemma 23. *Die Verteilungen δ der BLMQ-IBS, δ' der KH-IBS und δ'' der Simulation sind identisch bei beliebigen aber festen Masterschlüssel $s = \alpha$, Schlüsselpaar (S_0, P_0) und dem dazugehörigen Hashwert w_0 der Identität.*

Beweis. Wir werden zeigen, dass in allen drei Fällen die Werte R und h auf die gleiche Weise verteilt sind, obwohl diese unterschiedlich erzeugt werden und dass genau ein Element S existiert, das durch R und h eindeutig bestimmt ist. Bei allen drei Fällen gilt nämlich dieselbe Gleichung:

$$R = e\left(S, P_0\right) \cdot G^{-h}$$

Dadurch zeigen wir, dass die Verteilungen identisch sind.

1. Die Verteilungen δ von BLMQ-IBS:

$$\delta = \left\{ (R, h, S) \;\middle|\; \begin{array}{c} x \overset{R}{\leftarrow} \mathbb{Z}_p^* \\ R = G^x \in \mathbb{G}_T \backslash \{1\} \\ h = H_2\left(m, R\right) \in \mathbb{Z}_p^* \\ S = {S_0}^{x+h} \in \mathbb{G}_1 \end{array} \right\}$$

   - Es gilt $R = G^x$. Da x ein zufälliges, gleichverteiltes Element aus $\mathbb{Z}_p^*$ ist und G ein Generator aus der Gruppe $\mathbb{G}_T$ mit Primordnung p ist, folgt, dass R gleichverteilt aus $\mathbb{G}_T \backslash \{1\}$ ist.
   - Nach der Definition des Zufallsorakels sind die Werte h unabhängig von R und zufällig, gleichverteilt aus $\mathbb{Z}_p^*$.
   - Die Gleichung $R = e\left(S, P_0\right) \cdot G^{-h}$ gilt für alle gültigen Signaturen (siehe Korrektheit des Verfahrens).

2. Die Verteilungen δ' von KH-IBS:

$$\delta' = \left\{ (R,h,S) \;\middle|\; \begin{array}{c} x \overset{R}{\leftarrow} \mathbb{Z}_p^* \\ R = e\left(P,P_0\right)^x \in \mathbb{G}_T \backslash \{1\} \\ h = H_2\left(m,R\right) \in \mathbb{Z}_p^* \\ S = P^x \cdot {S_0}^h = P^{x+\frac{h}{H_1(ID)+s}} \in \mathbb{G}_1 \end{array} \right\}$$

   - Es gilt $R = e\left(P,P_0\right)^x$. Da x ein zufälliges, gleichverteiltes Element aus $\mathbb{Z}_p^*$ ist und $e\left(P,P_0\right)$ ein Generator von $\mathbb{G}_T$ mit Primordnung p ist, folgt, dass R gleichverteilt aus $\mathbb{G}_T \backslash \{1\}$ ist.
   - Nach der Definition des Zufallsorakels sind die Werte h unabhängig von R und zufällig, gleichverteilt aus $\mathbb{Z}_p^*$.
   - Die Gleichung $R = e\left(S,P_0\right) \cdot G^{-h}$ gilt für alle gültigen Signaturen (siehe Korrektheit des Verfahrens).

3. Betrachte nun die Wahrscheinlichkeitsverteilungen δ'' der Simulation:

$$\delta'' = \left\{ (R,h,S) \;\middle|\; \begin{array}{c} h \overset{R}{\leftarrow} \mathbb{Z}_p^* \\ S \overset{R}{\leftarrow} \mathbb{G}_1 \\ R = e\left(S,P_0\right) \cdot \widehat{G}^{-h} \\ R \neq 1 \end{array} \right\}$$

   Zur Erinnerung $P_0 = \widehat{Q}^{w_0+\alpha}$ und wir ignorieren den Fall, dass $w_0 + \alpha = 0$. Da w_0 und α fest sind, definiere der Einfachheit halber $t := (w_0 + \alpha) \in \mathbb{Z}_p^*$. Dann gilt:

   - Nach Konstruktion der Simulation ist h zufällig, gleichverteilt aus $\mathbb{Z}_p^*$.
   - Betrachte nun die Verteilung über R für ein gegebenes h. S können wir darstellen als $\widehat{P}^x$ für ein $x \in \mathbb{Z}_p$. Eine zufällige Wahl von S entspricht also einer zufälligen, gleichverteilten Wahl von x aus $\mathbb{Z}_p$. Somit gilt $R = \widehat{G}^{x \cdot t - h}$. Unter der Voraussetzung, dass der Fall $R = 1$ nicht auftritt folgt unmittelbar, dass $x \cdot t - h$ zufällig gleichverteilt aus $\mathbb{Z}_p^*$ ist und somit R zufällig, gleichverteilt aus $\mathbb{G}_T \backslash \{1\}$ ist.
   - Die Gleichung $R = e\left(S,P_0\right) \cdot G^{-h}$ gilt nach Konstruktion der Simulation.

□

Aus den beiden Lemmas ergibt sich folgendes Theorem:

Theorem 24. *Wenn es einen CMIDA Angreifer F_0 auf eins der vorgestellten IBS-Verfahren mit Laufzeit t_0 und Erfolgswahrscheinlichkeit ε_0 gibt, der q_{H_i} Anfragen an die Zufallsorakel H_i ($i = 1, 2$), q_E Extract-Anfragen sowie q_S Sign-Anfragen stellt, dann existiert ein Algorithmus B, der das q-SDH-Problem (für $q = q_{H_1} + q_E + q_S$)*

in Zeit $t \leq c \cdot t_0 + \mathcal{O}(q_S \cdot \tau_p) + \mathcal{O}\left(q^2 \cdot \tau_{exp_{\mathbb{G}_2}}\right)$ *(wobei* c *eine geeignete Konstante ist) mit Erfolgswahrscheinlichkeit*

$$\varepsilon \geq \frac{\varepsilon_0^2}{q_{H_1}^2 \cdot (q_{H_2} + q_S)} - \frac{q_S \cdot \varepsilon_0^2 + q_{H_1} \cdot \varepsilon_0}{2^{k-1} \cdot q_{H_1}^2}$$

löst. Dabei bezeichnet $\tau_{exp_{\mathbb{G}_2}}$ *die Kosten für die Berechnung der Potenzen in* $\mathbb{G}_2$ *und* τ_p *die Kosten für die Berechnung der Paarung.*

Korollar 25. Unter der Annahme, dass das q-SDH-Problem in den Abbildungsgruppen hart ist, sind die vorgestellten IBS-Verfahren *existentiell fälschungssicher* gegen einen adaptive-chosen-message-and-ID-Angreifer.

4 Identitätsbasierte Signcryption

In diesem Kapitel wird das Signcryptionverfahren aus der Originalarbeit [BLMQ05] in Abschnitt 4.1 vorgestellt. In Abschnitt 4.2 präsentieren wir den kompletten Beweis für IND-IBSC-CCA-Sicherheit des Verfahrens. In Abschnitt 4.3 wird nur die Idee des Beweises des existentiellen Fälschungssicherheit des Verfahrens vorgestellt, da diese den anderen Beweise sehr ähnelt.

4.1 Das Signcryptionverfahren

In diesem Abschnitt wird das in der Originalarbeit vorgeschlagene Signcryptionverfahren vorgestellt. Die Beschreibung des Verfahrens wurde erweitert, um ein besseres Verständnis zu gewährleisten.

Definition 26. Das BLMQ-IBSC-Verfahren Π = (Setup, Extract, Sign/Encrypt, Decrypt/Verify, Verify) ist definiert als:

- **Setup:** Gegeben sei der Sicherheitsparameter 1^k. Die vertrauenswürdige Stelle wählt bilineare Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ von Primordnung $p > 2^k$ mit der dazugehörigen bilinearen Abbildung $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ und dem dazugehörigen Isomorphismus $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Dann wird ein Generator aus der zweiten Gruppe gewählt und aus diesen zwei Generatoren für die anderen Gruppen berechnet:

  $$Q \overset{R}{\leftarrow} \mathbb{G}_2;\ P := \psi(Q) \in \mathbb{G}_1;\ G := e(P, Q) \in \mathbb{G}_T$$

  Der Masterschlüssel s wird aus $\mathbb{Z}_p^*$ gewählt:

  $$s \overset{R}{\leftarrow} \mathbb{Z}_p^*$$

  Der systemweite öffentliche Schlüssel Q_{pub} wird aus dem Masterschlüssel berechnet:

  $$Q_{pub} = Q^s \in \mathbb{G}_2$$

  Weiterhin werden im Verfahren drei Hashfunktionen benutzt. $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ und $H_2 : \{0,1\}^n \times \mathbb{G}_T \to \mathbb{Z}_p^*$ zur Berechnung der Hashwerte der Identitäten bzw. der Nachrichten sowie $H_3 : \mathbb{G}_T \to \{0,1\}^n$ für die Berechnung der

Verschlüsselung. n bezeichnet dabei die Länge der Nachricht. Die öffentlichen Parameter sind somit:

$$params := \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, G, e, \psi, Q_{pub}, H_1, H_2, H_3\}$$

- **Extract:** Für eine gegebene Identität ID berechnet die vertrauenswürdige Stelle den geheimen Schlüssel als:

$$S_{ID} = Q^{\frac{1}{H_1(ID)+s}} \in \mathbb{G}_2$$

Der dazugehörige öffentliche Schlüssel kann folgendermaßen aus ID berechnet werden:[1]

$$P_{ID} = Q^{H_1(ID)} \cdot Q_{pub} = Q^{H_1(ID)+s}$$

Man beachte, dass somit gilt:

$$e\left(\psi\left(P_{ID}\right), S_{ID}\right) = e\left(\psi\left(S_{ID}\right), P_{ID}\right) = e\left(P, Q\right)^{\frac{H_1(ID)+s}{H_1(ID)+s}} = G \tag{4.1}$$

Auch für dieses Verfahren gilt, dass für eine $\widetilde{ID}$ mit $H_1\left(\widetilde{ID}\right) = -s$ kein geheimer Schlüssel $S_{\widetilde{ID}}$ berechnet werden kann. Wir ignorieren diesen Fall wieder und gehen davon aus, dass $H_1\left(ID\right) + s$ ein gleichverteiltes Element aus $\mathbb{Z}_p^*$ ist.

- **Sign/Encrypt:** Gegeben sei eine Nachricht $m \in \{0,1\}^n$, der geheime Schlüssel des Senders S_{ID_S} und die Identität des Empfängers ID_R (bzw. dessen öffentliche Schlüssel P_{ID_R}). Führe folgende Schritte durch:

  1. Wähle $x \overset{R}{\leftarrow} \mathbb{Z}_p^*$ und berechne $R := G^x \in \mathbb{G}_T$.
  2. Berechne die Verschlüsselung $c := m \oplus H_3\left(R\right) \in \{0,1\}^n$.
  3. Setze $h := H_2\left(m, R\right) \in \mathbb{Z}_p^*$.
  4. Berechne $S := \psi\left(S_{ID_S}\right)^{x+h} \in \mathbb{G}_1$.
  5. Berechne $T := \psi\left(P_{ID_R}\right)^x \in \mathbb{G}_1$.
  6. Die Signcryption für m lautet:

$$\varsigma := (c, S, T) \in \{0,1\}^n \times \mathbb{Z}_p^* \times \mathbb{G}_1$$

- **Decrypt/Verify**: Gegeben sei eine Signcryption $\varsigma = (c, S, T)$, der geheime Schlüssel des Empfängers S_{ID_R} und die Identität des Senders ID_S (bzw. dessen öffentliche Schlüssel P_{ID_S}). Führe folgende Berechnungen durch:

[1] Wir verweisen wieder darauf, dass der öffentliche Schlüssel an dieser Stelle wie schon bei IBS-Verfahren eine andere Bedeutung hat, als bei asymmetrischen Verfahren. Dieser wird hier aus der Identität berechnet. Die Identität ist der eigentliche öffentlicher Schlüssel eines Benutzers.

1. $R := e\left(T, S_{ID_R}\right)$
2. $m := c \oplus H_3\left(R\right)$
3. $h := H_2\left(m, R\right)$
4. Falls die folgende Gleichung gilt:

$$R \stackrel{?}{=} e\left(S, P_{ID_S}\right) \cdot G^{-h}$$

gebe die Nachricht m und die Signatur $(h, S) \in \mathbb{Z}_p^* \times \mathbb{G}_1$ aus. Sonst gebe $\perp$ aus.

- **Verify**: Die Signatur $\sigma = (h, S)$ für eine Nachricht m und Identität des Senders ID_S wird akzeptiert, falls gilt:

$$h \stackrel{?}{=} H_2\left(m, e\left(S, P_{ID_S}\right) \cdot G^{-h}\right)$$

Beweis. [Korrektheit] Zeige die Korrektheit des Verfahrens in zwei Schritten. Erst zeigen wir, dass der $Decrypt/Verify$-Algorithmus die Nachricht korrekt entschlüsselt und dann zeigen wir, dass er dabei eine gültige Signatur des Senders für diese Nachricht berechnet:

- Entschlüsselung: Der Empfänger mit der Identität ID_R kann mit seinem geheimen Schlüssel korrekt R berechnen:

$$e\left(T, S_{ID_R}\right) = e\left(\psi\left(P_{ID_R}\right)^x, S_{ID_R}\right) \stackrel{4.1}{=} G^x = R$$

Mit einem korrekten Wert R kann die Nachricht m und der Hashwert h in den Schritten 2 und 3 offensichtlich korrekt aus R und c berechnet werden.

- Gültigkeit: Der Empfänger kennt also h aus dem dritten Schritt sowie S als Teil der Signcryption ς. Zeige nun, dass die Gleichung $R = e\left(S, P_{ID_S}\right) \cdot G^{-h}$ gilt:

$$e\left(S, P_{ID_S}\right) \cdot G^{-h} = e\left(\psi\left(S_{ID_S}\right)^{x+h}, P_{ID_S}\right) \cdot G^{-h} \stackrel{4.1}{=} G^{x+h-h} = R$$

Somit liefert der Decrypt/Verify-Algorithmus eine korrekte Signatur (h, S) für die Nachricht m, denn für $R = e\left(S, P_{ID_S}\right) \cdot G^{-h}$ gilt offensichtlich

$$H_2\left(m, e\left(S, P_{ID_S}\right) \cdot G^{-h}\right) = H_2\left(m, R\right) = h$$

Somit ist das Verfahren korrekt.

□

Irreflexivitätseigenschaft

Bei den Sicherheitsbeweisen werden wir die Irreflexivitätseigenschaft benutzen. Diese besagt, dass der Sender nie eine Nachricht mit seiner eigenen Identität als Identität des Empfängers signiert:

$$ID_S \neq ID_R$$

Somit führt er auch nie den Decrypt/Verify-Algorithmus für $ID_S = ID_R$ aus. Alle Beweise gelten nur unter dieser Annahme.

Unter Umständen kann es aber sinnvoll sein eine Signcryption für sich selbst zu berechnen. Man kann diese Funktionalität den Benutzern anbieten, wie Xavier Boyen in seiner Arbeit [Boy03] erklärt hat, indem man ihnen z.B. eine zweite Identität mit dem dazugehörigen geheimen Schlüssel zur Verfügung stellt, die in solchen Fällen als Identität des Empfängers benutzt werden kann. Eine andere Möglichkeit ist die Identität des Benutzers in zwei Teile zu trennen und für jeden von diesen einen geheimen Schlüssel zu erzeugen. Der zweite Schlüssel wird nur zum Erzeugen der Signcryption für sich selbst eingesetzt.

4.2 Die Sicherheitsanalyse I

In diesem Abschnitt zeigen wir, dass das BLMQ-IBSC-Verfahren *IND-IBSC-CCA-sicher* ist.

Theorem 27. *Angenommen es existiert ein IND-IBSC-CCA-Angreifer A gegen das vorgestellte BLMQ-IBSC-Verfahren mit dem Vorteil ε und mit der Laufzeit t im IND-IBSC-CCA $_{A,\Pi}(k)$-Spiel, der jeweils q_{H_i} Anfragen an die Zufallsorakel H_i ($i = 1,2,3$), q_E Extract-Anfragen, q_{SE} Sign/Encrypt-Anfragen und q_{DV} Decrypt/Verify-Anfragen stellt. Dann existiert ein Algorithmus B, der das $q-BDHI$-Problem für $q = q_{H_1} + q_E + 2 \cdot q_{SE} + 2 \cdot q_{DV}$ mit Erfolgswahrscheinlichkeit:*

$$\varepsilon' \geq \frac{\varepsilon_0}{q \cdot (2 \cdot q_{H_2} + q_{H_3} + 2 \cdot q_{SE})} \cdot \left(1 - q_{SE} \cdot \frac{q_{SE} + q_{H_2}}{2^k}\right) \cdot \left(1 - \frac{q_{DV}}{2^k}\right)$$

und in Zeit:

$$t' \leq t + \mathcal{O}\left((q_{SE} + q_{DV}) \cdot \tau_p\right) + \mathcal{O}\left(q_{H_2} \cdot \tau_{exp_{\mathbb{G}_T}}\right) + \mathcal{O}\left(q^2 \cdot \tau_{exp_{\mathbb{G}_2}}\right)$$

löst. Dabei stehen $\tau_{exp_{\mathbb{G}_T}}$ und $\tau_{exp_{\mathbb{G}_2}}$ jeweils für die Zeit der Berechnung der Potenzen in $\mathbb{G}_T$ und $\mathbb{G}_2$. τ_p ist die Zeit für die Berechnung der Paarungen.

Die Idee des Beweises basiert darauf, dass bei einer gegebenen Identität des Empfängers ID_R zu jedem Wert T der Signcryption (c, S, T) ein eindeutiger Wert $R = e(T, S_{ID_R})$ gehört (siehe Korrektheit des Verfahrens). Weiterhin geht in die Berechnung von c der Wert $H_3(R)$ ein und solange der Angreifer diesen Wert nicht

anfragt, kann er keine Informationen aus c gewinnen. S und T stehen dagegen über den Wert $h = H_2(m, R)$ in Relation zueinander und solange dieser nicht angefragt wurde, kann der Angreifer nicht erkennen, ob die Werte gültig sind. Also auch wenn der Angreifer den Wert R korrekt berechnet, muss er mindestens einen der Hashwerte $H_3(R)$ oder $H_2(m, R)$ anfragen. Ansonsten sehen für ihn alle drei Werte der Signcryption wie zufällige Elemente aus und sein Vorteil im *IND-IBSC-CCA* $_{A,\Pi}(k)$-Spiel ist gleich 0.

Beweis. Gegeben seien bilineare Abbildungsgruppen $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Der Algorithmus B nimmt als Eingabe ein $(q+2)$-Tupel $\left(P, Q, Q^{\alpha}, Q^{\alpha^2}, ..., Q^{\alpha^q}\right)$ mit Generatoren $Q \in \mathbb{G}_2$ und $P = \psi(Q) \in \mathbb{G}_1$ sowie einem unbekannten Element $\alpha \in \mathbb{Z}_p^*$ und versucht den Wert $e(P, Q)^{1/\alpha}$ zu berechnen. Der Beweis ist wie folgt strukturiert:

- In der Setup-Phase wird der Masterschlüssel implizit auf $s = -\alpha - w_0$ für ein zufällig gewähltes w_0 gesetzt. Die entsprechenden öffentlichen Parameter werden in Abhängigkeit von α berechnet. Außerdem werden $q-1$ Paare $\left(w_i, \widehat{Q}^{\frac{1}{w_i+s}}\right)$ berechnet. Der erste Wert kann als Hashwert einer Identität und der zweite Wert als geheime Schlüssel für diese Identität ausgegeben werden.
- In der Phase 1 zeigen wir, wie mit Hilfe von ausgerechneten Parametern die Anfragen des Angreifers im *IND-IBSC-CCA* $_{F,\Pi}\left(1^k\right)$-Spiel simuliert werden können.
- In der Challenge-Phase ignorieren wir die Nachrichten vom Angreifer und geben ihm eine Signcryption $\left(c, S, \widehat{T}\right)$ vor. Der Wert $\widehat{T}$ wird dabei so konstruiert, dass aus dem dazugehörigen eindeutig bestimmten Wert $\widehat{R}$ die Lösung des Problems berechnet werden kann.
- Die Ausgabe des Angreifers ignorieren wir und wählen eine der direkt oder indirekt gestellten $H_2\left(\widetilde{R}\right)$- oder $H_3\left(m, \widetilde{R}\right)$-Anfragen und berechnen für $\widetilde{R}$ die Lösung. Mit bestimmter Wahrscheinlichkeit raten wir die richtige Anfrage, so dass $\widetilde{R} = \widehat{R}$ gilt und berechnen die korrekte Lösung des Problems.
- Im letzten Teil des Beweises wird gezeigt wie aus $\widehat{R}$ die Lösung des Problems berechnet werden kann.

Zusatzannahmen o.B.d.A. (Die Annahmen entsprechen den Zusatzannahmen bei Lemma 21 und werden deswegen nicht weiter erörtert.)

1. A macht für jede ID maximal eine Anfrage des Hashwertes $H_1(ID)$.
2. Vor jeder Anfrage, in die eine Identität ID involviert ist, macht der Angreifer die Anfrage des Hashwertes $H_1(ID)$. Es können also maximal $q = q_{H_1} + q_E + 2 \cdot q_{SE} + 2 \cdot q_{DV}$ Anfragen an H_1-Orakel gemacht werden.
3. Der Angreifer fragt den Hashwert der Identität des Empfängers ID_R, für die er die Signcryption in der Challenge-Phase bekommt, irgendwann nach.

Setup-Phase: Die öffentlichen Parameter des Verfahrens werden fast genau so berechnet, wie im Lemma 22 auf Seite 35. Die ersten drei Schritte entsprechen der Berechnung in dem Lemma und werden kurz zusammengefasst.

1. Wähle $v_1, ..., v_{q-1} \overset{R}{\leftarrow} \mathbb{Z}_p^*$. Die Funktion $f(z)$ wird definiert als $f(z) := \prod_{i=1}^{q-1}(z + v_i)$. Daraus berechnen wir die Werte $c_0, ..., c_{q-1} \in \mathbb{Z}_p^*$:

$$f(z) := \sum_{i=0}^{q-1} c_i \cdot z^i$$

2. Die Generatoren $\widehat{Q} \in \mathbb{G}_2$, $\widehat{P} \in \mathbb{G}_1$ und $\widehat{G} \in \mathbb{G}_T$ werden berechnet als:

$$\widehat{Q} := Q^{f(\alpha)};\ \widehat{P} := P^{f(\alpha)};\ \widehat{G} := e\left(\widehat{P}, \widehat{Q}\right)$$

   Ein weiteres Element U aus $\mathbb{G}_2$ wird berechnet als:

$$U := \widehat{Q}^{\alpha}$$

   (dieser entspricht $\widehat{Q_{pub}}$ im Lemma 22)

3. Für alle v_i bestimme das Element U_i:

$$U_i := \widehat{Q}^{\frac{1}{\alpha + v_i}}$$

   (entspricht der Berechnung von geheimen Schlüssel im Lemma 22)

4. Wähle nun $w_0 \overset{R}{\leftarrow} \mathbb{Z}_p^*$ und berechne den öffentlichen Schlüssel $\widehat{Q_{pub}} \in \mathbb{G}_2$ als

$$\widehat{Q_{pub}} := U^{-1} \cdot \widehat{Q}^{-w_0} = \widehat{Q}^{-\alpha - w_0}$$

   Somit setzen wir implizit den Masterschlüssel auf

$$s = -\alpha - w_0$$

5. $\forall i \in \{1, ..., q-1\}$ berechne:

$$w_i := w_0 - v_i$$

6. Nun können wir für alle w_i mit $i \in \{1, ..., q-1\}$ den geheimen Schlüssel S_i sowie den öffentlichen Schlüssel P_i berechnen:

$$S_i := U_i^{-1} = \widehat{Q}^{-\frac{1}{\alpha + v_i}} = \widehat{Q}^{\frac{1}{(s + w_0) + (w_i - w_0)}} = \widehat{Q}^{\frac{1}{w_i + s}}$$

$$P_i := \widehat{Q}^{w_i} \cdot \widehat{Q_{pub}} = \widehat{Q}^{w_i + s}$$

   Für w_0 berechnen wir analog den öffentlichen Schlüssel P_0:

$$P_0 := \widehat{Q}^{w_0 + s}$$

   Wir ignorieren wieder den Fall, dass ein w_i gibt, für den $w_i + s = 0$ gilt, denn dann kennen wir α und können das Problem ohne weitere Berechnungen lösen. Der Fall tritt, wie bei IBS schon erklärt, mit einer vernachlässigbaren Wahrscheinlichkeit auf.

7. Wähle $d \overset{R}{\leftarrow} \{1, ..., q\}$ und benenne die Paare $(w_1, S_1), ..., (w_{q-1}, S_{q-1})$ in $(w_1, S_1), ..., (w_{d-1}, S_{d-1}), (w_{d+1}, S_{d+1}), ..., (w_q, S_q)$ um.

Somit haben wir ohne Kenntnis von α die Parameter wie folgt definiert:

- Die Generatoren $\widehat{P} = P^{f(\alpha)} \in \mathbb{G}_1$, $\widehat{Q} = Q^{f(\alpha)} \in \mathbb{G}_2$ und $\widehat{G} = e\left(\widehat{P}, \widehat{Q}\right) \in \mathbb{G}_T$.
- Der Masterschlüssel: $s = -\alpha - w_0 \in \mathbb{Z}_p^*$
- Der systemweite öffentliche Schlüssel $\widehat{Q_{pub}} = \widehat{Q}^s$

Das entspricht der Definition von IBSC. Außerdem kennen wir $q - 1$ Tripel $\left(w_i, \widehat{Q}^{\frac{1}{w_i+s}}, \widehat{Q}^{w_0+s}\right)$ die als $(H(ID), S_{ID}, P_{ID})$ definiert werden können, wobei w_i nach Konstruktion zufällig, gleichverteilt gewählt wurden. Für w_0 kennen wir den öffentlichen Schlüssel P_0.

Phase 1:

B simuliert A mit den zuvor berechneten öffentlichen Parametern:

$$params := \left\{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \widehat{P}, \widehat{Q}, \widehat{G}, e, \psi, \widehat{Q_{pub}}, H_1, H_2, H_3\right\}$$

Bezeichne die Identität, die bei der i-ten Anfrage an das H_1-Orakel als Eingabe dient als ID_i. Das ist möglich auf Grund von den Zusatzannahmen 1 und 2.

Beantworte somit die Anfragen von A wie folgt:

- $H_1(ID_i) := \begin{cases} w_i & \text{falls } i \neq d \\ w_0 & \text{falls } i = d \end{cases}$
- $H_2(m, R)$: Suche in der Liste L_2 nach dem Wert h_2 für m und R und gebe diesen falls gefunden zurück. Sonst gebe einen zufälligen Wert $h_2 \overset{R}{\leftarrow} \mathbb{Z}_p^*$ zurück. Zusätzlich berechne $h_3 := H_3(R)$, wie später beschrieben, und speichere $\left(m, R, h_2, c = m \oplus h_3, \gamma = R \cdot \widehat{G}^{h_2}\right)$ in L_2.
- $H_3(R)$: Suche in der Liste L_3 nach dem Wert für R und gebe diesen falls gefunden zurück. Sonst gebe einen zufälligen Wert $h_3 \overset{R}{\leftarrow} \{0,1\}^n$ zurück und speichere das Paar (R, h_3) in der Liste L_3.
- $Extract(ID_i) := \begin{cases} S_i & \text{falls } i \neq d \\ \text{breche ab und gebe } \perp \text{ aus} & \text{falls } i = d \end{cases}$
- $Sign/Encrypt(m, ID_S, ID_R)$: Für $ID_S \neq ID_d$ kennt B den geheimen Schlüssel S_{ID_S} des Senders sowie den öffentlichen Schlüssel P_{ID_R} des Empfängers und kann somit die Signcryption wie im Verfahren definiert berechnen.
  Sei also $ID_S = ID_d$. Nach der Irreflexivitätseigenschaft gilt dann $ID_R \neq ID_d$.

Wir kennen also die beiden Schlüssel S_{ID_R} und P_{ID_R} des Empfängers sowie den öffentlichen Schlüssel $P_{ID_S} = P_0$ des Senders.

Berechne eine gültige Signcryption wie folgt:

1. Wähle $h \overset{R}{\leftarrow} \mathbb{Z}_p^*$.
2. Wähle $x \overset{R}{\leftarrow} \mathbb{Z}_p$ und berechne $T := \psi(P_0)^x \cdot \psi(P_{ID_R})^{-h}$. Falls $T = 1$ gilt, führe den Schritt neu durch.[2]
3. Berechne $S := \psi(S_{ID_R})^x$.
4. Berechne $R := e(T, S_{ID_R})$ und $c := m \oplus H_3(R)$, wobei $H_3(R)$ als zusätzliche Anfrage aufgefasst wird.
5. Falls der Wert für m und R in der Liste L_2 bereits existiert, breche die Berechnung ab und gebe $\perp$ zurück. Sonst speichere in der Liste L_2 den Wert $\left(m, R, h, c, R \cdot \widehat{G}^h\right)$. Dadurch definieren wir explizit $H_2(m, R) := h$.
6. Gebe den Tripel (c, S, T) als Antwort auf die Anfrage zurück.

Beweis. [Korrektheit der Simulation des Sign/Encrypt-Algorithmus für $ID_S = ID_d$]

Nach Konstruktion gilt:

$$R = e(T, S_{ID_R}) = \left(\psi(P_0)^x \cdot \psi(P_{ID_R})^{-h}, S_{ID_R}\right) \overset{4.1}{=} (\psi(P_0)^x, S_{ID_R}) \cdot \widehat{G}^{-h} =$$

$$(\psi(S_{ID_R})^x, P_0) \cdot \widehat{G}^{-h} = e(S, P_0) \cdot \widehat{G}^{-h}$$

Die berechnete Signcryption ist somit gültig, denn bei der Korrektheit des Verfahrens wurde gezeigt, dass $R = e(T, S_{ID_R})$ gilt. Somit berechnet der Decrypt/Verify-Algorithmus nach Konstruktion des Punktes 4 der Simulation korrekt den Wert R. Aus R werden m und h korrekt bestimmt. Die Gültigkeitsüberprüfung $R \overset{?}{=} e(S, P_{ID_S}) \cdot G^{-h}$ funktioniert, da die obere Gleichung gilt.

Wir müssen noch zeigen, dass die Wahrscheinlichkeitsverteilungen über (c, h, h_3, R, S, T) des Verfahrens und der Simulation identisch sind. Siehe dafür Lemma 29 am Ende des Beweises. □

- *Decrypt/Verify* $(\varsigma = (c, S, T), ID_S, ID_R)$: Analog zu Sign/Encrypt-Anfragen können wir alle Anfragen mit $ID_R \neq ID_d$ wie im Verfahren definiert berechnen und beantworten. Nehme also an, dass $ID_R = ID_d$ und nach der Irreflexivitätseigenschaft $ID_S \neq ID_d$ gilt. Wir kennen also $P_{ID_R} = P_0$, P_{ID_S} und S_{ID_S}. Simuliere die Anfrage wie folgt:

[2] Der Fall tritt mit einer vernachlässigbaren Wahrscheinlichkeit $\frac{1}{2^{k-1}}$ und wird nicht weiter betrachtet.

1. Berechne $\gamma' := e\left(S, P_{ID_S}\right)$.
2. Suche in der Liste L_2 nach Einträgen $(m_i, R_i, h_i, c_i, \gamma_i)$ mit $c_i = c$ und $\gamma_i = \gamma'$. Falls kein Eintrag gefunden wurde, beantworte die Anfrage mit $\perp$.
3. Unter den gefundenen Einträgen suche den, bei dem die folgende Gleichung erfühlt wird:

$$\frac{e\left(T, S_{ID_S}\right)}{e\left(S, P_0\right)} = e\left(\psi\left(P_0\right), S_{ID_S}\right)^{-h_{2,i}}$$

   Falls kein Eintrag gefunden wurde, beantworte die Anfrage mit $\perp$. Ansonsten finden wir, wie bei der Korrektheit gezeigt wird, genau einen solchen Eintrag $(m_i, R_i, h_{2,i}, c_i, \gamma_i)$. Beantworte die Anfrage mit $(m := m_i, \sigma := (h_{2,i}, S))$.

Beweis. [Korrektheit der Simulation des Decrypt/Verify-Algorithmus für $ID_R = ID_d$]

Gegeben sei die Signcryption $\varsigma = (c, S, T)$. Seien dann R', m' und h' wie im Decrypt/Verify-Algorithmus definiert (wir können diese aber ohne S_{ID_R} nicht ausrechnen):

- $R' = e\left(T, S_{ID_R}\right)$
- $m' = c \oplus H_3\left(R'\right)$
- $h' = H_2\left(m', R'\right)$

Für eine gültige Signcryption muss nach Konstruktion des Decrypt/Verify-Algorithmus gelten:

$$e\left(S, P_{ID_S}\right) \cdot \widehat{G}^{-h'} = R' = e\left(T, S_{ID_R}\right)$$

Falls der Angreifer die Anfrage $H_2\left(m', R'\right)$ nicht stellt und dieser Wert durch die Sign/Encrypt-Anfrage nicht definiert wurde, kennt er h' nicht. Somit kann er in diesem Fall den Wert $G^{-h'}$ nicht bestimmen und die Gültigkeit der oberen Gleichung nicht überprüfen. Somit kann er die Werte nur raten.

Aus diesem Grund können wir den Fall, dass der Wert $H_2\left(m', R'\right)$ durch eine H_2- oder Sign/Encrypt-Anfrage nicht definiert wurde, vernachlässigen und weisen die Signcryption einfach ab (entweder im Schritt 2 oder 3 der Simulation). Den Fall, dass der Angreifer eine gültige Signcryption rät und wir diese abweisen, werden wir bei der Erfolgswahrscheinlichkeit betrachten.

Falls $H_2\left(m', R'\right)$ definiert wurde und die Signcryption gültig ist, existiert ein Eintrag $(m_i, R_i, h_i, c_i, \gamma_i)$ in der Liste L_2 mit:

- $R_i = e\left(T, S_{ID_R}\right)$

- $m_i = c \oplus H_3(R_i)$
- $h_i = H_2(m_i, R_i)$
- $R_i = e(S, P_{ID_S}) \cdot \widehat{G}^{-h_i}$

Es können keine zwei solche Einträge existieren, denn diese sind durch R_i und m_i nach Konstruktion der Simulation eindeutig bestimmt.

Die oberen Gleichungen sind äquivalent zu:

- $c = m_i \oplus H_3(R_i)$
- $h_i = H_2(m_i, R_i)$
- $R_i \cdot \widehat{G}^{h_i} = e(S, P_{ID_S})$
- $e(T, S_{ID_R}) = e(S, P_{ID_S}) \cdot \widehat{G}^{-h'}$

Diese Gleichungen werden bei der Simulation der Decrypt/Verify-Anfragen überprüft, denn

- Nach Konstruktion der Liste L_2 gilt $c_i = m_i \oplus H_3(R_i)$ und bei der Simulation wird $c \stackrel{?}{=} c_i$ überprüft.
- Nach Konstruktion der Liste L_2 gilt immer $h_i = H_2(m_i, R_i)$.
- Die dritte Gleichung wird durch $e(S, P_{ID_S}) \stackrel{?}{=} \gamma_i$ überprüft.
- Die letzte Gleichung ist äquivalent zu

  $$e(T, S_{ID_S}) = e(S, P_{ID_R}) \cdot e(\psi(P_{ID_R}), S_{ID_S})^{-h'}$$

  Wir zeigen das im Lemma 28 am Ende des Beweises. Diese Gleichung wird im Schritt 3 der Simulation überprüft.

Also finden wir den richtigen Eintrag in der Liste und können somit die Anfrage korrekt beantworten.

□

Challenge:

A wählt zwei Nachrichten m_0 und m_1 und zwei Identitäten (ID_S, ID_R). Wenn $ID_R \neq ID_d$ gilt, bricht B ab und gibt $\perp$ aus. Sonst wähle $\xi \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, $c \stackrel{R}{\leftarrow} \{0,1\}^n$ und $S \stackrel{R}{\leftarrow} \mathbb{G}_1$. Berechne $T = \widehat{P}^{-\xi}$ und gebe $\sigma^* = (c, S, T)$ dem Angreifer.

Zeige, dass für den dazugehörigen Wert $x \in \mathbb{Z}_p^*$ mit $R = \widehat{G}^x$ gilt: $x = \xi/\alpha$.

- Für $s = -\alpha - w_0$ gilt:

  $$T = \widehat{P}^{-\xi} = \widehat{P}^{-\alpha \cdot x} = \widehat{P}^{(w_0+s)\cdot x}$$

- Andererseits gilt für T nach Konstruktion des Verfahrens und bei $H_1(ID_R) = w_0$:

$$T = \widehat{P}^{(w_0+s)\cdot x}$$

Somit gilt für den zu dem Wert T dazugehöriger Wert R:

$$R = \widehat{G}^x = \widehat{G}^{\xi/\alpha}$$

A kann nicht erkennen, dass σ^* keine echte Signcryption für eine von seinen Nachrichten ist, es sei denn A kennt mindestens einen der Werte $H_2(m_i, R)$ oder $H_3(R)$ für $R = \widehat{G}^{\xi/\alpha}$. Ohne $H_2(m_i, R)$ kann der Angreifer nicht feststellen, ob die Abhängigkeiten zwischen S und T korrekt sind (wie bei Decrypt/Verify-Anfragen schon erklärt) und ohne $H_3(R)$ kann der Angreifer nicht feststellen, ob der Wert c korrekt ist.

Phase 2 und Guess:

In der zweiten Phase werden die Anfragen von A wie in Phase 1 beantwortet und seine Ausgabe ignoriert. Die dem Angreifer vorgelegte Signcryption können wir zwar nicht selbst entschlüsseln, dieser darf die Anfrage aber auch gar nicht stellen. Somit brechen wir in dem Fall einfach ab.

Berechnung der Lösung des q-BDHI-Problems

Um die Lösung für das Problem zu berechnen, wählt B zufällig einen Eintrag (m, R, h_2, c, γ) oder $(R, \cdot)$ aus den Listen L_2 bzw. L_3. Die Liste L_2 hat nach Konstruktion nicht mehr als $q_{H_2} + q_{SE}$ Einträge und die Liste L_3 nicht mehr als $q_{H_2} + q_{H_3} + q_{SE}$ Einträge. Mit Wahrscheinlichkeit $\frac{1}{2\cdot q_{H_2} + q_{H_3} + 2\cdot q_{SE}}$ beinhaltet der gewählte Eintrag das richtige Element $R = \widehat{G}^{\frac{\xi}{\alpha}} = e\left(\widehat{P}, \widehat{Q}\right)^{\frac{\xi}{\alpha}} = e(P,Q)^{\frac{f(\alpha)^2\cdot\xi}{\alpha}}$. Falls das der Fall ist, können wir das gesuchte Element $\gamma^* = e(P,Q)^{\frac{1}{\alpha}}$ für die Lösung des q-BDHI-Problems wie folgt berechnen:

$$\gamma^* = \left[R^{\frac{1}{\xi}} \cdot e\left(\widehat{P} \cdot P^{c_0}, \prod_{i=0}^{q-2} \left(Q^{\alpha^i}\right)^{c_{i+1}} \right)^{-1} \right]^{\frac{1}{c_0^2}}$$

Um diese Behauptung zu beweisen zeige, dass folgende Gleichung gilt:

$$(\gamma^*)^{c_0^2} \cdot e\left(\widehat{P} \cdot P^{c_0}, \prod_{i=0}^{q-2} \left(Q^{\alpha^i}\right)^{c_{i+1}} \right) = e(P,Q)^{\frac{f(\alpha)^2}{\alpha}} = R^{\frac{1}{\xi}}$$

Aus dieser Gleichung folgt durch einfache Umformung die Behauptung:

$$e\left(\widehat{P}\cdot P^{c_0},\prod_{i=0}^{q-2}\left(Q^{\alpha^i}\right)^{c_{i+1}}\right)=e\left(P^{f(\alpha)+c_0},Q^{\sum_{i=0}^{q-2}c_{i+1}\alpha^i}\right)=e\left(P,Q\right)^{(f(\alpha)+c_0)\cdot\sum_{i=0}^{q-2}c_{i+1}\alpha^i}$$

Weiterhin gilt:

$$\sum_{j=0}^{q-2}c_{j+1}\alpha^j=\frac{1}{\alpha}\cdot\sum_{j=0}^{q-2}c_{j+1}\alpha^{j+1}=\frac{1}{\alpha}\cdot\sum_{j=1}^{q-1}c_j\alpha^j=\frac{1}{\alpha}\cdot\left(f\left(\alpha\right)-c_0\right)=\frac{f\left(\alpha\right)-c_0}{\alpha}$$

Für den linken Teil der obigen Gleichung bekommen wir also:

$$(\gamma^*)^{c_0^2}\cdot e\left(\widehat{P}\cdot P^{c_0},\prod_{i=0}^{q-2}\left(Q^{\alpha^i}\right)^{c_{i+1}}\right)=e\left(P,Q\right)^{\frac{c_0^2}{\alpha}}\cdot e\left(P,Q\right)^{(f(\alpha)+c_0)\cdot\frac{f(\alpha)-c_0}{\alpha}}=$$

$$e\left(P,Q\right)^{\frac{c_0^2}{\alpha}+\frac{f(\alpha)^2}{\alpha}-\frac{c_0^2}{\alpha}}=e\left(P,Q\right)^{\frac{f(\alpha)^2}{\alpha}}$$

Somit wird der gesuchte Wert $\gamma^*=e\left(P,Q\right)^{\frac{1}{\alpha}}$ korrekt berechnet.

Erfolgswahrscheinlichkeit

Definiere die Ereignisse:

- E_0 - A ist erfolgreich. Wir nehmen o.B.d.A an, dass $P\left(E_0\right)>\frac{1}{2}$ gilt.[3] Somit gilt nach Definition des Spiels $P\left(E_0\right)-\frac{1}{2}\geq\varepsilon_0$.
- E_1 - A wählt die Identität ID_d als Identität des Empfängers in der Challenge-Phase.
- E_2 - A stellt keine Extract-Anfrage von ID_d und keine verbotene Sign/Encrypt-Anfrage in der Phase 2.
- E_3 - Es kommt zu keiner Kollision durch die Definition der H_2-Werte bei Sign/Encrypt-Anfragen.
- $E=E_1\wedge E_2\wedge E_3$ - B bricht die Berechnung nicht ab (nach Konstruktion).
- E_4 - A stellt bei seiner Berechnung mindestens eine der Anfragen $H_2\left(m_i,R\right)$ oder $H_3\left(R\right)$ für $R=\widehat{G}^{\xi/\alpha}$ oder diese wurden während einer der Sign/Encrypt-Anfrage definiert.
- E_5 - B weist keine gültige Signcryption bei Decrypt/Verify-Anfragen ab.
- E_6 - B rät die richtige Anfrage mit $R=\widehat{G}^{\xi/\alpha}$.

[3] Für jeden Algorithmus A kann man einen Algorithmus A' angeben, der das gleiche tut, wie A und eine andere Vermutung, als A ausgibt.

Zeichne bei der Analyse alle Wahrscheinlichkeiten für Ereignisse, die im in der Definition angegebenen Spiel betrachtet werden, mit Pr^* aus. Die Wahrscheinlichkeit für E_0 in diesem Spiel kann wie folgt abgeschätzt werden:

$$Pr^*(E_0) = Pr^*(E_0 | E_4) \cdot Pr^*(E_4) + Pr^*\left(E_0 \left| \overline{E_4}\right.\right) \cdot Pr^*\left(\overline{E_4}\right) \leq$$

$$Pr^*(E_0 \wedge E_4) + \frac{1}{2} \cdot 1$$

denn wenn der Angreifer die Anfragen für R nicht stellt, kann es nichts aus der Signcryption lernen. Daraus folgt:

$$Pr^*(E_0 \wedge E_4) \geq Pr^*(E_0) - \frac{1}{2} \geq \varepsilon_0$$

und somit

$$Pr^*(E_4) \geq Pr^*(E_0 \wedge E_4) \geq \varepsilon_0$$

Nun betrachten wir die Simulation. Nehme o.B.d.A. an, dass der Angreifer nie die verbotenen Anfragen stellt. (Man kann aus jedem Angreifer einen konstruieren, der an Stelle von solchen Anfragen abbricht.)

Unter dieser Zusatzannahme folgt, dass wenn E_1 auftritt, dass dann auch E_2 auftritt.

Weiterhin gilt nach Konstruktion:

Betrachten wir nun die Simulation bis zum ersten Zeitpunkt, an dem eine Anfrage mit dem Wert $R = \widehat{G}^{\xi/\alpha}$ gestellt wird. Dann gilt:

$$Pr(E_4 | E_1 \wedge E_3 \wedge E_5) =$$

$$Pr(E_4 \mid \text{Simulation ist perfekt bis das Ereignis } E_4 \text{ zum ersten mal auftritt}) =$$

$$Pr^*(E_4)$$

denn die rechte Seite der Gleichung gilt, da der Angreifer bis zu diesem Zeitpunkt die Simulation von dem in der Definition angegebenen Spiel nicht unterscheiden kann. Die linke Seite gilt, weil wir die Anfragen des Angreifers wie im Verfahren definiert beantworten, wenn $E_1 \wedge E_3 \wedge E_5$ bis zu dem angegebenen Zeitpunkt gilt. Eine formale Analyse von solchen Fällen kann z.B. in Folgerung 1 von [BF01] nachgeschlagen werden.

Betrachte nun unabhängige Ereignisse E_1, E_3, E_5, E_6 ($2^k - 1$ schätzen wir direkt mit 2^k ab).

Es gilt, da d zufällig, gleichverteilt aus $\{1, ..., q\}$ gewählt wird:

$$Pr(E_1) = \frac{1}{q}$$

Die Wahrscheinlichkeit für eine Kollision bei einer Sign/Encrypt-Anfrage ist höchstens $\frac{q_{SE}+q_{H_2}}{2^k}$, da maximal $q_{SE} + q_{H_2}$ Werte der Hashfunktion H_2 definiert werden. Somit gilt für das ganze Spiel:

$$Pr\left(E_3\right) \geq 1 - q_{SE} \cdot \frac{q_{SE} + q_{H_2}}{2^k}$$

Die Wahrscheinlichkeit, dass keine gültige Decrypt/Verify-Anfrage zurückgewiesen wird, ist über das ganze Spiel:

$$Pr\left(E_5\right) \geq 1 - \frac{q_{DV}}{2^k}$$

Insgesamt werden weiterhin höchstens $\frac{1}{2 \cdot q_{H_2} + q_{H_3} + 2 \cdot q_{SE}}$ Werte für die Hashfunktionen H_2- und H_3 definiert. Falls E_4 gilt, rät B die richtige Anfrage mit Wahrscheinlichkeit:

$$Pr\left(E_6 \,|E_4\right) \geq \frac{1}{2 \cdot q_{H_2} + q_{H_3} + 2 \cdot q_{SE}}$$

Nach Konstruktion des Algorithmus berechnen wir die Lösung des Problems, wenn das Ereignis $E_1 \wedge E_2 \wedge E_3 \wedge E_4 \wedge E_5 \wedge E_6$ auftritt. Die Erfolgswahrscheinlichkeit des Algorithmus ist also:

$$\varepsilon' = Pr\left(E_1 \wedge E_2 \wedge E_3 \wedge E_4 \wedge E_5 \wedge E_6\right) =$$

$$Pr\left(E_1 \wedge E_3 \wedge E_4 \wedge E_5\right) \cdot Pr\left(E_6 \,|E_4\right) =$$

$$Pr\left(E_4 \,|E_1 \wedge E_3 \wedge E_5\right) \cdot Pr\left(E_1 \wedge E_3 \wedge E_5\right) \cdot Pr\left(E_6 \,|E_4\right) =$$

$$Pr\left(E_4 \,|E_1 \wedge E_3 \wedge E_5\right) \cdot Pr\left(E_1\right) \cdot Pr\left(E_3\right) \cdot Pr\left(E_5\right) \cdot Pr\left(E_6 \,|E_4\right) \geq$$

$$\frac{\varepsilon_0}{q \cdot (2 \cdot q_{H_2} + q_{H_3} + 2 \cdot q_{SE})} \cdot \left(1 - q_{SE} \cdot \frac{q_{SE} + q_{H_2}}{2^k}\right) \cdot \left(1 - \frac{q_{DV}}{2^k}\right)$$

Laufzeit

Betrachte die Laufzeiten für die jeweilige Teile des Algorithmus.

- Die Setup-Phase entspricht weitgehend der Berechnung des IG-Algorithmus in Lemma 22. Die Laufzeit wird durch Schritt 3 bestimmt, durch die Berechnung von den Elementen U_i. Insgesamt braucht der dritte Schritt $\mathcal{O}\left[q^2 \cdot \left(\tau_{exp_{\mathbb{G}_2}} + \tau_{mult_{\mathbb{G}_2}}\right)\right]$ Zeit. Da die Berechnung von Potenzen aufwändiger ist, bekommen wir insgesamt Laufzeit von:

$$\mathcal{O}\left(q^2 \cdot \tau_{exp_{\mathbb{G}_2}}\right)$$

- In der Phase 1 und 2 sind die Laufzeiten für die H_2-, Sign/Encrypt- und Decrypt/Verify-Anfragen entscheidend. Bei den beiden letzten werden konstant viele Paarungen ausgerechnet und konstant viele andere Operationen in den Gruppen. Bei H_2-Anfragen wird eine Multiplikation und eine Potenz ausgerechnet. Dazu kommt noch die Laufzeit von A. Insgesamt benötigt man also:
$$t + \mathcal{O}\left((q_{SE} + q_{DV}) \cdot \tau_p\right) + \mathcal{O}\left(q_{H_2} \cdot \tau_{exp_{\mathbb{G}_T}}\right)$$
- Die Challenge-Phase und die Berechnung der Lösung brauchen asymptotisch weniger Zeit und können vernachlässigt werden.

Daraus folgt die im Theorem angegebene Laufzeit von B.

□

Eigenschaft von gültigen Signcryptions

Im dem folgenden Lemma zeigen wir wie die Werte S und T einer gültigen Signcryption zusammenhängen.

Lemma 28. *Für jede gültige Signcryption $\varsigma = (c, S, T)$ mit den Werten R, m und h, wie im Decrypt/Verify-Algorithmus definiert, gilt:*

$$e\left(T, S_{ID_S}\right) = e\left(S, P_{ID_R}\right) \cdot e\left(\psi\left(P_{ID_R}\right), S_{ID_S}\right)^{-h}$$

Beweis. Nach Konstruktion des Decrypt/Verify-Algorithmus hängen die Werte S und T einer gültigen Signcryption über den Wert R zusammen. Der Wert R wird nämlich als der linke Teil der folgenden Gleichung ausgerechnet und dann bei der Überprüfung der Gültigkeit mit dem rechten Teil verglichen:

$$e\left(T, S_{ID_R}\right) = R = e\left(S, P_{ID_S}\right) \cdot G^{-h}$$

Durch einsetzen der Schlüssel und einfache Umformung erhalten wir:

$$e\left(T, Q^{\frac{1}{H_1(ID_R)+s}}\right) = e\left(S, Q^{H_1(ID_S)+s}\right) \cdot e\left(P, Q\right)^{-h} \Big| \wedge \frac{H_1\left(ID_R\right) + s}{H_1\left(ID_S\right) + s}$$

$$e\left(T, Q^{\frac{1}{H_1(ID_S)+s}}\right) = e\left(S, Q^{H_1(ID_R)+s}\right) \cdot e\left(P^{H_1(ID_R)+s}, Q^{\frac{1}{H_1(ID_S)+s}}\right)^{-h} \Longleftrightarrow$$

$$e\left(T, S_{ID_S}\right) = e\left(S, P_{ID_R}\right) \cdot e\left(\psi\left(P_{ID_R}\right), S_{ID_S}\right)^{-h}$$

Somit haben wir gezeigt, dass für jede gültige Verschlüsselung die Behauptung gilt.

□

Analyse der Wahrscheinlichkeitsverteilung der Simulation von Sing/Encrypt-Anfragen

Im folgenden Lemma zeigen wir, dass die Wahrscheinlichkeitsverteilungen über die Werte (c, h, h_3, R, S, T) des Verfahrens und der Simulation der Sign/Encrypt-Anfragen identisch sind.

Lemma 29. *Die Verteilungen δ der BLMQ-IBSC und δ' der Simulation der Sign/Encrypt-Anfragen über die Werte (c, h, h_3, R, S, T) sind für eine gegebene Nachricht m, einen gegebenen Masterschlüssel s und gegebene Senderidentität ID_S und Empfängeridentität ID_R identisch.*

Wir zeigen erst, dass bei beiden Verteilungen der Wert R durch T eindeutig bestimmt sind. Dann betrachten wir die Verteilung über die Werte h, h_3 und T und zeigen, dass diese gleich sind und dass h und h_3 durch T eindeutig bestimmt wird. Die Verteilungen über h und h_3 müssen wir betrachten, da diese erst während der Berechnung als $H_2(m, R)$ bzw. H_3 definiert werden. Am Ende zeigen wir, dass c und S eindeutig durch T bestimmt sind. Somit wird gezeigt, dass die Verteilung der Simulation identisch zu der Verteilung des Verfahrens ist.

Beweis. Beachte, dass wir bei der Betrachtung der Sing/Encrypt-Anfragen schon gezeigt haben, dass die Simulation eine gültige Signcryption berechnet.

- Betrachte zuerst den Wert R. Bei der Korrektheit des BLMQ-IBSC-Verfahrens haben wir gezeigt, dass für jede gültige Signcryption folgendes gilt:

  $$R = e(T, S_{ID_R})$$

  Somit ist R sowohl bei der Simulation als auch beim BLMQ-IBSC-Verfahren bei einem gegebenen Wert T gleich.

- Betrachte die Verteilungen über die Elementen h, h_3 und T.

  Die Verteilungen δ von BLMQ-IBSC:

  $$\delta = \left\{ (h, h_3, T) \;\middle|\; \begin{array}{c} x \overset{R}{\leftarrow} \mathbb{Z}_p^* \\ T = \psi(P_{ID_R})^x \in \mathbb{G}_1 \backslash \{1\} \\ h = H_2(m, G^x) \\ h_3 = H_3(G^x) \end{array} \right\}$$

  - Der Wert x wird zufällig, gleichverteilt aus $\mathbb{Z}_p^*$ gewählt und $\psi(P_{ID_R})$ ist ein Generator aus der Gruppe $\mathbb{G}_1$ mit Primordnung p. Daraus folgt, dass T gleichverteilt aus $\mathbb{G}_T \backslash \{1\}$ ist.
  - Die Werte h und h_3 sind im Zufallsorakel-Modell zufällige, gleichverteilte Elemente aus $\mathbb{Z}_p^*$.

Die Verteilung δ' der Simulation:

$$\delta' = \left\{ (h, h_3, T) \;\middle|\; \begin{array}{c} h, h_3 \overset{R}{\leftarrow} \mathbb{Z}_p^* \\ x \overset{R}{\leftarrow} \mathbb{Z}_p \\ T = \psi\left(P_{ID_S}\right)^x \cdot \psi\left(P_{ID_R}\right)^{-h} \\ T \neq 1 \end{array} \right\}$$

- Bei der Simulation werden die Werte h und h_3 zufällig, gleichverteilt aus $\mathbb{Z}_p^*$ gewählt.
- Den Wert T kann man darstellen als

  $$T = P^{x \cdot k_1 - h \cdot k_2}$$

  für feste $k_1, k_2 \in \mathbb{Z}_p^*$. Weiterhin ist h schon festgelegt und somit ist $-h \cdot k_2$ ein festes Element aus $\mathbb{Z}_p^*$. Der Wert x wird zufällig, gleichverteilt aus $\mathbb{Z}_p$ gewählt. Unter der Voraussetzung, dass der Fall $T = 1$ nicht auftritt folgt unmittelbar, dass $x \cdot k_1 - h \cdot k_2$ ein zufälliges gleichverteiltes Element aus $\mathbb{Z}_p^*$ ist und somit T ein zufälliges, gleichverteiltes Element aus $\mathbb{G}_1 \backslash \{1\}$ ist.

  Weiterhin werden die Werte h und h_3 während der Simulation als $H_2(m, R)$ bzw. $H_3(R)$ definiert. Somit sind die Verteilungen über die Elemente gleich und die Werte h und h_3 bei gegebener Nachricht m und gegebenen Schlüsseln eindeutig durch T bestimmt, und zwar als $h = H_2(m, e(T, S_{ID_R}))$ und $h_3 = H_3(e(T, S_{ID_R}))$.

- Die Berechnung von c bei der Simulation entspricht der Berechnung $c := m \oplus h_3$ im BLMQ-IBSC-Verfahren. Somit ist dieser für gegebene m und T auch gleich.
- Weiterhin haben wir im Lemma 28 gezeigt, dass für eine gültige Signcryption gilt:

  $$e(T, S_{ID_S}) = e(S, P_{ID_R}) \cdot e\left(\psi\left(P_{ID_R}\right), S_{ID_S}\right)^{-h}$$

  Aus dieser Gleichung folgt, dass S durch T und h bei gegebenen Schlüsseln eindeutig bestimmt ist.

Somit haben wir gezeigt, dass die Verteilungen über h, h_3 und T identisch sind und dass alle Elemente durch T eindeutig bestimmt sind. Somit ist die Verteilung der Simulation identisch zu der Verteilung des Verfahrens.

□

Aus dem Theorem 27 folgt das folgende Korollar.

Korollar 30. Unter der Annahme, dass das q-BDHI-Problem in den Abbildungsgruppen hart ist, ist das BLMQ-IBSC-Verfahren *IND-IBSC-CCA-sicher*.

4.3 Die Sicherheitsanalyse II

Der Sicherheitsbeweis dafür, dass das IBSC-Verfahren *existentiell fälschungssicher* ist, läuft analog zu den schon vorgestellten Sicherheitsbeweisen für IBS- und IBSC-Verfahren. Deswegen wird in diesem Abschnitt nur kurz erklärt, wie der Beweis funktioniert.

Wir wollen das q-SDH-Problem auf den CMIDA-Angreifer für IBSC reduzieren:

$$\text{q-SDH} \leq \text{CMIDA}$$

Diese Reduktion wird wie bei IBS-Verfahren in zwei Schritten durchgeführt. Im ersten Schritt reduzieren wir den CMIDA-Angreifer F_0 auf den CMA-Angreifer F_1:

$$\text{CMA} \leq \text{CMIDA}$$

Der Beweis ist analog zum Lemma 21. F_1 rät am Anfang die Nummer der H_1-Anfrage, bei der F_0 den Hashwert der Senderidentität ID_S anfragt, für die er später die Fälschung ausgibt. Alle Anfragen, die diese Identität betreffen ersetzt F_1 durch die ihm vorgegebene Identität. Falls F_0 genau für ID_S die Fälschung berechnet, dann berechnet er tatsächlich eine Fälschung für die vorgegebene Identität und somit gewinnt F_1 sein Spiel.

Im zweiten Schritt wird analog zum Lemma 22 die folgende Reduktion durchgeführt:

$$\text{q-SDH} \leq \text{CMA}$$

Diese Reduktion unterscheidet sich kaum von der entsprechenden Reduktion für IBS-Verfahren. Die Struktur ist komplett analog. Es wird wieder das allgemeine Forking-Lemma angewendet.

Ein kleiner Unterschied beim IG-Algorithmus liegt daran, dass die geheimen Schlüssel bei $IBSC$ aus der Gruppe $\mathbb{G}_2$ sind. D.h. dass auf den entsprechenden Stelle die Funktion ψ nicht angewendet wird. Weiterhin werden bei der Simulation auch die öffentliche Schlüssel gebraucht, die analog zu P_0 ausgerechnet werden können.

Die größte Änderung im Vergleich zu IBS-Verfahren liegt im Simulationsalgorithmus S_A. Allerdings entspricht dieser der Simulation des Angreifers im vorherigen Theorem 27. Dabei entspricht die Identität ID_d in dem Theorem der Identität des Senders, für die der Angreifer die Signcryption berechnen muss. Wenn der Angreifer eine gültige Signcryption für die vorgegebene Identität des Senders ID^* und eine von ihm gewählte Identität des Empfängers ID_R ausgibt, kann S_A daraus eine gültige Signatur für ID^* berechnen, denn der geheime Schlüssel des Empfängers ist nach Irreflexivitätseigenschaft bekannt. S_A berechnet also die gleiche Ausgabe wie bei IBS-Verfahren.

Die Erfolgswahrscheinlichkeit von S_A entspricht weitgehend der Erfolgswahrscheinlichkeit von A. Nur Kollisionen bei der Definition der H_2-Werten in Sign/Encrypt-Anfragen und Abweisen von gültigen Signcryptions bei den Decrypt/Verify-Anfragen müssen berücksichtigt werden.

Die Anwendung des Forking-Lemmas und die Berechnung einer Lösung für das q-SDH-Problem funktionieren dann also analog, wobei wieder berücksichtigt werden muss, dass die geheimen Schlüssel aus der anderen Gruppe sind, als bei IBS-Verfahren.

5 Das Sakai-Kasahara-IBE

In diesem Kapitel wird das Sakai-Kasahara-Verschlüsselungsverfahren [SK03] präsentiert und darüber diskutiert, wie sich das Signcryptionverfahren so anpassen lässt, dass dieses sich als Sakai-Kasahara-Verschlüsselungsverfahren anwenden lässt. In diesem Kapitel wird davon ausgegangen, dass die Sicherheitskonzepte für Verschlüsselungsverfahren bekannt sind.

5.1 Das Sakai-Kasahara-Verschlüsselungsverfahren

In der Praxis will man oft nicht für jede Funktionalität ein neues Verfahren implementieren. So kann man sich gut vorstellen, dass in einem System, in dem ein Signcryptionverfahren eingesetzt wird, auch einfache Verschlüsselung benötigt wird. Dabei würde man also gerne Teile des Signcryptionverfahrens wiederverwenden. Als ein Teil dieser Arbeit sollte analysiert werden, ob das vorgestellte Signcryptionverfahren so angepasst werden kann, dass eine einfache Verschlüsselung für eine Nachricht berechnet werden kann und ob man dabei an Effizienz im Vergleich zum Signcryptionverfahren gewinnt. Das Verschlüsselungsverfahren sollte CCA-sicher sein.

Es hat sich herausgestellt, dass einfaches Weglassen des Parameters S der Signcryption zwar zu einer korrekten Verschlüsselung führt, diese ist allerdings nicht CCA-sicher. Der Angreifer kann nämlich jede Verschlüsselung c entschlüsseln, indem er z.B. eine Decrypt-Anfrage für den Wert $c \oplus 1^n$ stellt und aus dem Ergebnis m' die ursprüngliche Nachricht als $m' \oplus 1^n$ berechnet. Der Wert S in der Signcryption ist also nicht nur ein Teil der Signatur. Dieser Wert spielt eine entscheidende Rolle in der Sicherheit des Verfahrens und kann nicht einfach weggelassen werden.

Es ist aber trotzdem möglich auf der Basis des vorgestellten IBSC-Verfahrens eine CCA-sichere Verschlüsselung zu konstruieren. Und zwar ist das IBSC-Verfahren sehr ähnlich zum Sakai-Kasahare-IBE-Verfahren (im Folgenden SK-IBE).[1] Man kann also große Teile des IBSC-Verfahrens wieder verwenden. In diesem Abschnitt wird das Verfahren präsentiert und darauf eingegangen, was zusätzlich zum IBSC-Verfahren notwendig ist und welche Änderungen vorgenommen werden müssen.

Definition 31. Das SK-IBE-Verfahren Π = (Setup, Extract, Encrypt, Decrypt) ist definiert als:

[1] „Identity-based encryption"

- **Setup:** Entspricht dem Setup-Algorithmus von IBSC mit dem Unterschied, dass der systemweite öffentliche Schlüssel P_{pub} aus der Gruppe $\mathbb{G}_1$ als P^s berechnet wird. Weiterhin werden andere Hashfunktionen benutzt (die Nummerierung wurde im Vergleich zur Originaldefinition an IBSC angepasst).
  - $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$
  - $H_2 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_p^*$
  - $H_3 : \mathbb{G}_T \to \{0,1\}^n$
  - $H_4 : \{0,1\}^n \to \{0,1\}^n$

  Die öffentlichen Parameter sind somit:

$$params := \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, G, e, \psi, P_{pub}, H_1, H_2, H_3, H_4\}$$

- **Extract:** Entspricht dem Extract-Algorithmus von IBSC. Der „öffentliche" Schlüssel wird als $P_{ID} = P^{H_1(ID)+s}$ berechnet.
- **Encrypt:** Gegeben sei eine Nachricht $m \in \{0,1\}^n$ und der „öffentliche" Schlüssel P_{ID} eines Benutzers. Führe folgende Schritte durch:
  1. Wähle $x \xleftarrow{R} \{0,1\}^n$ und berechne $h := H_2(m, x)$.
  2. Berechne $R := G^h$.
  3. Berechne die Verschlüsselung als:

$$c := \left(P_{ID}^h, x \oplus H_3(R), m \oplus H_4(x)\right)$$

- **Decrypt**: Gegeben sei die Verschlüsselung $c = (U, V, W)$ und der geheime Schlüssel S_{ID} des Benutzers. Führe folgende Schritte durch:
  1. Berechne $R := e(U, S_{ID})$ und $x := V \oplus H_3(R)$.
  2. Berechne $m := W \oplus H_4(x)$ und $h = H_2(m, x)$.
  3. Falls die folgende Gleichung gilt:

$$U \stackrel{?}{=} P_{ID}^h$$

     gebe die Nachricht m aus. Sonst gebe $\perp$ aus.

Beweis. [Korrektheit] Zeige die Korrektheit des Verfahrens in zwei Schritten. Erst zeigen wir, dass der Decrypt-Algorithmus korrekt die Nachricht entschlüsselt und dann, dass die Gültigkeitsüberprüfung erfolgreich durchläuft.

- Entschlüsselung: Falls die Verschlüsselung korrekt ist, kann mit dem geheimen Schlüssel S_{ID} der Wert R korrekt berechnet werden:

$$e\left(U, S_{ID}\right) = e\left(P_{ID}^{h}, S_{ID}\right) = G^{h} = R$$

  Mit einem korrekten Wert R kann x und daraus m und h korrekt berechnet werden.

- Gültigkeit: Die Gültigkeitsüberprüfung funktioniert dann offensichtlich, da der Wert h korrekt berechnet ist.

Somit ist das Verfahren korrekt.

□

Sicherheitsbeweis für SK-IBE

Der Sicherheitsbeweis dafür, dass das SK-IBE-Verfahren CCA-sicher ist, wurde in [CC05] durchgeführt. An dieser Stelle wird die Idee des Beweises kurz erläutert. Dabei wird davon ausgegangen, dass die entsprechenden Sicherheitsdefinitionen für IBE-Verfahren bekannt sind.

Der Beweis besteht aus drei Schritten. Im ersten Schritt wird eine Reduktion durchgeführt, bei der ein nicht identitätsbasiertes IBE-Verfahren $BasicPub^{hy}$ aus dem SK-IBE-Verfahren konstruiert wird und ein IND-CCA-Angreifer für dieses Verfahren auf einen IND-ID-CCA-Angreifer für SK-IBE-Verfahren reduziert wird:

$$\text{IND-CCA-BasicPub}^{hy} \leq \text{IND-ID-CCA-SK-IBE}$$

Dadurch müssen in weiteren Schritten keine Identitätsbasierte Verfahren mehr betrachtet werden.

Man beachte, dass diese Idee bei dem Sicherheitsbeweis für IBS-Verfahren angewendet werden könnte. In diesem Fall könnte man auch das in der Originalarbeit angewendete Forking-Lemma benutzen, wenn erst eine ähnliche Reduktion auf ein Standardsignaturverfahren durchgeführt wäre. In dieser Arbeit wurde aber das allgemeine Forking-Lemma angewendet, da diese den Beweis besser verständlich macht und man spätestens beim Signcryptionverfahren das Forking-Lemma aus der Originalarbeit nicht mehr anwenden könnte.

Im zweiten Schritt der Reduktion definiert man ein weiteres IBE-Verfahren $BasicPub$ und reduziert einen schwächeren IND-CPA-Angreifer für dieses Verfahren auf einen IND-CCA-Angreifer für $BasicPub^{hy}$:

$$\text{IND-CPA-BasicPub} \leq \text{IND-CCA-BasicPub}^{hy}$$

Der Beweis für diesen Reduktionsschritt basiert auf einem Theorem zur Fujisaki-Okamoto-Transformation [FO99], denn das Sakai-Kasahara-IBE auf dieser basiert.

Im dritten Schritt wird dann das q-BDHI-Problem auf den IND-CPA-Angreifer für *BasicPub* reduziert:

$$q - BDHI \leq \text{IND-CPA-BasicPub}$$

Diese Reduktion ist ähnlich zu den Reduktionen, die wir durchgeführt haben. Man berechnet aus der Instanz des q-BDHI-Problems die öffentliche Parameter des Verfahrens auf gleiche Art und Weise, wie auch in dieser Arbeit und simuliert dann die Anfragen des Angreifers.

Insgesamt zeigt man also die folgende Reduktion:

$$q - BDHI \leq \text{IND-ID-CCA-SK-IBE}$$

und somit ist das SK-IBE-Verfahren IND-ID-CCA-sicher, wenn das q-BDHI-Problem hart in den bilinearen Abbildungsgruppen ist.

Anpassung des IBSC-Verfahrens

Aus der Definition des SK-IBE-Verfahrens wird schon deutlich, dass die Setup- und Extract-Algorithmen kaum angepasst werden müssen.

Beim Setup-Algorithmus kann der systemweite öffentliche Schlüssel P_{pub} aus Q_{pub} als $\psi(Q_{pub})$ berechnet werden. Somit muss dieser Algorithmus kaum angepasst werden. Außerdem werden vier Hashfunktionen verwenden, wobei die Funktionen H_1 und H_3 denen aus IBSC-Verfahren entsprächen. Bei der H_2-Hashfunktion von IBSC ist der zweite Parameter aus der Gruppe $\mathbb{G}_T$. Dieser wird in der Praxis sowieso erst auf ein Element aus $\{0,1\}^n$ gehasht. Somit benötigt man im Vergleich zum IBSC-Verfahren nur eine weitere Funktion H_4.

Der Extract-Algorithmus muss nicht angepasst werden, wenn man bedenkt, dass der Element P_{ID} nur dafür definiert wurde, um die Beschreibung der Verfahren zu erleichtern.

Die Encrypt- und Decrypt-Algorithmen sind anders, als beim Signcryptionverfahren. Allerdings sind auch hier die Operationen die gleichen, wie bei den anderen Verfahren.

Insgesamt muss man also nicht so viele Anpassungen vornehmen, um das SK-IBE anzuwenden. Dieses Verfahren ist weiterhin sehr effizient. Bei der Verschlüsselung werden nur zwei Potenzen ausgerechnet und sonst nur Hashfunktionen ausgewertet und XOR-Operationen durchgeführt. Bei der Entschlüsselung wird eine einzige Paarung und eine Potenz berechnet.

Somit lässt sich wohl kein effizienteres CCA-sicheres Verschlüsselungsverfahren für das vorgestellte IBSC-Verfahren finden, für das nur solche kleine Änderungen an den Algorithmen notwendig sind.

6 Fazit

In dieser Arbeit wurden im Wesentlichen vier identitätsbasierte Verfahren vorgestellt. Dabei wurde der Fokus darauf gelegt die vorgestellten Beweise gründlich zu analysieren und keine Beweisschritte wegzulassen, wie es in der Originalarbeit der Fall war. Insgesamt kann man sagen, dass die Originalarbeit an vielen Stellen große Lücken hatte, die geschlossen werden mussten. Angefangen schon bei den Sicherheitsdefinitionen für identitätsbasierte Signcryptions, die unpräzise und teilweise auch inkorrekt waren. Aus diesem Grund mussten die Sicherheitsdefinitionen korrigiert werden und sogar daraufhin kleine Änderungen in der Definition des IBSC-Verfahrens gemacht werden. Der Verify-Algorithmus wurde dabei definiert.

Der Beweis für IBS-Verfahren musste komplett überarbeitet werden, denn die Autoren haben ein Forking-Lemma (Lemma 13) aus [PS00] angewendet, das nicht für identitätsbasierte Verfahren, sondern für Standardsignaturverfahren formuliert und bewiesen wurde. Wenn man das Lemma an dieser Stelle anwenden wollte, müsste man noch einen Zwischenschritt in den Beweis einbauen und erst eine Reduktion auf einen Angreifer für ein auf Basis des IBS-Verfahrens konstruierten Standardsignaturverfahren durchführen. Eine ähnliche Reduktion wurde z.B. im Lemma 1 von [CC05] für das SK-IBE-Verfahren gemacht. In dieser Arbeit habe ich mich aber dafür entschieden das allgemeine Forking-Lemma anzuwenden, das den Beweis besser verständlich macht und spätestens bei den Signcryptionverfahren unverzichtbar ist.

Weiterhin gehen die Autoren der Originalarbeit an vielen Stellen von bestimmten Zusatzannahmen, die aber nicht angegeben worden sind, sondern erst aus den Beweisen erarbeitet werden mussten. Zusätzlich fehlen an einigen Stellen viele wichtige Beweisschritte, wie die Betrachtung von Wahrscheinlichkeitsverteilungen der Simulationen. Die Analysen der Wahrscheinlichkeitsverteilungen sind in dieser Arbeit erarbeitet worden, was dazu geführt hat, dass die Simulationen auch angepasst werden mussten.

Diese unvorhersehbare Schwierigkeiten haben dazu geführt, dass nicht alle für diese Arbeit geplante Aufgaben, wie das Effizienzvergleich mit den neuesten Verfahren, betrachtet werden konnten. Auch bei dem Sakai-Kasahara-IBE-Verfahren musste die Analyse der Sicherheitsbeweise weggelassen werden, da diese den Rahmen dieser Bachelorarbeit sprengen würden.

Insgesamt wurde gezeigt, dass mit Hilfe vom vorgestellten Signcryptionverfahren sowohl die Vertraulichkeit als auch die Verbindlichkeit effizient erreicht werden kann.

Weiterhin kann das Verfahren auch so eingesetzt werden, dass die Nachricht nur signiert wird, indem man das vorgeschlagene BLMQ-Signaturverfahren anwendet. Andererseits kann es auch als reines Verschlüsselungsverfahren eingesetzt werten, wenn man das Sakai-Kasahara-Verschlüsselungsverfahren anwendet. Somit kann ein kryptographisches System aufgebaut werden, das alle drei Funktionalitäten zur Verfügung stellt und dabei nur ein Verfahren mit kleinen Anpassungen an die jeweiligen Funktionalitäten implementiert werden müsste.

Literaturverzeichnis

[BB04] Boneh, Dan und Xavier Boyen: *Short Signatures Without Random Oracles.* In: *EUROCRYPT*, Band 3027 der Reihe *Lecture Notes in Computer Science*, Seiten 56–73, 2004.

[BF01] Boneh, Dan und Matthew K. Franklin: *Identity-Based Encryption from the Weil Pairing.* In: *CRYPTO*, Band 2139 der Reihe *Lecture Notes in Computer Science*, Seiten 213–229, 2001.

[BLMQ05] Barreto, Paulo S. L. M., Benoît Libert, Noel McCullagh und Jean-Jacques Quisquater: *Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps.* In: *ASIACRYPT*, Band 3788 der Reihe *Lecture Notes in Computer Science*, Seiten 515–532, 2005.

[BN06] Bellare, Mihir und Gregory Neven: *Multi-signatures in the plain public-Key model and a general forking lemma.* In: *ACM Conference on Computer and Communications Security*, Seiten 390–399, 2006.

[Boy03] Boyen, Xavier: *Multipurpose Identity-Based Signcryption (A Swiss Army Knife for Identity-Based Cryptography).* In: *CRYPTO*, Band 2729 der Reihe *Lecture Notes in Computer Science*, Seiten 383–399, 2003.

[CC03] Cha, Jae Choon und Jung Hee Cheon: *An Identity-Based Signature from Gap Diffie-Hellman Groups.* In: *Public Key Cryptography*, Band 2567 der Reihe *Lecture Notes in Computer Science*, Seiten 18–30, 2003.

[CC05] Chen, Liqun und Zhaohui Cheng: *Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme.* In: *IMA Int. Conf.*, Seiten 442–459, 2005.

[FO99] Fujisaki, Eiichiro und Tatsuaki Okamoto: *Secure Integration of Asymmetric and Symmetric Encryption Schemes.* In: *CRYPTO*, Band 1666 der Reihe *Lecture Notes in Computer Science*, Seiten 537–554, 1999.

[KL07] Katz, Jonathan und Yehuda Lindell: *Introduction To Modern Cryptography.* Chapman & Hall/CRC, 2007.

[PS00] Pointcheval, David und Jacques Stern: *Security Arguments for Digital Signatures and Blind Signatures.* J. Cryptology, 13(3):361–396, 2000.

[Sha84] Shamir, Adi: *Identity-Based Cryptosystems and Signature Schemes.* In: *CRYPTO*, Band 196 der Reihe *Lecture Notes in Computer Science*, Seiten 47–53, 1984.

[SK03] Sakai, Ryuichi und Masao Kasahara: *ID based Cryptosystems with Pairing on Elliptic Curve.* In: *SCIS'03*, 2003.

[Zhe97] Zheng, Yuliang: *Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption).* In: *CRYPTO*, Band 1294 der Reihe *Lecture Notes in Computer Science*, Seiten 165–179, 1997.