TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG

**Ausarbeitung eines Seminarvortrags**

# Eine allgemeine Konstruktion für gleichzeitiges Signieren und Verschlüsseln

cand. inform. Lutz Wachsmann

08. Januar 2007


Institut für Theoretische Informatik
Prof. Dr. Dietmar Wätjen

# Inhaltsverzeichnis

**1 Einleitung** **1**

**2 Grundlagen** **3**
2.1 Notationen . . . . . 3
2.2 Probabilistische Polynomialzeitalgorithmen . . . . . 3
2.3 Das Random Oracle Model . . . . . 4
2.4 Signaturverfahren . . . . . 4
2.5 Mechanismus zur Schlüsselkapselung . . . . . 5
2.6 Einmalverschlüsselungsverfahren mit symmetrischen Schlüsseln . . . . 7

**3 Gleichzeitiges Signieren und Verschlüsseln** **8**

**4 Sicherheitsaspekte für gleichzeitiges Signieren und Verschlüsseln** **10**
4.1 Ununterscheidbarkeit von Verschlüsselungen . . . . . 10
4.2 Fälschungssicherheit . . . . . 11
4.3 Sicherheit . . . . . 11

**5 Eine mögliche Instanzierung** **12**

**6 Diskussion** **13**
6.1 Weitere Sicherheitsbegriffe . . . . . 13
6.2 Instanzierung . . . . . 15
6.3 Sicherheitsanforderungen an SIG . . . . . 16

**Literatur** **17**

**A Beweis des Theorems** **18**

# 1 Einleitung

Verschlüsselungs- und Signaturverfahren sind Basiswerkzeuge heutiger Public-Key-Kryptographie, um Datenschutz und Authentizität zu bieten. Ursprünglich wurden sie als wichtige, aber voneinander getrennte, Primitive für die Benutzung in Protokollen höherer Ebenen betrachtet. Es gibt jedoch viele Anwendungsfälle, in denen die Dienste beider Verfahren benötigt werden, wie zum Beispiel beim sicheren eMail-Verkehr. Hier sollten Nachrichten verschlüsselt und signiert werden, um die Vertraulichkeit und Authentizität zu gewährleisten. In diesem Fall ist es natürlich möglich, ein Verschlüsselungsverfahren in Kombination mit einem digitalen Signaturverfahren zu benutzen. In früheren Arbeiten von An und anderen wurde beobachtet, dass es häufig Raffinessen in der Benutzung solcher Kombinationen gibt [1]. Eine Möglichkeit der Ausnutzung dieser Merkmale besteht darin, sowohl Effizienz als auch Funktionalität der Authentifizierung und Verschlüsselung zu steigern. Angeregt durch diese Überlegungen, wurde in den letzten Jahren viel auf dem Gebiet der Verfahren und Methoden gleichzeitiger Signierung und Verschlüsselung geforscht.

Der erste Vorschlag zur Konstruktion eines Verfahrens mit der gemeinsamen Funktionalität eines Verschlüsselungsverfahrens und eines Signaturverfahrens erschien in einem Artikel von Zheng. Die Motivation dieser Arbeit lag darin, im Vergleich zur separaten Verschlüsselung und Signierung einen gewissen Effizienzgewinn zu erreichen. Zheng nannte diese Kombination der Verfahren *Signcryption* [10]. Später wurden viele Verfahren entwurfen, die auf Zhengs ursprünglicher Idee aufbauten. Einige von ihnen wurden formell mit komplexitätstheoretischen Reduktionen untersucht. Diese Untersuchungen verlassen sich allerdings auf das Random Oracle Model.

Die erste formale Sicherheitsuntersuchung von Signcryption wurde von An und anderen durchgeführt. Im Unterschied zu der Arbeit von Zheng hatte er nicht die Intention, Effizienz zu erreichen. Das Ziel lag vielmehr darin, einen strengen Rahmen zur Analyse beliebiger Verfahren oder zusammengesetzter Methoden mit kombinierter Funktionalität von Verschlüsselung und Signierung anzubieten. Einige Sicherheitsbegriffe wurden eingeführt: *Insider*- und *Outsider*-Sicherheit, sowie *Zweibenutzer*- und *Mehrbenutzer*-Sicherheit [1]. Diese Begriffe werden später genauer erläutert.

Auf den oben genannten Ergebnissen setzt die Arbeit von John Malone-Lee auf, dessen Ausarbeitungen die Basis des vorliegenden Dokumentes bilden. Es wird eine allgemeine Konstruktion empfohlen, die immer dann benutzt werden kann, wenn eine Nachricht verschlüsselt und signiert werden soll. Diese Konstruktion bedient sich der KEM-DEM - Methodik von Cramer und Shoup [4] in Kombination mit einem sicheren Signaturverfahren. Es wird bewiesen, dass die Sicherheit dieser Konstruktion auf der Sicherheit der einzelnen Komponenten beruht, aus denen sie zusammengesetzt ist. Die Konstruktion ist der „zuerst signieren, dann verschlüsseln“ - Methode ähnlich.

Es wird gezeigt, wie man in bestimmten Situationen eine effizientere Lösung mit dem KEM-DEM - Ansatz erreicht [5].

Es wird weiterhin eine mögliche Instanzierung des Verfahrens beschrieben, die auf dem Signaturverfahren von Boneh und Boyen und dem KEM von Cramer und Shoup basiert. Wenn Zhengs ursprüngliche Definition von Signcryption benutzt wird, um einen Effizienzgewinn mit kombinierter Verschlüsselung und Signatur zu erzielen, kann entgegengebracht werden, dass die hier beschriebene Konstruktion als erstes Signcryptionverfahren angesehen werden kann, das beweisbar sicher ist, ohne sich auf das Random Oracle Model zu stützen.

In Kapitel 2 werden die Primitiven beschrieben, auf denen die Konstruktion aufgebaut ist. Die Konstruktion selbst wird in Kapitel 3 eingeführt. Ihre Sicherheit wird in Kapitel 4 diskutiert. In Kapitel 5 wird beschrieben, wie die Konstruktion instanziert werden könnte. Das letzte Kapitel gibt eine Abgrenzung zwischen der bisherigen Forschungsarbeit und dieser Arbeit.

# 2 Grundlagen

In diesem Kapitel werden die grundlegenden Verfahren besprochen, auf denen die Konstruktion aufgebaut ist. Zuvor werden noch einige Notationen vorgestellt, die in diesem Dokument verwendet werden.

## 2.1 Notationen

Seien $\mathbb{Z}$ der Ring der ganzen Zahlen und $\mathbb{Z}_{\geq 0}$ der Ring der nichtnegativen ganzen Zahlen. Für eine positive Ganzzahl $a$ sei $\mathbb{Z}_a$ der Ring der ganzen Zahlen modulo a. $\mathbb{Z}_a^*$ sei die entsprechende Multiplikative Gruppe der Elemente.

Ist $S$ eine Menge, ist $v \leftarrow$ S die Abfrage aus der Gleichverteilung von S und die Zuweisung des Ergebnisses an die Variable v. Beinhaltet die Menge S ein Element $s$, wird kurz v $\leftarrow$ s an Stelle von v $\leftarrow$ {s} geschrieben.

Sei $A$ ein PPT Algorithmus. Die Zuweisung des Ergebnisses, das sich aus der Ausführung des Algorithmus auf eine Eingabe I ergibt, an die Variable v wird mit v $\leftarrow$ A(I) beschrieben. Eine Menge möglicher Ausgaben von A(I) wird mit [A(I)] bezeichnet.

Sei $E$ ein Ereignis in einem Wahrscheinlichkeitsraum. Dann wird die Wahrscheinlichkeit, dass das Ereignis E eintritt, mit Pr[$E$] benannt. Es wird angenommen, dass der Wahrscheinlichkeitsraum aus dem Kontext ersichtlich wird.

Die unäre Darstellung einer Zahl k wird mit $1^k$ beschrieben. Wird eine Zahl k unär dargestellt, ist die Länge von $1^k$ zugleich k.

## 2.2 Probabilistische Polynomialzeitalgorithmen

Die Abkürzung PP steht für Probabilistische Polynomialzeit, bzw. PPT für probabilistic polynomial-time. In der Komplexitätstheorie ist PP die Klasse der Entscheidungsprobleme, die von einer probabilistischen Turingmaschine (TM) in Polynomialzeit gelöst werden können. Die Bedingung, dass die Maschine akzeptiert, lautet, dass mehr als die Hälfte der Berechnungspfade akzeptiert werden. Dabei werden die Übergänge zwischen den möglichen Zuständen gemäss einer Wahrscheinlichkeitsverteilung zufällig gewählt. Im Falle gleicher Wahrscheinlichkeiten für Zustandsübergänge kann die TM entweder durch eine zusätzliche Instruktion zum Schreiben (0 oder 1) oder ein zusätzliches Band voller zufälliger Nullen und Einsen, „random tape“ genannt, erweitert werden. Als Konsequenz daraus kann eine probabilistische TM stochastische Werte liefern. Für eine gegebene Eingabe kann sie verschiedene Laufzeiten haben.

Sie kann halten oder auch nicht. Bei einer Ausführung akzeptiert sie die Eingabe, bei einer anderen Ausführung kann sie dieselbe Eingabe zurückweisen [7].

## 2.3 Das Random Oracle Model

Ein *Random Oracle* (RO) ist ein theoretisches Konzept, das durch keine echte Funktion implementieren werden kann. Es kann aber als mathematische Funktion $\{0,1\}^k \leftarrow \{0,1\}^*$ (für ein bestimmtes k) aufgefasst werden, bei der die Ausgabe für jede Eingabe „echt zufällig“ ist, aber bei gleicher Eingabe soll auch die gleiche Ausgabe erfolgen. Anderesseits ist ein RO kollisionsresistent, was bedeutet, dass es (bei genügend grossem $k$) nicht effizient möglich ist, zwei Eingaben zu finden, für die das RO den gleichen Wert ausgibt. An ein RO wird keine Anforderung der Vertraulichkeit gestellt. Dies bedeutet, dass jeder[1] auf beliebige Werte des RO zugreifen kann. Die Auswertung beliebiger Stellen der Ausgabe geben keinen Aufschluss über noch nicht ausgewertete Stellen.

Random Oracles sind in vielen kryptographischen Verfahren und Protokollen sehr nüzlich. So kann z.B. die Sicherheit einiger digitaler Signaturverfahren bewiesen werden, wenn die Nachricht vor Anwendung der eigentlichen Signaturtransformation durch Aufruf eines RO (statt einer Hashfunktion) gehasht wird. Solche Sicherheitsbeweise nennt man dann „Beweise im Random Oracle Model (ROM)“. Es gibt viele kryptographische Verfahren, die im ROM als sicher bewiesen werden können, unter der Annahme, dass sich die verwendete Hashfunktion wie ein RO verhält [2].

## 2.4 Signaturverfahren

Ein Signaturverfahren SIG besteht aus den folgenden drei Algorithmen.

- Ein PPT *Schlüsselerzeugungsalgorithmus* SIG.KeyGen mit der Eingabe $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$. Seine Ausgabe ist ein öffentlich/geheimes Schlüsselpaar ($\mathsf{PK}_s$,$\mathsf{SK}_s$). Die Strukur der Schlüssel hängt vom jeweiligen Verfahren ab.

- Ein Polynomialzeit-*Signieralgorithmus* SIG.Sig, der als Eingabe ein $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$, einen geheimen Schlüssel $\mathsf{SK}_s$ und eine Nachricht $m \in \{0,1\}^*$ hat. Die Ausgabe ist eine Signatur $\sigma$. Der Algorithmus SIG.Sig kann probabilistisch oder deterministisch sein.

- Ein Polynomialzeit-*Verifikationsalgorithmus* SIG.Ver, der als Eingabe ein $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$, einen öffentlichen Schlüssel $\mathsf{PK}_s$, eine Nachricht $m$ und eine für $m$ genannte Signatur $\sigma$ besitz. Die Ausgabe ist $\top$, wenn $\sigma$ wirklich eine Signatur zu $m$ für $\mathsf{PK}_s$ ist. Ansonsten gibt der Algorithmus $\perp$ zurück.

[1] inklusive ein potenzieller Gegner

**Sicherheit von Signaturverfahren: Starke existenzielle Fälschungssicherheit** Der übliche Begriff der Sicherheit eines Signaturverfahrens ist die *existenzielle Fälschungssicherheit gegenüber einem Angriff mit angepaßten Nachrichten* (existential unforgeability under adaptive chosen message attack) [8]. Hier kann der Angreifer jeweils die bisher erhaltenen Kryptotexte analysieren und abhängig vom Ergebnis einen neuen Klartext zum Verschlüsseln wählen. In diesem Abschnitt soll der etwas stärkere Begriff der *starken existenziellen Fälschungssicherheit* (strong existential unforgeability) für Signaturverfahren eingeführt werden. Um den Begriff zu erläutern, wird ein Angreifer A betrachtet, der eine probabilistische Orakel-Abfragemaschine mit polynomialem Zeitbedarf ist. Als Eingabe hat sie einen Sicherheitsparameter $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$. Der Angriff für diesen Sicherheitsbegriff läuft wie folgt ab.

**Phase 1:** Der Angreifer befragt ein *Schlüsselerzeugungs-Orakel* mit $1^\kappa$. Dies berechnet ($\mathsf{PK}_s$,$\mathsf{SK}_s$) $\leftarrow$ SIG.KeyGen($1^\kappa$) und antwortet mit $\mathsf{PK}_s$.

**Phase 2:** Der Angreifer befragt in einer Serie von höchstens $n_s$ Abfragen das *Signier-Orakel*. Für jede Abfrage $m_i$ mit $i \in \{1,...,n_s\}$ berechnet das Signier-Orakel eine Signatur $\sigma_i \leftarrow$ SIG.Sig($1^\kappa$,$\mathsf{SK}_s$,$m$) und antwortet mit $\sigma_i$. Der Angreifer darf seine Abfragen $m_i$ anhand der bisher gewonnenen Ergebnisse seiner Abfragen anpassen.

**Phase 3:** Der Angreifer versucht, ein Paar ($m$,$\sigma$) herauszubekommen, so dass gilt:
1. ($m$,$\sigma$) $\notin \{(m_1,\sigma_1),...,(m_i,\sigma_i)\}$ und
2. $x = \top$ mit $x \leftarrow$ SIG.Ver($1^\kappa$,$\mathsf{PK}_s$,$m$,$\sigma$)

Man sagt, der Angreifer *gewinnt*, wenn er dies schafft.

Es wird $\mathrm{AdvEF}_{SIG,A}(\kappa)$ als Pr[A gewinnt] definiert mit der obigen Definition von *gewinnt*. Die Wahrscheinlichkeit beruht auf den zufälligen Auswahlen von A und auf denen von A's Orakeln.

## 2.5 Mechanismus zur Schlüsselkapselung

Ein *Mechanismus zur Schlüsselkapselung* KEM (key encapsulation mechanism) besteht aus den folgenden Algorithmen.

- Ein PPT *Schlüsselerzeugungsalgorithmus* KEM.KeyGen mit der Eingabe $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$. Seine Ausgabe ist ein öffentlich/geheimes Schlüsselpaar ($\mathsf{PK}_r$,$\mathsf{SK}_r$). Die Strukur der Schlüssel hängt vom jeweiligen Verfahren ab.

- Ein PPT *Verschlüsselungsalgorithmus* KEM.Enc, der als Eingabe ein $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$ und einen öffentlichen Schlüssel ($\mathsf{PK}_r$) erwartet. Die Ausgabe ist ein Paar (K,$\psi$), wobei *K* ein Schlüssel und $\psi$ ein Chiffretext ist.
  Für jede Ausgabe ($\mathsf{PK}_r$,$\mathsf{SK}_r$) von KEM.KeyGen sei angenommen, dass

$$\max \mathsf{Pr}[\psi^* = \psi : (K^*, \psi^*) \leftarrow \mathsf{KEM.Enc}(1^\lambda, \mathsf{PK}_r)] \leq \frac{1}{2^{\lambda-1}}$$

gilt, wobei $\frac{1}{2^{\lambda-1}}$ durch eine beliebige venachlässigbare Funktion von $\lambda$ mit geeigneter Anpassung an diese Sicherheitsanalyse ersetzt werden kann.
Ein Schlüssel *K* ist eine Bitfolge der Länge KEM.KeyLen($\lambda$) mit KEM.KeyLen($\lambda$) als weiterer Parameter von KEM.

- Ein deterministischer *Entschlüsselungsalgorithmus* KEM.Dec mit polynomialer Laufzeit hat als Eingabe $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einen privaten Schlüssel $\mathsf{SK}_r$ und einen Chiffretext $\psi$. Die Ausgabe ist entweder ein Schlüssel *K* oder das Symbol $\perp$, wenn der eingegebe Chiffretext ungültig war.

**Korrektheit** Der Begriff der *Korrektheit* wird für den KEM benötigt. Man sagt, dass ein öffentlich/privates Schlüsselpaar ($\mathsf{PK}_r$,$\mathsf{SK}_r$) *schlecht* ist, wenn für einige $(K,\psi) \in$ [KEM.Enc($1^\lambda$,$\mathsf{PK}_r$)] ein $x \neq K$ findet mit *x* ist Ausgabe von KEM.Dec($1^\lambda$,$\mathsf{SK}_r$,$\psi$).
Sei $\mathsf{BadKP}_{KEM}(\lambda)$ die Wahrscheinlichkeit, dass ein Schlüsselerzeugungsalgorithmus ein *schlechtes* Schlüsselpaar für ein gegebenes $\lambda$ ausgibt. Die Anforderung an den KEM ist, dass $\mathsf{BadKP}_{KEM}(\lambda)$ vernachlässigbar in $\lambda$ wächst.

**Echte oder zufällige Sicherheit gegenüber passiven Attacken** Hier wird der schwache Begriff der Sicherheit des KEM eingeführt: *Echte oder zufällige Sicherheit gegenüber passiven Attacken* (Real or Random Security Against Passive Attack). Obwohl der schwache Begriff der Sicherheit für diese Anwendung ausreichend ist, kann er bei anderen Anwendungen durchaus durch stärkere Notationen ersetzt werden [4].

Ein Angreifer A, der einen passiven Angriff gegen den KEM versucht, ist ein PPT-Algorithmus, der als Eingabe $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$ hat. Im Folgenden wird ein Angriff beschrieben, um die Sicherheit gegenüber einer passiven Attacke zu definieren.

**Phase 1:** Der Angreifer befragt ein *Schlüsselerzeugungs-Orakel* mit $1^\lambda$. Dieses berechnet ($\mathsf{PK}_r$,$\mathsf{SK}_r$)$\leftarrow$ KEM.KeyGen($1^\lambda$) und antwortet mit $\mathsf{PK}_r$.

**Phase 2:** Der Angreifer befragt ein *challenge encryption oracle*. Dies wählt aus zwei Möglichkeiten eine aus.

$$(K^*, \psi^*) \leftarrow \mathsf{KEM.Enc}(1^\lambda, \mathsf{PK}_r); K^+ \leftarrow \{0,1\}^{l_k}; b \leftarrow \{0,1\};$$
$$\text{if } b = 0, K^\dagger \leftarrow K^*; \text{ else } K^\dagger \leftarrow K^+;$$

mit $l_k = \mathsf{KEM.KeyLen}(\lambda)$. Es antwortet mit $(K^\dagger,\psi^*)$.

**Phase 3:** Der Angreifer gibt $b' \in \{0,1\}$ aus.

Wenn A den obigen Angriff ausführt, definieren wir

$$\mathsf{AdvRR}_{KEM,A}(\lambda) := |\mathsf{Pr}[b' = 1|b = 1] - \mathsf{Pr}[b' = 1|b = 0]|. \tag{2.1}$$

Die Wahrscheinlichkeit beruht auf den zufälligen Auswahlen von A und auf denen von A's Orakeln.

## 2.6 Einmalverschlüsselungsverfahren mit symmetrischen Schlüsseln

Ein *Einmalverschlüsselungsverfahren mit symmetrischen Schlüsseln* SKE (One-Time Symmetric-Key Encryption) [4] besteht aus den folgenden beiden Algorithmen.

- Ein deterministischer Polynomialzeit-*Verschlüsselungsalgorithmus* SKE.Enc, der als Eingabe ein $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einen Schlüssel $K$ und eine Nachtticht $m \in \{0,1\}^*$ erwartet. Die Ausgabe ist ein Chiffretext $\chi \in \{0,1\}^*$.
  Ein Schlüssel $K$ ist eine Bitfolge der Länge SKE.KeyLen($\lambda$) mit SKE.KeyLen($\lambda$) als weiterer Parameter von SKE.
- Ein deterministischer Polynomialzeit-*Entschlüsselungsalgorithmus* SKE.Dec hat als Eingabe $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einen Schlüssel $K$ und einen Chiffretext $\chi$. Die Ausgabe ist entweder eine Nachricht $m \in \{0,1\}^*$ oder das Symbol $\perp$, wenn der eingegeben Ciphertext ungültig war.
  Der Schlüssel $K$ ist eine Bitfolge der Länge SKE.KeyLen($\lambda$).

**Korrektheit** Das Einmalverschlüsselungsverfahren mit symmetrischen Schlüsseln SKE muss die folgende Bedingung der Korrektheit erfüllen: Für alle $\lambda \in \mathbb{Z}_{\geq 0}$, für alle $K \in \{0,1\}^{SKE.KeyLen(\lambda)}$ und für alle $m \in \{0,1\}^*$ gilt

$$x = m \text{ mit } x \leftarrow \mathsf{SKE.Dec}(1^\lambda, K, \mathsf{SKE.Enc}(1^\lambda, K, m)).$$

**Ununterscheidbarkeit der Verschlüsselungen bei passiven Angriffen** Der schwache Begriff der Sicherheit von SKE wird hier eingeführt: *Ununterscheidbarkeit der Verschlüsselungen bei passiven Angriffen* (indistinguishability of encryptions under passive attack). Ein Angreifer, der einen passiven Angriff auf die SKE plant, ist ein PPT Algorithmus mit der Eingabe $1^\lambda$ mit dem Sicherheitsparameter $\lambda \in \mathbb{Z}_{\geq 0}$. Im Folgenden wird der Ablauf des Angriffes beschrieben, um diesen Begriff der Sicherheit zu definieren.

**Phase 1:** Der Angreifer wählt zwei Nachrichten gleicher Länge $m_0$ und $m_1$ und übergibt diese an ein *Verschlüsselungs-Orakel*.

**Phase 2:** Dies berechnet

$$K \leftarrow \{0,1\}^{l_s}; b \leftarrow \{0,1\}; \chi^* = \mathsf{SKE.Enc}(1^\lambda, K, m_b)$$

mit $l_s = \mathsf{SKE.KeyLen}(\lambda)$. Es antwortet mit $\chi^*$.

**Phase 3:** Der Angreifer gibt $b' \in \{0,1\}$ aus.

Wenn A den obigen Angriff ausführt, definieren wir

$$\mathsf{AdvIND}_{SKE,A}(\lambda) := |\mathsf{Pr}[b' = 1 | b = 1] - \mathsf{Pr}[b' = 1 | b = 0]|.$$

Die Wahrscheinlichkeit beruht auf den zufälligen Auswahlen von A und auf denen von A's Orakeln.

# 3 Gleichzeitiges Signieren und Verschlüsseln

Eine Konstruktion SSE für gleichzeitiges Signieren und Verschlüsseln (simultaneous signing and encrypting) besteht aus den folgenden vier Algorithmen.

- Ein PPT *Schlüsselerzeugungsalgorithmus* des Senders SSE.SKeyGen mit der Eingabe $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$. Seine Ausgabe ist ein öffentlich/geheimes Schlüsselpaar ($\mathsf{PK}_s$,$\mathsf{SK}_s$). Die Strukur der Schlüssel hängt vom jeweiligen Verfahren ab.

- Ein PPT *Schlüsselerzeugungsalgorithmus* des Empfängers SSE.RKeyGen mit der Eingabe $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$. Seine Ausgabe ist ein öffentlich/geheimes Schlüsselpaar ($\mathsf{PK}_r$,$\mathsf{SK}_r$). Die Strukur der Schlüssel hängt vom jeweiligen Verfahren ab.

- Ein PPT *Signier/Verschlüsselungs-Algorithmus* SSE.SigEnc, der als Eingabe ein $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$, ein $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einen geheimen Schlüssel des Senders $\mathsf{SK}_s$, einen öffentlichen Schlüssel des Empfängers $\mathsf{PK}_r$ und eine Nachricht $m \in \{0,1\}^*$ erwartet. Die Ausgabe ist ein Chiffretext *C*.

- Ein deterministischer *Entschlüsselungs/Verifizier-Algorithmus* SSE.DecVer mit polynomialer Laufzeit, der als Eingabe ein $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$, ein $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einen öffentlichen Schlüssel des Senders $\mathsf{PK}_s$, einen privaten Schlüssel des Empfängers $\mathsf{SK}_r$ und einen vermutlichen Chiffretext $C$. Die Ausgabe ist entweder eine Nachricht $m$ oder das Symbol $\perp$, wenn $C$ nicht gültig war.

Die ersten beiden Algorithmen dienen der Erzeugung der Sender- bzw. Empfängerschlüsselpaare. Der Algortihmus SSE.SigEnc signiert und verschlüsselt ein Nachricht, indem er die KEM-DEM - Methodik von Cramer und Shoup verwendet. Dafür wird zuerst ein symmetrischer Schlüssel $K$ und dessen asymmetrische Chriffrierung $\psi$ erzeugt (KEM: Key Encapsulating Mechanism). Dann wird mittels des privaten Senderschlüssels und der Nachricht $m$ konkateniert mit $\psi$ die Signatur $\sigma$ erstellt. Zuletzt wird mit dem Schlüssel $K$ und der Nachricht $m$ konkateniert mit $\sigma$ der Chiffretext $\chi$ erstellt (DEM: Data Encapsulaing Mechanism). Versendet wird das Chiffrepaar $(\psi, \chi)$.

Der Algorithmus SSE.DecVer dient der Entschlüsselung und der Verifikation der Nachricht. Nach der Wiederherstellung des symmetrischen Schlüssels $K$ ist der Ablauf entgegengesetzt zu SSE.SigEnc. Zuerst wird mittels $K$ und $\chi$ die Signatur $\sigma$ wiederhergestellt, um anschließend mit Hilfe des öffentlichen Senderschlüssels und der Nachricht konkateniert mit $\psi$ die Signatur zu verifizieren.

In Abbildung 3.1 wird beschrieben, wie die allgemeine Konstruktion arbeitet. Grundlage sind SIG, KEM und SKE, die in 2 beschrieben werden. Zur Vereinfachung wird angenommen, dass $\mathsf{KEM.KeyLen}(\lambda) = \mathsf{SKE.KeyLen}(\lambda)$ gilt.

**SKeyGen:** Für eine Eingabe $1^\kappa$ für $\kappa \in \mathbb{Z}_{\geq 0}$ werden folgende Schritte ausgeführt:

1. $(\mathsf{PK}_s, \mathsf{SK}_s) \leftarrow \mathsf{SIG.KeyGen}(1^\kappa)$
2. Rückgabe des öffentlichen Schlüssels $\mathsf{PK}_s$ und des privaten Schlüssels $\mathsf{SK}_s$

**RKeyGen:** Für eine Eingabe $1^\lambda$ für $\lambda \in \mathbb{Z}_{\geq 0}$ werden folgende Schritte ausgeführt:

1. $(\mathsf{PK}_r, \mathsf{SK}_r) \leftarrow \mathsf{KEM.KeyGen}(1^\lambda)$
2. Rückgabe des öffentlichen Schlüssels $\mathsf{PK}_r$ und des privaten Schlüssels $\mathsf{SK}_r$

**SigEnc:** Für eine Eingabe von $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$, $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einem geheimen Schlüssel $\mathsf{SK}_s$, einem öffentlichen Schlüssel $\mathsf{PK}_r$ und einer Nachricht $m \in \{0,1\}^*$ werden folgende Schritte ausgeführt:

**SE1:** $(K, \psi) \leftarrow \mathsf{KEM.Enc}(1^\lambda, \mathsf{PK}_r)$
**SE2:** $\sigma \leftarrow \mathsf{SIG.Sig}(1^\kappa, \mathsf{SK}_s, m||\psi)$
**SE3:** $\chi \leftarrow \mathsf{SKE.Enc}(1^\lambda, K, m||\sigma)$
**SE4:** Rückgabe des Chiffretextes $(\psi, \chi)$

**DecVer:** Für eine Eingabe von $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$, $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$, einem öffentlichen Schlüssel $\mathsf{PK}_s$, einem geheimen Schlüssel $\mathsf{SK}_r$ und einem Chiffretext $(\psi, \chi)$ werden folgende Schritte ausgeführt:

**DV1:** $K \leftarrow \mathsf{KEM.Dec}(1^\lambda, \mathsf{SK}_r, \psi)$
**DV2:** $m||\sigma \leftarrow \mathsf{SKE.Dec}(1^\lambda, K, \chi)$
**DV3:** $x \leftarrow \mathsf{SIG.Ver}(1^\kappa, \mathsf{PK}_s, m||\psi, \sigma)$
**DV4:** Falls $x = \top$, wird $m$ zurückgegeben, andernfalls $\perp$

Abbildung 3.1: Eine allgemeine Konstruktion für gleichzeitiges Signieren und Verschlüsseln

# 4 Sicherheitsaspekte für gleichzeitiges Signieren und Verschlüsseln

## 4.1 Ununterscheidbarkeit von Verschlüsselungen

In diesem Abschnitt wird der Begriff der *Ununterscheidbarkeit von Verschlüsselungen gegenüber ausgewählten Klartext- und ausgewählten Chiffretext-Attacken* (ICPCA, indistinguishability of encryptions under an adaptive chosen-plaintext and chosen-ciphertext attack) für ein SSE-Verfahren SSE beschrieben. Es ist anaolog zur IND-CCA2 - Sicherheit der Standard-Public-Key-Verschlüsselung [6]. Um den Begriff zu beschreiben, wird ein Angreifer A angenommen, der eine PPT Orakel-Abfragemaschine ist mit der Eingabe $1^\kappa$ mit $\kappa \in \mathbb{Z}_{\geq 0}$ und $1^\lambda$ mit $\lambda \in \mathbb{Z}_{\geq 0}$. Der Angriff läuft wie folgt ab.

**Phase 1:** Der Angreifer befragt ein *Sender-Schlüsselerzeugungs-Orakel* mit $1^\kappa$. Dies berechnet ($\mathsf{PK}_s$,$\mathsf{SK}_s$) $\leftarrow$ SSE.SKeyGen($1^\kappa$) und antwortet mit $\mathsf{PK}_s$.

**Phase 2:** Der Angreifer befragt ein *Empfänger-Schlüsselerzeugungs-Orakel* mit $1^\lambda$. Dies berechnet ($\mathsf{PK}_r$,$\mathsf{SK}_r$) $\leftarrow$ SSE.RKeyGen($1^\lambda$) und antwortet mit $\mathsf{PK}_r$.

**Phase 3:** Der Angreifer befragt in einer Serie von Anfragen zwei Orakel:
*Ein Signier/Verschlüsselungs-Orakel* und ein *Entschlüsselungs/Verifizier-Orakel*.
Während der Phasen 3 bis 5 macht der Angreifer
höchstens $n_s$ Anfragen an das Signier/Verschlüsselungs-Orakel und
höchstens $n_d$ Anfragen an das Entschlüsselungs/Verifizier-Orakel.
Für jede Anfrage $m$ berechnet das Signier/Verschlüsselungs-Orakel

$$C \leftarrow \mathsf{SSE.SigEnc}(1^\kappa,1^\lambda,\mathsf{SK}_s,\mathsf{PK}_r,m)$$

und antwortet mit $C$.
Für jede Anfrage $C$ berechnet das Entschlüsselungs/Verifizier-Orakel

$$x \leftarrow \mathsf{SSE.DecVer}(1^\kappa,1^\lambda,\mathsf{PK}_s,\mathsf{SK}_r,C)$$

und antwortet mit $x$.
Der Angreifer wählt seine Anfragen an diese Orakel basierend auf deren Antworten auf vorangangene Anfragen aus.

**Phase 4:** Der Angreifer wählt zwei Nachrichten gleicher Länge $m_0$ und $m_1$. Er übergibt diese an ein *challange oracle*, das folgendes leistet.

$$b \leftarrow \{0,1\}; C^* \leftarrow \mathsf{SigEnc}(1^\kappa,1^\lambda,\mathsf{SK}_s,\mathsf{PK}_r,m_b)$$

Die Rückgabe an den Angreifer ist $C^*$.

**Phase 5:** Der Angreifer fährt mit der Abfrage der Orakel aus Phase 3 fort, vorausgesetzt, dass er das Entschlüsselungs/Verifizier-Orakel nicht mit $C^*$ befragt.

**Phase 6:** Der Angreifer gibt $b' \in \{0, 1\}$ aus.

Wenn A den obigen Angriff ausführt, definieren wir

$$\mathsf{AdvICPCA}_{SSE,A}(\kappa,\, \lambda) := |\mathsf{Pr}[b' = 1|b = 1] - \mathsf{Pr}[b' = 1|b = 0]|.$$

Die Wahrscheinlichkeit beruht auf den zufälligen Auswahlen von A und auf denen von A's Orakeln.

## 4.2 Fälschungssicherheit

Da die Konstruktion ein Signaturverfahren verwendet, das angenommen existenziell fälschungssicher unter Attacken mit ausgewählten Nachrichten (existential unforgeable under adaptive chosen message attack) ist, wird hier die Fälschungssicherheit nicht explizit behandelt. Es soll gezeigt werden, wie ein schwaches Verschlüsselungsverfahren benutzt und trotzdem hohe Sicherheit beibehalten werden kann. Durch die benutzte Fälschungssicherheit des Signaturverfahrens ist die Fälschungssicherheit der Konstruktion gegeben [1].

## 4.3 Sicherheit

Hier wird das Ergebnis der Sicherheitsuntersuchung für die Konstruktion mit Hilfe der Definition aus 4.1 angegeben.

**Theorem** *Sei SSE eine Instanz der Konstruktion, die SIG, KEM und SKE verwendet. Sei A ein Angreifer von SSE, der eine „ausgwählter Klartext- und ausgewählter Chiffretext-Attacke" benutzt, um zu versuchen, Verschlüsselungen von SSE zu unterscheiden. Angenommen, A macht höchstens $n_s$ Abfragen an Signieren/Verschlüsseln und höchstens $n_d$ Abfragen an Entschlüsseln/Verifizieren. Es ist möglich, von diesem Angreifer weitere Angreifer $A_1$, $A_2$ und $A_3$ zu konstruieren, deren Laufzeiten im Wesentlichen der von A gleichen, so dass*

$$\begin{aligned} AdvICPCA_{SSE,A}(\kappa,\lambda) \leq \quad & 2AdvEF_{SIG,A_1}(\kappa) + 2BadKP_{KEM}(\lambda) \\ & + 2AdvRR_{KEM,A_2}(\lambda) + AdvIND_{SKE,A_3}(\lambda) + \frac{n_s}{2^{\lambda-2}} \end{aligned}$$

*gilt.*

Der Beweis des Theorems wird im Anhang geführt. Zunächst soll eine mögliche Instanzierung der Konstruktion vorgestellt werden.

# 5 Eine mögliche Instanzierung

In diesem Kapitel wird kurz eine sehr effiziente Instanzierung der Konstruktion beschrieben. Diese Instanzierung kann als erstes Signcryption-Verfahren der Literatur angesehen werden, das beweisbar sicher ist, ohne das Random Oracle Model zu benutzen. Die Behauptung wird in Kapitel 6 begründet.

Das für unsere Konstruktion empfohlene Signaturverfahren wurde von Boneh und Boyen [3] vorgeschlagen. Dieses Verfahren benutzt eine bilineare Gruppe, die häufig zur Konstruktion von identity-based - Kryptosystemen verwendet wird. Es ist stark fälschungssicher gegenüber Attacken mit ausgewählten Nachrichten (strong existential unforgeable under adaptive chosen message attack), wodurch das Kriterum der Sicherheit für unsere Konstruktion erfüllt wird. Dies kann unter einer Annahme, Diffie-Hellmann-Annahme genannt, bewiesen werden, ohne die Heuristik des Zufallsorakels zu benutzen. Das Verfahren ist sehr effizient: Das Signieren kostet eine Gruppenexponentiation und das Verifizieren kostet zwei Gruppenexponentiationenm sowie zwei pairing-Berechungen.

Zur Instanzierung des Schlüsselkapselung-Mechanismus wird das Hashed-El-Gamal - Verfahren (HEG) von Cramer und Shoup vorgeschlagen [4]. Dafür wurde bereits bewiesen, dass es sicher gegenüber Attacken mit ausgewähltem Chiffretext ist, sofern das Gap-Diffie-Hellman-Problem hart ist. Aus den Abschnitten 2.5 und 4 wird klar, dass diese Konstruktion keinen so starken Begriff der Sicherheit von seinem Schlüsselkapselung-Mechanismus fordert. Ferner folgt aus den Ergebnissen von Tsiounis und Yung [9], dass solche auf ElGamal basierenden KEMs sicher gegenüber passiven Angriffen, wie benötigt, unter der Decisional-Diffie-Hellman-Annahme sind. Dieses Ergebnis wird auch ohne die Benutzung eines Random Oracle Model erzielt.

Der Vorteil der Benutzung eines solchen KEMs ist, dass die Sicherheit dieser Konstruktion unter den bereits aufgestellten Definitionen auch dann gewährleistet ist, wenn kein Random Oracle Model benutzt wird. Wird stärkere Sicherheit benötigt (siehe Kapitel 6), kann einfach das Ergebnis von Cramer und Shoup verwendet werden, das auf dem Random Oracle Model beruht und diese Sicherheit erreicht, ohne den KEM zu ändern.

Als Instanz des Einmalverschlüsselungsverfahrens mit symmetrischen Schlüsseln wird ein Blockchiffre-Verfahren wie AES im geeigneten Modus empfohlen. Obwohl es nicht möglich ist, die Sicherheit des AES zu beweisen, wird die Korrektheit auf Grund vieler Analysen weitläufig angenommen.

# 6 Diskussion

In diesem Kapitel werden weitere Sicherheitsaspekte erläutert, die in früheren Arbeiten von An und anderen [1] definiert wurden, um die Arbeit in einem Kontext zu sehen und um einige Entscheidungen innerhalb dieser Konstruktion zu begründen.

## 6.1 Weitere Sicherheitsbegriffe

Wie in der Einleitung erwähnt, wurden von An und anderen einige Sicherheitsbegriffe für Verfahren mit gleichzeitiger Signierung und Verschlüsselung definiert. Diese werden nun beschrieben.

**Outsider-Sicherheit** Die bisherige Definition ist eine Instanz der Outsider-Sicherheit: Angenommen, ein Angreifer hat Zugriff auf ein Signier/Verschlüsselungs-Orakel für $SK_s/PK_r$ und auf ein Entschlüsselungs/Verifikations-Orakel für $PK_s/SK_r$. Er ist immer noch ein Außenstehender des Systems und kennt $SK_s$ nicht[1].

**Insider-Sicherheit** Wenn einem Angreifer $SK_s$ innerhalb der Konstruktion bekannt ist, erhält man ein Standard-Public-Key-Verschlüsselungsverfahren. Denn damit ist der Angreifer in der Lage, Chiffretexte zu produzieren. In einer Arbeit [1] wird dann von einem *induzierten* Verschlüsselungsverfahren gesprochen. Die Konstruktion soll Insider-Sicherheit bieten, wenn das induzierte Verschlüsselungsverfahren IND-CCA2 - sicher ist.

An und andere erklären, dass in vielen Situationen Outsider-Sicherheit ausreichen dürfte. Es wird darauf hingewiesen, dass Chiffretexte, die vom Besitzer des $SK_s$ für einen Besitzer von $PK_r$ erstellt wurden, garantiert vertraulich bleiben, selbst wenn $SK_s$ im Weiteren kompromittiert wird. Viele Signcryption-Verfahren besitzen diese Eigenschaft jedoch nicht.

Hier wird nun skizziert, was nötig ist, damit die Konstruktion Insider-Sicherheit gewährleistet. Zuerst muss beachtet werden, dass die stark existenzielle Fälschungssicherheit (strong existential unforgeability) des Signaturverfahrens nicht benutzt werden kann wie im Theorem der Sicherheitsbetrachtungen. Der Grund dafür ist die Herausgabe von $SK_s$ an einen Angreifer. Falls der Angreifer $SK_s$ kennt, kann er Signaturen für jede beliebige Nachricht erstellen, ohne etwas fälschen zu müssen!

Es wird eine Methode zur Simulation des Entschlüsselungs/Verifizier-Orakels benötigt, die nicht die existenzielle Unfälschbarkeit des Signaturverfahrens dieser Konstruktion aus-

[1] Würde er $SK_r$ kennen, könnte er die Entschlüsselung selbst durchführen. Somit wäre die Ununterscheidbarkeit der Verschlüsselungen unmöglich!

nutzt. Dafür werden stärkere Begriffe der Sicherheit des KEM und SKE benötigt als bisher verwendet.

In Abschnitt 2.5 wurde der Begriff der echten oder zufälligen Sicherheit für KEM eingeführt. Die Forderung lautete, dass der KEM selbst dann sicher ist, wenn ein Angreifer Zugriff auf ein Entschlüsselungsorakel hat, das $SK_r$ benutzt. Dies ist der Begriff für KEM gegenüber Attacken mit ausgewähltem Chiffretext, wie in Kaptiel 4 erwähnt. Es sei nochmals angemerkt, dass der HEG KEM von Cramer und Shoup für diese Konstruktion empfohlen wird, da die Sicherheit für diesen stärkeren Begriff der Sicherheit bereits bewiesen wurde, wenn auch mit dem Random Oracle Model [4].

Der Begriff der Sicherheit, der für SKE benötigt wird, ist ähnlich wie in Abschnitt 2.6. Der Unterschied besteht darin, dass dem Angreifer Zugriff auf ein Entschlüsselungsorakel gewährt wird, das mit dem Schlüssel $K$ arbeitet. Cramer und Shoup zeigen, wie ein gegenüber passiven Attacken sicheres Verschlüsselungsverfahren - wie in Abschnitt 2.6 - sicher gemacht werden kann für den stärkeren Begriff der Sicherheit mit Hilfe eines MACs (message authentication code) [4].

Angenommen, es gibt die Grundverfahren KEM und SKE, die diesem stärkeren Begriff der Sicherheit, wie oben beschrieben, genügen. Eine Simulation für einen Insider-Angriff auf diese Konstruktion würde wie folgt arbeiten. Jemand führt zuerst SIG.KeyGen aus und übergibt dem Angreifer den resultierenden Signierschlüssel. Es ist wie gefordert möglich, das Entschlüsselungs/Verifikations-Orakel zu simulieren, indem dem Angreifer von KEM und SKE Zugriff auf die Entschlüsselungsorakel gewährt wird.

**Zweibenutzer-Konfiguration** Die bisherige Definition dieser Konstruktion ist eine Instanz der Sicherheit für eine Zweibenutzer-Konfiguration (Two-User Setting): Der Angreifer möchte eine Verschlüsselung, die mittels $SK_s$ und $PK_r$ erzeugt wurde, unterscheiden. Dafür nutzt er den Zugriff auf ein Signier/Verschlüsselungs-Orakel für $SK_s/PK_r$ und auf ein Entschlüsselungs/Verifizier-Orakel für $PK_s/SK_r$. Dabei gilt es zu beachten, dass diese Orakel jeweils fest sind für $PK_r$ und $PK_s$.

**Mehrbenutzer-Konfiguration** Anders als bei der Zweibenutzer-Konfiguration ist der Angreifer bei der Mehrbenutzer-Konfiguration (Multi-User Setting) in der Lage, die öffentlichen Schlüssel als Eingabe für die Orakel selbst auszuwählen. Er kann z.B. Verschlüsselungen erhalten, die mittels $SK_s$ und einem beliebigen öffentlichen Schlüssel seiner Wahl produziert wurden.

An und andere zeigen, dass entsprechende Sicherheit in der Mehrbenutzer-Konfiguration entscheidend ist, um Probleme von Identitätsbetrug zu vermeiden. Als Beispiel wird eine Konstruktion betrachtet, bei der eine Nachricht mit Bobs öffentlichen Schlüssel verschlüsselt und der resultierende Chiffretext mit dem geheimen Schlüssel von Alice signiert wird. Diese Konstruktion ist sicher in der Zweibenutzer-Konfiguration, wenn ihre beiden Komponenten sicher sind. Aber angenommen, Oskar sei ein Benutzer, der Nachrichten abfangen und modifizieren kann. Dann kann er die Signatur von Alice entfernen und gegen seine eigene austauschen. In diesem Fall würde Bob davon überzeugt sein, die von ihm emp-

fangene verschlüsselte Nachricht käme von Oskar, obwohl sie in Wirklichkeit von Alice stammt. Dies kann geschehen, ohne die Sicherheit irgendeiner Komponente zu verletzen.

In der Mehrbenutzer-Konfiguration kann die starke existenzielle Fälschungssicherheit des Signaturverfahrens nicht für diesen Sicherheitsbeweis benutzt werden. Das Entschlüsselungs/Verifikations-Orakel im Beweis des Theorems nutzt die Tatsache, dass es, falls das Orakel mit einem gültigen Chiffretext $(\psi, \chi)$ versorgt wurde, innerhalb dieses Chiffretextes eine Signatur für $\psi$ geben muss. Weil der Simulator die Schlüssel $K$ kennt, mit denen entsprechend alle $\psi$s signiert wurden, kann er entschlüsseln und verifizieren. Dies funktioniert nur, wenn die Chiffretexte hinsichtlich eines spezifischen öffentlichen Schlüssels eines Senders $\mathrm{PK}_r$ entschlüsselt und verifiziert werden - nicht im Falle der Mehrbenutzer-Konfiguration.

Es wird klar, dass die oben herausgearbeiteten hohen Sicherheitsanforderungen für KEM und SKE in der Insider-Sicherheit ebenso für die Mehrbenutzer-Konfiguration ausreichen: Die Simulation des Entschlüsselungs/Verifizier-Orakels benötigt nur Entschlüsselungsorakel für KEM und SKE. Diese beiden sind darüberhinaus noch unabhängig von irgendwelchen dem Sender gehörenden Schlüsseln. Um das Signier/Verschlüsselungs-Orakel zu simulieren, benötigt man nur ein für $\mathrm{PK}_s$ gültiges Signierorakel. Die Generierung von $\psi$ und die letzte Verschlüsselung kann mittels des vom Angreifer bereitgestellten öffentlichen Schlüssels erledigt werden.

## 6.2 Instanzierung

Bis heute müßte jede Konstruktion für gleichzeitiges Signieren und Verschlüsseln, das beweisbare Sicherheit ohne das Random Oracle Model bietet, ein komplett IND-CCA2 - sicheres Verschlüsselungsverfahren und ein existenziell fälschungssicheres Signaturverfahren gegenüber Attacken mit ausgewählten Nachrichten [1] benutzen. Die Ergebnisse des Verschlüsselungsverfahrens und des Signaturverfahrens müssten eindeutig auch im Standardmodell beibehalten werden. Alle anderen Lösungen verlassen sich mit ihren Sicherheitsbeweisen auf das Random Oracle Model. Obwohl diese Konstruktion immer noch ein existenziell fälschungssicheres Signaturverfahren benötigt, kann die Effizienz bei der Verschlüsselung stark verbessert werden: Der HEG KEM des Sicherheitsbegriffes im Theorem benötigt nur zwei Gruppenexponentiationen zur Verschlüsselung und eine Gruppenexponentiation für das Entschlüsseln. Analoge Methoden zur Sicherstellung der vollen IND-CCA2 - Sicherheit [4] benötigen dahingegen weit mehr Rechenaufwand.

Der Autor kann für sich beanspruchen, das erste Signcryption-Verfahren in Zhengs Sinne mit beweisbarer Sicherheit ohne das Random Oracle Model entworfen zu haben. Zur Zeit wird mit dieser Random Oracle - freien Version nur die Outsider-Sicherheit in der Zweibenutzer-Konfiguration gewährleistet. Eine Erweiterung der Ergebnisse auf andere Modelle [1] ist noch offen.

## 6.3 Sicherheitsanforderungen an SIG

Es wurde lieber der Begriff der starken existenziellen Fälschungssicherheit (strong existential unforgeability) [1] statt des Standardbegriffs der existenziellen Fälschungssicherheit (existential unforgeability) benutzt. Es folgen einige Erläuterungen zur Intention des Autors. Der herausgeforderte (challenge) Chiffretext $(\psi^*, \chi^*)$ ist derart, dass $\chi^*$ die Verschlüsselung von $m_b||\sigma^*$ ist und $\sigma^*$ die Signatur von $(m_b, \psi^*)$. Falls das Signaturverfahren nicht stark existenziell fälschungssicher ist, scheidet die Möglichkeit nicht aus, dass ein Angreifer ein $\sigma' \neq \sigma^*$ findet mit $\sigma'$ eine gültige Signatur für $m_b||\psi^*$. Darüberhinaus kann das Verfahren gebrochen werden, wenn es ein Angreifer schafft, ein derartiges $\chi' \neq \chi^*$ zu finden, das den herausgeforderten Chiffretext entschlüsselt und somit $m_b||\sigma'$ freigibt, und er schließlich $(\psi^*, \chi')$ an das Entschlüsselungs/Verifizier-Orakel übergibt.

An und andere geben in einer Arbeit an, dass starke existenzielle Fälschungssicherheit ebenso in ihrer „erst verschlüsseln - dann signieren“ - Konstruktion benutzt wird, um die Sicherheitsanforderungen an das Verschlüsselungsverfahren herabzusetzen. Bei diesem Modell wird letztendlich der Chiffretext anstatt der Nachricht signiert [1].

# Literatur


[1] An, J. H. ; Dodis, Y. ; Rabin, T.: On the security of joint signature and encryption. In: *Advances in Cryptology - EUROCRYPT 2002, volume 2332 of Lecture Notes in Computer Science* (2002), S. 83–107 1, 11, 13, 15, 16

[2] Bellare, M. ; Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: *1st ACM Conference on Computer and Communications Security* (1993), S. 62–73 4

[3] Boneh, D. ; Boyen, X.: Short signatures without random oracles. In: *Advances in Cryptology - EUROCRYPT 2004, volume 3027 of Lecture Notes in Computer Science* (2004), S. 56–73 12

[4] Cramer, R. ; Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In: *SIAM Journal on Computing, 33*1 (2003), S. 167–226 1, 6, 7, 12, 14, 15

[5] Dent, A. W.: A designers guide to KEMs. In: *Cryptography and Coding, volume 2898 of Lecture Notes in Computer Science* (2005), S. 133–151 2

[6] King, T. ; Lucks, S.: Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In: *Seminararbeit an der Universität Mannheim* (2002) 10

[7] Neukamm, S.: Complexity-Theoretic Cryptography. In: *Joint Advanced Student School* (2005) 4

[8] Schwenk, J. ; Manulis, M.: Digitale Signaturen. In: *Übungen zur Vorlesung Kryptographische Protokolle, Übung 2* (2006) 5

[9] Tsiounis, Y. ; Yung, M.: On the security of ElGamal based encryption. In: *Public Key Cryptography - PKC 1998, volume 1431 of Lecture Notes in Computer Science* (1998), S. 117–134 12

[10] Zheng, Y.: Digital signcryption or how to achieve cost(signature & encryption) « cost(signature) + cost(encryption). In: *Advances in Cryptology - CRYPTO '97, volume 1294 of Lecture Notes in Computer Science* (1997), S. 165–179 1

# A Beweis des Theorems

Die Beweisstrategie lautet wie folgt: Es wird eine Reihe von modifizierten Angriffspielen $G_0, \ldots, G_7$ geben. Der einzige Unterschied zwischen den Spielen ist, wie die Umgebung auf die Orakelanfragen von A antwortet. Für jedes $0 \leq i \leq 7$ soll $S_i$ das Ereignis $b' = b$ in Spiel $G_i$ sein. Dabei ist $b$ das vom Orakel gewählte und $b'$ das von A ausgegebene Bit. Die Wahrscheinlichkeit beruht auf den zufälligen Auswahlen von A und auf denen von A's Orakeln.

Das folgende Lemma wird häufig benötigt.

**Lemma 1** *Seien E, F und G Ereignisse im Wahrscheinlichkeitsraum Pr*$[E \wedge \neg G] =$ *Pr*$[F \wedge \neg G]$*. Dann gilt*

$$|Pr[E] - Pr[F]| \leq Pr[G].$$

**Spiel $G_0$** Es wird die Sicht eines Angreifers in einer echten Attacke simuliert, indem die entsprechenden Schlüsselerzeugungsalgorithmen ausgeführt und die resultierenden Schlüssel benutzt werden. A hat den gleichen Standpunkt wie in einer echten Attacke, und er hat die Wahrscheinlichkeit

$$Pr[S_0] = \frac{1}{2}(AdvICPCA_{SSE,A}(\kappa, \lambda) + 1). \tag{A.1}$$

**Spiel $G_1$** In diesem Spiel werden die Anworten der Orakel, mit denen A interagiert, nicht geändert. Die einzige Änderung ist die Einführung einer Liste $L_s$ für die nächsten Spiele. Diese Liste ist anfangs leer. Für jede Anfage an das Signieren/Verschlüsseln-Orakel wird das entsprechende $(K, m||\psi, \sigma)$ der Liste hinzugefügt. Wird das challenge oracle befragt, wird die Liste dementsprechend angepasst.

Diese Änderung hat keinerlei Auswirkung auf Angreifer, also gilt

$$Pr[S_1] = Pr[S_0]. \tag{A.2}$$

**Spiel $G_2$** Nun wird die Generation des challenge-Chiffretextes abgeändert. Zu Beginn der Simulation wird, nach der Abfrage des Empfängerschlüssel-Erzeugungsorakel durch A, ein $(K, \psi) \leftarrow$ KEM.Enc($1^\lambda$,$PK_r$) berechnet, $(K^*, \psi^*) \leftarrow (K, \psi)$ gesetzt und $(K^*, \psi^*)$ für die Erzeugung des challenge-Chiffretextes benutzt. Auch diese Änderung hat keinerlei Auswirkung auf die Sicht von A, so dass gilt

$$Pr[S_2] = Pr[S_1]. \tag{A.3}$$

**Spiel $G_3$** In diesem Spiel wird die Art des Signieren/Verschlüsseln-Orakels etwas abgeändert, wie es Abfragen von A antwortet. Es wird eine Regel **SE1'** zwischen **SE1** und **SE2** hinzugefügt.

**SE1'**: Falls $\psi = \psi^*$, wird die Simulation beendet.

Die Chancen, dass das Signieren/Verschlüsseln-Orakel ein $\psi^*$ auf eine Signieren/Verschlüsseln-Anfrage generiert, liegen bei $\frac{q_s}{2^{\lambda-1}}$. Die Spiele $G_2$ und $G_3$ laufen in gleicher Weise bis hierhin ab. Nach A gilt

$$|Pr[S_3] - Pr[S_2]| \leq \frac{q_s}{2^{\lambda-1}}. \tag{A.4}$$

**Spiel $G_4$** Bei diesem Spiel wird die Art, wie das Entschlüsselungs/Verifikations-Orakel auf die Anfragen von A antwortet, geändert. Dazu wird eine Regel **DV2'** zwischen **DV2** und **DV3** eingefügt. Sei $(\psi, \chi)$ der Chiffretext, der zu dem Orakel gesendet wurde. Die neue Regel leistet folgendes.

**DV2'**: Falls $(\hat{K}, \hat{m}||\psi, \hat{\sigma}) \notin L_s$ für jedes $\hat{K}, \hat{m}, \hat{\sigma}$, wird $\perp$ zurückgegeben. Falls $\psi = \psi^*$, wird $\perp$ zurückgegeben. Ansonsten wird mit **DV3** fortgefahren.

Sei nun $R_4$ das Ereignis, dass in Spiel A ein Chiffretext übermittelt wurde, den das Entschlüsselungs/Verifikations-Orakel zurückgewiesen hat, aber der in Spiel A nicht zurückgewiesen worden wäre. Da die Ereignisse $S_4 \wedge \neg R_4$ und $S_3 \wedge \neg R_4$ identisch sind, gilt nach Lemma 1

$$|Pr[S_4] - Pr[S_3]| \leq Pr[R_4]. \tag{A.5}$$

Pr[$R_4$] wird im folgenden Lemma bewiesen.

**Lemma 2** *Es existiert ein PPT Algorithmus $A_1$, dessen Laufzeit im Wesentlichen der Laufzeit von A entspricht, so dass gilt*

$$Pr[R_4] \leq \mathrm{AdvEF}_{SIG,A_1}(\kappa). \tag{A.6}$$

Die Beweise der Lemmata werden nach dem Hauptbeweis geführt.

**Spiel $G_5$** In diesem Angriffsspiel wird die Art, wie das Entschlüsselungs/Verifikations-Orakels auf eine Anfrage $(\psi, \chi)$ von A antwortet, angepasst. Dafür wird die Regel **DV1** durch folgende ersetzt.

**DV1**: Durchsuche die Liste $L_s$, bis ein Eintrag $(\hat{K}, \hat{m}||\psi, \hat{\sigma})$ für ein $\hat{K}, \hat{m}, \hat{\sigma}$ gefunden oder bis das Ende der Liste erreicht wird. Wird ein Eintrag gefunden, soll $\hat{K}$ in **DV2** benutzt werden, ansonsten wird $\perp$ zurückgegeben.

Wenn kein Eintrag $(\hat{K}, \hat{m}||\psi, \hat{\sigma})$ in der Liste $L_s$ auftaucht, wird jeder beliebige Chiffretext, dessen erste Komponente $\psi$ ist, in Spiel A auf jeden Fall zurückgewiesen. Deshalb bringt diese Modifikation keine Änderung mit sich aus Sicht von A, es sei denn, $\hat{K}' \neq \hat{K}$ mit $\hat{K}'$ ist das Ergebnis der Ausführung von KEM.Dec$(1^\lambda, SK_r, \psi)$. Hierfür gilt nach Lemma 1

$$|Pr[S_5] - Pr[S_4]| \leq \mathrm{BadKP}_{KEM}(\lambda). \tag{A.7}$$

**Spiel $\mathbf{G}_6$** In diesem Spiel gibt es eine Änderung, wie der challenge-Chiffretext generiert wird. Dafür wird Stufe **S3** durch folgende Regel **SE3**$^*$ ersetzt.

**SE3**$^*$: $K^* \leftarrow 0,1^{l_k}; \chi^* \leftarrow \mathsf{SKE.Enc}(1^\lambda, K^*, m_b||\sigma^*)$

Da Regel **DV2'** in Angriffsspiel $\mathbf{G}_4$ eingeführt wurde, muss $K^*$ für den weiteren Verlauf der Simulation nicht bekannt sein. Es wird daher $(-, m_b||\psi^*, \sigma^*)$ anstatt $(K^*, m_b||\psi^*, \sigma^*)$ der Liste $L_s$ hinzugefügt[1]. Außerdem muss $\mathsf{SK}_r$ für die Simulation von A nicht mehr bekannt sein, weil **DV1** in $\mathbf{G}_5$ umgeschrieben worden ist.

**Lemma 3** *Es existiert ein PPT Algorithmus $A_2$, der im Wesentlich dieselbe Laufzeit wie A hat, so dass gilt*

$$|Pr[S_6] - Pr[S_5]| = \mathrm{AdvRR}_{KEM,A_2}(\lambda). \tag{A.8}$$

**Spiel $\mathbf{G}_7$** Im letzten Spiel $\mathbf{G}_7$ wird nochmal die Generierung des challenge Chiffretextes modifiziert. Dazu wird Regel **SE**$^*$ aus Spiel $\mathbf{G}_6$ durch die folgende ausgetauscht.

**SE**$^*$: $K^* \leftarrow 0,1^{l_k}; str \leftarrow 0,1^{l_m}$ mit $l_m$ ist die Bitlänge von $(m_b||\sigma^*)$;

$\chi \leftarrow \mathsf{SKE.Enc}(1^\lambda, K^*, str)$

Da nun der challenge Chiffretext und alles andere in der Simulation nicht mehr von $b$ abhängt, gilt

$$Pr[S_7] = \frac{1}{2}. \tag{A.9}$$

Um den Beweis abzuschliessen, wird noch folgendes Lemma benötigt.

**Lemma 4** *Es existiert ein PPP Algorithmus $A_3$, der im Wesentlichen dieselbe Laufzeit wie A hat, so dass gilt*

$$|Pr[S_7] - Pr[S_6]| = \mathrm{AdvIND}_{SKE,A_3}(\lambda). \tag{A.10}$$

Das Ergebnis folgt nun aus (A.1), (A.2), (A.3), (A.4), (A.5), (A.6), (A.7), (A.8), (A.9) und (A.10).

**Beweis von Lemma 2** Um Lemma 2 zu beweisen, wird gezeigt, wie ein Angreifer $\mathsf{A}_1$ von SIG konstruiert werden kann, der die angenommene starke existenzielle Unfälschbarkeit verletzt. Dieser Angreifer erfüllt die Bedingung der Schranke aus A.6.

Der Angreifer $\mathsf{A}_1$ wird bei der Ausführung von Angreifer A konstruiert. Auf die Abfragen von A wird wie folgt geantwortet.

- Wenn A die Abfrage an das Orakel zur Schlüsselerzeugung des Senders macht, befragt $\mathsf{A}_1$ sein Sender-Schlüsselerzeugungsorakel, um $\mathsf{PK}_s$ zu erhalten, und gibt $\mathsf{PK}_s$ zurück an A.

[1] $-$ soll darstellen, dass an dieser Stelle kein Zeichen eingegeben wird

- Wenn A die Abfrage an das Orakel zur Schlüsselerzeugung des Empfängers macht, führt $A_1$ KEM.KeyGen aus, um $PK_r$,$SK_r$ zu erhalten. Er hält $SK_r$ geheim und gibt $PK_r$ zurück an A. Zu diesem Zeitpunkt führt $A_1$ auch KEM.Enc($1^\lambda$,$PK_r$) aus, um $(K^*, \psi^*)$ zu erhalten. Dies wird geheimgehalten.

- Um auf die Abfrage von A an das Signieren/Verschlüsseln-Orakel für eine Nachricht zu antworten, beschafft sich $A_1$ zuerst $(K^*, \psi^*)$ für $PK_r$. Er prüft, ob $\psi = \psi^*$ und beendet im Falle ihrer Gleichheit die Simulation. Angenommen, es wird nicht abgebrochen, führt er anschließend eine Abfrage an sein Signier-Orakel aus, um eine Signatur $\sigma$ für $m||\psi$ zu erhalten. Danach erhält $A_1$ $\chi$, indem er SKE.Enc mit $m||\sigma$ und dem Schlüssel $K$ aufruft. Er antwortet dem Angreifer A mit $(\psi, \chi)$ und fügt $(K, m||\psi, \sigma)$ in eine ursprünglich leere Liste $L_s$ ein.

- Um auf die Abfrage von A an das Entschlüsseln/Verifikations-Orakel für den Chiffretext $(\psi, \chi)$ zu antworten, benutzt $A_1$ den zuvor erstellten Schlüssel $SK_r$, um durch Ausführung von KEM.Dec und SKE.Dec $m||\sigma$ zu erhalten. Für jede Entschlüsseln/ Verifikations-Anfrage, aus der eine gültige Nachricht resultiert, und führt er $A_1$, bevor er dem Angreifer A antwortet, noch Folgendes durch. Zuerst überprüft er, ob $(\hat{K}, \hat{m}||\psi, \hat{\sigma})$ in $L_s$ für ein beliebiges $\hat{K}, \hat{m}, \hat{\sigma}$ vorhanden ist. Es sind zwei Fälle zu betrachten.
  1. Falls es nicht vorhanden ist, stoppt $A_1$ die Simulation und gibt $(m||\psi, \sigma)$ aus.
  2. Andernfalls stoppt $A_1$ die Simulation ebenfalls, gibt aber $(\hat{m}||\psi^*, \hat{\sigma})$ aus.

- Angreifer $A_1$ antwortet auf die challenge-Anfrage $(m_0, m_1)$ von A, indem er sich zuerst ein Bit $b \leftarrow \{0, 1\}$ auswählt und anschließend sein Signier-Orakel befragt, um eine Signatur $\sigma^*$ für $m_b||\psi^*$ zu erhalten[2]. $A_1$ erhält $\chi^*$ durch die Verschlüsselung von $(m_b||\sigma^*)$ mit dem Schlüssel $K^*$ durch Aufruf von SKE.Enc. Zuletzt antwortet er A mit $(\psi^*, \chi^*)$.

A wird in der obigen Simulation durch $A_1$ genauso ausgeführt wie in den beiden Angriffsspielen $\mathbf{G}_3$ und $\mathbf{G}_4$, bis die Simulation in einem der beiden Fälle hält.

Es soll hier gezeigt werden, dass $A_1$ in beiden Fällen eine gültige Fälschung ausgibt. Fall 1 ist klar. Für Fall 2 sei angenommen, dass solch eine Anfrage gemacht wird, bevor der challenge-Chifftetext an A übergeben wird. In diesem Fall hat $A_1$ keine Anfrage an das Signieren-Orakel getätigt, in der $\psi^*$ beteiligt war. Damit ist $(\hat{m}||\psi^*, \hat{\sigma})$ eindeutig eine gültige Fälschung. Angenommen, diese Anfrage wird erst nach der Übergabe des challenge-Chiffretextes an A durchgeführt. Dann gilt entweder $\hat{m} \neq m_b$ oder $\hat{\sigma} \neq \sigma^*$, denn ansonsten wäre der challenge-Chiffretext selbst an das Entschlüsselungs/Verifikations-Orakel gesendet worden. Außerdem wurde insgesamt nur eine Signier-Abfrage von $A_1$ getätigt, in der $psi^*$ beteiligt war. Daraus kann geschlossen werden, dass $(\hat{m}||\psi^*, \hat{\sigma})$ eine gültige Fälschung ist. Alles zusammengefasst ergibt dies wie gewünscht

$$Pr[R_4] \leq \mathrm{AdvEF}_{SIG,A_1}(\kappa).$$

[2] $(K^*, \psi^*)$ wurde etwas früher in der Simulation erstellt

**Beweis von Lemma 3** Für den Beweis dieses Lemmas wird gezeigt, wie ein Angreifer A benutzt werden kann, um einen Angreifer $A_2$ von KEM zu konstruieren, der die echte oder zufällige Ununterscheidbarkeit brechen kann. Algorithmus $A_2$ arbeitet, indem eine Umgebung definiert wird, in der A ausgeführt wird. Er antwortet auf die Anfragen von A wie folgt.

- Zuerst befragt $A_2$ sein Empfängerschlüssel-Erzeugungsorakel, um $PK_r$ zu erhalten. Anschließend wird das challenge oracle befragt, um ein $(K^\dagger, \psi^*)$ zu bekommen.
- Wenn A seine Anfrage an das Schlüsselerzeugungsorakel des Senders tätigt, führt $A_2$ SIG.KeyGen aus, um ($PK_s$,$SK_s$) zu erhalten. Dabei hält er $SK_s$ geheim und gibt $PK_s$ an A.
- Wenn A seine Anfrage an das Schlüsselerzeugungsorakel des Empfängers tätigt, gibt $A_2$ $PK_r$ zurück an A.
- Angreifer $A_2$ antwortet auf Anfragen von A an das Signier/Entschlüsselungs- und an das Entschlüsselungs/Verifikations-Orakel in fast gleicher Weise wie in Angriffsspiel $\mathbf{G}_5$. Der einzige Unterschied liegt darin, dass Signaturen nun mit $SK_s$ anstelle eines Signier-Orakels erstellt werden. Der zu $PK_r$ entsprechende geheime Schlüssel ist wie im Spiel $\mathbf{G}_5$ nicht erforderlich, um $A_2$ auszuführen.
- Wenn A zwei Nachrichten $(m_0, m_1)$ an das challenge oracle sendet, wählt $A_2$ zuerst ein Bit $b \leftarrow \{0,1\}$ aus. Er benutzt $SK_s$, um Signaturen $\sigma^*$ für $m_b||\psi^*$ zu erstellen. Außerdem benutzt $A_2$ SKE.Enc, um die Verschlüsselung $\chi^*$ von $m_b||\sigma^*$ mit dem Schlüssel $K^\dagger$ zu erhalten. Zuletzt antwortet er A mit $(\psi^*, \chi^*)$.
- Zum Schluss der Simulation gibt A ein Bit $b'$ aus. Falls $b' = b$, gibt $A_2$ 1 aus, ansonsten 0.

Sei $d$ das interne Bit des challenge oracle von $A_2$ und $d'$ das von $A_2$ ausgegebene Bit. Durch Konstruktion ist folgendes gegeben

$$\begin{aligned} Pr[S_5] &= Pr[b' = b | d = 1] = Pr[d' = 1 | d = 1], \\ Pr[S_6] &= Pr[b' = b | d = 0] = Pr[d' = 1 | d = 0] \end{aligned}$$

und per Definition gilt

$$|Pr[S_6] - Pr[S_5]| = \mathrm{AdvRR}_{KEM,A_2}(\lambda).$$

**Beweis von Lemma 4** Der Beweis für dieses Lemma läuft in zwei Phasen ab. Zuerst wird die Konstruktion eines Angreifers $A_3$' gezeigt, der die Sicherheit von SKE bricht unter einer neuen Definition, die im Folgenden noch beschrieben wird. Anschließend soll gezeigt werden, wie dieser Angreifer benutzt werden kann, um einen neuen Angreifer $A_3$ zu konstruieren, der die angenommene Ununterscheidabrkeit der Verschlüsselung von SKE brechen kann.

Es soll ein neuer Sicherheitsbegriff für SKE eingeführt werden: *real or random indistinguishability under passive attack*. Ein Angreifer $A_3$', der einen Angriff gegen SKE plant, ist ein PPT Algorithmus mit der Eingabe $1^\lambda$ mit dem Sicherheitsparameter $\lambda \in \mathbb{Z}_{\geq 0}$. Das Angriffsspiel für diesen Sicherheitsbegriff läuft wie folgt ab.

**Phase 1:** Der Angreifer wählt eine Nachricht $m$ der Länge $l_m$ und übergibt diese Nachricht an ein Verschlüsselungsorakel.

**Phase 2:** Das Verschlüsselungsorakel leistet Folgendes.

$$\begin{aligned} K &\leftarrow \{0,1\}^{\mathsf{SKE.KeyLen}(\lambda)}; str \leftarrow \{0,1\}^{l_m}; b \leftarrow \{0,1\}; \\ \text{falls } b &= 1, \chi^* \leftarrow \mathsf{SKE.Enc}(1^\lambda, K, m); \\ \text{falls } b &= 0, \chi^* \leftarrow \mathsf{SKE.Enc}(1^\lambda, K, str); \end{aligned}$$

Es antwortet mit $\chi^*$.

**Phase 3:** Der Angreifer gibt $b' \in \{0,1\}$ aus.

Wenn $A_3$' dieses Angriffsspiel ausführt, definieren wir

$$\mathrm{AdvRR}_{SKE,A_3'}(\lambda) := |Pr[b' = 1|b = 1] - Pr[b' = 1|b = 0]|.$$

Die Wahrscheinlichkeit beruht auf den zufälligen Auswahlen von $A_3$' und auf denen von $A_3$'s Orakeln.

Nun soll gezeigt werden, wie solch ein Angreifer $A_3$' mit A konstruiert werden kann. Die Simulation von $A_3$' arbeitet wie folgt.

- $A_3$' führt zuerst KEM.KeyGen aus, um ($PK_r$,$SK_r$) zu erhalten. Er behält $PK_r$ und verwirft $SK_r$. Anschließend führt er noch KEM.Enc aus, um $(K^*, \psi^*)$ zu erhalten. Er behält $\psi^*$ und verwirft $K^*$.

- Wenn A seine Abfrage an das Sender-Schlüsselerzeugungsorakel stellt, führt $A_3$' SIG.KeyGen aus, um ($PK_s$,$SK_s$) zu erhalten. Er gibt $PK_r$ an A zurück und behält $SK_r$ geheim.

- Wenn A seine Abfrage an das Empfänger-Schlüsselerzeugungsorakel stellt, gibt $A_3$' $PK_r$ an A zurück.

- Angreifer $A_3$' antwortet auf Anfragen von A an das Signier/Entschlüsselungs- und an das Entschlüsselungs/Verifikations-Orakel in fast gleicher Weise wie in Angriffsspiel $\mathbf{G}_6$. Der einzige Unterschied liegt darin, dass Signaturen nun mit $SK_s$ anstelle eines Signier-Orakels erstellt werden.

- Wenn A zwei Nachrichten $(m_0, m_1)$ an das challenge oracle sendet, wählt $A_3$' zuerst ein Bit $b \leftarrow \{0,1\}$ aus. Er benutzt $SK_s$, um Signaturen $\sigma^*$ für $m_b||\psi^*$ zu erstellen. Außerdem befragt er sein challenge oracle, um die Verschlüsselung $\chi^*$ von $m_b||\sigma^*$ zu erhalten. Zuletzt antwortet er A mit $(\psi^*, \chi^*)$.

- Zum Schluss der Simulation gibt A ein Bit $b'$ aus. Falls $b' = b$, gibt $A_3$' 1 aus, andernfalls 0.

Sei $d$ das interne Bit des challenge oracle von $A_3$' und $d'$ das von $A_3$' ausgegebene Bit. Durch Konstruktion ist folgendes gegeben

$$\begin{aligned} Pr[S_6] &= Pr[b' = b|d = 1] = Pr[d' = 1|d = 1], \\ Pr[S_7] &= Pr[b' = b|d = 0] = Pr[d' = 1|d = 0] \end{aligned}$$

und per Definition gilt

$$|Pr[S_7] - Pr[S_6]| = \mathrm{AdvRR}_{SKE,A_3'}(\lambda).$$

Um den Beweis abzuschließen, wird gezeigt, wie aus Angreifer $A_3$' der Angreifer $A_3$ im Sinne der Ununterscheidbarkeit der Verschlüsselungen konstruiert werden kann. Die Konstruktion von $A_3$ läuft wie folgt ab.

- $A_3$' wird ausgeführt, bis er eine Nachricht $m$ der Länge $l_m$ ausgibt.
- Eine Zufallsnachricht $str$ der Länge $l_m$ wird ausgewählt ($str \leftarrow \{0,1\}^{l_m}$).
- Seien $m_0 \leftarrow str$ und $m_1 \leftarrow m$.
- $(m_0, m_1)$ wird an das Verschlüsselungsorakel von $A_3$' gesendet und der Chiffretext $\chi^*$ empfangen.
- $\chi^*$ wird an $A_3$' zurückgegeben.
- Am Schluss der Simulation gibt $A_3$' ein Bit $b'$ aus, das zurückgegeben wird.

Durch die Konstruktion oben ist klar, dass jeder Vorteil von $A_3$' direkt in einen Vorteil von $A_3$ übertragen werden kann. Es gilt wie gewünscht

$$\mathrm{AdvRR}_{SKE,A_3'}(\lambda) \leq \mathrm{AdvIND}_{SKE,A_3}(\lambda).$$