Interne Angreifer in der Kryptographie: Digitale Signatur und Schlüsselaustausch

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

genehmigte

Dissertation

von

Jens-Matthias Bohli

aus Kiel

Tag der mündlichen Prüfung: 2. Februar 2007
Erster Gutachter: Dr. Jörn Müller-Quade
Zweiter Gutachter: Prof. Dr. Roland Vollmar

Dank

Mein erster Dank gilt Herrn Professor Dr. Thomas Beth (1949-2005). Leider konnte er die Fertigstellung dieser Arbeit nicht mehr miterleben. Für die Betreuung dieser Arbeit und die fachliche Zusammenarbeit danke ich Herrn Dr. Jörn Müller-Quade. Herrn Professor Dr. Roland Vollmar danke ich für seine Unterstützung und für die Übernahme des Korreferats. Ein herzlicher Dank gilt Herrn Professor Dr. Rainer Steinwandt, der durch zahlreiche Diskussionen und gemeinsame Arbeiten wesentlich zum Gelingen dieser Arbeit beigetragen hat.

Bei Frau Professor Dr. María Isabel González Vasco und meinen Kollegen Dr. Willi Geiselmann, Dr. Dennis Hofheinz, Stefan Röhrich und Dr. Dominique Unruh bedanke ich mich für die vielen fachlichen Gespräche und bei allen weiteren Kollegen am Institut für Algorithmen und Kognitive Systeme bedanke ich mich für das angenehme Arbeitsumfeld. Für wertvolle Kommentare zu einer frühen Version dieser Arbeit danke ich Dr. Thomas Decker.

Ein besonderer Dank gilt meinen Eltern für ihre Unterstützung während meiner Ausbildung und Promotion.

Inhaltsverzeichnis

1 Einleitung **1**

2 Angriffe auf Digitale Signaturen **3**
2.1 Digitale Signaturen 3
2.1.1 Definitionen 3
2.1.2 Diskussion der Sicherheitsdefinition 5
2.2 Schlüsselersetzungsangriffe 8
2.2.1 Das Station-to-Station-Protokoll 8
2.2.2 Ein verallgemeinertes Szenario 9
2.2.3 Modellierung von Schlüsselersetzungsangriffen 10
2.2.4 Beispiele für Schlüsselersetzungsangriffe 11
2.2.5 Eine Abwehrmaßnahme auf Protokollebene 13
2.3 Schlüsselersetzungsangriffe auf etablierte Signaturverfahren 15
2.3.1 Das Signaturverfahren RSA-PSS 15
2.3.2 Das Signaturverfahren DSA 16
2.3.3 Die Signaturverfahren EC-DSA und EC-GDSA 17
2.3.4 Das Signaturverfahren EC-KCDSA 20
2.4 Weitere Angriffe auf Signaturverfahren 22
2.5 Zusammenfassung und offene Fragen 23

3 Subliminale Kanäle **25**
3.1 Geschichte der subliminalen Kanäle 26
3.2 Subliminale Kanäle heute 27
3.3 Formalisierung von subliminalen Kanälen 29
3.3.1 Signatur mit subliminalem Kanal 29
3.3.2 Kapazität eines subliminalen Kanals 32
3.3.3 Codierung der subliminalen Nachricht 33
3.4 Subliminale Kanäle in deterministischen Signaturverfahren 34
3.4.1 Ein zopfgruppenbasiertes Verfahren 34
3.4.2 Ein polynombasiertes Verfahren: SFLASH^{v3} 35
3.4.3 Ein faktorisierungsbasiertes Verfahren: RSA-PSS 37
3.4.4 Ein faktorisierungsbasiertes Verfahren: ESIGN-D 39
3.5 Formalisierung von Subliminalfreiheit 41

3.5.1 Subliminalfreie Signaturverfahren 41
3.5.2 Eine subliminalfreie Variante von RSA-PSS 42
3.5.3 Verallgemeinerung: Subliminalfrei mit Wächter 44
3.5.4 Eine subliminalfreie Variante von EC-DSA 47
3.6 Beziehungen zu weiteren Eigenschaften 56
3.7 Zusammenfassung und offene Fragen 57

4 Schlüsselaustausch für Gruppen 59
4.1 Grundlagen 60
4.1.1 Sicherheitsmodell 63
4.1.2 Diskussion und Einordnung des Sicherheitsmodells 67
4.2 Angriffe interner Protokollteilnehmer 67
4.2.1 Angriffe auf das Protokoll von Katz und Yung 68
4.2.2 Angriffe auf das Protokoll von Kim, Lee und Lee 70
4.3 Erweiterte Sicherheitseigenschaften 73
4.4 Ein Rahmenwerk für sichere Protokolle 76
4.4.1 Ein gegen interne Angreifer sicheres Protokoll 76
4.4.2 Key-Confirmation 78
4.4.3 Die Sitzungsnummer 79
4.4.4 Beschreibung des Rahmenwerks 81
4.4.5 Sicherheit 82
4.5 Anwendungen des Rahmenwerks 85
4.6 Ein effizientes sicheres Protokoll 86
4.7 Zusammenfassung und offene Fragen 92

5 Zusammenfassung 93

6 Ausblick 95

A Definitionen 97

Literaturverzeichnis 99

Eigene Veröffentlichungen 113

Index 115

1. Einleitung

Ein Großteil von Angriffen auf Computersysteme geht von internen Angreifern aus [52, 35]. Da kryptographische Sicherheitsmechanismen häufig ein wesentlicher Baustein in der Sicherheitsarchitektur sind, stellt sich die Frage, inwiefern schon sie anfällig gegen interne Angreifer sind (vgl. [68]). Dazu werden in dieser Arbeit mit Signaturverfahren und Schlüsselaustauschprotokollen die verbreitetsten Anwendungen der Public-Key-Kryptographie untersucht. Es werden jeweils verbesserte Sicherheitsdefinitionen entwickelt, etablierte Verfahren und Protokolle in den neuen Modellen beweisbar untersucht und gegebenenfalls neue Protokolle präsentiert.

Eine Signatur soll ein Dokument in nicht abstreitbarer Weise an die Person binden, die das Dokument signiert hat. Die nach derzeitigem Kenntnisstand wichtigsten Angriffe auf Signaturverfahren, die nicht durch die zur Zeit etablierte Sicherheitsanforderung erfasst werden, sind sogenannte Schlüsselersetzungsangriffe, welche die Zuordnung der Signatur zu einer Person schwächen. In der bisherigen Modellierung der Schlüsselersetzungsangriffe wurde angenommen, dass sich der Unterzeichner eines Dokuments ehrlich verhält. Jedoch kann der Unterzeichner ein Interesse daran haben, einen solchen Angriff herbeizuführen, um eine Signatur abzustreiten. Diese Arbeit betrachtet daher als neue Angriffe auf Signaturverfahren Schlüsselersetzungsangriffe, bei denen der Unterzeichner unehrlich sein kann. In diesem Szenario werden die relevanten Signaturverfahren untersucht, Angriffe aufgezeigt und sichere Varianten beschrieben.

Ein anderer wichtiger Sicherheitsaspekt bei Signaturen sind subliminale Kanäle. Subliminale Kanäle stellen keinen unmittelbaren Angriff auf das Signaturverfahren dar, sondern einen Missbrauch des Signaturverfahrens. Sie ermöglichen, dass zwei Parteien durch den Austausch von Signaturen kommunizieren können, ohne dass der Informationsaustausch von Anderen bemerkt wird. Signaturen können dadurch überraschenderweise ein ganzes System unsicher machen: Dies war beispielsweise bei einem geplanten System zur Kontrolle der nuklearen Rüstungsbegrenzung der Fall. Auf der einen Seite ist eine Signatur zur Authentifizierung und Sicherung der Integrität nötig,

andererseits ermöglicht ein unachtsames Einfügen der Signatur dem Unterzeichner einen verdeckten Kanal, über den er unbemerkt geheime Informationen versenden kann. Solche subliminalen Kanäle werden in der vorliegenden Arbeit formal gefasst. Bei der Analyse wichtiger Signaturverfahren werden neue subliminale Kanäle identifiziert. Darüber hinaus werden sichere Verfahren entwickelt, bei denen ein solcher Missbrauch unmöglich ist.

Es folgt die Untersuchung von Schlüsselaustauschprotokollen als zweiter Schwerpunkt der Arbeit. Schlüsselaustausch zwischen zwei Parteien ist relativ gut untersucht und wird beispielsweise im Internet oft eingesetzt. Aktuell besteht zunehmendes Interesse nach Schlüsselaustauschprotokollen für mehr als zwei Parteien, sogenannten Gruppenschlüsselaustauschprotokollen. Solche Protokolle können beispielsweise bei Video-Konferenzen oder Pay-TV-Übertragungen eingesetzt werden. Es stellt sich heraus, dass für Gruppenschlüsselaustauschprotokolle neuartige Angriffe durch interne Angreifer relevant sind, die im Zweiparteienfall nicht möglich sind. Es wird gezeigt, dass etablierte Protokolle gegen diese Angriffe anfällig sind, weil im bisher üblicherweise zugrundeliegenden Sicherheitsmodell solche Angriffe nicht betrachtet werden. In der vorliegenden Arbeit wird das Sicherheitsmodell erweitert, und es werden Protokolle vorgestellt, die sicher gegen Angriffe interner Benutzer sind. Darüber hinaus werden wichtige Merkmale sicherer Protokolle identifiziert und ein Protokoll-Rahmenwerk entwickelt, das es ermöglicht, beliebige Protokolle so zu verbessern, dass sie sicher gegen interne Angriffe sind.

2. Angriffe auf Digitale Signaturen

Eine digitale Signatur soll – ähnlich einer handschriftlichen Unterschrift – die Authentizität eines elektronischen Dokuments in einer für den Unterzeichner nicht abstreitbaren Weise sicherstellen. Als Sicherheitsdefinition für digitale Signaturverfahren hat sich der Sicherheitsbegriff EUF-CMA („*existentially unforgeable under adaptive chosen-message attacks*“) durchgesetzt. Dieser Sicherheitsbegriff genügt allerdings unter Umständen nicht, um die geforderte Nichtabstreitbarkeit sicherzustellen: mittels sogenannter Schlüsselersetzungsangriffe konnte beispielsweise ein auf digitalen Signaturen aufbauendes Schlüsselaustauschprotokoll erfolgreich angegriffen werden. Die bisher bekannten Schlüsselersetzungsangriffe geben dem Angreifer jedoch nicht alle Freiheiten, die er plausibel haben kann. Dieses Kapitel betrachtet ein erweitertes Szenario der Schlüsselersetzungsangriffe, in dem auch der Unterzeichner an dem Angriff mitwirken kann. Nach einer Motivation und Formalisierung der Angriffe werden zusätzliche Schlüsselersetzungsangriffe auf Signaturverfahren aufgezeigt und etablierte Signaturverfahren auf ihre Sicherheit in dem neuen Szenario untersucht. Die Ergebnisse dieses Kapitels sind größtenteils in Bohli, Röhrich und Steinwandt [22] veröffentlicht.

2.1 Digitale Signaturen

Zunächst werden die etablierten Definitionen eines Signaturverfahrens und seiner Sicherheit vorgestellt.

2.1.1 Definitionen

Definition 2.1 (Signaturverfahren)**.** *Ein* Signaturverfahren $S = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *ist ein Tripel von Algorithmen mit den folgenden Eigenschaften:*

- *Der* Schlüsselgenerierungsalgorithmus Gen *ist ein PPT-Algorithmus.*[1] *Die Eingabe ist der Sicherheitsparameter ℓ und die Ausgabe ist ein Schlüsselpaar (PK, SK) mit einem öffentlichen Verifikationsschlüssel PK und einem geheimen Signaturschlüssel SK.*

- *Der* Signieralgorithmus Sig *ist ein PPT-Algorithmus. Die Eingaben sind eine Nachricht M sowie der geheime Signaturschlüssel SK und die Ausgabe ist eine gültige Signatur σ für die Nachricht M bezüglich des öffentlichen Schlüssels PK.*

- *Der* Verifikationsalgorithmus Ver *ist ein deterministischer, polynomiell beschränkter Algorithmus. Die Eingaben sind ein öffentlicher Schlüssel PK, eine Nachricht M und eine Signatur σ. Die Ausgabe ist* valid *– mit der Bedeutung, dass σ eine gültige Signatur für die Nachricht M unter dem Verifikationsschlüssel PK ist – oder* invalid.

Die *Korrektheit* für Signaturverfahren fordert, dass eine von Sig berechnete Signatur auf jeden Fall valid bezüglich Ver ist. Die formale *Sicherheit* von Signaturverfahren wurde eingehend in Goldwasser, Micali und Rivest [65, 66] untersucht. Angriffe auf Signaturverfahren wurden nach dem Ziel des Angreifers und den ihm zur Verfügung stehenden Hilfsmitteln klassifiziert. Die Ziele des Angreifers sind für eine gegebene Instanz des Signaturverfahrens:

- „*total break*“: der geheime Signaturschlüssel des Benutzers wird berechnet

- „*universal forgery*“: zu jeder vorgegebenen Nachricht kann mit einer signifikanten Wahrscheinlichkeit eine Signatur berechnet werden

- „*existential forgery*“: ein Nachricht/Signatur-Paar kann berechnet werden

Dem Angreifer stehen, um sein Ziel zu erreichen, verschiedene Hilfsmittel zur Verfügung. Er darf in einem „*no-message*“-Angriff nur auf den öffentlichen Schlüssel zurückgreifen. In einem „*known-message*“-Angriff erhält er auch Nachricht/Signatur-Paare und in einem „*chosen-message*“-Angriff darf er die Nachrichten dazu selber auswählen. Als stärkste Angriffsart wurde in [65] ein „*adaptive chosen-message*“-Angriff betrachtet, in dem der Angreifer Zugriff auf ein Signaturorakel hat und damit *jederzeit* gültige Signaturen zu beliebig gewählten Nachrichten erhalten kann.

Der stärkste Sicherheitsbegriff ergibt sich nun aus der Kombination des „leichtesten“ Ziels mit den mächtigsten Hilfsmitteln und ist der zur Zeit etablierte Begriff *existential unforgeability under adaptive chosen-message attacks* (EUF-CMA). Er besagt, dass es für jeden PPT-Angreifer selbst mit Zugriff auf ein Signaturorakel unmöglich ist, für einen gegebenen öffentlichen Schlüssel ein gültiges Nachricht/Signatur-Paar für diesen Schlüssel auszugeben, wobei der Angreifer für die ausgegebene Nachricht keine Anfrage an das Signaturorakel gestellt haben darf. Die formale Definition von EUF-CMA ist die folgende:

[1] siehe Definition A.2

Definition 2.2 (EUF-CMA)**.** *Ein Signaturverfahren* $S = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *ist* EUF-CMA*, falls für jeden PPT-Algorithmus* $\mathcal{A}$ *eine im Sicherheitsparameter* ℓ *vernachlässigbare*[2] *Funktion* $\mathrm{negl}(\ell)$ *existiert, so dass für das Experiment*

$$\begin{aligned} &\textit{Experiment } \mathbf{Exp}^{\mathrm{euf-cma}}_{S,\mathcal{A}}(\ell): \\ &\quad (PK, SK) \leftarrow \mathsf{Gen}(1^\ell); \\ &\quad (M, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sig}(SK,\cdot)}(1^\ell, PK); \\ &\quad \textit{return } \mathsf{Ver}(PK, M, \sigma); \end{aligned}$$

mit der Einschränkung, dass $\mathcal{A}$ *die Nachricht* M *nie an sein Signaturorakel* $\mathsf{Sig}(SK, \cdot)$ *gegeben haben darf, gilt, dass*

$$\Pr(\mathbf{Exp}^{\mathrm{euf-cma}}_{S,\mathcal{A}}(\ell) \rightarrow \mathsf{valid}) = \mathrm{negl}(\ell).$$

Eine wichtige Erweiterung von EUF-CMA ist *strongly EUF-CMA* oder SEUF-CMA. Dabei darf es dem Angreifer zusätzlich auch nicht möglich sein, eine *neue* Signatur für eine bereits signierte Nachricht zu generieren. In dem oben erwähnten Experiment muss also die Wahrscheinlichkeit selbst dafür vernachlässigbar sein, dass der Angreifer (M, σ) mit einer gültigen Signatur σ für M ausgibt und das Signaturorakel zwar nach einer Signatur für M gefragt wird, die Antwort aber nicht σ ist. Formal muss in dem Experiment $\mathbf{Exp}^{\mathrm{seuf-cma}}_{S,\mathcal{A}}(\ell)$ die Einschränkung also lauten, dass der Angreifer mit Ausgabe (M, σ) nicht σ als Antwort des Signaturorakels auf eine Anfrage $\mathsf{Sig}(SK, M)$ erhalten hat. Ein Verfahren, das zwar EUF-CMA sicher ist, jedoch nicht SEUF-CMA, wird auch „*malleable*“ genannt.

Goldwasser, Micali und Rivest [65, 66] weisen darauf hin, dass ihre Liste von Angriffen nicht notwendigerweise vollständig ist und es Angriffe auf Signaturverfahren geben kann, die selbst durch SEUF-CMA nicht verhindert werden. Fälschlicherweise wird aber häufig angenommen, dass EUF-CMA die *bestmögliche* Sicherheit für digitale Signaturen bietet (vgl. [82, 2, 131]). Dennoch bieten EUF-CMA-sichere Verfahren sehr starke Garantien, die in vielen Einsatzszenarien auch ausreichend sind.

2.1.2 Diskussion der Sicherheitsdefinition

Eine Frage, die Definition 2.1 und Definition 2.2 offenlassen, ist die Bindung des öffentlichen Schlüssels und der Signaturen an Personen in der realen Welt. Üblicherweise wird die Bindung eines öffentlichen Schlüssels an eine Person durch Zertifikate erreicht, die der Person von einer Zertifizierungsstelle ausgestellt werden. Ein *Zertifikat* enthält im Wesentlichen die Identität einer Person sowie einen öffentlichen Schlüssel für ein spezifiziertes Signaturverfahren und ist durch eine Zertifizierungsstelle unterschrieben. Die Zertifizierungsstelle garantiert damit, dass der im Zertifikat genannte Schlüssel tatsächlich der dort genannten Person gehört.

[2] siehe Definition A.1

Für die Bindung zwischen einer digitalen Signatur und einer Person spielt es auch eine Rolle, wie ein Benutzer ein Zertifikat zu seinem Schlüssel erhält bzw. wie er überhaupt sein Schlüsselpaar erhält. Um ein Schlüsselpaar und ein Zertifikat zu erhalten, gibt es prinzipiell zwei Möglichkeiten:

- Der Benutzer führt selbst den Algorithmus Gen aus, um sich ein Schlüsselpaar zu erzeugen. Anschließend kann sich der Benutzer bei einer Zertifizierungsstelle ein Zertifikat ausstellen lassen, das den öffentlichen Schlüssel an ihn bindet.
- Die Zertifizierungsstelle erzeugt ein Schlüsselpaar und das Zertifikat und lässt dem Benutzer seinen geheimen Schlüssel und sein Zertifikat auf einem sicheren Weg zukommen.

Eine Schwäche der erstgenannten Methode ist, dass eine unehrliche, d. h. von Gen abweichende, Schlüsselerzeugung möglich ist. Bevor eine Zertifizierungsstelle in diesem Fall einen Schlüssel als gültig anerkennt, sollte also sichergestellt sein, dass der Schlüssel ehrlich generiert wurde und dem Benutzer der passende geheime Schlüssel bekannt ist. Dazu wird ein Algorithmus oder ein Protokoll GenVer benötigt, das – möglicherweise unter aktiver Beteiligung des Schlüsselinhabers – beweist, dass der Schlüssel für das gegebene Verfahren gültig und der geheime Schlüssel bekannt ist.

Definition 2.3 (Schlüsselverifikation)**.** *Zu einem Signaturverfahren S mit den Algorithmen* Gen, Sig *und* Ver *ist die* Schlüsselverifikation GenVer *ein Zwei-Parteien-Zero-Knowledge-Protokoll, das einem Beweiser mit einem Schlüsselpaar (PK, SK) ermöglicht, einen Verifizierer, der nur PK kennt, davon zu überzeugen,*[3] *dass PK ein gültiger Schlüssel zu dem ihm bekannten geheimen Schlüssel SK im Signaturverfahren S ist.*

Eine andere Betrugsmöglichkeit in diesem Szenario wäre, dass ein Benutzer einen bereits (für einen anderen Benutzer) existierenden Schlüssel nimmt und versucht, sich dafür ein Zertifikat ausstellen zu lassen. Falls er den geheimen Schlüssel nicht kennt, kann das ebenfalls durch die Verpflichtung, mittels GenVer der Zertifizierungsstelle Korrektheit und Kenntnis des geheimen Schlüssels zu beweisen, verhindert werden. Jedoch hilft das nicht, falls beide Benutzer kooperieren und den geheimen Schlüssel teilen. Dagegen hilft nur, dass eine Zertifizierungsstelle zusätzlich prüft, ob der zu zertifizierende Schlüssel noch nicht für einen anderen Benutzer registriert ist. Jedoch wird das mit steigender Anzahl existierender Zertifizierungsstellen allmählich unmöglich, wie in der folgenden Bemerkung festgehalten wird.

Bemerkung 2.4. *Falls mehrere Zertifizierungsdienste existieren, ist es plausibel, dass sie untereinander nicht abgleichen, ob ein zu zertifizierender öffentlicher Schlüssel nicht bereits zuvor unter einem anderen Namen bei einer anderen Zertifizierungsstelle zertifiziert wurde. Dann ist es möglich, dass zwei unterschiedliche Nutzer Zertifikate*

[3] Das bedeutet, das der Verifizierer einen Zustand annimmt, den er nur mit vernachlässigbarer Wahrscheinlichkeit annimmt, falls die Aussage nicht zutrifft.

für einen identischen *öffentlichen Schlüssel haben und sich möglicherweise den gleichen geheimen Signaturschlüssel teilen.*

Falls die Zertifizierungsstelle den Schlüssel generiert, ist die Gefahr einer unehrlichen Schlüsselerzeugung gering. Der Zertifizierungsstelle muss bezüglich des Zertifizierens von Schlüsseln ohnehin vertraut werden. Die Zertifizierungsstelle muss daran interessiert sein, den Schlüssel ehrlich zu erzeugen, um die Glaubwürdigkeit der Zertifikate zu erhalten. Wenn die Zertifizierungsstelle den Schlüssel erzeugt, muss das Vertrauen jedoch dahingehend höher sein, dass sie den geheimen Schlüssel nach der Übergabe an den Benutzer nicht behält, sondern sicher löscht. Eine weitere Schwierigkeit liegt darin, dem Benutzer den geheimen Schlüssel vertraulich zukommen zu lassen.

Das Signaturgesetz [107] kennt sogenannte *qualifizierte* und *fortgeschrittene* Signaturen. Es verlangt beispielsweise für eine *qualifizierte Signatur*, die gesetzlich einer handschriftlichen Signatur gleichgestellt wird, dass die Signaturen auf einem *qualifizierten Zertifikat* beruhen und mit einer *sicheren Signaturerstellungseinheit* erzeugt werden. Dabei ist ein qualifiziertes Zertifikat von einer Zertifizierungsstelle, die gewisse gesetzlich vorgegebene Standards erfüllt, ausgestellt. Eine sichere Signaturerstellungseinheit ist im Allgemeinen eine Chipkarte, welche die Signatur korrekt ausführt und ein Auslesen des geheimen Schlüssels verhindert. Das Schlüsselpaar kann entweder direkt von der Karte generiert werden oder zunächst außerhalb der Karte generiert und dann auf die Karte übertragen werden. Die Zertifizierungsstelle muss garantieren, dass ein eventuell außerhalb der Karte erzeugter Signaturschlüssel sicher gelöscht wird, so dass der Signaturschlüssel letztendlich nur auf der Chipkarte in einem sicheren Speicher vorhanden ist. Durch eine solche vertikale Interoperabilität mit mehreren Sicherheitsniveaus sind qualifizierte Signaturen in der Tat nicht von den hier aufgeführten Angriffen betroffen, da die Schlüsselerzeugung und Signaturberechnung in einer vertrauenswürdigen Umgebung stattfinden.

Für die weiter verbreiteten *fortgeschrittenen Signaturen* müssen Angriffe, die trotz EUF-CMA möglich sind, jedoch beachtet werden. Fortgeschrittene Signaturen müssen nach dem Signaturgesetz nicht von einer Chipkarte berechnet werden, der geheime Schlüssel darf dem Inhaber bekannt sein. Auch die hohen Anforderungen an die Zertifikatserstellung bei den qualifizierten Signaturen fallen für fortgeschrittene Signaturen weg. Dennoch verlangt das Signaturgesetz, dass eine fortgeschrittene Signatur „ausschließlich dem Signaturschlüssel-Inhaber zugeordnet ist“. Die Bindung zwischen Signatur und einer Person scheint in diesem Fall durch EUF-CMA nicht garantiert zu sein, wie die in diesem Kapitel untersuchten sogenannten Schlüsselersetzungsangriffe zeigen werden.

Es bleibt anzumerken, dass auch in dem UC-Modell für Universelle Komponierbarkeit [44], das üblicherweise zu sehr starken Sicherheitsbegriffen führt, die aktuelle Formalisierung für digitale Signaturen von Backes und Hofheinz [4] oder Canetti [45] „nur“ äquivalent zu EUF-CMA ist. Sämtliche Angriffe, die in diesem Kapitel vorgestellt werden, werden momentan auch im UC-Modell nicht berücksichtigt. Es

konnte aber in einer Diplomarbeit gezeigt werden, dass sich die Ergebnisse dieses Kapitels auch in das UC-Modell übertragen lassen und die Formulierung der idealen Funktionalität entsprechend angepasst werden kann, um selbst das erweiterte Szenario der Schlüsselersetzungsangriffe mit unehrlichem Unterzeichner zu berücksichtigen (siehe GÜTTINGER [70]).

2.2 Schlüsselersetzungsangriffe

Bei einem Schlüsselersetzungsangriff im Sinne von [14] ist ein unter einem Schlüssel PK gültiges Nachricht/Signatur-Paar (M, σ) gegeben und der Angreifer schafft es, einen öffentlichen Schlüssel $\overline{PK}$ zu konstruieren, so dass (M, σ) auch bezüglich $\overline{PK}$ gültig ist. Ein Schlüsselersetzungsangriff hat ähnliche Auswirkungen wie die Situation vor und in Bemerkung 2.4. Es gibt zwei Benutzer denen eine Signatur zugeordnet werden kann. Bei einem Schlüsselersetzungsangriff haben die beiden Benutzer allerdings unterschiedliche Schlüssel, so dass keine Möglichkeit besteht, durch Vergleich der bereits zertifizierten Schlüssel den Angriff zu bemerken. Die Relevanz von Schlüsselersetzungsangriffen kann anhand des folgenden Schlüsselaustauschprotokolls demonstriert werden.

2.2.1 Das Station-to-Station-Protokoll

Das Station-to-Station-Protokoll ist ein Schlüsselaustauschprotokoll, das als Grundlage viele andere Protokolle beeinflusst hat, beispielsweise die für IPSEC verwendete IKE Protokollfamilie. Anhand dieses Protokolls werden in BLAKE-WILSON UND MENEZES [14] Schlüsselersetzungsangriffe eingeführt. Es wird gezeigt, dass in dem Station-to-Station-Protokoll eine Schlüsselersetzung einen sogenannten „*unknown key-share*“ Angriff ermöglicht. Eine Übersicht über das Station-to-Station-Protokoll zeigt Bild 2.1.

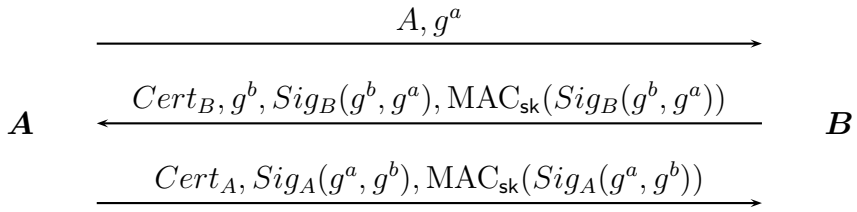

Bild 2.1: Das Station-to-Station-Protokoll

In der dritten Nachricht authentifiziert sich A bei B, indem A die beiden Werte g^a, g^b signiert und die Signatur mit einem „*message authentication code*“ (MAC) bezüglich des Sitzungsschlüssels $\mathsf{sk} = g^{ab}$ authentifiziert. Die Aussage dieser Nachricht

ist scheinbar: „Wer die Signatur erstellt hat, der ist der Partner, der den Sitzungsschlüssel kennt." Ein Angreifer C kann jedoch die erste Nachricht von A durch C, g^a ersetzen und die dritte Nachricht durch

$$Cert_C, Sig_C(g^a, g^b), \mathrm{MAC}_{\mathsf{sk}}(Sig_A(g^a, g^b)).$$

Dann ist der MAC weiterhin gültig, falls ihm ein Schlüsselersetzungsangriff gelingt, d. h., falls er einen öffentlichen Schlüssel generieren kann und ein Zertifikat $Cert_C$ bekommt, so dass seine Signatur $Sig_C(g^a, g^b) = Sig_A(g^a, g^b)$ mit As Signatur übereinstimmt, und B glaubt *fälschlicherweise*, den Schlüssel mit C ausgetauscht zu haben. C benötigt einen Schlüsselersetzungsangriff, da er den MAC bezüglich des ihm unbekannten sk nicht fälschen kann.

2.2.2 Ein verallgemeinertes Szenario

Um die Frage zu beantworten, ob es weitere relevante Angriffe geben kann, die nicht durch EUF-CMA verhindert werden, haben Menezes und Smart [91] Signaturen in einem Mehrparteien-Szenario betrachtet. Dabei wurde untersucht, welche Angriffe auftreten können, wenn mehrere Schlüssel existieren, es also nicht offensichtlich ist, zu welchem Schlüssel eine Signatur gehört. Sie haben dabei die Schlüsselersetzungsangriffe („*key substitution attacks*"), sowie MKS-Angriffe (für „*message and key substitution*") als einzige zusätzliche relevante Angriffe in ihrem Modell identifiziert. Ein MKS-Angriff ist gegeben, wenn eine gültige Signatur σ für M unter dem Schlüssel PK ebenfalls eine gültige Signatur für eine Nachricht $\overline{M} \neq M$ unter dem Schlüssel $\overline{PK} \neq PK$ ist. Für $M = \overline{M}$ geht ein MKS-Angriff in einen Schlüsselersetzungsangriff über. MKS-Angriffe sind jedoch von geringerer Bedeutung als Schlüsselersetzungsangriffe, da nur selten Signaturen unabhängig von einer Nachricht gespeichert werden, es also in der Regel klar ist, welche Nachricht signiert wurde. Diese Angriffe werden hier nicht weiter behandelt.

Es wurden in [91] allerdings nur Szenarien betrachtet, in denen der Unterzeichner sein Schlüsselpaar (PK, SK) ehrlich berechnet, alle Signaturen ehrlich erzeugt und seinen Signaturschlüssel SK geheim hält.

Dieses Szenario hat durchaus seine Berechtigung, die folgenden Betrachtungen gehen aber weiter als in [91] und beziehen auch unehrliche Unterzeichner mit ein. In einigen Fällen mag es auch für den ursprünglichen Unterzeichner der Nachricht vorteilhaft sein, an der Schlüsselersetzung mitzuwirken, indem er beispielsweise seinen geheimen Schlüssel in die Berechnung des neuen Schlüssels einfließen lässt oder indem er schon bei der Signaturberechnung abweicht, um den Angriff vorzubereiten. Ein motivierendes Beispiel aus [120] kann entsprechend weiterentwickelt werden: Der Unterzeichner eines elektronischen Lottoscheins hat einen bestimmten Anspruch auf etwas. Falls ein Schlüsselersetzungsangriff dazu führt, dass zusätzlich auch die angreifende Partei in den Genuss des Lottoscheins kommt, läge der Schaden ausschließlich beim Anbieter. In diesem Szenario kann es plausibel sein, dass der Unterzeichner

von Beginn an kooperiert und beispielsweise eine „schwache“ Signatur produziert, die einen Schlüsselersetzungsangriff erleichtert. Es ist sogar möglich, dass der Unterzeichner schon sein Schlüsselpaar im Hinblick auf die Schlüsselersetzung wählt oder bereit ist, seinen geheimen Signaturschlüssel mit dem Angreifer zu teilen.

Auch Rosa [105] hat Schlüsselersetzungsangriffe betrachtet. Seine Definition eines *kooperativen* Schlüssel-Kollisions-Suchalgorithmus geht weiter als die Definition aus [91], indem sie unehrliche Unterzeichner zulässt, die bereit sind, Information über ihren geheimen Schlüssel dem Angreifer zu offenbaren. Jedoch nutzen die Angriffe auf DSA und EC-DSA in [105] nur das nicht-kooperative Szenario. Das hier betrachtete Szenario ist noch allgemeiner, da es auch eine unehrliche Schlüsselgenerierung und Signaturberechnung des ersten Benutzers zulässt.

Es konnte für zahlreiche Verfahren gezeigt werden, dass sie anfällig für Schlüsselersetzungsangriffe – noch im Modell mit ehrlichem Unterzeichner – sind [14, 91, 120, 64]. Menezes und Smart [91] konnten jedoch auch für verbreitete Signaturverfahren wie DSA und EC-DSA mit geeigneten Parametern zeigen, dass sie unter gewissen Annahmen sicher gegen Schlüsselersetzungsangriffe sind – solange der Unterzeichner ehrlich ist. In Abschnitt 2.3 soll die Gültigkeit dieser Bedingungen in einem Szenario mit unehrlichem Unterzeichner untersucht werden.

2.2.3 Modellierung von Schlüsselersetzungsangriffen

Die Aufgabe des Angreifers, um im Modell aus [91][4] einen Schlüsselersetzungsangriff herbeizuführen, ist, aus gegebener Nachricht M und Signatur σ für einen Verifikationsschlüssel PK einen neuen Verifikationsschlüssel $\overline{PK}$ zu berechnen, so dass σ auch eine Signatur für M bezüglich $\overline{PK}$ ist. In einer Variante des Angriffs muss der Angreifer zu dem neuen öffentlichen Schlüssel $\overline{PK}$ auch einen geheimen Signaturschlüssel $\overline{SK}$ berechnen können. Im Falle eines *unehrlichen* Unterzeichners bekommt der Angreifer keine Eingabe, sondern hat die Freiheit, beide Schlüssel PK und $\overline{PK}$ sowie die Nachricht M und Signatur σ selbst zu wählen. Es folgt eine formale Definition der Schlüsselersetzungsangriffe, wobei direkt die Möglichkeit eines betrügenden Unterzeichners miteinbezogen wird.

Definition 2.5 (Schlüsselersetzungsangriff)**.** *Ein* Schlüsselersetzungsangriff („*key substitution attack*“) *auf ein Signaturverfahren* $S = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *ist ein PPT-Algorithmus* $\mathcal{A}$*, der zwei öffentliche Schlüssel* PK*,* $\overline{PK}$ *und ein Nachricht/Signatur-Paar* (M, σ) *mit* $\mathsf{Ver}(M, \sigma, PK) = \mathsf{true}$ *und* $\mathsf{Ver}(M, \sigma, \overline{PK}) = \mathsf{true}$ *mit signifikanter*[5] *Wahrscheinlichkeit ausgibt.*

Der Schlüsselersetzungsangriff heißt schwach*, falls* $\mathcal{A}$ *auch geheime Signaturschlüssel* SK *und* $\overline{SK}$*, die zu* PK *beziehungsweise* $\overline{PK}$ *passen (und, falls definiert,* GenVer *erfüllen), ausgeben muss, sonst wird der Angriff* stark *genannt.*

[4] also bei *ehrlichem* Unterzeichner

[5] siehe Anhang A

Falls Zertifikate benötigt werden, steht dem Angreifer $\mathcal{A}$ ein Zertifizierungsorakel zur Verfügung.

Bevor im Abschnitt 2.3 für etablierte Signaturverfahren die Anfälligkeit gegenüber Schlüsselersetzungsangriffen zusammengefasst wird, sollen zunächst zur Verdeutlichung zwei Schlüsselersetzungsangriffe auf weniger verbreitete Signaturverfahren vorgestellt werden. Zuerst ein Schlüsselersetzungsangriff auf ein kürzlich vorgestelltes Signaturverfahren von BONEH UND BOYEN [25], der es offensichtlich macht, dass Schlüsselersetzungsangriffe noch nicht genug Beachtung finden und EUF-CMA keinerlei Garantien in dieser Richtung bietet. Anschließend wird NTRUSign [72] betrachtet, da bei diesem Verfahren ein unehrlicher Unterzeichner besonders effektiv ist.

2.2.4 Beispiele für Schlüsselersetzungsangriffe

Ein Verfahren von Boneh und Boyen

Das Signaturverfahren von BONEH UND BOYEN [25] ist in Algorithmus 1 zusammengefasst, eine detailliertere Beschreibung findet der Leser im Originalpapier.

Das Verfahren ist ein sehr aktueller Vorschlag eines Signaturverfahrens und wird in einer Variante als EUF-CMA-sicher bewiesen. Das Verfahren bietet aber keinerlei Schutz gegenüber Schlüsselersetzungsangriffen.

Proposition 2.6. *Auf das Signaturverfahren in Algorithmus 1 existiert ein starker Schlüsselersetzungsangriff bei ehrlichem Unterzeichner. Im Falle eines unehrlichen Unterzeichners ist sogar ein schwacher Schlüsselersetzungsangriff erfolgreich.*

Beweis. Sei eine Signatur (r, s) für eine Nachricht $M \in \{0,1\}^*$ bezüglich eines öffentlichen Schlüssels $PK = (g_1, g_2, g_2^x, g_2^y)$ gegeben. Dann wählt der Angreifer beliebig $\overline{y} \in \mathbb{Z}_p$ mit $g_2^{\overline{y}} \neq g_2^y$ und berechnet

$$\overline{u} := g_2^x \cdot (g_2^y \cdot g_2^{-\overline{y}})^r.$$

Dann ist (r, s) auch eine gültige Signatur der Nachricht M bezüglich des öffentlichen Schlüssels $(g_1, g_2, \overline{u}, g_2^{\overline{y}})$, wie einfach nachprüfbar ist:

$$\begin{aligned} e(s, \overline{u} \cdot g_2^{H(M)} \cdot (g_2^{\overline{y}})^r) &= e(s, g_2^x \cdot (g_2^y \cdot g_2^{-\overline{y}})^r \cdot g_2^{H(M)+\overline{y}r}) \\ &= e(s, g_2^{x+yr+H(M)}) \\ &= e(s, g_2^x \cdot g_2^{H(M)} \cdot (g_2^y)^r) \\ &= e(g_1, g_2), \end{aligned}$$

wobei der letzte Schritt folgt, da $\sigma = (r, s)$ eine gültige Signatur für den Schlüssel $PK = (g_1, g_2, g_2^x, g_2^y)$ ist.

Der neue Schlüssel $\overline{PK} = (g_1, g_2, \overline{u}, g_2^{\overline{y}})$ unterscheidet sich durch das Konstruktionsverfahren vom ursprünglichen Schlüssel und der zugehörige geheime Schlüssel ist $\overline{SK} = (x + r \cdot (y - \overline{y}), \overline{y})$. Sofern dem Schlüsselersetzungsangreifer $SK = (x, y)$ bekannt ist, kann der geheime Schlüssel einfach berechnet werden. □

Algorithmus 1 Ein Signaturverfahren von BONEH UND BOYEN [25]

Domainparameter:[6]

p	Primzahl
$\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$	zyklische Gruppen der Ordnung p
$\psi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$	Isomorphismus
$e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$	eine nicht entartete Bilinearform, d. h. für alle $h_1 \in \mathbb{G}_1$, $h_2 \in \mathbb{G}_2$ und $a, b \in \mathbb{Z}$ gilt $e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$ und $e(g_1, g_2) \neq 1$.
$H : \{0,1\}^* \longrightarrow \mathbb{Z}_p$	eine kollisionsresistente Hashfunktion

Schlüsselerzeugung Gen: Geheimer und öffentlicher Schlüssel sind:

$SK = (x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ zufällig gewählt.
$PK = (g_1, g_2, g_2^x, g_2^y) \in G_1 \times G_2^3$ für zufällig gewähltes $g_2 \in \mathbb{G}_2$ und $g_1 = \psi(g_2)$.

Signaturberechnung Sig: Gegeben der Signaturschlüssel $SK = (x, y)$, berechnet sich die Signatur σ einer Nachricht $M \in \{0,1\}^*$ in folgender Weise:

1. Wähle zufällig $r \in \mathbb{Z}_p$ mit $p \nmid (x + H(M) + yr)$.
2. Berechne $s := g_1^{(x+H(M)+yr)^{-1} \bmod p}$.

Die Signatur ist $\sigma = (r, s)$.

Signaturverifikation Ver: Gegeben ist ein öffentlicher Verifikationsschlüssel $PK = (g_1, g_2, g_2^x, g_2^y)$, eine Signatur $\sigma = (r, s)$ und eine Nachricht $M \in \{0,1\}^*$. Der Algorithmus Ver liefert valid, falls

$$e(s, g_2^x \cdot g_2^{H(M)} \cdot (g_2^y)^r) = e(g_1, g_2),$$

ansonsten liefert der Algorithmus invalid.

NTRUSign.

Als weiteres Beispiel für einen Schlüsselersetzungsangriff dient das Signaturverfahren NTRUSign [72]. Das Verfahren basiert auf der Schwierigkeit des „*closest vector*"-Problems in einem Gitter. Informell ist der öffentliche Schlüssel eine lange Basis des Gitters und der geheime Schlüssel eine kurze Basis, bezüglich der das „*closest vector*"-Problem einfach zu lösen ist. Eine Nachricht M wird auf einen Punkt abgebildet und

[6] Formal korrekt müssen die Domainparameter als Familien von Domainparametern, indiziert mit dem Sicherheitsparameter ℓ, aufgefasst werden.

mittels der geheimen Basis kann ein Gitterpunkt gefunden werden, der „nah“ an der Nachricht liegt.

Hier wird nur die einfache Variante von NTRUSign im sogenannten transpose-Gitter und ohne zusätzliche Verrauschung des Nachrichtenpunktes vor der Signaturberechnung vorgestellt. Für eine genauere Analyse und Varianten von NTRUSign sei auf [72, 98] verwiesen. Die Grundoperationen werden im Quotientenring $R := \mathbb{Z}[X]/(X^N - 1)$ ausgeführt, wobei N eine Primzahl ist. Elemente von R, die nur Koeffizienten aus $\{0, 1\}$ haben, werden *binär* genannt. NTRUSign verwendet die zentrierte Euklidische Norm der Koeffizienten, um die Größe eines Elements zu bestimmen, d. h. für ein Polynom $a(X) = a_0 + a_1 X + \cdots + a_{N-1} X^{N-1} \in R$ ist die Norm $\|a\|$ gegeben durch $\sum_{i=0}^{N-1} a_i^2 - (1/N)(\sum_{i=0}^{N-1} a_i)^2$. Eine Beschreibung des Verfahrens findet sich in Algorithmus 2.

Bemerkung 2.7. *Ein schwacher Schlüsselersetzungsangriff auf NTRUSign ist möglich. Im Modell mit einem ehrlichen Unterzeichner ist immer noch ein starker Schlüsselersetzungsangriff möglich.*

Für einen modifizierten öffentlichen Schlüssel $\overline{PK} = \overline{h}$ kann die Signaturverifikation einer Signatur (s, r) nur dann erfolgreich sein, falls die Bedingung $\|(s, s\overline{h} - H(M\|r) \bmod q)\| < \mathcal{N}$ erfüllt ist. Das ist der Fall, wenn der Punkt $(s, s\overline{h} \bmod q)$ nahe an $(s, sh \bmod q)$ liegt. Die Wahrscheinlichkeit dafür ist hoch, falls $\overline{h}$ nahe an h liegt. Ein starker Schlüsselersetzungsangriff auf NTRUSign mit ehrlichem Unterzeichner kann dadurch geschehen, dass ein Koeffizient von h leicht verändert wird, z. B. $\overline{h} = h + X$, oder $\overline{h} = h + \alpha$ mit einem $\|\alpha\| = 0$.

Im Gegensatz zu den anderen betrachteten Verfahren hat der neue Schlüssel bei NTRUSign die Eigenschaft, dass er für viele Signaturen, die mit dem Originalschlüssel h berechnet wurden, ebenfalls erfolgreich verifiziert. Daraus ergibt sich auch die Angriffsmöglichkeit mit einem kooperierenden Unterzeichner: Ein Schlüsselersetzungsangreifer kann die öffentlichen Schlüssel $PK = h$ und $\overline{PK} = \overline{h}$ jeweils zusammen mit demselben geheimen Schlüssel SK ausgeben und durch Erhöhen von r im Signieralgorithmus ein geeignetes Nachricht/Signatur-Paar (M, σ), das für beide Schlüssel gültig ist, ausgeben. Beim Berechnen der Signatur kann der Unterzeichner überprüfen, ob die Signatur $\sigma = (r, s)$ der Nachricht M für beide Schlüssel $h, \overline{h}$ die Verifikationsbedingung erfüllt und gegebenenfalls eine neue Signatur durch Erhöhen von r berechnen, bis er tatsächlich eine für beide Schlüssel gültige Signatur erhält. Da SK ebenfalls ein passender geheimer Schlüssel zu $\overline{h}$ ist, ist ein schwacher Schlüsselersetzungsangriff gegeben. Bei diesem Signaturverfahren kann es also zu einem geheimen Schlüssel mehrere öffentliche Schlüssel geben.

2.2.5 Eine Abwehrmaßnahme auf Protokollebene

Es gibt eine einfache Maßnahme, um ein Signaturverfahren gegen Schlüsselersetzungsangriffe zu sichern: Jeder Signatur wird der öffentliche Verifikationsschlüssel des Unterzeichners vorangestellt, d. h. eine Signatur ist definiert als ein Tupel bestehend aus

Algorithmus 2 Das Signaturverfahren NTRUSign.

Domainparameter:

$N \in \mathbb{N}$	Primzahl für die Dimension
R	der Ring $\mathbb{Z}[X]/(X^N - 1)$
$q \in \mathbb{N}$	Modulus
$d_f, d_g \in \mathbb{N}$	Parameter für die Schlüsselgröße
$\mathcal{N} \in \mathbb{R}$	Schranke für die Verifikation
$H : \{0,1\}^* \to R$	eine kollisionsresistente Hashfunktion

Schlüsselerzeugung Gen: Zunächst werden zufällig binäre $f, g \in R$ mit d_f bzw. d_g Einsen gewählt. Daraufhin müssen kleine $F, G \in R$ gefunden werden, so dass $fG - Fg = q$. Schließlich wird $h = f^{-1}F \bmod q$ berechnet. Dabei ist f^{-1} das Inverse von f im Ring $(\mathbb{Z}/q\mathbb{Z})[X]/(X^N - 1)$. Die Schlüssel sind schließlich

$$SK = (f, g, h) \quad \text{und} \quad PK = h$$

Signaturberechnung Sig: Die Signaturberechnung startet mit $r = 0$ und erhöht r, bis eine gültige Signatur gefunden wird. Die Signatur einer Nachricht M ist ein Paar (s, r), so dass der Gitterpunkt $(s, s \cdot h \bmod q)$ nahe bei $(0, H(M\|r))$ liegt, d. h. $\|(s, sh - H(M\|r) \bmod q)\| < \mathcal{N}$. Dazu wird

$$s = \lfloor -\frac{1}{q} H(M\|r) g \rceil f + \lfloor \frac{1}{q} H(M\|r) f \rceil g$$

berechnet, wobei $\lfloor a \rceil$ die am nächsten bei a liegende ganze Zahl bezeichnet. Falls die Signatur $\sigma = (s, r)$ gültig ist, wird sie ausgegeben, andernfalls wird r erhöht und erneut s berechnet.

Signaturverifikation Ver: Eine Signatur $\sigma = (s, r)$ ist genau dann valid, wenn

$$\|(s, sh - H(M\|r) \bmod q)\| < \mathcal{N}.$$

dem öffentlichen Schlüssel und der eigentlichen Signatur. Der Verifikationsalgorithmus Ver muss dann zunächst testen, ob die Signatur mit dem gegebenen Schlüssel beginnt. Damit ist trivialerweise gewährleistet, dass Signaturen zu unterschiedlichen Verifikationsschlüsseln auch eine unterschiedliche Form haben. Aus Effizienzgründen ist es zu bevorzugen, nicht den Verifikationsschlüssel, sondern nur einen Hashwert des Schlüssels oder des Zertifikats der Signatur voranzustellen. Das Verwenden des Zertifikats statt des öffentlichen Schlüssels hat den Vorteil, dass auch der in Bemerkung 2.4 erwähnte Angriff abgewehrt werden kann (wie bei dem zertifikatsbasierten EC-KCDSA in Abschnitt 2.3.4).

Die in [91, Theorem 6] vorgeschlagene Variante, die Nachricht M so zu definieren, dass sie mit dem öffentlichen Schlüssel des Unterzeichners beginnt, reicht im Allgemeinen nicht: Falls nur die ursprüngliche Nachricht M_{orig} zusammen mit der Signatur abgespeichert wird und nur zur Signaturberechnung beispielsweise die Nachricht $M_A = (A, M_{orig})$ berechnet wird, wurde der Schlüsselersetzungsangriff auf einen MKS-Angriff zurückgeführt, der in diesem Fall auch relevant wäre. Für viele Verfahren, die eine kollisionsresistente Hashfunktion verwenden, ist das resultierende Verfahren sicher gegen Schlüsselersetzungsangriffe, jedoch nicht generell. Ein Beispiel, das die prinzipielle Möglichkeit des Angriffs zeigt, wird in Bohli u. a. [22] präsentiert. Falls die neue Nachricht M_A zusammen mit der Signatur gespeichert wird, sieht die Situation ähnlich wie bei der zuvor vorgeschlagenen Lösung aus: Der Unterschied ist nur die Zuordnung der Identität zur Nachricht bzw. zur Signatur.

2.3 Schlüsselersetzungsangriffe auf etablierte Signaturverfahren

Im Folgenden werden etablierte Signaturverfahren genauer im Hinblick auf Schlüsselersetzungsangriffe mit unehrlichem Unterzeichner untersucht. Ausgewählt wurden die Signaturalgorithmen (mit Appendix), die von der Bundesnetzagentur im Algorithmenkatalog [15] für eine qualifizierte elektronische Signatur genehmigt wurden. Im Einzelnen werden die Verfahren RSA-PSS, DSA, EC-DSA, EC-GDSA und EC-KCDSA betrachtet. Dieser Abschnitt beinhaltet nur eine kurze Zusammenfassung des jeweiligen Signaturverfahrens. Eine ausführlichere Beschreibung ist in den entsprechenden Standards PKCS #1 v2.1 [103] für RSA, FIPS 186-2 [59] für DSA sowie zusätzlich ISO/IEC 15946 [78, 79] für die auf elliptische Kurven basierenden Verfahren zu finden.

2.3.1 Das Signaturverfahren RSA-PSS

Das Signaturverfahren RSA-PSS („*probabilistic signature scheme*“) ist eine beweisbar sichere Variante des bekannten RSA-Signaturalgorithmus. Es wird zunächst eine PSS-Codierung der Nachricht M berechnet und anschließend die Codierung mit dem öffentlichen RSA-Exponenten potenziert. Das PSS-Codierungsverfahren wird im Zusammenhang mit subliminalen Kanälen in Abschnitt 3.4.3 detailliert beschrieben. Das RSA-Verfahren wird in Algorithmus 3 beschrieben.

Ein schwacher Schlüsselersetzungsangriff auf RSA ist einfach durchzuführen, falls der Exponent e frei gewählt werden kann, der Algorithmus des Angriffs findet sich in [14]. Für einen festgelegten Exponenten ist RSA vermutlich sicher gegen Schlüsselersetzungsangriffe im Modell mit ehrlichem Unterzeichner, ein Beweis dafür existiert bisher jedoch nicht [14, 91]. Ähnlich sieht es im Modell mit unehrlichem Unterzeichner aus: Es konnte bisher weder ein besserer Angriff noch ein Sicherheitsbeweis angegeben werden.

Algorithmus 3 Das Signaturverfahren RSA-PSS.

Schlüsselerzeugung Gen: Der Algorithmus wählt zwei Primzahlen p und q und berechnet $n = pq$. Ein Wert $e \in \{2, \ldots, (p-1)(q-1)\}$ wird beliebig gewählt – gewöhnlich werden mögliche Werte für e als Domainparameter festgelegt oder finden sich im Standard. Es kann aber auch ein Wert e zufällig gewählt werden. Falls $\mathrm{ggT}(e, (p-1)(q-1)) \neq 1$ wird der Algorithmus erneut gestartet, sonst wird $d = e^{-1} \bmod (p-1)(q-1)$ berechnet. Die Schlüssel sind

$$PK = (n, e) \quad \text{und}$$
$$SK = (n, d).$$

Signaturberechnung Sig: Die Signatur σ einer Nachricht $M \in \{0,1\}^*$ ist $\sigma = (PSS(M))^d$, wobei $PSS(M)$ die PSS-Codierung (siehe Abschnitt 3.4.3) von M ist.

Signaturverifikation Ver: Eine Signatur σ ist für M genau dann valid, wenn σ^e eine zulässige PSS-Codierung von M ist.

2.3.2 Das Signaturverfahren DSA

Der *Digital Signature Algorithm* (DSA) ist ein Standard der US-Regierung für digitale Signaturen. Der Algorithmus baut auf dem ElGamal-Signaturverfahren [57] auf. Als Domainparameter werden Primzahlen p, q mit $q|(p-1)$ benötigt und ein Element $g \in \mathbb{Z}_p^\times$ der Ordnung q. Das Verfahren arbeitet dann in der von g erzeugten zyklischen Gruppe und wird in Algorithmus 4 genauer beschrieben.

Menezes und Smart beweisen in [91, Theorem 8], dass DSA in ihrem Modell selbst starken Schlüsselersetzungsangriffen widersteht, sofern die Funktion $f : \langle g \rangle \to \mathbb{Z}_q$ definiert durch $f(g^x \bmod p) = (g^x \bmod p) \bmod q$ für die DSA-Parameter g, p, q kollisionsresistent ist. Der Beweis hängt nicht davon ab, dass sich der Unterzeichner ehrlich verhält. Daher überträgt er sich unmittelbar auch in das hier verwendete Modell mit unehrlichem Unterzeichner.

Proposition 2.8. *Wenn $f(k) = g^k \bmod p \bmod q$ kollisionsresistent ist, dann ist DSA sicher gegen starke Schlüsselersetzungsangriffe.*

Beweis. Angenommen ein Angreifer findet eine Nachricht $M \in \{0,1\}^*$ mit einer zugehörigen Signatur $(r, s) \in \mathbb{Z}_q \times \mathbb{Z}_q$, die für zwei öffentliche Schlüssel $y, \overline{y} \in \langle g \rangle$ gültig ist. Dann liefert die Verifikationsgleichung

$$r = (g^{s^{-1}H(M)} y^{s^{-1}r} \bmod p) \bmod q = (g^{s^{-1}H(M)} \overline{y}^{s^{-1}r} \bmod p) \bmod q,$$

und unter der Annahme, dass f kollisionsresistent ist, folgt

$$g^{s^{-1}H(M)} y^{s^{-1}r} = g^{s^{-1}H(M)} \overline{y}^{s^{-1}r},$$

Algorithmus 4 Das Signaturverfahren DSA.

Domainparameter:

p, q	Primzahlen mit $q\|(p-1)$
$g \in \mathbb{Z}_p^\times$	mit $\mathrm{ord}(g) = q$
H	kollisionsresistente Hashfunktion

Schlüsselerzeugung Gen: Der Algorithmus wählt als geheimen Schlüssel SK eine Zufallszahl $x \in \{1, \ldots, q-1\}$ und berechnet $y = g^x \bmod p$ als öffentlichen Schlüssel PK.

Signaturberechnung Sig: Die Signatur σ einer Nachricht $M \in \{0,1\}^*$ ist ein Paar $(r, s) \in \mathbb{Z}_q \times \mathbb{Z}_q$ mit

$$r = (g^k \bmod p) \bmod q, \quad \text{für ein zufälliges } k \in \{1, \ldots, q-1\} \text{ und}$$
$$s = k^{-1}(H(M) + xr) \bmod q.$$

Signaturverifikation Ver: Eine Signatur $\sigma = (r, s) \in \mathbb{Z}_q \times \mathbb{Z}_q$ ist für M genau dann valid, wenn

$$r = (g^{s^{-1}H(M)} y^{s^{-1}r}) \bmod q.$$

also $y = \overline{y}$. □

Der nächste Abschnitt wird zeigen, dass für die DSA-Variante auf elliptischen Kurven – trotz der engen Verwandtschaft zum DSA – die Situation deutlich komplizierter ist. Auch die Domainparameterwahl spielt eine größere Rolle und bedarf einer größeren Sorgfalt.

2.3.3 Die Signaturverfahren EC-DSA und EC-GDSA

Die Signaturverfahren EC-DSA und die Variante EC-GDSA werden in diesem Abschnitt zusammen behandelt, da sie sich sehr ähnlich sind. Beide Verfahren arbeiten in einer zyklischen Untergruppe einer elliptischen Kurve $E(\mathbb{F}_q)$ über einem endlichen Körper $\mathbb{F}_q$. EC-DSA verläuft analog zum DSA, während es ein Designziel des EC-GDSA war, die modulare Inversion beim Signieren zu vermeiden. Daher sind der öffentliche Schlüssel sowie die Algorithmen zum Signieren und Verifizieren beim EC-GDSA leicht variiert. Der Ablauf beider Verfahren ist in Algorithmus 5 zusammengefasst.

Das Signaturverfahren EC-DSA ist auf einfache Weise malleable und damit nicht SEUF-CMA. Da die Funktion π nur von der x-Koordinate eines Punktes abhängt, gilt $\pi(P) = \pi(-P)$. Aus der Verifikationsgleichung ist ersichtlich, dass, falls (r, s) eine gültige Signatur ist, auch $(r, -s)$ eine gültige Signatur ist, die als Zwischenergebnis

Algorithmus 5 Die Signaturverfahren EC-DSA und EC-GDSA.

Domainparameter:

$\mathbb{F}_q$	endlicher Körper
$E(\mathbb{F}_q)$	eine elliptische Kurve über $\mathbb{F}_q$ mit $\#E(\mathbb{F}_q) = n \cdot h$ für eine Primzahl n
$G \in E(\mathbb{F}_q)$	Punkt der Ordnung n auf der elliptischen Kurve
$H(\cdot)$	kollisionsresistente Hashfunktion
$\pi : E(\mathbb{F}_q) \to \mathbb{F}_q$	Projektion eines Punktes auf seine x-Koordinate

Schlüsselerzeugung Gen: Eine Zufallszahl $d \in \{1, \ldots, n-1\}$ wird als geheimer Schlüssel SK gewählt. Als öffentlicher Schlüssel PK wird der Punkt $P = d \cdot G$ für EC-DSA (bzw. $P = (d^{-1} \bmod n) \cdot G$ für EC-GDSA) berechnet.

Signaturberechnung Sig: Die Signatur σ einer Nachricht $M \in \{0,1\}^*$ ist ein Paar (r, s), mit

$$\begin{aligned} r &= \pi(k \cdot G) \bmod n, \quad \text{für ein zufälliges } k \in \{1, \ldots, n-1\} \text{ und} \\ s &= k^{-1}(dr + H(M)) \bmod n \\ (\text{bzw.} \quad s &= d(kr - H(M)) \bmod n \quad \text{für EC-GDSA}). \end{aligned}$$

Signaturverifikation Ver: Eine Signatur $\sigma = (r, s)$ für eine Nachricht M ist genau dann valid, wenn

$$\begin{aligned} r &= \pi((s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P) \bmod n, \\ (\text{bzw. } r &= \pi((r^{-1}H(M) \bmod n) \cdot G + (r^{-1}s \bmod n) \cdot P) \bmod n \quad \text{für EC-GDSA}). \end{aligned}$$

einen inversen Punkt des ursprünglich verwendeten berechnet. Es kann überprüft werden, dass

$$\pi(((-s)^{-1}H(M)) \cdot G + ((-s)^{-1}r) \cdot P) = \pi(-(s^{-1}H(M)) \cdot G + (s^{-1}r) \cdot P).$$

Im Allgemeinen ist die „*malleability*“ von Signaturverfahren keine Gefahr, es muss jedoch nach Einsatzbereich genau geprüft werden, ob nicht ein Signaturverfahren benötigt wird, das SEUF-CMA ist. Bei EC-GDSA ist dieser Angriff durch das „inverse k“ nicht möglich.

Menezes und Smart [91] beweisen, dass EC-DSA gegen schwache Schlüsselersetzungsangriffe ohne unehrliche Unterzeichner sicher ist, sofern die Domainparameterwahl passend eingeschränkt wird. Insbesondere fordern sie $n > q$. Ihr Beweis nutzt aus, dass der Angreifer den geheimen Schlüssel des Unterzeichners nicht kennt, was für das hier verwendete Modell nicht mehr gilt. Tatsächlich ist es möglich, mithilfe eines unehrlichen Unterzeichners einen Schlüsselersetzungsangriff auf EC-DSA durchzuführen.

Proposition 2.9. *Ein schwacher Schlüsselersetzungsangriff auf EC-DSA und EC-GDSA ist möglich.*

Beweis. Der Beweis betrachtet nur EC-DSA, für EC-GDSA verläuft der Angriff ganz analog. Gegeben sei ein öffentlicher Schlüssel $PK = P$ und eine Signatur $\sigma = (r, s)$ für eine Nachricht $M \in \{0,1\}^*$. Die Verifikation der Signatur (r, s) geschieht durch Prüfen der Gleichung

$$r = \pi((s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P) \bmod n. \tag{2.1}$$

Im Gegensatz zur Situation bei DSA, wo das zusätzliche „mod q“ nicht zu effizient berechenbaren Kollisionen der Funktion f (siehe Proposition 2.8) führt, existiert für die Funktion π, die nur von der x-Koordinate eines Punktes abhängt, eine triviale Kollision durch die Invertierung der y-Koordinate:

$$r = \pi(\pm((s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P)).$$

Daher ist der Punkt $\overline{P}$ mit

$$(s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot \overline{P} = \\ -((s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P)$$

ein weiterer öffentlicher Schlüssel $\overline{PK}$ für den die Verifikationsgleichung (2.1) erfüllt ist. Zusammen mit dem geheimen Schlüssel $SK = d$ kann zu $\overline{PK}$ auch ein geheimer Schlüssel $\overline{SK} = \overline{d}$ ausgerechnet werden:[7]

$$\overline{d} = -(2r^{-1}H(M) + d) \bmod n$$

und $\overline{P} = \overline{d} \cdot G$. Ohne Einschränkung kann $H(M) \neq -dr \bmod n$ und $\overline{d} \neq 0$ angenommen werden (sonst kann ein neues Nachricht/Signatur-Paar $(M, (r, s))$ gewählt werden). Dann ist $\overline{SK} = \overline{d}$ ein gültiger geheimer Schlüssel, der sich von $SK = d$ unterscheidet ebenso wie auch $PK \neq \overline{PK}$. □

Bemerkung 2.10. *Der Angriff mit einer „negativen y-Koordinate“ wie in Proposition 2.9 kann auch mit ehrlichem Unterzeichner durchgeführt werden, jedoch nur als starker Schlüsselersetzungsangriff. Für EC-DSA kann ein neuer öffentlicher Schlüssel mittels $(sr^{-1} \bmod n) \cdot (X - (s^{-1}H(M) \bmod n) \cdot G)$ berechnet werden, wobei X ein Punkt ist, dessen x-Koordinate r ist. (Falls die elliptische Kurve nur einen Punkt mit der x-Koordinate r enthalten sollte, muss ein anderes Nachricht/Signatur-Paar $(M, (r, s))$ benutzt werden).*

Elliptische Kurven mit $n < q$ können einen weiteren Schlüsselersetzungsangriff ermöglichen. In einem Körper $\mathbb{F}_q$ mit einer Charakteristik ungleich zwei oder drei,

[7] In dem Modell mit unehrlichem Unterzeichner gilt das in [91] genannte Argument, dass der Schlüssel SK dem Angreifer unbekannt sei, nicht.

kann eine elliptische Kurve $E(\mathbb{F}_q)$ durch eine Gleichung der Form $y^2 = x^3 + ax + b$ mit $a, b \in \mathbb{F}_q$ ausgedrückt werden. Es sei ein Nachricht/Signatur-Paar (M, σ) bezüglich eines öffentlichen Schlüssels $PK = P$ gegeben. Falls $n < q$ ist, ist der folgende Schlüsselersetzungsangriff denkbar: Der Verifikationsalgorithmus Ver überprüft die Gleichung

$$r = \pi((s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P) \bmod n$$

nur modulo n. Seien (x, y) die Koordinaten des Punktes $((s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P)$. Ein weiterer öffentlicher Schlüssel kann gefunden werden, falls ein $\overline{x} := x + l \cdot n$ gefunden wird, wobei $l \geq 1$, so dass $\overline{x}^3 + a\overline{x} + b$ ein Quadrat ist und $\overline{x} < q$. Dann existiert ein Punkt T auf der Kurve, dessen x-Koordinate $\overline{x}$ ist, und der zusätzliche öffentliche Schlüssel wird definiert als $\overline{P} := (r^{-1}s \bmod n) \cdot (T - (s^{-1}H(M) \bmod n) \cdot G)$.

Beispiel 2.11. *Die Kurve mit den folgenden Kurvenparametern (aus [6])*

$$\begin{aligned}
q &= 5331369449787147210725799182247363710162558951080875268627741810017386746493149\\
a &= -3\\
b &= 8808842535654359666556380433494756852865478556389536949350714056479398028333100\\
G &= (5081731805997595463647968776224307067547060629202974902735739659928355409323038, 8291339351528036362764097908361461263815578639057850151667575229518322514900 88)\\
n &= 1332842362446786802681449795561840927540869571121424054459187605073720511411159
\end{aligned}$$

ist eine elliptische Kurve, welche die starken Bedingungen der qualifizierten Signaturen [15] erfüllt (Die Länge von q ist 262 bit und die Länge von n 260 bit). Mit einem Computeralgebrasystem (beispielsweise Magma [26]) kann nun einfach verifiziert werden, dass für diese Kurve (zusätzlich zu dem Angriff aus Proposition 2.9) auch ein starker Schlüsselersetzungsangriff mittels der erwähnten „$n < q$-Methode“ möglich ist.

Auch wenn bisher kein Algorithmus bekannt ist, der darauf einen schwachen Schlüsselersetzungsangriff aufbaut, erscheint es dennoch sinnvoll, diese Parameter auszuschließen. Es ist erwähnenswert, dass in der Algorithmenempfehlung der Bundesnetzagentur [15] diese Einschränkung nicht gemacht wird, obwohl andere Einschränkungen der Parameter durchaus vorhanden sind.

2.3.4 Das Signaturverfahren EC-KCDSA

Der zertifikatbasierte Digitale Signaturalgorithmus (KCDSA) ist ein Standard des koreanischen Amts für Informationssicherheit. Hier soll die Variante mit elliptischen Kurven EC-KCDSA betrachtet werden. Ein wesentlicher Punkt bei den Signaturverfahren KCDSA und EC-KCDSA ist die Verwendung der Zertifikate im Signaturprozess. Sehr vereinfacht geht der Algorithmus aus dem DSA hervor, wenn statt der

Nachricht M die Verkettung $z\|M$ von Zertifikat z und Nachricht M signiert wird. Dadurch ist, im Gegensatz zum DSA, der KCDSA beweisbar sicher (siehe [33, 104]). Wie bei EC-GDSA wird bei der Signaturberechnung die modulare Inversion vermieden. Der EC-KCDSA verwendet eine kollisionsresistente Hashfunktion, um im Signieralgorithmus r aus dem Punkt kG zu berechnen, so dass die im letzten Abschnitt ausgenutzte Schwäche des EC-DSA nicht existiert. Auch im Hinblick auf eine Situation wie in Bemerkung 2.4, wenn ein Benutzer einen bereits existierenden öffentlichen Schlüssel zertifizieren lassen will, bietet KCDSA den Vorteil, dass unterschiedliche Nutzer immer unterschiedliche Zertifikate haben – die Signaturen der beiden Benutzer mit demselben Schlüssel werden also nicht identisch sein und der erwähnte Angriff ist nicht möglich. Der Algorithmus EC-KCDSA wird in Algorithmus 6 beschrieben. Das Verfahren bietet von den in diesem Kapitel betrachteten Signaturverfahren die beste Sicherheit.

Algorithmus 6 Das Signaturverfahren EC-KCDSA.

Domainparameter:

$\mathbb{F}_q$	endlicher Körper
$E(\mathbb{F}_q)$	eine elliptische Kurve über $\mathbb{F}_q$ mit $\#E(\mathbb{F}_q) = n \cdot h$ für eine Primzahl n
$G \in E(\mathbb{F}_q)$	Punkt der Ordnung n auf der elliptischen Kurve
$H(\cdot)$	kollisionsresistente Hashfunktion

Schlüsselerzeugung Gen: Eine Zufallszahl $d \in \{1, \ldots, n-1\}$ wird als geheimer Schlüssel SK gewählt. Als öffentlicher Schlüssel PK wird der Punkt $P = (d^{-1} \bmod n) \cdot G$ berechnet. Zusätzlich wird ein Hashwert z des Zertifikats des Unterzeichners berechnet. Das Zertifikat enthält mindestens eine eindeutige Identifikation des Benutzers, die Domainparameter und den öffentlichen Schlüssel P.

Signaturberechnung Sig: Die Signatur σ einer Nachricht $M \in \{0,1\}^*$ ist ein Paar (r, s) mit

$$r = H(k \cdot G), \quad \text{für ein zufälliges } k \in \{1, \ldots, n-1\} \text{ und}$$
$$s = d(k - r \oplus H(z\|M)) \bmod n.$$

Signaturverifikation Ver: Eine Signatur $\sigma = (r, s)$ für eine Nachricht M ist genau dann valid, wenn

$$r = H((r \oplus H(z\|M) \bmod n) \cdot G + s \cdot P).$$

Proposition 2.12. *Das Signaturverfahren EC-KCDSA ist im Random-Oracle-Modell EUF-CMA sicher und sicher gegen starke Schlüsselersetzungsangriffe.*

Beweis. Die EUF-CMA Sicherheit von EC-KCDSA wurde von BRICKELL U. A. [33] bewiesen. Hier bleibt noch die Sicherheit gegen starke Schlüsselersetzungsangriffe zu beweisen.

Ein Schlüsselersetzungsangriff ist gegeben, wenn ein PPT-Algorithmus existiert, der M, $(P_1, z_1) \neq (P_2, z_2)$, und eine Signatur $\sigma = (r, s)$ ausgibt, die für beide Schlüssel gültig ist, also:

$$\begin{aligned} r &= H((r \oplus H(z_1 \| M) \bmod n) \cdot G + s \cdot P_1) \\ &= H((r \oplus H(z_2 \| M) \bmod n) \cdot G + s \cdot P_2). \end{aligned} \tag{2.2}$$

Der Wert r kann nicht beliebig gewählt werden, da $(r \oplus H(z_i \| M) \bmod n) \cdot G + s \cdot P_i$ für $i \in \{1, 2\}$ ein Urbild von r unter H ist; die Wahrscheinlichkeit, ein Urbild von r zu finden, ist für einen PPT-Algorithmus jedoch vernachlässigbar. Daher muss der Punkt $k \cdot G$ mit $r = H(k \cdot G)$ bekannt sein, also:

$$k \cdot G = (H(k \cdot G) \oplus H(z_i \| M) \bmod n) \cdot G + s \cdot P_i, \quad i = 1, 2.$$

In dieser Gleichung kommen die Variablen k, P_1, P_2 (via z_1, z_2) und M als Eingabe für Hashfunktionen vor. Jede Modifikation einer dieser Variablen resultiert daher in einem neuen gleichverteilt zufälligen Wert $H(k \cdot G) \oplus H(z_i \| M)$. Daher ist für jeden festgelegten Punkt $X \in E(\mathbb{F}_q)$ die Wahrscheinlichkeit für $X = (H(k \cdot G) \oplus H(z_i \| M) \bmod n) \cdot G$ vernachlässigbar. Also ist die Wahrscheinlichkeit, eine Lösung von Gleichung (2.2) durch Anpassen einer der Variablen k, P_1, P_2 und M zu finden, vernachlässigbar.

Es bleibt daher nur, die Variable s als letztes anzupassen, nachdem die anderen Variablen festgelegt wurden. Die Wahrscheinlichkeit, dass beide Gleichungen (2.2) dieselbe Lösung s haben, ist vernachlässigbar, weil s unter Kenntnis eines einzelnen geheimen Schlüssels d_i gefunden werden kann und damit einen „*existential forgery*“ bezüglich des anderen Schlüssels darstellen würde. □

2.4 Weitere Angriffe auf Signaturverfahren

Es gibt neben den erwähnten Schlüsselersetzungs- und MKS-Angriffen weitere Angriffe auf Signaturverfahren, die nicht von dem Sicherheitsbegriff EUF-CMA berücksichtigt werden. Einige davon sind ebenfalls Ersetzungsangriffe, allerdings auf andere Zielwerte.

So untersucht beispielsweise VAUDENAY [126] Domainparameterersetzungsangriffe. Dabei gehen der Unterzeichner und der Verifizierer von unterschiedlichen Domainparametern aus, unter denen das Signaturverfahren genutzt wird. Falls der Verifikationsschlüssel nicht klar einem Domainparameter zugeordnet ist, können auch Signaturfälschungen möglich sein: Beispielsweise bei DSA ist der Verifikationsschlüssel $y = g^x$, der geheime Signaturschlüssel ist der diskrete Logarithmus von y bezüglich g. Wenn nun ein Angreifer den Schlüssel y in ein System in derselben Gruppe, jedoch mit der Basis $\overline{g} = y^\alpha$, überträgt, ist ihm der diskrete Logarithmus von y bezüglich

$\overline{g}$ natürlich bekannt und er kann beliebige Nachrichten in diesem System signieren. Diese Angriffe sind jedoch einfacher zu verhindern als die Schlüsselersetzungsangriffe. Beispielsweise sollten im Zertifikat die Domainparameter, unter denen der öffentliche Schlüssel gilt, erwähnt werden.

Es gibt auch Grenzfälle zwischen Schlüsselersetzungsangriffen und den Domainparameterersetzungsangriffen. So werden in [14, Abschnitte 4.4 & 4.5] und [105, Abschnitt 4] Schlüsselersetzungsangriffe auf DSA-Varianten betrachtet, bei denen der Erzeuger der zyklischen Gruppe ersetzt wird. Wird der Erzeuger der zyklischen Gruppe jedoch als Domainparameter aufgefasst, so fallen diese Angriffe in die Domainparameterersetzungsangriffe und passen nicht in das hier betrachtete Modell der Schlüsselersetzungsangriffe.

Ein Problem, das VAUDENAY [124, 125] aufzeigt, ist eine unehrliche Erzeugung der Domainparameter. Beispielsweise können bei DSA zwei Nachrichten M und $\overline{M}$ so gewählt werden, dass die Differenz ihrer SHA-1-Hashwerte SHA-1(M) − SHA-1$(\overline{M})$ eine 160-bit-Primzahl ergibt. Dann kann der Parameter $q =$ SHA-1(M) − SHA-1$(\overline{M})$ gewählt werden mit dem Effekt, dass die Nachrichten M und $\overline{M}$ für alle Benutzer in diesem System die gleichen Signaturen haben. Es könnte nun versucht werden, einen Benutzer dazu zu bewegen, die harmlose Nachricht M zu unterschreiben, um eine Signatur für die schädliche Nachricht M' zu erhalten. Auch dieser Angriff ist praktisch weniger relevant (vgl. auch [15]). Bei EC-DSA ist ein solcher Angriff mit zwei Nachrichten, die identische Signaturen besitzen auch bei der Schlüsselerzeugung möglich [118]. Der Angriff nutzt ebenfalls die einfache Kollision der Funktion π aus, die sich auch bezüglich der Schlüsselersetzungsangriffe als Schwäche herausstellte.

Weitere „*unknown key-share*“ Angriffe auf Schlüsselaustauschprotokolle werden von BAEK UND KIM [5] präsentiert. Ein neuer Sicherheitsbegriff „*completely non-malleable*“ für Signaturverfahren wird in FISCHLIN [60] eingeführt. Der Begriff deckt EUF-CMA und Schlüsselersetzungsangriffe mit ehrlichem Unterzeichner ab, berücksichtigt aber die hier betrachteten Schlüsselersetzungsangriffe mit unehrlichem Unterzeichner nicht.

2.5 Zusammenfassung und offene Fragen

Dieses Kapitel zeigt, dass der für Signaturverfahren übliche Sicherheitsbegriff EUF-CMA nicht alle Anforderungen an eine Authentifikation berücksichtigt und die erforderliche Bindung der Signatur an eine Person nicht zufriedenstellend garantiert. Dadurch kann letztendlich auch die Nichtabstreitbarkeit von Signaturen in Frage gestellt werden. Als bisher unberücksichtigter Angriff, der auf EUF-CMA-sichere Verfahren möglich ist, wurden Schlüsselersetzungsangriffe in ein Szenario übertragen, in dem sich auch der Unterzeichner unehrlich verhält. Durch unehrliche Schlüsselerzeugung, unehrliche Signaturberechnung oder Veröffentlichen des geheimen Schlüssels werden zusätzliche Schlüsselersetzungsangriffe ermöglicht, die bisher nicht betrachtet wurden. Für etablierte Signaturverfahren wurde die Sicherheit in diesem neuen

Szenario analysiert. Die Verfahren EC-DSA und EC-GDSA stellten sich dabei als anfällig gegen Schlüsselersetzungsangriffe heraus, während EC-KCDSA sowie auch DSA beweisbar nicht betroffen sind. Bei dem Verfahren EC-KCDSA wird dadurch, dass das Zertifikat zusammen mit der Nachricht gehasht wird, auch ein möglicher Angriff durch ein gemeinsames Benutzen eines Schlüsselpaars durch mehrere Teilnehmer verhindert. Diese Maßnahme kann für die Entwicklung neuer Signaturverfahren empfohlen werden. Durch eine geeignete Parameterwahl können aber auch beim EC-DSA Schlüsselersetzungsangriffe erschwert werden, jedoch sind in aktuellen Vorschlägen zur Parameterwahl Schlüsselersetzungsangriffe noch nicht berücksichtigt.

Schlüsselersetzungsangriffe können durch eine geeignet gesicherte Schlüsselerzeugung und Zertifizierung auf einer höheren Ebene ausgeschlossen werden. Dadurch sind bei qualifizierten Signaturen, bei denen die Schlüsselerzeugung und Signaturberechnung von einer vertrauenswürdigen Instanz bzw. von sicherer Hardware durchgeführt werden, keine Schlüsselersetzungsangriffe möglich. Bei den sogenannten *fortgeschrittenen* Signaturen, die verlangen, dass die Signatur eindeutig an eine Identität gebunden ist, aber die Schlüsselerzeugung und Signaturberechnung nicht einschränken, stellt sich die Frage, ob Signaturen, die anfällig gegenüber Schlüsselersetzungsangriffen sind, ausreichend sind. Eine rechtliche Bewertung der Schlüsselersetzungsangriffe im Hinblick auf die fortgeschrittene Signatur wäre daher interessant, liegt aber außerhalb des Rahmens dieser Arbeit.

Zusammengefasst müssen beim Einsatz von EUF-CMA-sicheren Signaturverfahren in einer Anwendung die nötigen Anforderungen an die Signatur überprüft werden und gegebenenfalls ein Signaturverfahren mit weiterreichenden Sicherheitsgarantien gewählt werden. Verfahren, die neben EUF-CMA auch sicher gegen Schlüsselersetzungsangriffe sind, sind aus derzeitiger Sicht für einen Großteil der Anwendungen ausreichende Verfahren. Als offenes Problem bleibt eine umfassende Modellierung und Klassifikation von Signaturverfahren bezüglich der nicht von EUF-CMA erfassten Angriffe.

3. Subliminale Kanäle

Dieses Kapitel behandelt einen anderen Aspekt von Angriffen auf Signaturverfahren: einen Missbrauch des Verfahrens zur verborgenen Kommunikation. Ein sogenannter subliminaler Kanal in einem Signaturverfahren ermöglicht es dem Unterzeichner[1], in der Signatur eine Nachricht zu verbergen, die nur ein autorisierter Empfänger lesen kann. Ohne das zusätzliche Wissen des autorisierten Empfängers ist selbst das Vorhandensein der Nachricht nicht zu erkennen – die Signatur ist von Signaturen ununterscheidbar[2], in denen keine Nachricht eingebettet wurde. Damit sind die subliminalen Kanäle ein Teilgebiet der *Steganographie*. Die Steganographie beschäftigt sich mit dem Verbergen einer Nachricht in einem gegebenen Medium, ohne dass für Unbefugte das Vorhandensein einer Nachricht erkennbar ist. Ein steganographisches Verfahren gilt dann als sicher, wenn es keinem Angreifer gelingt die Existenz einer verborgenen Nachricht festzustellen. Verbreitete Verfahren nutzen als Medien, beispielsweise Bilder oder Musikstücke, um darin die Nachricht zu verstecken. Ein *subliminaler Kanal* bezeichnet nun einen *sicheren* steganographischen Kanal, der als Medium ein kryptographisches Objekt benutzt.

Es wird nun zunächst eine Motivation und ein Überblick über die Entdeckung subliminaler Kanäle gegeben. Abschnitt 3.3 gibt eine formale Untersuchung der subliminalen Kanäle. Anschließend werden in Abschnitt 3.4 beweisbar sichere subliminale Kanäle in deterministischen Signaturverfahren – unter anderem in ESIGN-D und RSA-PSS – aufgezeigt. Die Untersuchung der subliminalfreien Verfahren beginnt im Abschnitt 3.5 und zeigt, dass das Verfahren RSA-PSS, wenn es in einer deterministischen Variante benutzt wird und mit einer geeigneten Schlüsselüberprüfung GenVer verbunden wird, beweisbar subliminalfrei ist. In einem verallgemeinerten Szenario für Subliminalfreiheit kann schließlich in Abschnitt 3.5.3 auch eine Variante von EC-DSA als Beispiel für ein Signaturverfahren der DSA-Familie als *subliminalfrei mit Wächter* bewiesen werden. Diese Ergebnisse zu subliminalen Kanälen in Signaturverfahren

[1] bzw. letztendlich dem Algorithmus, der die Signatur berechnet

[2] siehe Definition A.3

wurden in Bohli und Steinwandt [23] sowie Bohli, González Vasco und Steinwandt [20] veröffentlicht.

3.1 Geschichte der subliminalen Kanäle

Ende der 70er und Anfang der 80er Jahre beschäftigte sich Simmons [109, 110, 108] mit der Authentifizierung von Daten ohne den Einsatz von Verschlüsselung. Geplante Einsatzmöglichkeiten dazu gibt es bei der Überwachung von Verträgen, wie später noch genauer erläutert wird. Dabei entdeckte Simmons [111] schließlich, dass in dem Authentifikator selbst eine verborgene Kommunikation möglich ist, was zuvor übersehen worden war. Simmons nannte diesen verdeckten Kanal *subliminaler Kanal* und veranschaulichte die Auswirkungen des Kanals anhand des von ihm vorgestellten Gefangenenproblems. Bei dem Problem sitzen zwei Gefangene in unterschiedlichen Zellen, wollen sich aber über einen Ausbruchsversuch abstimmen. Ein Wächter erlaubt ihnen schriftlich zu kommunizieren, jedoch keine Verwendung von Kryptographie, weil er hofft, in den Briefen Hinweise auf den Ausbruchsversuch zu entdecken. Da die Gefangenen fürchten, der Wächter könne Nachrichten einschleusen oder verändern, bestehen sie darauf, die Nachrichten digital zu signieren, was ihnen vom Wächter auch erlaubt wird. Nun können sie einen subliminalen Kanal in der Signatur verwenden, um unbemerkt ihre Ausbruchspläne auszutauschen. Sie brauchen dazu einen gemeinsamen Schlüssel, den sie aber schon lange Zeit vorher vereinbart haben können. In diesem Beispiel sind natürlich auch andere steganographische Verfahren möglich, beispielsweise kann die offene Nachricht so formuliert werden, dass sie aufgrund des geteilten Wissens der Gefangenen eine versteckte Bedeutung erhält. In Abschnitt 3.2 werden aber weitere Beispiele präsentiert, bei denen der subliminale Kanal der einzig mögliche verdeckte Kanal ist.

Simmons [113] erklärte 10 Jahre später den Hintergrund seines Interesses für subliminale Kanäle und die möglichen Auswirkungen. Bei den SALT-Verträgen (Strategic Arms Limitation Talks) zur nuklearen Rüstungsbegrenzung in den 70er Jahren ging es auch um die Beschränkung der einsatzbereiten Trägersysteme. Um nicht mit weniger Raketensilos Opfer eines Erstschlags zu werden, hatten die Amerikaner die Idee eines „*missile shell games*“. Es sollten beispielsweise 100 Raketen auf 1000 Silos verteilt werden, wobei so viele Transporte und scheinbare Transporte durchgeführt werden sollten, dass auf feindlicher Seite keine Information mehr über die tatsächlichen Standorte der Raketen vorhanden war. Nun waren die USA jedoch verpflichtet, der UdSSR nachzuweisen, dass tatsächlich nicht mehr als 100 Silos belegt waren. Auf ein statistischen Verfahren wollte man sich damals nicht verlassen und daher wurde nach einer technologischen Lösung gesucht: Es war möglich, mit Sensoren festzustellen ob sich eine Rakete in einem Silo befindet. In jedem Silo sollte der UdSSR nun erlaubt werden, einen solchen Sensor anzubringen, der unverschlüsselte, authentifizierte Nachrichten verschickt. Eine eindeutige Kennzeichnung des Sensors sollte erst nachträglich eingefügt werden, so dass jeder Sensor für die Kontrolleure eindeutig identifizierbar

ist, seine Position aber nicht bekannt ist. Der nun entdeckte subliminale Kanal würde es jedoch erlauben, dass eine zuvor gewählte geheime Identifikations-Nummer der Sensoren unbemerkt durch die Authentifizierung übertragen werden kann, so dass die Geheimhaltung der Raketenstandorte nicht erreicht wird. Weitere Details finden sich in einem Vortragsprotokoll von SIMMONS [116] und in dem bereits erwähnten Artikel [113].

Mit der Betrachtung von digitalen Signaturverfahren wurde die Möglichkeit der subliminalen Kommunikation konkreter. Die ersten subliminalen Kanäle in Signaturverfahren hat SIMMONS im ElGamal-Verfahren [111] und später [114] im „*Digital Signature Algorithm*“ (DSA) aufgezeigt. In den folgenden Jahren konnten weitere subliminale Kanäle in Signaturverfahren gefunden werden [3, 71, 130].

Da subliminale Kanäle in der Regel ein unerwünschter Effekt sind, stellt sich auch die Frage nach Subliminalfreiheit, der Abwesenheit subliminaler Kanäle, wie es auch Simmons' ursprüngliches Ziel war. SIMMONS [112] schlug einen interaktiven Signaturalgorithmus für DSA vor, der zwischen dem Unterzeichner und dem Wächter ausgeführt wird, um die Signatur zu berechnen, ohne dass der Unterzeichner den subliminalen Kanal nutzen kann. DESMEDT [53] machte darauf aufmerksam, dass trotz dieses Verfahrens unter Umständen eine verborgene Kommunikation stattfinden kann, falls der Unterzeichner nur dann die erhaltene Signatur akzeptiert, falls sie zufällig die passende subliminale Nachricht enthält. Jedoch ist es fragwürdig, inwieweit ein solches Vorgehen für den Unterzeichner überhaupt möglich ist, beispielsweise könnte direkt der Wächter die Signatur versenden [115]. Es stellt sich in diesem Zusammenhang auch die Frage, wann ein Verfahren subliminalfrei genannt werden soll. Eine allgemeine Definition für subliminalfreie Kryptographie ist ein äußerst schwieriges Ziel. Für interaktive Zero-Knowledge-Beweise wurde eine Formalisierung von Subliminalfreiheit von BURMESTER U. A. [39] vorgeschlagen, in dieser Arbeit wird ein subliminalfreies Signaturverfahren formalisiert.

3.2 Subliminale Kanäle heute

Gerade in letzter Zeit nehmen die Anwendungen der digitalen Signaturen zu. Als Beispiel seien E-Mail-Authentifizierung, elektronische Reisepässe oder, im Zusammenhang mit der Gesundheitskarte, elektronische Rezepte genannt. Subliminale Kanäle können in diesen Einsatzszenarien durchaus eine Gefahr oder einen unerwünschten Nebeneffekt darstellen. Ein alltägliches motivierendes Szenario für die Gefahr subliminaler Kanäle in Signaturverfahren ist die von YOUNG UND YUNG [127, 128] beschriebene „*Secretly Embedded Trapdoor with Universal Protection (SETUP)*“, eine Hintertür, die folgendermaßen funktioniert: Ein böswilliger Code-Designer kann beispielsweise in einem Programm zum Signieren von E-Mails einen Signaturalgorithmus für ein Signaturverfahren mit subliminalem Kanal so implementieren, dass es den geheimen Signaturschlüssel des Benutzers als subliminale Nachricht in der Signatur versteckt. Wenn der Code-Designer dann auf eine mit seinem Programm signierte

Nachricht trifft, kann er aus der Signatur den geheimen Schlüssel extrahieren und von dort an Nachrichten im Namen dieses Benutzers unterzeichnen. Steganographie im Nachrichtentext kommt hier natürlich nicht in Frage, da das Signaturprogramm die vom Benutzer vorgegebene Nachricht nicht unbemerkt verändern kann.

Eine andere Situation, bei der subliminale Kommunikation in Signaturen benutzt werden kann, sind elektronische Ausweise. Solche Dokumente enthalten die Daten einer Person in digitaler Form, von einer ausstellenden Behörde signiert, um die Integrität zu gewährleisten. Auch hier ist unter Umständen kein steganographisches Verfahren auf dem Nachrichtentext möglich, da die Behörde hier keinen Einfluss auf die Daten hat, die ja durch den späteren Inhaber des Ausweises in Form von Name, Adresse etc. vorgegeben sind.[3] Der subliminale Kanal kann hier genutzt werden, um Kommentare zu der betreffenden Person oder Einschränkungen des Ausweises für autorisierte Stellen dezentral zu speichern. Je nach Szenario kann ein solcher Gebrauch des subliminalen Kanals unerwünscht sein.

Der Standardbegriff EUF-CMA für die Sicherheit digitaler Signaturverfahren geht nicht auf subliminale Kanäle ein und ebensowenig hat die Sicherheit gegen Schlüsselersetzungsangriffe mit subliminalen Kanälen zu tun. Auch in anderen Sicherheitsmodellen, wie beispielsweise in dem Modell für Universelle Komponierbarkeit [44] berücksichtigen die aktuellen Formalisierungen für digitale Signaturen [45, 4] keine subliminalen Kanäle.

In praktischen Anwendungen werden üblicherweise nicht-interaktive Signaturverfahren benötigt, bei denen weder die Signaturgenerierung noch die Signaturverifizierung Kommunikation benötigen. Das von SIMMONS [112] vorgeschlagene interaktive Signaturprotokoll eignet sich daher nicht für einen breiten Einsatz. Wenn ein interaktives Signaturprotokoll in einen nicht-interaktiven Algorithmus übertragen wird, indem der Unterzeichner die zweite Partei (den Wächter) simuliert, wird in der Regel die Subliminalfreiheit verloren gehen. Ein Wächter in einem interaktiven Protokoll kann einige Maßnahmen ergreifen, um die Kapazität des Kanals weiter zu verringern [115], bei einem unter Zuhilfenahme eines Pseudozufallsgenerators simulierten Wächter kann der Unterzeichner das Zurückweisen und Neuberechnen der Signatur jedoch unbemerkt durchführen.

Eine scheinbar offensichtliche Methode, um subliminale Kommunikation in einer Signatur zu verhindern, ist die Verwendung eines deterministischen Signaturverfahrens: Wenn es keine Zufallswahlen gibt, hat der Signieralgorithmus keine Freiheiten um eine subliminale Nachricht zu verstecken. Das Argument, dass ohne Freiheit keine subliminale Nachricht versteckt werden kann, ist grundsätzlich richtig, jedoch reicht ein deterministisches Verfahren alleine dazu noch nicht aus, da der Unterzeichner beispielsweise einen probabilistischen Signieralgorithmus finden kann, dessen Benutzung von der des deterministischen nicht zu unterscheiden ist.[4] Die einzige uns bekannte

[3] Falls zusätzliche Daten, wie beispielsweise ein Zeitstempel, ein Photo oder ein Fingerabdruck in die Signatur mit einbezogen werden, kann es eine Möglichkeit steganographischer Kommunikation durch diese zusätzlichen Daten vorhanden sein.

[4] solange jede Nachricht nur einmal signiert wird

öffentliche Anmerkung zu diesem Aspekt findet sich in einem kurzen Abschnitt eines Protokolls zu einer Sitzung der Institution of Electrical Engineers (IEE) [74], in dem darauf hingewiesen wird, dass der Verifizierer keine Möglichkeit hat, zu entscheiden, ob die Signatur deterministisch erzeugt wurde oder nicht. Im selben Absatz wird auch auf die Einschränkung von subliminalen Kanälen in deterministischen Verfahren eingegangen, nämlich dass für jede Nachricht nur eine subliminale Nachricht eingebettet werden kann, da sonst zwei Signaturen zu einer Nachricht veröffentlicht werden können und ein Missbrauch des deterministischen Verfahrens somit offensichtlich wird.

3.3 Formalisierung von subliminalen Kanälen

Zunächst werden subliminale Kanäle in einem strengen Sinn formalisiert, motiviert durch den Wunsch subliminale Kommunikation als besondere Eigenschaft sicher zu verwenden. Für den Fall, dass subliminale Kommunikation unerwünscht ist, müssen neben den hier definierten subliminalen Kanälen auch weitere subliminale Kommunikationsmöglichkeiten ausgeschlossen werden. Es gibt zwischen Signaturverfahren mit subliminalem Kanal und subliminalfreien Signaturverfahren also eine Lücke.

Ein subliminaler Kanal soll ein sicherer verdeckter Kanal in einem kryptographischen Objekt sein. Am Beispiel von digitalen Signaturen wird nun ein formales Modell für subliminale Kanäle vorgestellt. Der subliminale Kanal wird üblicherweise genutzt, indem Zufallswerte, die in den Signaturprozess einfließen, durch eine entsprechend gewählte zufällig aussehende Verschlüsselung der subliminalen Nachricht ersetzt werden. Der beabsichtigte Empfänger muss in der Lage sein, den verwendeten „Zufall“ zu extrahieren, und muss den Schlüssel kennen, um die subliminale Nachricht zu entschlüsseln. Eine Möglichkeit eine solche Verschlüsselung der Nachricht zu erhalten wird in Abschnitt 3.3.3 beschrieben.

3.3.1 Signatur mit subliminalem Kanal

Zunächst sei erneut an die Definition 2.1 eines Signaturverfahrens erinnert. Ein nichtinteraktives Signaturverfahren ist formal ein Tupel von Algorithmen Gen, Sig, Ver zur Schlüsselgenerierung, Signaturberechnung beziehungsweise Signaturverifikation. Aus dieser Definition wird nun eine Formalisierung für Signaturverfahren mit subliminalem Kanal abgeleitet. Ein subliminaler Kanal soll dabei jeweils mit überwältigender Wahrscheinlichkeit die Kommunikation ermöglichen, sowie sie vor nicht autorisierten Parteien verborgen halten.

Bei der Verwendung eines subliminalen Kanals soll der Anschein erweckt werden, dass das Signaturverfahren ehrlich benutzt wird. Zunächst sei ein beliebiges Signaturverfahren durch die Algorithmen Gen, Sig, Ver gegeben. Um den subliminalen Kanal zu definieren, werden zusätzliche Algorithmen für das Verbergen und Extrahieren der subliminalen Nachricht benötigt. Desweiteren muss die Schlüsselgenerierung neben dem Signatur- und Verifikationsschlüssel auch einen geheimen *Subliminalschlüssel* er-

zeugen, der nötig sein wird, um die Nachricht zu verbergen und zu extrahieren. Um keinen Verdacht zu erregen, sollte jedoch die Verteilung der Schlüsselpaare (PK, SK) von derjenigen des Algorithmus Gen nicht zu unterscheiden sein.[5]

Es ist erlaubt, dass der Subliminalschlüssel auch den geheimen Signaturschlüssel beinhaltet, also auch dem Empfänger der verborgenen Nachricht ermöglicht, Signaturen im Namen des eigentlichen Schlüsselinhabers zu berechnen. Das ist durch Szenarien motiviert, in denen die verborgene Kommunikation das eigentliche Ziel der Teilnehmer ist und das Signaturverfahren letztendlich nur zur Tarnung eingesetzt wird.

Der subliminale Kanal muss einige Bedingungen erfüllen, um wirklich subliminal zu sein: Es ist ein Wächter als Algorithmus gegeben, der versucht, die subliminale Kommunikation zu entdecken. Bei einem subliminalen Kanal darf ihm das höchstens mit einer vernachlässigbaren Wahrscheinlichkeit gelingen, selbst wenn er die verborgene Nachricht kennt oder gar selbst vorgeben kann. Gleichzeitig muss der autorisierte Empfänger die Nachricht aber mit überwältigender Wahrscheinlichkeit rekonstruieren können. Die Definition der Verborgenheit wird durch ein Experiment formuliert, ganz ähnlich der Definition des Sicherheitsbegriffs IND-CCA („*indistinguishability of ciphertexts under chosen-ciphertext attacks*“) für Public-Key-Verschlüsselungssysteme.

Definition 3.1 (Subliminaler Kanal)**.** *Ein* Signaturverfahren mit subliminalem Kanal S *für einen Subliminalnachrichtenraum* $\mathcal{M}_{sub} = \mathcal{M}_{sub}(\ell)$ *wird durch ein Tupel von Algorithmen definiert:*

$$S = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{SubGen}, \mathsf{SubSig}, \mathsf{SubVer})$$

Die Algorithmen Gen, Sig, Ver *bilden ein Signaturverfahren gemäß Definition 2.1, die übrigen Algorithmen haben die folgenden Eigenschaften:*

- SubGen *ist ein PPT-Algorithmus. Die Eingabe ist der Sicherheitsparameter* ℓ *und die Ausgabe ein Schlüsselpaar* (PK, SK)*, sowie ein geheimer Subliminalschlüssel* K_{sub} *für die verborgene Kommunikation. Die Verteilung der durch* SubGen *erzeugten Schlüsselpaare* (PK, SK) *muss von der Verteilung der von* Gen *erzeugten Schlüsselpaare ununterscheidbar sein.*

- SubSig *ist ein PPT-Algorithmus, der die alternative Signaturberechnung mit verborgener Nachricht ermöglicht. Der Algorithmus nimmt als Eingabe eine Nachricht* M*, eine zu verbergende Nachricht* $m \in \mathcal{M}_{sub}$*, den Signaturschlüssel* SK *und Zustandsinformation* s*, die in der letzten Aktivierung von* SubSig *ausgegeben wurde (bei der ersten Aktivierung ist* $s := K_{sub}$*). Die Ausgabe ist ein neuer Zustand* s' *sowie eine Signatur* σ *für die Nachricht* M*, so dass* σ *die Nachricht* m *verbirgt. Die Gründe für den Zustand* s *werden in Bemerkung 3.2 näher erläutert.*

[5] Eine schwächere Forderung wäre, dass nur der öffentliche Schlüssel PK von einem mit Gen erzeugten ununterscheidbar ist. Dann kann der subliminale Kanal jedoch offenbar werden, falls der Unterzeichner gezwungen wird, seinen geheimen Schlüssel offenzulegen.

- SubVer *ist ein deterministischer polynomieller Algorithmus, der als Eingabe eine Nachricht M, eine Signatur σ, den öffentlichen Verifikationsschlüssel PK und den Subliminalschlüssel K_{sub} nimmt. Der Algorithmus* SubVer *gibt genau dann* invalid *aus, wenn auch* Ver *auf Eingabe (M, σ, PK)* invalid *ausgibt. Falls die Signatur gültig ist und von* SubSig *erzeugt wurde, gibt* SubVer valid *und die verborgene Nachricht $m \in \mathcal{M}_{sub}$ aus.*

Die verborgene Kommunikation soll mit überwältigender Wahrscheinlichkeit funktionieren, wobei ausschließlich gültige Signaturen als Medium erlaubt sind:
$\forall (PK, SK, K_{sub}) \leftarrow \mathsf{SubGen}(1^\ell), \forall m \in \mathcal{M}_{sub}$ *und* $\forall M \in \{0,1\}^*$*:*

$$\Pr\left[\begin{array}{l}(\sigma, s') \leftarrow \mathsf{SubSig}(M, m, SK, K_{sub}) : \\ \mathsf{SubVer}(M, \sigma, PK, K_{sub}) = (\mathsf{valid}, m)\end{array}\right] \geq 1 - \mathrm{negl}(\ell).$$

Die Wahrscheinlichkeit wird hier über die Zufallswahlen des Algorithmus SubSig *gebildet.*

Die Kommunikation soll gegenüber einem Wächter, der den Austausch der Signaturen überwacht, verborgen bleiben. Für alle Sicherheitsparameter ℓ muss der Subliminalnachrichtenraum mindestens zwei verschiedene Nachrichten enthalten[6] *und für alle PPT-Algorithmen W soll gelten,*

$$\left|P[\mathbf{Exp}_{S,W}^{\mathrm{ward-ind-1}}(\ell) = 1] - P[\mathbf{Exp}_{S,W}^{\mathrm{ward-ind-0}}(\ell) = 1]\right| \leq \mathrm{negl}(\ell)$$

mit den folgenden Experimenten $\mathbf{Exp}_{S,W}^{\mathrm{ward-ind-b}}(\cdot)$ für $b \in \{0,1\}$:

Experiment $\mathbf{Exp}_{S,W}^{\mathrm{ward-ind-0}}(\ell)$:	*Experiment* $\mathbf{Exp}_{S,W}^{\mathrm{ward-ind-1}}(\ell)$:
$(PK, SK) \leftarrow \mathsf{Gen}(1^\ell)$;	$(PK, SK, K_{sub}) \leftarrow \mathsf{SubGen}(1^\ell)$;
$d \leftarrow W^{\mathcal{S}_{SK}(\cdot,\cdot)}(PK)$;	$d \leftarrow W^{\mathcal{S}_{SK,s}(\cdot,\cdot)}(PK)$;
return d;	*return* d.

Hier ist $\mathcal{S}_{SK}(\cdot,\cdot)$ ein Orakel, das auf Eingabe (M, m) die Signatur $\sigma \leftarrow \mathsf{Sig}(M, SK)$ zurückgibt und $\mathcal{S}_{SK,s}(\cdot,\cdot)$ ist ein Orakel, das auf die Eingabe (M, m) eine Signatur σ mit $(\sigma, s) \leftarrow \mathsf{SubSig}(M, m, SK, s)$ zurückgibt und seinen internen Zustand auf s setzt (der Anfangszustand ist $s = K_{sub}$).

Bemerkung 3.2. *Falls der subliminale Kanal in einem deterministischen Signaturverfahren existiert, ist die Zustandshaltung des Algorithmus* SubSig *wesentlich: Wenn eine subliminale Nachricht $m_1 \in \mathcal{M}_{sub}$ in einer Signatur σ einer Nachricht M versteckt ist, dann kann die Nachricht M nicht mehr benutzt werden, um $m_2 \in \mathcal{M}_{sub} \setminus \{m_1\}$ zu verstecken. Sonst wäre das Resultat eine andere Signatur $\sigma' \neq \sigma$ für M und der subliminale Kanal oder zumindest ein unehrliches Verhalten des Unterzeichners wäre offensichtlich (siehe auch [74]).*

[6] Sonst kann über den Kanal keine Information übertragen werden

3.3.2 Kapazität eines subliminalen Kanals

Ein wichtiges Merkmal eines subliminalen Kanals ist seine Kapazität. Der Kanal kann für eine vorgesehene Anwendung möglicherweise nicht genutzt werden, falls er eine geringe Kapazität hat, also nur sehr kurze Nachrichten mit einer Länge von wenigen Bits versteckt werden können. Auch wenn die subliminale Kommunikation unerwünscht ist, mag ein Kanal mit geringer Kapazität noch tolerierbar sein. Eine hohe Kapazität dagegen vergrößert die Gefahr die von einem subliminalen Kanal ausgeht. Beispielsweise benötigt ein böswilliger Signieralgorithmus mit SETUP dann weniger Signaturen, um den geheimen Signaturschlüssel zu verbreiten.

Es wird die Kapazität pro Signatur betrachtet. Da die Signaturen mit subliminaler Nachricht ununterscheidbar von Signaturen sein müssen, bei denen keine Nachricht eingebettet wurde, kann die Länge der Signatur nicht mit der Länge der subliminalen Nachricht wachsen. Der Subliminalnachrichtenraum ist daher endlich. SIMMONS [113] unterschied zwischen Schmalband- und Breitbandkanälen. Dabei nannte er Kanäle, bei denen der benutzte Zufall effizient komplett durch eine subliminale Nachricht ersetzt werden kann, *Breitbandkanal* und Kanäle, bei denen der Aufwand, um die Signatur mit subliminaler Nachricht zu berechnen, exponentiell mit der Nachrichtenlänge steigt, *Schmalbandkanal*. Die Information, die über den subliminalen Kanal übertragen werden kann, wird in der folgenden Definition quantifiziert:

Definition 3.3 (Kapazität)**.** *Sei $\mathcal{M}_{sub} = \mathcal{M}_{sub}(\ell)$ der Subliminalnachrichtenraum eines Signaturverfahrens mit subliminalem Kanal S und $P_{\mathcal{M}_{sub}}$ eine Wahrscheinlichkeitsverteilung auf $\mathcal{M}_{sub}$. Dann ist die durchschnittliche Information pro subliminaler Nachricht*

$$H(\mathcal{M}_{sub}) = -\sum_{m \in \mathcal{M}_{sub}} P_{\mathcal{M}_{sub}}(m) \log_2(P_{\mathcal{M}_{sub}}(m)) \quad \textit{(in bit pro Signatur),}$$

die Entropie in der Quelle der Subliminalnachrichten. Sei Σ_S die Menge der Signaturverfahren mit subliminalem Kanal, die bis auf den Subliminalnachrichtenraum $\mathcal{M}_{sub}$ identisch zu S sind.

Wenn es in einer polynomiellen Anzahl von Ausführungen des Signieralgorithmus SubSig *mit überwältigender Wahrscheinlichkeit gelingt, eine Nachricht zu verstecken,*[7] *dann wird mit $C(S) := min_{\Sigma_S} H(\mathcal{M}_{sub})$ die* Kapazität *des subliminalen Kanals bezeichnet. Den Bezeichnungen aus [115] folgend, heißt der subliminale Kanal* Breitbandkanal*, falls die Kapazität wenigstens linear im Sicherheitsparameter ℓ wächst, d. h. $C(S) \in \Omega(\ell)$. Der subliminale Kanal heißt* Schmalbandkanal*, falls seine Kapazität höchstens logarithmisch in ℓ wächst, d. h., $C(S) \in \mathrm{O}(\log_2(\ell))$.*

Für ein deterministisches Signaturverfahren stellt sich die Situation anders da. Die Information, die subliminal übertragen werden kann, hängt von den Nachrichten ab,

[7] Deterministische Verfahren erfüllen das nur, falls die Wahrscheinlichkeit, dass zweimal dieselbe Nachricht signiert wird, vernachlässigbar ist.

die signiert werden, da jede Nachricht nur einmal subliminal signiert werden kann, ohne durch dem Wächter aufzufallen. Gegeben eine Verteilung $P_{\mathcal{M}}$ der Nachrichten, die signiert werden, beschreibt die Folge $B = (B_i)_{i\in\mathbb{N}} \in \mathbb{R}^{\mathbb{N}}$ die Information, die in der $i$ten Signatur durchschnittlich übertragen werden kann:

$$B_i = C(S) \cdot \sum_{M\in\mathcal{M}} (1 - P_{\mathcal{M}}(M))^{i-1} P_{\mathcal{M}}(M)$$

Praktisch wird es jedoch fast nie nötig sein, dieselbe Nachricht doppelt zu unterzeichnen, da in Anwendungen üblicherweise ein frischer Wert wie beispielsweise ein Zeitstempel mitsigniert wird.

3.3.3 Codierung der subliminalen Nachricht

In diesem Abschnitt wird genauer untersucht, wie die subliminale Nachricht verschlüsselt werden muss, bevor sie in einer Signatur versteckt werden kann. Zunächst werden subliminale Kanäle in probabilistischen Verfahren betrachtet. Es stellt sich heraus, dass Bemerkung 3.2 in gewisser Weise dual zu probabilistischen Verfahren ist, die superpolynomiell viele Signaturen für eine Nachricht ermöglichen: In diesem Fall wäre es verdächtig, wenn zwei Signaturausführungen dieselbe Signatur liefern. Anders ausgedrückt, für probabilistische Signaturverfahren mit einer superpolynomiellen Anzahl an Signaturen ist es wichtig, dass wiederholte Signaturanfragen für dieselbe Nachricht immer unterschiedliche Signaturen liefern, selbst wenn jeweils die gleiche subliminale Nachricht verborgen werden soll. Die Verschlüsselung der subliminalen Nachricht muss also probabilistisch sein und dem Signieralgorithmus genügend verschiedene „Zufallswerte“ zur Verfügung stellen.

Typischerweise soll die Zufallszahl r, die in dem Signieralgorithmus verwendet wird und in der Regel eine lineare Länge im Sicherheitsparameter hat, durch den Chiffretext der subliminalen Nachricht ersetzt werden. Bei wiederholten Signieranfragen muss es immer wieder möglich sein, einen neuen, wieder wie eine Zufallsfolge aussehenden Chiffretext derselben Nachricht zu produzieren. Von der Anwendung abhängig sind unterschiedliche Verfahren denkbar, um solche Chiffretexte zu erhalten. Ein Public-Key-Verfahren, das pseudozufällige Chiffretexte produziert, wird beispielsweise in [94] vorgestellt. Für die hier verwendete Modellierung subliminaler Kanäle reichen jedoch Secret-Key-Verfahren, da der Subliminalschlüssel K_{sub} gemeinsam beim Sender und Empfänger der subliminalen Nachricht vorhanden ist.

Eine Möglichkeit dazu ist die folgende: Sei Δ ein Bitstring, dessen Länge superpolylogarithmisch im Sicherheitsparameter ist. Dieser String wird als Teil des geheimen Subliminalschlüssels K_{sub} festgelegt. Desweiteren sei $\mathrm{MGF}(\cdot,\cdot)$ eine „*mask generation function*“ (MGF), d. h. eine Funktion, die aus einer Eingabe z und einer natürlichen Zahl ℓ eine pseudozufällige Bitstring der Länge ℓ berechnet, um Werte maskieren zu können. Für praktische Zwecke finden sich in [103, Appendix B.2] Konstruktionen von MGFs. Eine subliminale Nachricht m der Länge $|m|$ wird dann als

$$c = \Xi \| [m \oplus \mathrm{MGF}(\Delta \| \Xi, |m|)]$$

verschlüsselt, wobei Ξ ein gleichverteilt zufällig gewählter Bitstring superpolylogarithmischer Länge im Sicherheitsparameter ist.

Die Entschlüsselung für den Empfänger der subliminalen Nachricht ist entsprechend: Zunächst wird aus dem Chiffretext c der Zufallswert Ξ gelesen und anschließend die Maske $\mathrm{MGF}(\Delta \| \Xi, |m|)$ berechnet und von der Nachricht abgezogen. Unter der Annahme, dass sich die Funktion $\mathrm{MGF}(\cdot, |m|)$ wie ein Random-Oracle[8] verhält, ist es nicht schwer einzusehen, dass ein PPT-Algorithmus nicht zwischen einem Chiffretext und einer echt zufälligen Zeichenfolge der Länge $|\Xi| + |m|$ unterscheiden kann.

Wenn der Wächter so beschränkt wird, dass er das Signaturorakel nur einmal für jede Nachricht M aufrufen darf, wird die Variable Ξ nicht benötigt, als Eingabe für MGF genügt dann nur die nun eindeutige Nachricht M. In einem Szenario, in dem von vornherein klar ist, dass jede Nachricht garantiert nur einmal signiert werden muss, braucht die Anzahl der möglichen Signaturen pro Nachricht also nicht im Sicherheitsparameter ℓ zu wachsen. Entsprechend ist die Situation in diesem Szenario für deterministische Verfahren, da eine subliminale Nachricht ohnehin nur bei der ersten Signaturberechnung einer Nachricht verborgen werden darf.

Eine Alternative, die superpolynomielle Anzahl der Signaturen zu erhalten, ist es, dem Algorithmus SubVer zu erlauben einen fortdauernden Zustand zu haben. Falls darüber hinaus der Empfänger der subliminalen Nachricht die Ordnungsnummer jeder Signatur erkennen kann, kann beispielsweise eine One-Time-Pad-Verschlüsselung verwendet werden.

3.4 Subliminale Kanäle in deterministischen Signaturverfahren

Es werden nun subliminale Kanäle in einigen deterministischen Signaturverfahren aufgezeigt. Es werden hauptsächlich Verfahren, die im NESSIE Portfolio [97] empfohlen werden, betrachtet, um eine Relevanz der untersuchten Verfahren zu gewährleisten. Doch zunächst soll ein Verfahren betrachtet werden, dessen Sicherheit auf dem Konjugationsproblem in Zopfgruppen beruht.

3.4.1 Ein zopfgruppenbasiertes Verfahren

In Ko u. a. [87] wird ein neues interessantes Signaturverfahren basierend auf Zopfgruppen vorgestellt. Obwohl die Arbeit einen konzeptuellen Charakter trägt und mit Sicherheit weit von einer Anwendung entfernt ist, dient es der Veranschaulichung von subliminaler Kommunikation in deterministischen Signaturverfahren aufgrund der Art wie die Nachricht versteckt wird und wie eine subliminale Kommunikation eventuell

[8] Ein *Random-Oracle* ist eine Funktion, die jeder Eingabe eine echt zufällige Ausgabe aus dem Bildbereich zuordnet; vgl. [11, 46].

detektiert werden kann. Es handelt sich bei dieser subliminalen Kommunikationsmöglich jedoch nicht um einen subliminalen Kanal wie in Definition 3.1.

Gegeben sei eine Zopfgruppe B_n. Dann heißen zwei Elemente $x, y \in B_n$ zueinander *konjugiert*, geschrieben $x \sim y$, falls ein Element $a \in B_n$ existiert, so dass $y = a^{-1}xa$ ist. Für Elemente $x, y \in B_n$ ist das „*conjugacy decision problem*“ (CDP) das Problem, zu entscheiden, ob $x \sim y$, und das „*conjugator search problem*“ (CSP) ist das Problem, ein Element $a \in B_n$ zu finden, so dass $y = a^{-1}xa$ ist.

Das Signaturverfahren basiert nun auf einer sogenannten Gap-Annahme, dass das CDP in der benutzten Zopfgruppe effizient lösbar ist, während CSP hart ist. Zusätzlich wird eine kryptographische Hashfunktion $h : \{0,1\}^* \longrightarrow B_n$ benötigt, die eine Nachricht auf ein Element der Zopfgruppe abbildet. Der geheime Schlüssel für das Signaturverfahren ist ein Element $a \in B_n$ und der öffentliche Schlüssel ein bezüglich CSP hartes Paar $(x, x') \in B_n^2$ mit $x' = a^{-1}xa$. Schließlich wird eine Nachricht $M \in \{0,1\}^*$ signiert, indem $\sigma = a^{-1}ya$ berechnet wird, wobei $y = h(M)$ der Hashwert von M ist. Die Signaturverifikation überprüft, ob $\sigma \sim y$ und $x'\sigma \sim xy$ und ergibt genau dann valid, wenn beide Konjugationen erfüllt sind.

Das Signaturverfahren ist vom Konstruktionsprinzip her deterministisch. Es ist jedoch nicht subliminalfrei nach der Definition 3.5: Sei M eine Nachricht mit $y = h(M)$, die nicht mit einem gewählten Element x kommutiert. Statt der Signatur $\sigma = a^{-1}ya$ kann der Unterzeichner auch $\sigma' = (xa)^{-1}y(xa)$ berechnen. σ' ist tatsächlich eine gültige Signatur für M, weil $\sigma' \sim y$ und

$$x'\sigma' = a^{-1}xaa^{-1}x^{-1}yxa = a^{-1}yxa = a^{-1}x^{-1}xyxa = (xa)^{-1}xy(xa) \sim xy.$$

Ein autorisierter Empfänger kann aber einfach zwischen σ und σ' unterscheiden, falls er den geheimen Signaturschlüssel a kennt. Das Signaturverfahren bietet dennoch keinen subliminalen Kanal nach Definition 3.1: Falls einem Wächter zwei Signaturen bekannt sind, in denen unterschiedliche subliminale Nachrichten versteckt sind, kann der Wächter die subliminale Kommunikation bemerken. Für zwei ehrlich berechnete Signaturen σ_1 von M_1 und σ_2 von M_2, mit $y_1 = h(M_1)$ und $y_2 = h(M_2)$ ergibt sich das Produkt $\sigma_1\sigma_2 = a^{-1}y_1aa^{-1}y_2a \sim y_1y_2$. Falls jedoch beispielsweise die erste Signatur als $\sigma_1' = (xa)^{-1}y_1(xa)$ gewählt wurde, dann ist das Produkt $\sigma_1'\sigma_2 = a^{-1}x^{-1}y_1xaa^{-1}y_2a = a^{-1}x^{-1}y_1xy_2a$ und wird im Allgemeinen nicht mit y_1y_2 konjugiert sein.

3.4.2 Ein polynombasiertes Verfahren: SFLASH^{v3}

SFLASH^{v2} ist ein Signaturverfahren, das als Empfehlung in das NESSIE Portfolio [97] aufgenommen wurde. Das Verfahren wurde von den Autoren zunächst nicht mehr empfohlen und zu SFLASH^{v3} [51] weiterentwickelt, mittlerweile wird jedoch wieder SFLASH^{v2} empfohlen.

SFLASH^{v3} passt nicht in die Modelle der beweisbaren Sicherheit und zeigt Grenzen dieser Definitionen auf: SFLASH^{v3} ist nicht für einen „skalierbaren“ Sicherheitsparameter ℓ, sondern nur mit festen Parametern definiert. Das wird weder von der

Algorithmus 7 Das Signaturverfahren SFLASH^{v3}

Domainparameter:

$K \simeq \mathbb{F}_{128}$	endlicher Körper mit 128 Elementen
$L \simeq \mathbb{F}_{128^{67}}$	Körpererweiterung von K vom Grad 67
$\pi : \{0,1\}^7 \longrightarrow K$	Bijektion
$\varphi : K^{67} \longrightarrow L$	Bijektion

Schlüsselerzeugung Gen: Der geheime Schlüssel SK setzt sich aus einem 80 bit langen Bitstring Δ und zwei affinen Bijektionen $s, t : K^{67} \longrightarrow K^{67}$ zusammen. Die Funktion $G(X) = [(t \circ \varphi^{-1} \circ F \circ \varphi \circ s)(X)]$ mit $F : L \longrightarrow L$, $\alpha \mapsto \alpha^{128^{33}+1}$ und $(f \circ g)(x) := f(g(x))$ kann auch durch Polynome

$$(P_0(X_0, \ldots, X_{66}), \ldots, P_{66}(X_0, \ldots, X_{66}))$$

mit Polynomen P_i vom Gesamtgrad ≤ 2 und Koeffizienten aus K geschrieben werden. Der öffentliche Schlüssel besteht nun aus den ersten 56 Polynomen $PK = (P_0, \ldots, P_{55})$.

Signaturberechnung Sig: Um eine Nachricht M zu signieren, wird zunächst ein 392-bit-String V durch mehrere Benutzungen von SHA-1 aus M berechnet. Mittels π wird aus V der Vektor

$$Y := (\pi([V]_{0\to 6}), \quad \pi([V]_{7\to 13}), \ldots, \pi([V]_{385\to 391})) \in K^{56},$$

berechnet, wobei die Notation $[\cdot]_{a\to b}$ die Bits an den Stellen a bis b auswählt. Nun wird SHA-1 auf die Konkatenation von V und Δ angewendet und aus den ersten 77 bit des Hashwerts $W = \text{SHA-1}(V||\Delta)$ wieder durch π ein Vektor

$$R := (\pi([W]_{0\to 6}), \quad \pi([W]_{7\to 13}), \ldots, \pi([W]_{70\to 76})) \in K^{11}$$

berechnet. Die Konkatenation von Y und R wird nun auf einen Vektor $X := (s^{-1} \circ \varphi^{-1} \circ F^{-1} \circ \varphi \circ t^{-1})(Y||R) \in K^{67}$ abgebildet. Aus X kann schließlich durch Anwendung von π^{-1} der 469-bit-String σ, der die Signatur von M bildet, berechnet werden.

Signaturverifikation Ver: Um eine Signatur σ einer Nachricht M zu verifizieren, wird aus σ zunächst mittels π ein Vektor $X \in K^{67}$ berechnet. Durch Auswertung der 56 öffentlichen Polynome aus PK an X wird ein Vektor $Y \in K^{56}$ berechnet. Die Verifikation der Signatur σ gibt genau dann valid zurück, wenn Y mit dem Wert Y' übereinstimmt, der wie im Signieralgorithmus aus der Nachricht M berechnet wird.

Sicherheitsdefinition für Signaturverfahren, noch von der Definition der subliminalen Kanäle erfasst. Aus praktischer Sicht hat ein Verfahren mit festen Parametern jedoch durchaus seine Berechtigung. Ebenso ist ein intuitiv klar erkennbarer subliminaler Kanal vorhanden und soll daher hier beschrieben werden.

Details zu SFLASH^{v3} finden sich in COURTOIS U. A. [51], zur Sicherheit des Verfahrens sollte auch DING UND SCHMIDT [56] beachtet werden. Hier reicht eine grobe Zusammenfassung des Verfahrens aus, wie sie für das Verständnis des subliminalen Kanals nötig ist. Das Verfahren wird in Algorithmus 7 beschrieben.

Signieren mit subliminaler Nachricht. Die Verifikation von SFLASH^{v3} überprüft nicht, ob der Wert R aus dem Signieralgorithmus wie vorgegeben berechnet wurde. An Stelle von R kann also die subliminale Nachricht eingefügt werden. Die Funktion SubSig unterscheidet sich von Sig also dadurch, dass R nicht mittels SHA-1 aus Δ und V berechnet wird, sondern aus dem 77-bit-Chiffretext $c = (c_0, \ldots, c_{76})$ der subliminalen Nachricht durch

$$R = (\pi([c]_{0\to 6}), \quad \pi([c]_{7\to 13}), \ldots, \pi([c]_{70\to 76})) \in K^{11}.$$

Das Ergebnis ist eine gültige Signatur von M. Die maximale Bandbreite des subliminalen Kanals beträgt 77 bit.

Damit der autorisierte Empfänger der Nachricht R rekonstruieren kann, braucht er zusätzlich zum öffentlichen Schlüssel PK mit den Polynomen $P_0, \ldots, P_{55}$ auch die Polynome $P_{56}, \ldots, P_{66}$, um die 77-bit-Zahl R – also die verschlüsselte, subliminale Nachricht – zu berechnen. Den Vektor $X \in K^{67}$ wertet er in SubVer an allen Polynomen $P_0, \ldots, P_{66}$ aus und erhält dadurch sowohl Y als auch R.

3.4.3 Ein faktorisierungsbasiertes Verfahren: RSA-PSS

RSA-PSS [103] ist ein Signaturverfahren, das auf dem RSA-Algorithmus (siehe Algorithmus 3) basiert. Das Verfahren benutzt ein im Allgemeinen randomisiertes Padding, die sogenannte PSS-Codierung. Das Padding-Verfahren PSS („*probabilistic signature scheme*“) wurde ursprünglich von BELLARE UND ROGAWAY [13] eingeführt. Das Signaturverfahren bietet einen Breitbandkanal, der dem Kanal in DSA weit überlegen ist, weil der autorisierte Empfänger keinen Vorteil hat, eine Signatur zu fälschen.

PSS-Codierung. Für die PSS-Codierung werden eine kryptographische Hashfunktion $H(\cdot)$ und eine „*mask generation function*“ $\mathrm{MGF}(\cdot,\cdot)$ benötigt. Um eine Nachricht M zu codieren, ist zunächst ein zufälliger Bitstring salt von einer vorgegebenen Länge für die Randomisierung des Verfahrens zu wählen. Die PSS-Spezifikation legt keine bestimmte Länge für das salt fest, eine typische Länge ist *hLen*, die Ausgabelänge der Hashfunktion H, oder 0 für ein deterministisches Padding. Die Länge der PSS-codierten Nachricht entspricht der Länge des Modulus n, der mindestens linear im Sicherheitsparameter ℓ wächst. Daher ist es vernünftig, auch die Länge des salt linear

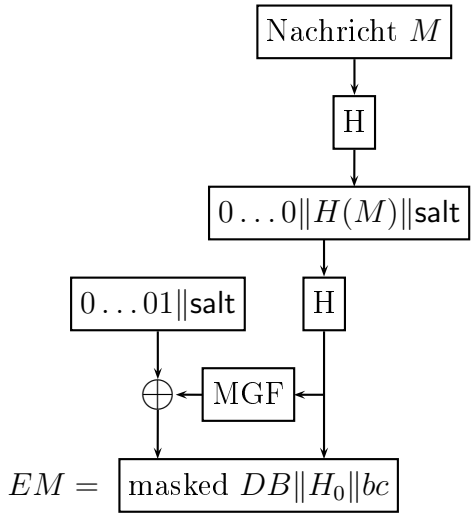

Bild 3.1: Schema der PSS-Codierung.

in ℓ zu wählen. Das verursacht jedoch die Entstehung des subliminalen Breitbandkanals. Die PSS-Codierung EM einer Nachricht M wird folgendermaßen berechnet:

$$EM = [(0\ldots0\|1\|salt) \oplus \mathrm{MGF}(H_0, emLen - hLen - 1)]\|H_0\|bc,$$

wobei

$$H_0 = H(\underbrace{0\ldots0}_{8\text{ Byte}}||H(M)\|\mathsf{salt})$$

ein Hashwert der Länge $hLen$ ist, $emLen$ die Länge des Strings EM in Byte und bc ein Byte mit dem hexadezimalen Wert bc. Ein Schema der PSS-Codierung zeigt Bild 3.1 Die Signatur σ wird durch Anwendung des RSA-Signaturalgorithmus (Algorithmus 3) auf EM berechnet.

Zur Verifikation einer Signatur σ der Nachricht M muss der Verifizierer zuerst die PSS-codierte Nachricht durch Anwendung des RSA-Algorithmus berechnen: $EM = \sigma^e \bmod n$. Als nächstes muss er prüfen, ob das berechnete EM tatsächlich eine gültige PSS-Codierung der Nachricht M ist. Zuerst kann H_0 unmittelbar aus EM extrahiert werden. Anschließend ist es möglich, $\mathrm{MGF}(H_0, emLen - hLen - 1)$ zu berechnen, woraus der Zufallswert salt extrahiert werden kann. Mit dem salt ist es schließlich einfach, aus der Nachricht $H_0' = H(0\ldots0\|H(M)\|\mathsf{salt})$ zu berechnen und mit H_0 zu vergleichen.

Ein subliminaler Breitbandkanal in (probabilistischem) RSA-PSS. Es wurde schon von Bao und Wang [7] gezeigt, dass die randomisierte PSS-Codierung einen subliminalen Breitbandkanal beherbergt. Der öffentliche Schlüssel des Unterzeichners genügt, um die PSS-Codierung der Nachricht sowie ebenfalls den Zufallsstring, der beim

Signieren verwendet wurde, zu berechnen. Im Gegensatz zu den angesprochenen subliminalen Kanälen im DSA benötigt ein Empfänger der subliminalen Nachricht keinen Zugriff auf den geheimen Signaturschlüssel und erhält auch keinen Vorteil, eine Signatur zu fälschen. Zudem hat der Kanal eine hohe Bandbreite: Für die PSS-Codierung ist es empfohlen, einen Zufallsstring in der Länge der Hashfunktionsausgabe zu verwenden. Da geplant ist, Hashfunktionen mit 256-bit-Ausgabe zu verwenden, kann im optimalen Fall[9] auch eine subliminale Nachricht mit 256 bit verborgen werden. Damit sind beispielsweise SETUP-Angriffe mit diesem Verfahren äußerst gefährlich [129].

Proposition 3.4. *Unter der Annahme, dass die Länge des* salt *linear im Sicherheitsparameter wächst, bietet das RSA-PSS-Signaturverfahren einen subliminalen Breitbandkanal.*

Beweis. Wie bereits erwähnt, erfordert die Verifizierung einer Signatur die Berechnung des verwendeten Zufallsstrings salt. Es genügt also, die subliminale Nachricht so zu verschlüsseln, dass sie von einen Zufallsstring nicht zu unterscheiden ist – beispielsweise mit dem in Abschnitt 3.3.3 beschriebene Verfahren. Falls die Länge des salt linear im Sicherheitsparameter wächst, ist der Kanal ein Breitbandkanal. Es muss beachtet werden, dass der Schlüssel $|\Xi|$ superpolylogarithmisch im Sicherheitsparameter ℓ ist. □

3.4.4 Ein faktorisierungsbasiertes Verfahren: ESIGN-D

Bevor auf ESIGN-D eingegangen wird, soll zunächst eine Zusammenfassung von ESIGN [61] und dem darin vorhandenen subliminalen Kanal präsentiert werden. ESIGN ist ein effizientes Signaturverfahren, das auf dem Faktorisierungsproblem und dem Berechnen näherungsweiser $e$ten Wurzeln[10] in einer Gruppe unbekannter Ordnung basiert. Der öffentliche Schlüssel ist ein Paar (n, e), wobei $e \geq 8$ und $n = p^2q$ für ℓ-bit-Primzahlen p, q, so dass n eine Bitlänge von 3ℓ hat. Im Gegensatz zu RSA muss e nicht koprim zu $\phi(n)$ sein. Der geheime Schlüssel ist das Paar (p, pq). Desweiteren ist eine Hashfunktion $H(\cdot)$ mit Ausgabelänge $k-1$ gegeben. Das Verfahren berechnet als Signatur einer Nachricht M effizient eine Zahl, die eine $e$te Wurzel von $H(M)\|\cdot^{2\ell-1}$ ist, einer Zahl, deren höchstwertige Bits $H(M)$ sind, während die restlichen $2\ell - 1$ Bits beliebige Werte annehmen dürfen. Das Verfahren wird genauer in Algorithmus 8 beschrieben.

Ein subliminaler Kanal in ESIGN

Kuwakado und Tanaka [88] beschreiben einen subliminalen Kanal in ESIGN. Anstatt auf die sehr effiziente Weise die $e$te Wurzel eines in gewissen Grenzen zufälligen Wertes zu berechnen, kann ein bestimmtes Element gewählt werden, das die subliminale Nachricht codiert, und dann eine exakte $e$te Wurzel dieses Elements berechnet

[9] falls keine randomisierte Verschlüsselung der Nachricht nötig ist

[10] im Gegensatz zu RSA, das exakte $e$te Wurzeln berechnet

Algorithmus 8 Das Signaturverfahren ESIGN.

Domainparameter:

ℓ Sicherheitsparameter
H kollisionsresistente Hashfunktion

Schlüsselerzeugung Gen: Der Algorithmus wählt zwei Primzahlen p und q der Länge ℓ bit und berechnet $n = p^2q$. Zusätzlich wird eine Zahl $e \geq 8$ gewählt. Die Schlüssel sind dann

$$PK = (n, e) \quad \text{und} \quad SK = (p, pq).$$

Signaturberechnung Sig: Zunächst wird eine Zufallszahl $r < pq$ gewählt. Anschließend werden

$$\alpha \equiv H(M) - r^e \pmod{n},$$
$$w_0 = \lceil \frac{\alpha}{pq} \rceil \quad \text{und} \quad w_1 = w_0 \cdot pq - \alpha$$

berechnet. Falls $w_1 \geq 2^{2\ell-1}$ sein sollte, muss von vorne mit der Wahl einer neuen Zufallszahl r begonnen werden. Anderenfalls ist die Signatur durch

$$\sigma \equiv r + t \cdot pq \pmod{n}$$

gegeben, wobei $t \equiv w_0/(er^{e-1}) \pmod{p}$.

Signaturverifikation Ver: Die Verifikation geschieht durch Prüfen, ob σ eine $e$te Wurzel von $(H(M)\|\cdot^{2\ell-1}) \pmod{n}$ ist, d. h., es wird $\sigma^e \pmod{n}$ berechnet und geprüft, ob die ℓ höchstwertigen Bits $0\|H(M)$ entsprechen.

werden. Der subliminale Signieralgorithmus sieht folgendermaßen aus: Um die Nachricht M zu signieren und eine subliminale Nachricht m mit $|m| < |pq| - \ell - 1$ zu verstecken, wird zunächst eine Verschlüsselung von m in einen zufällig aussehenden Bitstring c der Länge $|pq| - 1$ berechnet. Falls $H(M) \cdot 2^\ell + c$ keine $e$te Wurzel hat[11], muss eine neue Verschlüsselung von m berechnet werden. Anderenfalls berechnet sich die Signatur σ von M als $e$te Wurzel aus $H(M) \cdot 2^\ell + c$. Das kann praktikabel berechnet werden [63], aber die Effizienz, die ESIGN auszeichnet, ist verloren. Die Signatur ist gültig, wie die Verifikation

$$\sigma^e \bmod n \equiv H(M) \cdot 2^\ell + c = 0\|H(M)\|c$$

beweist – die ℓ höchstwertigen Bits von $\sigma^e \pmod{n}$ sind $0\|H(M)$. Um die subliminale Nachricht zu extrahieren, muss der autorisierte Empfänger die niederwertigsten Bit von $\sigma^e \pmod{n}$ mit seinem Subliminalschlüssel entschlüsseln.

[11] Das kann passieren, falls e und n nicht teilerfremd sind.

ESIGN-D

Der Sicherheitsbeweis, der ursprünglich für ESIGN gegeben war, war nicht korrekt, weil angenommen wurde, dass das Signaturorakel zu einer Nachricht immer dieselbe Signatur zurückgibt, sich also deterministisch verhält [118]. Ohne diese Annahme ließ sich das Originalverfahren nicht beweisen. Daher schlägt GRANBOULAN [69] vor, ESIGN deterministisch zu verwenden, und nennt diese Variante ESIGN-D. ESIGN-D ist mit ESIGN identisch, bis auf die Zufallswahl der Zahl r im Signieralgorithmus. Anstatt einer Zufallszahl wird bei ESIGN-D die Ausgabe einer Pseudozufallsfunktion $\Phi(\cdot,\cdot)$ benutzt, die im Beweis als Random-Oracle modelliert wird [100]. Für praktische Zwecke wird in der Spezifikation [99] konkret die Verwendung der „*mask generation function*“ MGF1 aus PKCS #1 v2.1 [103] vorgeschlagen. Der geheime Schlüssel enthält einen zusätzlichen ℓ-bit-Initialisierungswert Δ für die MGF, der sich nicht auf den Public Key auswirkt. Der Zufallswert r für die Signaturberechnung wird nun aus der pseudozufälligen Folge $\Phi(H(M)\|\Delta\|i, |pq|)$ für $i = 1, 2, \ldots$ berechnet, wobei i die Wiederholungen zählt, die nötig sind, wenn w_1 im Signieralgorithmus die notwendige Bedingung nicht erfüllt.

Der subliminale Kanal in ESIGN kann ebenso in ESIGN-D benutzt werden. Der Wächter ist nicht in der Lage, die Ausgabe der MGF, die ja als Random-Oracle modelliert wird, von echtem Zufall zu unterscheiden, da er dazu Δ kennen müsste. Die codierten Nachrichten in dem subliminalen Kanal sind ebenfalls nicht von echtem Zufall zu unterscheiden. Daher sind auch ehrliche Benutzung von ESIGN-D sowie subliminale Benutzung von ESIGN für ihn ununterscheidbar.

3.5 Formalisierung von Subliminalfreiheit

Nachdem subliminale Kanäle formal gefasst und charakterisiert wurden, sowie Besonderheiten bei der Verwendung der subliminalen Kanäle in unterschiedlichen Szenarien beschrieben wurden, bleibt die Frage nach subliminalfreien Signaturverfahren. Eine Möglichkeit, sich vor manipulierten Signieralgorithmen mit einer SETUP-Falltür zu schützen, ist es offensichtlich, ein subliminalfreies Signaturverfahren zu verwenden. Auch für die Inhaber einer ID-Karte, beispielsweise eines Reisepasses, mag es wünschenswert sein, dass ihr Ausweis keine subliminale Information enthält. Es kann also ein Interesse bestehen, dass auch Ausweise mit subliminalfreien Signaturverfahren unterschrieben werden. Dieser Abschnitt untersucht nun Verfahren, die subliminale Kommunikation verhindern.

3.5.1 Subliminalfreie Signaturverfahren

Zunächst werden Signaturverfahren betrachtet, bei denen es grundsätzlich nicht möglich ist, Information zu verstecken. Dazu wird ein Experiment definiert, bei dem ein in zwei Algorithmen aufgeteilter Angreifer versuchen muss, ein Bit mittels einer Signatur zu übertragen. Der Angreifer, der die subliminale Kommunikation starten möchte,

bekommt die Freiheit, andere als die vorgesehenen Algorithmen Gen und Sig zur Erzeugung der Schlüssel und zum Signieren zu verwenden, solange die Ausgaben seiner Algorithmen von einer ehrlichen Benutzung des Signaturverfahrens ununterscheidbar sind. Die folgende Definition versucht diese Intuition formal zu fassen:

Definition 3.5 (Subliminalfrei)**.** *Sei* $S = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *ein Signaturverfahren. Das Verfahren* S *heißt* subliminalfrei*, wenn für alle Tupel von PPT-Algorithmen*

$$\mathcal{A} = (\mathsf{SubGen}, \mathsf{GetCoverMessage}, \mathsf{BadSign}, \mathsf{BadVerify})$$

die Ausgaben der folgenden beiden Experimente

Experiment $\mathbf{Exp}_{S,\mathcal{A}}^{\text{signer}-b}$:
$(PK, SK, K_{sub}) \leftarrow \mathsf{SubGen}(1^\ell)$;
$(M, s) \leftarrow \mathsf{GetCoverMessage}(K_{sub}, PK)$; *(für bel. Information* s*)*
$\sigma \leftarrow \mathsf{BadSign}(M, b, SK, s)$ *mit* $\mathsf{Ver}(M, \sigma, PK) = \mathsf{valid}$;
$d \leftarrow \mathsf{BadVerify}(M, \sigma, PK, s)$;
return d;

für ein Bit $b = 0, 1$ *nicht unterscheidbar sind:*

$$\left| P[\mathbf{Exp}_{S,\mathcal{A}}^{\text{signer}-1}(\ell) = 1] - P[\mathbf{Exp}_{S,\mathcal{A}}^{\text{signer}-0}(\ell) = 1] \right| \leq \text{negl}(\ell).$$

3.5.2 Eine subliminalfreie Variante von RSA-PSS

Die PSS-Codierung kann auch deterministisch verwendet werden. Ohne Verwendung des Zufallsstrings salt entsteht eine deterministische Variante des Signaturverfahrens RSA-PSS. Die PSS-Codierung wirkt dann prinzipiell als eine „*Full-Domain-Hashfunktion*“, d. h. als Hashfunktion, deren Ausgabe der Bitlänge des RSA-Parameters n entspricht. Leider ist für diesen Fall die Reduktion einer Signaturfälschung auf das RSA-Problem nicht so „eng“ wie mit einer randomisierten Codierung, jedoch bleibt das Verfahren beweisbar sicher [13]. Das alleine genügt aber überraschenderweise noch nicht, um die Subliminalfreiheit zu erreichen.

Eine subliminale Kommunikation in RSA

Neben dem subliminalen Kanal im Signaturverfahren RSA-PSS gibt es auch eine subliminale Kommunikationsmöglichkeit im einfachen RSA-Algorithmus, die jedoch keinen subliminalen Kanal nach Definition 3.1 darstellt.

Die Schlüsselgenerierung in einem RSA-basierten Verfahren erfordert, dass ein öffentlicher Schlüssel (n, e) der Bedingung $\gcd(\phi(n), e) = 1$ genügt. Wenn jedoch nur das Paar (n, e) gegeben ist, ist nicht effizient überprüfbar, ob die Bedingung bei der Schlüsselgenerierung eingehalten wurde, also $\phi(n)$ und e tatsächlich koprim sind.

In dem Experiment $\mathbf{Exp}_{S,\mathcal{A}}^{\text{signer}-b}$ darf der Angreifer von der vorgesehenen Schlüsselgenerierung abweichen, solange der resultierende Schlüssel von einem ehrlich erzeugten

ununterscheidbar ist. Eine abweichende Schlüsselerzeugung kann beispielsweise einen öffentlichen Schlüssel (n, e) generieren, so dass $\gcd(\phi(n), e) = 3$ ist. Dann gilt für Elemente in $\mathbb{Z}_n^\times$, dass sie entweder drei oder keine $e$te Wurzeln haben. In diesem Fall existiert kein Exponent d mit $de \equiv 1 \pmod{\phi(n)}$, aber es ist effizient möglich, alle $e$ten Wurzeln zu berechnen – sofern sie existieren.

Da nur 1/3 der Elemente in $\mathbb{Z}_n^\times$ eine $e$te Wurzel hat, ist es mit einer deterministischen Codierung der Nachricht in ein Element prinzipiell nicht möglich, alle Nachrichten zu signieren. Mit einem randomisierten Padding-Verfahren gibt es jedoch eine hohe Chance, die Nachricht zu einem Element in $\mathbb{Z}_n^\times$ zu codieren, das eine $e$te Wurzel hat. Der Unterzeichner kann dann eine der drei möglichen Wurzeln (die kleinste, mittlere, größte) als Signatur auswählen. Ein autorisierter Empfänger der subliminalen Nachricht führt in Kenntnis[12] der Faktorisierung von n ebenfalls den Algorithmus aus, um die $e$ten Wurzeln zu finden, und kann dann nachvollziehen, welche Entscheidung der Unterzeichner getroffen hat. Es ist bei dieser subliminalen Kommunikationsmöglichkeit dem autorisierten Empfänger also möglich, selber Signaturen zu fälschen.

Subliminalfreiheit von RSA-PSS

Der Breitbandkanal in der PSS-Codierung ist in der deterministischen Variante verhindert, jedoch noch nicht die subliminale Kommunikation aus dem vorigen Absatz. Um solche eigentlich ungültigen Schlüssel zu verhindern, wird nun eine Methode GenVer zur Überprüfung der Schlüssel vorgeschlagen, die der Erzeuger der Schlüssel während der Schlüsselgenerierung vorbereiten muss. Der Algorithmus zur Schlüsselgenerierung Gen wird so modifiziert, dass sogenannte „*safe primes*“ p und q verwendet werden, also Primzahlen p und q, bei denen $(p-1)/2$ und $(q-1)/2$ wiederum Primzahlen sind. Der Modulus $n = pq$ ist dann das Produkt solcher „*safe primes*“ p und q, wodurch $\phi(n)$ nur die Primfaktoren 2, $(p-1)/2$ und $(q-1)/2$ besitzt. Es ist bekannt, dass ein Zero-Knowledge-Beweis, dass n tatsächlich das Produkt zweier „*safe primes*“ ist, effizient möglich ist [41]. Ein Wächter muss die Korrektheit des Beweises überprüfen und zusätzlich verifizieren, dass e keinen der Primfaktoren von $\phi(n)$ enthält. Das ist einfach möglich, wenn $e \neq 2$ als Primzahl gewählt wird. Dann ist offensichtlich, dass e nicht 2 ist, und durch Testen, ob $(2e+1) \nmid n$, kann ausgeschlossen werden, dass e entweder $(p-1)/2$ oder $(q-1)/2$ entspricht. Weil e prim ist, kann e auch keinen dieser Werte als Faktor enthalten, womit e koprim zu $\phi(n)$ sein muss. Dadurch gibt es für ein beliebiges $m \in \mathbb{Z}/n\mathbb{Z}$ nur eine Lösung $\sigma \in \mathbb{Z}/n\mathbb{Z}$ für die Gleichung $\sigma^e \equiv m \pmod{n}$.

Noch einmal zusammengefasst:

Schlüsselerzeugung Gen_{sf}: Der Algorithmus wählt zwei „*safe primes*“ p und q, Primzahlen der Länge $\ell/2$ bit mit $p = 2p' + 1$ und $q = 2q' + 1$ für geeignete Primzahlen p' und q'. Dann wird $n = pq$ berechnet und ein nicht-interaktiver Beweis t erzeugt, dass n das Produkt zweier „*safe primes*“ ist (vgl. [41]).

[12] Die Faktorisierung von n muss also Teil des Subliminalschlüssels sein.

Der öffentliche Exponent e wird als ungerade Primzahl $e \neq p', q'$ gewählt. Anschließend wird $d = e^{-1} \bmod (p-1)(q-1)$ berechnet. Die Schlüssel sind

$$PK = (n, e, t) \quad \text{und} \quad SK = (n, d).$$

Schlüsselverifikation GenVer**:** Bevor eine Signatur σ zu einem öffentlichen Schlüssel $PK = (n, e, t)$ akzeptiert wird, muss überprüft werden, ob

- n das Produkt zweier „*safe primes*“ ist. Das geschieht anhand des Zero-Knowledge-Beweises t.
- e eine ungerade Primzahl ist und $(2e+1) \nmid n$ gilt.

Signaturverifikation Ver_{sf}**:** Zunächst wird GenVer ausgeführt und anschließend die Signatur wie in Algorithmus 3 verifiziert.

Proposition 3.6. *RSA-PSS mit der Schlüsselgenerierung* Gen_{sf}*, sowie einem Verifikationsalgorithmus* Ver_{sf}*, der zunächst* GenVer *ausführt ist subliminalfrei.*

Beweis. Um die Proposition zu beweisen, ist zu zeigen, dass für eine Nachricht M und einen verifizierten RSA-Schlüssel $PK = (n, e, t)$ nur eine Signatur σ gültig ist.

Gegeben eine Signatur σ, so liefert die einfache RSA-Verifizierung eine PSS-Codierung $EM' = \sigma^e \pmod n$ der Nachricht M. Aufgrund der Schlüsselverifikation ist e teilerfremd zu $\phi(n)$, also gibt es nur eine mögliche Signatur $\sigma \in \{0, \ldots, n-1\}$, welche $\sigma^e \equiv EM' \pmod n$ erfüllt[13].

Es muss noch gezeigt werden, dass die Codierung EM' die eindeutige Codierung EM der Nachricht M ist. Dazu bezeichne *emLen* die Länge von EM' und *hLen* die Ausgabelänge der Hashfunktion jeweils in Byte. Zunächst wird geprüft, ob *emLen* innerhalb der zulässigen Grenzen liegt: $\lceil (\text{Bitlänge von } n)/8 \rceil$ und das letzte Byte von EM' dem hexadezimalen Wert *bc* entspricht. Bezeichne *maskedDB* die $emLen - hLen - 1$ vorderen Byte und H_0 die folgenden *hLen* Byte. Die Verifikation der PSS-Codierung ergibt genau dann valid, wenn $H_0 = H(M')$, wobei M' wegen des Weglassens des Zufallswertes *salt* einfach aus M ausgerechnet werden kann. Folglich gibt es nur ein mögliches H_0. Es wird überprüft, dass $maskedDB \oplus \mathrm{MGF}(H_0, emLen - hLen - 1)$ der String $0 \ldots 0 \| 1$ ist. Dann ist $maskedDB = (0 \ldots 0 \| 1) \oplus \mathrm{MGF}(H_0, emLen - hLen - 1)$, also ist auch $EM' = EM$ eindeutig bestimmt. □

3.5.3 Verallgemeinerung: Subliminalfrei mit Wächter

Es ist kein DSA-ähnliches Signaturverfahren bekannt, das gemäß Definition 3.5 subliminalfrei wäre. Jedoch muss die Forderung an die Subliminalfreiheit der Signatur in vielen Fällen nicht so strikt sein: Es muss im Allgemeinen nicht *jeder* davon überzeugt

[13] Es wird ohnehin schon verifiziert, ob σ in dem Bereich $\{0, \ldots, n-1\}$ liegt, also dass σ eindeutig modulo n ist.

sein, dass eine Signatur keine subliminale Nachricht beherbergt. Häufig ist ein einzelner Wächter vorhanden, der an der Subliminalfreiheit einer spezifischen Signatur interessiert ist. In diesem Fall ist es ausreichend, den Wächter davon zu überzeugen, dass der Algorithmus Sig ehrlich benutzt wurde. Der Wächter kann dann einen Beweis vom Unterzeichner erhalten, der die ehrliche Berechnung der Signatur beweist. Jedoch wird in dem Beweis selbst im Allgemeinen eine subliminale Kommunikationsmöglichkeit bestehen, daher ist dieses Verfahren nur möglich, wenn der Wächter vertrauenswürdig ist, und subliminale Nachrichten an den Wächter nicht verhindert werden müssen. Der Wächter muss zudem in einer Position sein, die Kommunikation stoppen zu können oder den Unterzeichner anklagen zu können, falls er sich unehrlich verhält und der Beweis ungültig ist. Bei den beiden in diesem Kapitel betrachteten Beispielszenarien ist das der Fall:

- Bei der Falltür SETUP verwendet ein Benutzer ein böswillig modifiziertes Programm, um eine Signatur zu berechnen. Durch einen subliminalen Kanal im Signaturverfahren kann dadurch sein geheimer Signaturschlüssel dem Designer des Programms zugespielt werden. In diesem Szenario ist der Benutzer des Programms und Inhaber des geheimen Schlüssels ein solcher Wächter, der den Beweis der Subliminalfreiheit überprüfen kann, bevor er die signierte Nachricht veröffentlicht.

- Im Falle des elektronischen Passes ist der Inhaber des Passes der Wächter, der überprüfen will, ob die Signatur seines Passes subliminalfrei ist. Falls der Beweis fehlerhaft ist, kann der Inhaber des Passes durch Veröffentlichen des Beweises das Vertrauen in die ausstellende Behörde schwächen, so dass diese darauf achten muss subliminalfreie Signaturen zu erstellen und korrekte Beweise auszuliefern.

Die nachfolgende Definition formalisiert diesen verallgemeinerten intuitiven Begriff der Subliminalfreiheit mit Wächter.

Definition 3.7 (Subliminalfrei mit Wächter)**.** *Ein Signaturverfahren, das* subliminalfrei mit Wächter *ist, ist ein Quadrupel* $S = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Chk})$ *von Algorithmen mit den folgenden Eigenschaften:*

- Gen*, der Schlüsselgenerierungsalgorithmus, ist ein PPT-Algorithmus. Die Eingabe ist der Sicherheitsparameter* ℓ *und die Ausgabe ist ein Schlüsselpaar* (PK, SK) *und ein Wert ci. Dabei ist* PK *der öffentliche Verifikationsschlüssel,* SK *der geheime Signaturschlüssel und ci ist eine Prüfinformation, die der Wächter für die Überprüfung der Signaturberechnung benötigt.*

- Sig *ist ein PPT-Algorithmus, der eine Nachricht* M *und einen geheimen Schlüssel* SK *als Eingabe erwartet und eine Signatur* σ *für* M *sowie einen Beweis* t *ausgibt.*

- Ver*, der Algorithmus für die Signaturverifikation ist vom selben Typ wie in Definition 2.1.*

- Chk *ist ein deterministischer, polynomiell beschränkter Algorithmus, der eine Nachricht M, eine Signatur σ, einen öffentlichen Schlüssel PK, die Prüfinformation ci und einen Beweis t als Eingaben erwartet und* true *oder* false *ausgibt.*

  Die Ausgabe true *setzt voraus, dass* $\mathsf{Ver}(M, \sigma, PK) = \mathsf{valid}$ *und bedeutet, dass die Signatur σ keine subliminale Nachricht verbirgt.*

Neben der Korrektheit, dass eine mit Sig *signierte Nachricht auf jeden Fall gültig ist, gilt die folgende Eindeutigkeit für eine Signatur σ bezüglich der Prüfung durch* Chk*: Für jeden PPT-Algorithmus $\mathcal{A}$, der den Sicherheitsparameter ℓ als Eingabe bekommt, ist die Wahrscheinlichkeit vernachlässigbar, dass $\mathcal{A}$ Werte $PK, SK, ci, M, \sigma_1, \sigma_2, t_1, t_2$ ausgibt, so dass die Verteilung (PK, SK) ununterscheidbar von der Ausgabeverteilung von* Gen *ist,*

$$\sigma_1 \neq \sigma_2 \quad \textit{und} \quad \mathsf{Chk}(M, \sigma_1, PK, ci, t_1) = \mathsf{Chk}(M, \sigma_2, PK, ci, t_2) = \mathsf{true}.$$

Ein Signaturverfahren mit eindeutigen Signaturen wie die besprochene Variante von RSA-PSS erfüllt natürlich ebenfalls die Definition von Subliminalfreiheit mit Wächter. Umgekehrt erlaubt Subliminalfreiheit mit Wächter aber, dass mehrere gültige Signaturen σ für eine Nachricht M und einen öffentlichen Schlüssel PK existieren, solange der Algorithmus Chk für jede Nachricht M nur eine davon akzeptiert. Der Wächter (z. B. der Inhaber des Passes) wird eine Signatur σ nur dann akzeptieren, wenn er einen gültigen Beweis t bekommt. Dann kann er sicher sein, dass kein Bit subliminale Information in σ vorhanden ist. In diesem Sinn erfüllt ein Verfahren, das subliminalfrei mit Wächter ist, die Anforderung von Subliminalfreiheit.

Die Prüfinformation ci und der Beweis t werden nur vom Wächter benötigt. Es ist wichtig, dass der Wächter nicht im Verdacht steht, Empfänger subliminaler Nachrichten zu sein. Im Allgemeinen ist der Wächter jedoch nicht vertrauenswürdig bezüglich der Sicherheit des Signaturverfahrens. Für einen elektronischen Reisepass, der subliminalfrei signiert werden soll, liegt der geheime Signaturschlüssel bei der ausstellenden Behörde. Die Fälschung einer Signatur würde es erleichtern, Ausweise zu fälschen; die Sicherheit der Signatur muss daher sehr gut geschützt werden. Die Wächter, die an der Subliminalfreiheit interessiert sind, sind letztendlich alle Bürger, die einen Reisepass erhalten. Es ist in diesem Szenario offensichtlich, dass keinerlei Vertrauen in den Wächter bezüglich der Sicherheit des Signaturverfahrens benötigt werden sollte.

Um den Unterzeichner vor dem Wächter zu schützen, wird angenommen, dass die Informationen ci und t dem EUF-CMA-Angreifer bekannt sind und er trotzdem nur mit vernachlässigbarer Wahrscheinlichkeit einen „*existential forgery*“ produzieren darf. Für spezielle Anwendungen kann angenommen werden, dass ci und t geheim bleiben, also nicht für einen Angriff auf die Eigenschaft EUF-CMA nutzbar sind: Beispielsweise ist der Wächter im SETUP-Szenario der Inhaber des Signierschlüssels, der das nicht vertrauenswürdige Programm zur Signaturberechnung überwacht. In diesem Fall kann angenommen werden, dass der EUF-CMA-Angreifer keinen Zugriff auf ci und t hat, ci und t müssen dann aber mit derselben Sorgfalt wie der geheime Schlüssel SK aufbewahrt werden.

3.5.4 Eine subliminalfreie Variante von EC-DSA

Es soll nun eine Variante von EC-DSA entwickelt werden, die subliminalfrei mit Wächter ist. Das Verfahren wird auf einen Einsatz ausgerichtet, der die Geheimhaltung von ci und t erfordert – beispielsweise das Reisepassszenario, in welchem dem Wächter der Signaturschlüssel SK des Unterzeichners nicht bekannt ist. Es geht in diesem Abschnitt weniger um die Möglichkeit einer solchen Variante von EC-DSA, als vielmehr darum, aufzuzeigen, dass eine solche Variante durchaus praktikabel ist.

Für eine Beschreibung des Signaturverfahrens EC-DSA sei noch einmal auf Algorithmus 5 in Abschnitt 2.3.3 verwiesen. Von EC-DSA vorgegeben sind als Domainparameter ein Körper $\mathbb{F}_q$, eine elliptische Kurve $E(\mathbb{F}_q)$ über $\mathbb{F}_q$ und ein Punkt $G \in E(\mathbb{F}_q)$ der Ordnung n_{ec} für eine Primzahl n_{ec}. Außerdem werden eine kollisionsfreie Hashfunktion H und eine Projektion π eines Punktes der elliptischen Kurve auf seine x-Koordinate benötigt. Ein geheimer Schlüssel ist ein Element $d \in \{1, \ldots, n_{ec} - 1\}$ und der zugehörige öffentliche Schlüssel ist der Punkt $P := d \cdot G$. Eine Signatur einer Nachricht $M \in \{0,1\}^*$ ist ein Paar (r, s) mit $r = \pi(k \cdot G) \bmod n_{ec}$ für ein gleichverteilt zufällig gewähltes $k \in \{1, \ldots, n_{ec} - 1\}$ und $s = k^{-1}(dr + H(M)) \bmod n_{ec}$.

Eine deterministische EC-DSA-Variante

Als erster Schritt zu einer Variante von EC-DSA, die subliminalfrei mit Wächter ist, wird zunächst eine deterministische Variante vorgestellt. Eine deterministische Variante entsteht, indem die Zufallseingabe k im Signieralgorithmus durch deterministisch gewonnenen Pseudozufall ersetzt wird (siehe auch [95]). Der Initialisierungswert der Pseudozufallsfunktion ist dabei Teil des geheimen Schlüssels und das Argument ist die zu signierende Nachricht. Jedoch genügt die Determinisierung des Verfahrens noch nicht, um subliminalfrei zu werden, weil nicht überprüfbar ist, ob tatsächlich der vorgesehene Pseudozufallswert benutzt wurde. Zahlreiche Beispiele dazu wurden in Abschnitt 3.4 vorgestellt.

Die EC-DSA-Variante muss es also ermöglichen, dem Wächter zu beweisen, dass der Unterzeichner den deterministischen Algorithmus korrekt ausgeführt hat. In einem Szenario wie bei dem beschriebenen elektronischen Reisepass muss gewährleistet sein, dass der Wert k geheim bleibt – eine Veröffentlichung von k würde den geheimen Signaturschlüssel des Unterzeichners offenlegen. Dadurch ist es nicht möglich, eine *verifizierbare* Pseudozufallsfunktion aus Micali u. a. [92] zu verwenden. Eine solche Funktion erlaubt es, die ehrliche Erzeugung des Pseudozufallswerts zu überprüfen, allerdings nur, sofern der Pseudozufallswert bekannt ist. Die hier vorgestellte Konstruktion basiert daher auf einer Pseudozufallsfunktion sowie einem separaten nichtinteraktiven Zero-Knowledge-Beweis. Speziell wird die effiziente Pseudozufallsfunktion von Naor und Reingold [96], die auf der DDH-Annahme[14] basiert, verwendet, weil es in diesem Umfeld eine Vielzahl effizienter nicht-interaktiver Zero-Knowledge-Beweise gibt.

[14] siehe Definition A.4

Die Pseudozufallsfunktion

Die Pseudozufallsfunktion von NAOR UND REINGOLD [96] ist durch zwei Primzahlen p_1, q_1, mit $q_1 \mid p_1 - 1$, ein Element $g_1 \in \mathbb{Z}_{p_1}^{\times}$ von Ordnung q_1 und $n+1$ Zufallszahlen $\vec{a} = (a_0, \ldots, a_n) \in \mathbb{Z}_{q_1}^{n+1}$ parametrisiert. Die Ausgabe der Funktion für eine n-bit-Eingabe $x = (x_1 \ldots x_n) \in \{0,1\}^n$ ist

$$f_{p_1,q_1,g_1,\vec{a}}(x) = g_1^{a_0 \prod_{x_i=1} a_i}.$$

Dieser Wert ist ein pseudozufälliges Element in der von g_1 generierten zyklischen Gruppe der Ordnung q_1.

Der n-bit-Wert x wird aus der zu signierenden Nachricht M berechnet. Dazu wird eine kollisionsresistente Hashfunktion $H(M)$ mit einer Ausgabe der Länge n bit auf die Nachricht M angewendet, bevor das pseudozufällige Gruppenelement $f_{p_1,q_1,g_1,\vec{a}}$ berechnet wird. Aus diesem Element muss eine Zahl $k \in \mathbb{Z}_{n_{ec}}$ berechnet werden, die im EC-DSA-Signieralgorithmus verwendet werden kann. Die Verteilung der generierten Elemente k soll möglichst nahe an der Gleichverteilung liegen. Das kann beispielsweise durch Anwendung einer universellen Hashfunktion auf das Gruppenelement $f_{p_1,q_1,g_1,\vec{a}}$ geschehen. Für die subliminalfreie EC-DSA-Variante wird die universelle Familie von Hashfunktionen[15] $\mathcal{H} = \{H_{a,b} | a, b \in \mathbb{Z}_{p_1}, a \neq 0\}$ aus [76] verwendet, wobei:

$$\begin{array}{rccc} H_{a,b}: & \mathbb{Z}_{p_1} & \longrightarrow & \mathbb{Z}_{n_{ec}} \\ & y & \longmapsto & a \cdot y + b \bmod p_1 \bmod n_{ec}. \end{array}$$

Die folgende Variante des Leftover-Hashlemmas [75, 76, 119] garantiert eine Verteilung, die sehr nahe an der Gleichverteilung in $\mathbb{Z}_{n_{ec}}$ liegt.

Lemma 3.8 (Leftover-Hashlemma)**.** *Mit den oben eingeführten Primzahlen n, p_1, q_1 sei X eine Teilmenge von $\mathbb{Z}_{p_1}$ mit Kardinalität $|X| \geq q_1$. Sei $e > 0$ und $\mathcal{H}$ eine fast universelle Familie von Hashfunktionen $H : \mathbb{Z}_{p_1} \to \mathbb{Z}_{n_{ec}}$ mit $\log n_{ec} = \log q_1 - 2e$. Dann weicht die Verteilung von $(H, H(x))$ um höchstens 2^{-e} von der Gleichverteilung ab, wobei $H \in \mathcal{H}$ und $x \in X$ beide unabhängig und zufällig gleichverteilt gezogen wurden.*

Der Beweis dieses Lemmas findet sich in [76]. Nur die verwendeten Mengen müssen entsprechend durch $\mathbb{Z}_{p_1}$ und $\mathbb{Z}_{n_{ec}}$ ersetzt werden.

EC-DSA-SF

Bevor die Konstruktion des Beweises t des Signaturverfahrens detailliert erläutert wird, zeigt Algorithmus 9 eine Zusammenfassung der subliminalfreien Variante EC-DSA-SF von EC-DSA.

[15] siehe Definition A.5

Algorithmus 9 Eine subliminalfreie Variante: EC-DSA-SF

Parameter:

$\mathbb{F}_q$	endlicher Körper
$E(\mathbb{F}_q)$	eine elliptische Kurve über $\mathbb{F}_q$ mit $\#E(\mathbb{F}_q) = n_{ec} \cdot h$
$G \in E(\mathbb{F}_q)$	Punkt der Ordnung n_{ec} auf der elliptischen Kurve für eine Primzahl n_{ec}
$\pi : E(\mathbb{F}_q) \to \mathbb{F}_q$	Projektion eines Punktes auf seine x-Koordinate
H	kollisionsresistente Hashfunktion

Schlüsselerzeugung Gen: Eine Zufallszahl $d \in \{1, \ldots, n-1\}$ wird als erster Teil des geheimen Schlüssels SK gewählt. Als öffentlicher Schlüssel PK wird der Punkt $P = d \cdot G$ berechnet. Der Algorithmus Gen berechnet zusätzlich für den deterministischen Signieralgorithmus und den Beweis t die folgenden Werte:

- Primzahlen p_1 und q_1 mit $q_1 | (p_1 - 1)$ und einen Erzeuger $g \in \mathbb{Z}_{p_1}^{\times}$ von Ordnung q_1 für die Pseudozufallsfunktion.
  (Für den Beweis t ist es notwendig, dass q_1 vernachlässigbar nahe bei einer Potenz von 2 liegt, um einfach beweisen zu können, dass ein festgelegter Wert in einem bestimmten Intervall liegt.)
- Gleichverteilt gezogene Zufallszahlen $\vec{a} = (a_0, \ldots, a_n) \in \mathbb{Z}_{q_1}^{n+1}$, die für die Pseudozufallsfunktion von Naor und Reingold benötigt werden. Die Werte $a_0, \ldots, a_n$ müssen vor dem EUF-CMA-Angreifer geheim gehalten werden, damit ihm der Wert k unbekannt bleibt.
- Gleichverteilt gezogene Zufallszahlen $a, b \in \mathbb{Z}_{p_1}$, die eine universelle Hashfunktion aus der Familie $\mathcal{H}$ bestimmen.
- Eine Menge PROOF von Domainparametern, die für den Beweis t benötigt werden. Die Menge wird im Abschnitt Beweiskonstruktion spezifiziert, beispielsweise wird PROOF Commitments auf die Werte $a_0, \ldots, a_n$ enthalten.

Der Wächter erhält die öffentliche Information $ci := (p_1, q_1, a, b, \mathsf{PROOF})$, also alle Parameter außer $\vec{a}$. Der Vektor $\vec{a}$ und die Prüfinformation ci bilden den zweiten Teil des geheimen Schlüssels SK. Benutzer, die nur Signaturen verifizieren müssen, brauchen keine zusätzlichen Parameter, da die Signaturverifikation Ver und der öffentliche Schlüssel PK gegenüber EC-DSA (Algorithmus 5) unverändert bleibt.

Signaturberechnung Sig: Der Signaturalgorithmus berechnet zunächst einen Wert $k \in \{1, \ldots, n_{ec} - 1\}$ als[16]

$$k = (a \cdot g^{a_0 \prod_{x_i=1} a_i} + b \bmod p_1) \bmod n_{ec} \quad (\text{mit } x = H(M)),$$

[16] Falls das Ergebnis $k = 0$ ist, kann die Nachricht nicht auf diese Weise signiert werden. Das tritt nur mit vernachlässigbarer Wahrscheinlichkeit auf und ist daher für die praktische Verwendung

und anschließend $r = \pi(k \cdot G) \bmod n_{ec}$ und $s = k^{-1}(dr + H(M)) \bmod n_{ec}$. Die Signatur einer Nachricht $M \in \{0,1\}^*$ ist dann das Paar (r, s).

Zusätzlich wird ein Beweis t für die korrekte Berechnung des pseudozufälligen k konstruiert. Der Beweis t muss dem Wächter zur Verfügung stehen, ist aber nicht Teil der Signatur.

Signaturverifikation Ver**:** Der Algorithmus Ver ist gegenüber EC-DSA unverändert.

Beweisverifikation Chk**:** Der Wächter überprüft anhand des Beweises t, ob für die Signatur $\sigma = (r, s)$ der Nachricht M tatsächlich

$$k = (a \cdot g^{a_0 \prod_{x_i=1} a_i} + b \bmod p_1) \bmod n_{ec} \quad (\text{mit } x = H(M))$$

verwendet wurde.

Bausteine für den Beweis t der ehrlichen Signaturberechnung

Es bleibt noch zu zeigen, wie der Beweis t im Einzelnen implementiert werden kann, bevor die Sicherheit des Verfahrens abschließend analysiert wird. Zunächst werden die Bestandteile des Beweises t beschrieben.

Commitment-Verfahren. Die Implementierung des Beweises benötigt ein Commitment-Verfahren. Ein Commitment-Verfahren bietet die beiden Algorithmen Commit und Verify. Gegeben ein String M, kann durch Commit(M) ein sogenanntes Commitment C und Aufdeckinformation D berechnet werden. Es wird davon gesprochen, dass M durch das Commitment C *festgelegt* ist. Durch Verify(C, D) erhält man die Nachricht M zurück. Ein sicheres Commitment-Verfahren muss *festlegend* sein, d. h. es ist in polynomieller Zeit unmöglich, ein D' auszurechnen, so dass Verify$(C, D') \neq M$ ist, und *verbergend*, d. h. es ist in polynomieller Zeit unmöglich, aus C die Nachricht M auszurechnen.

Um einen nicht-interaktiven Beweis t zu implementieren werden hier Pedersen-Commitments [101, 41] eingesetzt. Die Commitments benötigen als Parameter Primzahlen p und q mit $q|p-1$ sowie Elemente $g, h \in \mathbb{Z}_p^\times$, welche beide dieselbe Untergruppe der Größe q erzeugen. Für die Berechnung von g und h muss ein geeignetes Verfahren gewählt werden, so dass $\log_g h$ für jeden unbekannt ist. Das Commitment auf ein Element $u \in \mathbb{Z}_q$ ist dann

$$BC_{g,h}(u) = g^u h^v$$

---

des Verfahrens nicht relevant. Um alle Nachrichten subliminalfrei signieren zu können, kann in diesem Fall bewiesen werden, dass $k = 0$ ist und beispielsweise mit der Eingabe $H(M\|1)$ ein neuer Pseudozufallswert k berechnet werden.

für ein zufälliges $v \in \mathbb{Z}_q$. Wie in [101] gezeigt wird, ist dieses Commitment-Verfahren informationstheoretisch verbergend und ein Brechen der Festlegungseigenschaft ist so hart wie die Berechnung des diskreten Logarithmus von h zur Basis g.

Grundlegende Zero-Knowledge-Beweise. Der Beweis t wird aus Zero-Knowledge-Beweisen für modulare Arithmetik wie in [62, 41] und für die Lage eines festgelegten Wertes in einem Intervall wie in [27] zusammengesetzt. Die Notation für die Zero-Knowledge-Beweise stammt aus CAMENISCH UND STADLER [43] und CAMENISCH UND MICHELS [41]. Ein Beispiel für die Notation ist $PK\{(x) : g^x\}$. Dabei stehen zunächst die nur dem Beweiser bekannten Variablen (hier: x) über die etwas bewiesen wird, in Klammern, gefolgt von der bewiesenen Aussage über die Variablen. Im Beispiel hat ein bestimmtes, dem Verifizierer bekanntes Element g^x, den Logarithmus x bezüglich der Basis g.

Die nicht-interaktiven Zero-Knowledge-Beweise entstehen, indem von einem „*zero-knowledge argument*“ für einen ehrlichen Verifizierer ausgegangen wird und mittels Fiat-Shamir-Heuristik [58] die Zufallswahl des Verifizierers durch die Ausgabe einer als Random-Oracle modellierten Hashfunktion ersetzt wird. Die Beweise sind Zero-Knowledge im Random-Oracle-Modell, mit einer konkreten Hashfunktion bleiben sie *wittness hiding*, verbergen also weiterhin den Zeugen für die Aussage, sind allerdings übertragbar. Das ist in dieser Anwendung aber auch durchaus erwünscht (siehe „*signature of knowledge*“ [42]). Die wesentlichen Protokollbausteine aus [62, 41, 27] werden in folgender Weise notiert:

- Beweis der Kenntnis eines diskreten Logarithmus: $PK\{(x) : g^x\}$
- Beweis der Kenntnis des festgelegten Wertes: $PK\{(x, \alpha) : g^x h^\alpha\}$
- Beweis der Gleichheit von festgelegten Werten: $PK\{(x, \alpha, \beta) : g^x h^\alpha \wedge g^x h^\beta\}$
- Beweis einer Addition von Commitments:
  $PK\{(x, y, z, \alpha, \beta, \gamma) : g^x h^\alpha \wedge g^y h^\beta \wedge g^z h^\gamma \wedge z = x + y\}$
- Beweis einer Multiplikation von Commitments:
  $PK\{(x, y, z, \alpha, \beta, \gamma) : g^x h^\alpha \wedge g^y h^\beta \wedge g^z h^\gamma \wedge z = x \cdot y\}$
- Beweis einer Exponentiation von Commitments:
  $PK\{(x, y, z, \alpha, \beta, \gamma) : g^x h^\alpha \wedge g^y h^\beta \wedge g^z h^\gamma \wedge z = x^y\}$
- Beweis der Lage eines festgelegten Wertes in einem Intervall $[r_{min}, r_{max}]$:
  $PK\{(x, \alpha) : g^x h^\alpha \wedge r_{min} \leq x \leq r_{max}\}$

Beweis der Äquivalenz festgelegter Werte. Im Verlauf der Konstruktion des Beweises t wird die Gruppe, die für die Commitments verwendet wird, gewechselt, so dass sich die Gruppe, auf deren Elemente ein Commitment berechnet werden kann, entsprechend ändert. Hier wird ein Verfahren beschrieben, das für ein Commitment c_1 in einer

Gruppe von Ordnung q_1 ein Commitment c_2 in einer Gruppe von Ordnung $q_2 \neq q_1$ konstruiert und beweist, dass die darin festgelegten Werte gleich sind[17].

Dazu wird das Commitment zunächst in eine Gruppe der Ordnung q_0, die Untergruppen der Ordnung q_1 und q_2 besitzt, eingebettet. Für die Gruppe mit Ordnung q_0 gibt es zwei Erzeuger g_0 und h_0 – auch hier mit unbekanntem $\log_{g_0} h_0$. Dann sind die Erzeuger für die Untergruppen $g_1 = g_0^{q_2}$, $h_1 := {h_0}^{q_2}$ und analog $g_2 = g_0^{q_1}$, und $h_2 := {h_0}^{q_1}$.

Gegeben ein Commitment $c_1 = BC_{g_1,h_1}(x) = g_1^x h_1^{v_1}$, konstruiert der Beweiser ein Commitment $c_2 = BC_{g_2,h_2}(x) = g_2^x h_2^{v_2}$ mit gleichverteilt zufällig gewähltem $v_2 \in \mathbb{Z}_{q_2}$. Dann führt der Beweiser $\mathcal{P}$ das folgende Protokoll aus:

1. $\mathcal{P}$ beweist, dass er den Wert v_2 des Commitments c_2 kennt.
2. $\mathcal{P}$ berechnet $v_0 := \mathrm{CRT}^{-1}(v_1, v_2)$, wobei CRT^{-1} den inversen Isomorphismus des Chinesischen Restsatzes bezeichnet.
3. $\mathcal{P}$ berechnet und sendet $c_3 = BC_{g_0,h_0}(x) = g_0^x h_0^{v_0}$ an $\mathcal{V}$.
4. $\mathcal{P}$ beweist $x < \min(q_1, q_2)$ für das Commitment c_3.

Für den Beweis, dass ein festgelegter Wert genau in einem bestimmten Intervall liegt, ist bei Commitments in Gruppen von Primordnung kein effizienter Beweis bekannt. Es kann je nach Verfahren effizient bewiesen werden, dass der Wert mit einer gewissen Wahrscheinlichkeit in dem Intervall liegt oder dass der Wert in dem Intervall oder höchstens in einem Toleranzbereich um das Intervall liegt. In der Konstruktion des Beweises t wird explizit angegeben, welche Methode des Intervallbeweises verwendet wird und warum die Genauigkeit ausreicht.

5. $\mathcal{V}$ überprüft $BC_{g_0,h_0}(x)^{q_2} = BC_{g_1,h_1}(x)$ und $BC_{g_2,h_2}(x) = BC_{g_0,h_0}(x)^{q_1}$.

Beweiskonstruktion

Für die Commitments wird eine weitere Primzahl p_0 mit $q_1 p_1 n_{ec} | (p_0 - 1)$ benötigt. Zudem werden Erzeuger g_0 und h_0 einer zyklischen Untergruppe von $\mathbb{Z}_{p_0}^{\times}$ der Ordnung $q_1 p_1 n_{ec}$ ausgewählt. Die Erzeuger $g_1 = g_0^{p_1 n_{ec}}$, $g_2 = g_0^{q_1 n_{ec}}$ und $h_{ec} = g_0^{q_1 p_1}$ weiterer Untergruppen werden berechnet. Schließlich werden Commitments $BC(a_0), \ldots, BC(a_n)$ mit $BC(a_i) = g_1^{a_i} h_1^{v_i}$ auf die geheimen Initialisierungswerte $\vec{a} = (a_0, \ldots, a_n) \in \mathbb{Z}_{q_1}^{n+1}$ für die Pseudozufallsfunktion berechnet. Diese Werte bilden die zusätzlichen Parameter

$$\mathsf{PROOF} = (p_0, g_0, h_0, g_1, g_2, h_{ec}, BC(a_0), \ldots, BC(a_n)).$$

Die Parameter werden nur vom Unterzeichner und vom Wächter benötigt, sind aber als dem EUF-CMA-Angreifer bekannt anzusehen.

[17] Gleich heißt hier, dass für die festgelegten Werte x_1 in c_1 und x_2 in c_2 gilt, dass $x_1 \bmod q_1 = x_2 \bmod q_2$.

Der Beweis t, dass der Zufall für das Signieren der Nachricht M tatsächlich

$$k = \left(a g^{a_0 \prod_{x_i=1} a_i} + b \bmod p_1 \right) \bmod n_{ec}$$

mit $x = H(M)$ ist, wird durch die folgende Ergänzung des Signaturalgorithmus berechnet:

1. Es wird ein Commitment $c_1 = BC_{g_1,h_1}(z)$ auf das Produkt $z = a_0 \prod_{x_i=1} a_i \bmod q_1$ berechnet und ein nicht-interaktiver Beweis t angegeben:

$$PK\{(a_0, \ldots, a_n, z, v_0, \ldots, v_{n+1}) : g_1^{a_0} h_1^{v_0} \wedge \cdots \wedge g_1^{a_n} h_1^{v_n} \wedge g_1^z h_1^{v_{n+1}} \wedge z = a_0 \prod_{x_i=1} a_i \bmod q_1\}.$$

2. Es wird ein Commitment $c_2 = BC_{g_2,h_2}(z)$ berechnet und ein nicht-interaktiven Beweis angegeben, dass die Werte in den Commitments $BC_{g_1,h_1}(z)$ und $BC_{g_2,h_2}(z)$ gleich sind:

$$PK\{(z, v_{n+1}, v_{n+2}) : g_1^z h_1^{v_{n+1}} \wedge g_2^z h_2^{v_{n+2}}\}.$$

   Aufgrund der geeigneten Wahl der verwendeten Gruppen und Erzeuger – nämlich $g_1 = g_0^{p_1 n_{ec}}$ und $g_2 = g_0^{q_1 n_{ec}}$ und q_1 nahe bei einer Potenz von 2 – kann der Intervallbeweis über die Bitdarstellung durchgeführt werden. Dazu werden Commitments $BC_{g_0,h_0}(b_i)$ auf die Bits b_i von z für $0 \leq i \leq \lfloor \log_2(q_1) \rceil$ berechnet, und bewiesen, dass $b_i \in \{0, 1\}$ sowie $\sum_{i=0}^{\lfloor \log_2(q_1) \rceil} b_i 2^i = z$.

3. Es wird $g^z \bmod p_1$ und ein Commitment $c_3 = BC_{g_2,h_2}(g^z)$ auf diesen Wert berechnet. Es muss für jeden Schritt des Square-and-Multiply-Algorithmus bewiesen werden, dass der Wert korrekt berechnet wurde:

$$PK\{(z, x, v_{n+2}, v_{n+3}) : g_2^z h_2^{v_{n+2}} \wedge g_2^x h_2^{v_{n+3}} \wedge x = g^z\}.$$

   Für diesen Beweis kann die Bitdarstellung von z aus dem vorigen Schritt wiederverwendet werden. Die detaillierte Ausführung des Beweises wird in [41] beschrieben.

4. Es wird der Wert $k = (a \cdot g^z + b) \bmod p_1 \bmod n_{ec}$ und ein Commitment $c_4 = BC_{g_2,h_2}(k)$ auf k berechnet. Die Korrektheit der Berechnung von k und der Konstruktion des Commitments muss bewiesen werden:

$$PK\{(z, k, v_{n+3}, v_{n+4}) : g_2^{(g^z)} h_2^{v_{n+3}} \wedge g_2^k h_2^{v_{n+4}} \wedge k = a \cdot (g^z) + b \bmod p_1 \bmod n_{ec} \wedge 0 \leq k < 3n_{ec}\}.$$

   Dafür wird ein Intervallbeweis aus Brickell u. a. [34] benutzt. Dieser Beweis ist sehr effizient, es gibt aber einen Toleranzbereich um das Intervall: Ein Beweis

von $x < n_{ec}$ garantiert nur $x < 3n_{ec}$. Weil $3n_{ec} < p_1$, genügt diese Genauigkeit jedoch, um zu beweisen, das k korrekt mod p_1 reduziert wurde. Auch im nächsten Schritt wird dieser Beweis ausreichen, da die Reduktion mod n_{ec} durch den Gruppenwechsel implizit durchgeführt wird.

5. Schließlich wird ein Commitment $c_5 = BC_{g_{ec},h_{ec}}(k)$ auf k berechnet und die Äquivalenz mit c_4 bewiesen:

$$PK\{(k, v_{n+4}, v_{n+5}) : g_2^h h_2^{v_{n+4}} \wedge g_{ec}^k h_{ec}^{v_{n+5}}\}.$$

   Es ist kein erneuter Beweis über die Bitdarstellung nötig, da $k \bmod n_{ec}$ der gewünschte Wert ist, und die Reduktion mod n_{ec} durch Wechsel in die von g_{ec} und h_{ec} erzeugte Gruppe implizit erfolgt.

6. Als letzter Schritt muss noch bewiesen werden, dass der Wert k aus Commitment c_5 tatsächlich für die Berechnung der Signatur (r, s) verwendet wurde. Dazu wird ein weiteres Commitment $c_6 = BC_{G,H}(k) = kG + v_{n+6}H$ in der elliptischen Kurve $E(\mathbb{F}_q)$ berechnet. Als Erzeuger G wird der Erzeuger aus den EC-DSA Parametern benutzt, H ist ein weiterer Erzeuger derselben Untergruppe, so dass $\log_G H$ unbekannt ist. Für den Zero-Knowledge-Beweis

$$PK\{(k, v_{n+5}, v_{n+6}) : g_{ec}^k h_{ec}^{v_{n+5}} \wedge kG + v_{n+6}H\}$$

   wird kein Intervallbeweis benötigt, da die beiden Gruppen dieselbe Ordnung haben. Die Kenntnis der entsprechenden diskreten Logarithmen der beiden Faktoren kG und $v_{n+6}H$ von c_6 überzeugt den Verifizierer, dass der diskrete Logarithmus von kG tatsächlich der Wert k aus Commitment c_5 ist. Gegeben kG ist es schließlich einfach zu verifizieren, dass $r = \pi(kG) \bmod p$ $s \in \mathbb{Z}_{n_{ec}}$ und $kG = (s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P$.

Sicherheit von EC-DSA-SF

Die folgenden Propositionen fassen die EUF-CMA-Sicherheit und Subliminalfreiheit des Verfahrens zusammen.

Proposition 3.9. *EC-DSA-SF ist subliminalfrei mit Wächter.*

Beweis. Das folgt nun unmittelbar, da der nicht-interaktive Zero-Knowledge-Beweis als „*zero-knowledge-argument*“ mit überwältigender Wahrscheinlichkeit zeigt, dass für die Signatur tatsächlich der deterministisch berechnete Pseudozufallswert k verwendet wurde. Dem Wächter ist kG bekannt. Es gibt offensichtlich nur ein r, das $r = \pi(kG)$ erfüllt und ebenso kann es nur ein $s \in \mathbb{Z}_{n_{ec}}$ geben, das $kG = (s^{-1}H(M) \bmod n) \cdot G + (s^{-1}r \bmod n) \cdot P$ erfüllt. Damit ist klar, dass (r, s) die eindeutige Signatur nach dem deterministischen Signaturalgorithmus ist. □

Proposition 3.10. *Wenn das Signaturverfahren EC-DSA die Sicherheitsanforderung EUF-CMA erfüllt, dann ist auch die deterministische Variante EC-DSA-SF gemäß EUF-CMA sicher, selbst wenn dem Angreifer die Prüfinformation ci und der Beweis t bekannt sind.*

Beweisskizze. Eine Übersicht über Beweise für DSA und EC-DSA findet sich in [125]. Der Beweis konzentriert sich daher auf die folgenden drei Punkte, welche die Unterschiede von EC-DSA-SF zu EC-DSA bilden:

(i) die pseudozufällige Berechnung von k,

(ii) die zusätzliche öffentliche Information ci, die dem Angreifer zur Verfügung steht,

(iii) der zusätzliche Beweis t, der jeder Signatur beiliegt, und dem Angreifer zur Verfügung steht.

Weil nur die Berechnung von k bei der Ausführung von Sig geändert wurde, genügt es bezüglich (i) zu überprüfen, dass diese neue Berechnungsmethode ein hinreichend zufälliges k erzeugt. Die Pseudozufallsfunktion garantiert, dass k ununterscheidbar von einem gleichverteilten Gruppenelement ist. Die Hashfunktion $H_{a,b}$ bildet ein gleichverteiltes Gruppenelement auf eine (fast) gleichverteilte Zahl aus $\mathbb{Z}_{n_{ec}}$ ab. Die Verteilung von k in EC-DSA-SF kann daher von einem PPT-Algorithmus nicht von einem zufällig gewählten k unterschieden werden.

Auch der Beweis t und die Zusatzinformation ci geben dem Angreifer keinen Vorteil, eine Signatur zu fälschen: Einerseits ist t ein Zero-Knowledge-Beweis für die Korrektheit von k unter der Random-Oracle-Annahme. Darüber hinaus liefert die Menge ci nur einen vernachlässigbaren Vorteil, solange das Commitmentverfahren sicher ist.

Also kann ein Angreifer nicht mehr als vernachlässigbar erfolgreicher sein, eine Fälschung von EC-DSA-SF zu erreichen als im EC-DSA-Verfahren. □

Länge des Zero-Knowledge-Beweises

Dieser Abschnitt soll zeigen, dass es praktikabel ist, einen solchen Beweis an den Wächter zu übertragen. Dazu soll die Länge des Beweises grob abgeschätzt werden.

Zunächst muss die Größe der verwendeten Parameter festgelegt werden. Die Parameterwahl orientiert sich an den beim elektronischen Reisepass verwendeten Größen [36], da dies das motivierende Szenario ist. Für die Signatur des Reisepasses wird eine Gruppe mit der Ordnung einer 224-bit-Primzahl n_{ec} verwendet. Damit das Verfahren EC-DSA-SF dasselbe Sicherheitsniveau wie das für den elektronischen Reisepass verwendete EC-DSA-Verfahren erreicht, geht die Abschätzung in diesem Abschnitt davon aus, dass q_1 eine 356-bit- und p_1 eine 3.072-bit-Primzahl ist. Die Commitments benötigen dann eine Primzahl p_0 mit wenigstens 3.685 bit. Die Ausgabelänge der Hashfunktion H für die Eingabe in die Pseudozufallsfunktion sollte 256 bit betragen.

Daraus ergeben sich die folgenden Bitlängen der Bestandteile des Beweises. Vereinfachend wird angenommen, dass Gruppenelemente oder Commitments eine Länge von 4 kbit haben. Der Beweis der Kenntnis von festgelegten Werten oder der Äquivalenz festgelegter Werte umfasst dann höchstens ungefähr 16 kbit. Beweise für die Addition und Multiplikation benötigen dann beide in etwa 64 kbit. Diese Größen ergeben sich anhand der Protokolle in [62, 41, 27]. Für die einzelnen Schritte des Algorithmus heißt das:

1. Ein Commitment und 256 Multiplikationen summieren sich zu 16.400 kbit.

2. In diesem Schritt werden $\log_2(n_{ec})+1$ Beweise für Commitments und $\log_2(n_{ec})+1$ für Multiplikationen und Additionen benötigt. Zusammen mit einem Äquivalenzbeweis und entsprechenden Kenntnis-Beweisen für die Commitments ergeben sich ungefähr 37.000 kbit.

3. Der Square-and-Multiply-Algorithmus benötigt $2 \cdot 3.072$ Multiplikationen. Der Beweis verlängert sich dadurch um 6.144 kbit.

4. Dieser Schritt beweist ein Commitment und zwei Operationen, also 144 kbit. Dazu kommt ein Intervallbeweis, der durch die nötigen Wiederholungen ≈1.000 kbit einnimmt.

5. Hier kommt nur ein neues Commitment mit einem Äquivalenzbeweis hinzu: 32 kbit.

6. Im letzten Schritt werden nur Elemente der elliptischen Kurve angegeben, der Platzbedarf beschränkt sich auf wenige kbit.

Aufsummiert benötigt der Beweis weniger als 60.500 kbit, d. h., weniger als 8 MB. Die Beweiskonstruktion war nicht auf eine kurze Länge optimiert und auch für die Größenabschätzungen wurden grobe Vereinfachungen vorgenommen. Die Zahl stellt also nur eine obere Schranke dar. Dennoch zeigt sich, dass das Übermitteln des Beweises über das Internet oder verbreitete Datenträger problemlos möglich ist. Der Rechenaufwand zur Berechnung des Beweises ist hoch: Der Beweis besteht überwiegend aus Gruppenelementen, die durch eine modulare Exponentiation berechnet wurden, so dass die Anzahl modularer Exponentiationen sich in der Größenordnung 20.000 bewegt.

3.6 Beziehungen zu weiteren Eigenschaften

Es gibt in der Literatur sogenannte *eindeutige* und *invariante* Signaturverfahren. Diese Eigenschaften sind unabhängig von subliminalen Kanälen entstanden, stellen sich aber als eng verwandt zu den subliminalfreien Signaturen heraus. Goldwasser und Ostrovsky [67] definieren invariante Signaturverfahren („*invariant signature*

schemes“) aufgrund ihrer Äquivalenz zu nicht-interaktiven Zero-Knowledge-Beweisen. Bei einem invarianten Signaturverfahren existiert eine deterministische in polynomieller Zeit berechenbare Funktion Inv, so dass $\mathsf{Inv}(\sigma)$ für alle gültigen Signaturen σ zu einer Nachricht M identisch ist. Darüber hinaus muss $\mathsf{Inv}(\sigma)$ von einem zufälligen Bitstring ununterscheidbar sein, solange nur der öffentliche Schlüssel PK des Signaturverfahrens und die Nachricht M gegeben sind. Der invariante Wert $\mathsf{Inv}(\sigma)$ darf also insbesondere nicht einfach nur von der Nachricht M bestimmt werden.

Eine Möglichkeit, ein solches Verfahren zu konstruieren, ist es, mit einem Verfahren, das nur eine gültige Signatur pro Nachricht erlaubt, zu beginnen und die Funktion Inv so zu wählen, dass sie aus einer Signatur Zufallsbit extrahiert, um der Pseudozufälligkeit zu genügen. LYSYANSKAYA [90] hat diese Konstruktion betrachtet und dazu eindeutige Signaturverfahren („*unique signature schemes*“) definiert, bei denen jede Nachricht nur eine gültige Signatur besitzt.

Definition 3.11 (Eindeutiges Signaturverfahren)**.** *Ein* eindeutiges Signaturverfahren *ist ein Signaturverfahren, bei dem keine Werte PK, M, σ_1, σ_2 existieren, so dass*

$$\sigma_1 \neq \sigma_2 \quad \textit{und} \quad \mathsf{Ver}(M, \sigma_1, PK) = \mathsf{Ver}(M, \sigma_2, PK) = \mathsf{valid}.$$

Weil für ein eindeutiges Signaturverfahren nur genau eine gültige Signatur für eine gegebene Nachricht existiert, ist es nicht möglich in einer gültigen Signatur eine subliminale Nachricht zu verstecken und ein eindeutiges Signaturverfahren ist subliminalfrei nach Definition 3.5. Aufgrund eines subtilen Unterschieds sind jedoch subliminalfreie Signaturverfahren nicht unbedingt auch eindeutig: Subliminalfrei ist komplexitätstheoretisch definiert, es dürfen durchaus zwei gültige Signaturen σ_1, σ_2 zu einer Nachricht M existieren, sofern sie nicht von einem PPT-Algorithmus gefunden werden können.

Die Definition von subliminalfrei mit Wächter liegt zwischen den eindeutigen und invarianten Signaturen. Ein Signaturverfahren, das subliminalfrei mit Wächter ist, ist ein invariantes Signaturverfahren bezüglich der Verifikation durch den Algorithmus Chk. Das Paar aus Signatur und Beweis (σ, t) zu einer Nachricht M ist nicht eindeutig, aber alle Paare, welche Chk als gültig (true) einstuft, enthalten mit überwältigender Wahrscheinlichkeit dasselbe σ (vgl. Definition 3.7), das daher die Invariante liefert. Die Extraktionsfunktion $\mathsf{Inv}(\sigma, t)$ wendet auf den eindeutigen Teil σ von (σ, t) einen Zufallsextraktor wie in [92] an. Eine zusätzliche Anforderung, die Signaturen, die subliminalfrei mit Wächter sind, gegenüber invarianten Signaturen einschränkt, ist, dass der eindeutige Teil σ schon eine Signatur der Nachricht darstellt, die durch den Algorithmus Ver verifiziert werden kann.

3.7 Zusammenfassung und offene Fragen

In diesem Kapitel wurden subliminale Kanäle in digitalen Signaturverfahren untersucht. Erstmals wurden Signaturverfahren mit subliminalem Kanal formal definiert.

Es wurde gezeigt, dass auch in deterministischen Signaturverfahren ein subliminaler Breitbandkanal vorhanden sein kann. In den Verfahren ESIGN-D und SFLASH^{v3} wurden neue subliminale Kanäle entdeckt.

Auch die Frage nach subliminalfreien Signaturverfahren konnte beantwortet werden. Es wurde eine formale Definition für die Subliminalfreiheit gefunden, so dass für Signaturverfahren bewiesen werden kann, dass subliminale Kommunikation durch die Signaturen unmöglich ist. Für das praktisch eingesetzte Signaturverfahren RSA-PSS konnte gezeigt werden, dass eine deterministische Variante, die beweisbar sicher im Sinne von EUF-CMA ist, mit einer leicht modifizierten Schlüsselerzeugung und Verifikation auch subliminalfrei ist. In vielen Fällen muss ein Signaturverfahren jedoch nicht in diesem strengen Sinne subliminalfrei sein, es genügt, wenn ein Wächter davon überzeugt ist, dass keine subliminale Kommunikation stattfindet. Für diesen Fall wurde die Verallgemeinerung *subliminalfrei mit Wächter* eingeführt und formal gefasst. Eine Variante des Signaturverfahrens EC-DSA ist EUF-CMA und subliminalfrei mit Wächter. Bei ihrem Einsatz werden zusätzlich zahlreiche nicht-interaktive Beweise nötig, die in einem Szenario mit hohen Anforderungen an die Subliminalfreiheit allerdings durchaus noch eine vernünftige Größe haben. Es ist noch eine offene Frage, ob ein anderes, effizienteres DSA-ähnliches Signaturverfahren gefunden werden kann, das in dem Szenario aus dieser Arbeit Subliminalfreiheit mit oder sogar ohne Wächter bietet. Die beiden Begriffe *subliminalfrei* und *subliminalfrei mit Wächter* konnten ebenfalls zu anderen in der Literatur betrachteten Eigenschaften von digitalen Signaturen in Beziehung gesetzt werden.

Es gibt durchaus weitere subliminale Kommunikationsmöglichkeiten im Zusammenhang mit Signaturen. Beispielsweise ist eine subliminale Kommunikation im öffentlichen Schlüssel möglich. Von einem böswilligen Programm mit einer SETUP-Falltür kann eventuell schon bei der Schlüsselerzeugung der öffentliche Schlüssel so berechnet werden, dass der Programmdesigner daraus den geheimen Schlüssel ablesen kann. Subliminale Kanäle im öffentlichen Schlüssel wurden hier nicht betrachtet, verdienen aber durchaus noch eine intensivere Untersuchung. Beispielsweise ist für RSA ein Verfahren bekannt, um subliminalfreie Schlüssel zu erzeugen [81].

4. Schlüsselaustausch für Gruppen

In diesem Kapitel werden die Auswirkungen interner Angreifer auf Gruppenschlüsselaustauschprotokolle („*group key establishment protocols*“) untersucht. Ein Gruppenschlüsselaustauschprotokoll ermöglicht es einer Gruppe von zwei oder mehr Benutzern, einen gemeinsamen Sitzungsschlüssel zu erhalten. Gegenüber dem Zwei-Parteien-Schlüsselaustausch ergeben sich qualitativ neue Herausforderungen, wenn interne Angreifer betrachtet werden. Ein interner Angreifer in einem Zwei-Parteien-Protokoll führt dazu, dass nur ein ehrlicher Teilnehmer am Protokoll verbleibt; im Gegensatz dazu können bei einem Mehrparteien-Protokoll trotz interner Angreifer noch mehrere ehrliche Teilnehmer übrig bleiben. Dadurch gibt es zusätzliche Eigenschaften zwischen den ehrlichen Teilnehmern, die im Zwei-Parteien-Protokoll keine Rolle spielen, aber die für Gruppenschlüsselaustauschprotokolle möglichst auch mit internen Angreifern erhalten bleiben sollen. Als weiterer Grund kommt hinzu, dass mit der Anzahl der Parteien auch die Wahrscheinlichkeit steigt, dass sich einer der Teilnehmer unehrlich verhält.

Dieses Kapitel gibt zunächst eine Einführung in die Protokolle und Modelle des Gruppenschlüsselaustauschs. In Abschnitt 4.2 werden bisher unberücksichtigte Angriffe mit internen Angreifern auf etablierte Protokolle vorgestellt. Anschließend werden in Abschnitt 4.3 die Angriffe diskutiert und Eigenschaften von Gruppenschlüsselaustauschprotokollen identifiziert, die solche Angriffe unmöglich machen. Die letzten beiden Abschnitte beschäftigen sich mit der Konstruktion effizienter Protokolle, welche die zuvor aufgestellten Sicherheitsanforderungen erfüllen. Die Ergebnisse dieses Kapitels sind in unseren Arbeiten Bohli, González Vasco und Steinwandt [18] und Bohli [16] veröffentlicht.

4.1 Grundlagen

Der Schlüsselaustausch ist eines der wichtigsten Probleme der Public-Key-Kryptographie und bildet in DIFFIE UND HELLMAN [55] den Anfang der Public-Key-Kryptographie. Ein Gruppenschlüsselaustauschprotokoll, das sich eignet, um in einer Gruppe von mehr als zwei Benutzern einen gemeinsamen Sitzungsschlüssel zu etablieren, wird erstmals von INGEMARSSON U. A. [77] vorgestellt. Das Protokoll benötigt wie viele frühe Protokolle jedoch eine Rundenanzahl, die linear mit der Teilnehmeranzahl wächst, und eignet sich daher nicht für große Gruppen. Das erste Protokoll, das unabhängig von der Teilnehmerzahl mit einer konstanten Rundenanzahl auskommt, wird in BURMESTER UND DESMEDT [37] vorgestellt. Für letzteres Protokoll wurde zunächst kein Sicherheitsbeweis gegeben, erst in [38] wird die Sicherheit des Protokolls gegen passive Angreifer – Angreifer, die nur Nachrichten mithören, aber keine Nachrichten senden oder verändern – bewiesen.

Sicherheitseigenschaften von Schlüsselaustauschprotokollen

Zunächst sollen die wichtigsten Eigenschaften, welche die Sicherheit von Schlüsselaustauschprotokollen beschreiben, informell eingeführt werden. Im Sicherheitsmodell, das in Abschnitt 4.1.1 beschrieben wird, werden die Eigenschaften formalisiert. Eine gute Einführung zu verschiedenen Aspekten von Schlüsselaustauschprotokollen findet sich in dem Buch von BOYD UND MATHURIA [29].

Implizite Schlüsselauthentifizierung. Das ist die Haupteigenschaft jedes Schlüsselaustauschprotokolls. Sie garantiert die Geheimhaltung des Schlüssels und besagt, dass der Schlüssel nur den beabsichtigten Teilnehmern bekannt sein darf. Ein Aspekt der impliziten Schlüsselauthentifizierung, der häufig getrennt genannt wird, ist die *Frische* des Schlüssels, d. h. die Gewissheit, dass der Schlüssel während der Protokollausführung entstanden ist. Allerdings ist ein Schlüssel, der nicht frisch ist, prinzipiell als dem Angreifer bekannt anzunehmen, so dass dann auch die implizite Schlüsselauthentifizierung verletzt ist.

(Perfect) Forward-Secrecy. Diese Eigenschaft garantiert, dass nach einer Kompromittierung des langfristigen Geheimnisses, mittels dessen sich ein Benutzer authentifiziert, alle Sitzungsschlüssel aus abgeschlossenen Protokollausführungen weiterhin geheim bleiben.

Key-Confirmation. Key-Confirmation ist die Eigenschaft, dass ein Protokollteilnehmer die Zusicherung hat, dass seine Partner den Schlüssel korrekt berechnet haben. Zusammen mit der impliziten Schlüsselauthentifizierung ergibt sich die sogenannte *explizite Schlüsselauthentifizierung.*

Key-Agreement. Eine Unterklasse von Schlüsselaustauschprotokollen sind Key-Agreement-Protokolle[1], bei denen sich der Sitzungsschlüssel durch eine Funktion berechnet, zu der alle Teilnehmer eine Eingabe liefern. Es sollte möglichst gewährleistet sein, dass auch mehrere zusammenarbeitende Teilnehmer den Schlüssel nicht vorherbestimmen können, solange wenigstens ein Teilnehmer sich ehrlich verhält.

Beweisbare Sicherheit

Ein formales Sicherheitsmodell für Schlüsselaustauschprotokolle wird von Bellare und Rogaway [10] eingeführt. Ihr Modell definiert ein Experiment, in dem ein als PPT-Algorithmus modellierter Angreifer versuchen muss, einen tatsächlich durch einen Protokolldurchlauf etablierten Schlüssel von einem zufällig aus dem Schlüsselraum[2] $\mathcal{K}$ gezogenen Element zu unterscheiden. Gelingt ihm das nicht signifikant häufiger als durch Raten, wird das Protokoll als sicher bezeichnet. Das ursprüngliche Modell eignet sich nur für Zwei-Parteien-Schlüsselaustauschprotokolle. Es wird aber in darauf aufbauenden Arbeiten für andere Arten des Schlüsselaustauschs erweitert. Beispiele sind das „Drei-Parteien-Modell“ von Bellare und Rogaway [12], bei dem sich die beiden Benutzer symmetrische Schlüssel mit einem Server teilen, das Modell für passwortbasierten Schlüsselaustausch von Bellare u. a. [9], bei dem die Benutzer nur ein Passwort mit geringer Entropie zum Authentifizieren verwenden, und die Modelle für Gruppenschlüsselaustausch von Bresson u. a. [30, 32]. Das Modell für Gruppenschlüsselaustausch wurde in den folgenden Jahren benutzt, um zahlreiche Gruppenschlüsselaustauschprotokolle als sicher zu beweisen [31, 28, 85, 84]. Eine wichtige Arbeit zu beweisbar sicherem Gruppenschlüsselaustausch ist Katz und Yung [84]. Dort wird ein Protokollcompiler (Katz-Yung-Compiler) präsentiert, der ein (nicht-authentifiziertes) Gruppenschlüsselaustauschprotokoll, das gegen passive Angreifer sicher ist, in ein Protokoll, das auch gegen aktive Angreifer sicher ist, transformiert. Zusammen mit dem Protokoll von Burmester und Desmedt entsteht dadurch ein effizientes Gruppenschlüsselaustauschprotokoll in drei Runden. Einen aktuellen Überblick über die Unterschiede der genannten, auf Ununterscheidbarkeit basierenden Protokolle geben Choo u. a. [48].

Bellare u. a. [8] definieren ein Modell, das zum Ziel hat, den modularen Aufbau von Schlüsselaustauschprotokollen zu ermöglichen. Das Modell erwies sich aber als unpraktikabel und wurde von Canetti und Krawczyk [47] überarbeitet. Ein sehr ähnliches Modell, das auf [8] aufbaut stammt von Shoup [106]. Diese Modelle haben gemeinsam, dass die Sicherheitsdefinition simulationsbasiert ist. Jedoch sind die Modelle in der gegebenen Form nur für die Analyse von Zwei-Parteien-Schlüsselaustauschprotokollen geeignet.

Ein weiterer Ansatz, Schlüsselaustauschprotokolle zu modellieren, existiert in den Modellen für universelle Komponierbarkeit (UC-Modell) [44] bzw. reaktive Simulier-

[1] Key-Agreement wird in Abschnitt 4.3 genauer behandelt.

[2] Der Schlüsselraum $\mathcal{K}$ ist die Menge der möglichen Sitzungsschlüssel, die aus dem Protokoll hervorgehen können.

barkeit [102]. Eine Modellierung von Gruppenschlüsselaustauschprotokollen in diesem Kontext betrachtet beispielsweise STEINER [117]. In diesen Modellen ist jedoch Key-Confirmation zwingend erforderlich. Daher eignen sich die Modelle nicht unmittelbar, um in der Praxis etablierte Protokolle zu untersuchen. Es existieren für diese Modelle auch keine effizienten Zwei-Runden-Protokolle wie von KIM U.A. [85]. Letztendlich ist auch die Formulierung eines Key-Agreement in diesen Modellen unklar (siehe HOFHEINZ U. A. [73]). Zusammengefasst sind diese Modelle derzeit nicht gut zur Analyse von Schlüsselaustauschprotokollen geeignet und weitere Untersuchungen zur Modellierung von Schlüsselaustausch in diesen Modellen ist nötig.

Diese Arbeit stützt sich daher auf die erstgenannte, auf [10] zurückgehende Modellfamilie, mit den Modellen für Gruppenschlüsselaustausch von BRESSON U. A. [30, 32].

Interne Angreifer

Die etablierten Modelle [30, 32] für Gruppenschlüsselaustausch bieten keinerlei Garantien im Falle von internen Angreifern. Üblicherweise wird die Sicherheitsanalyse des Protokolls auf den Fall ehrlicher Teilnehmer eingeschränkt (siehe z. B. [28, 84, 49]). Daher wird in diesem Kapitel für ein auf [30, 32] basierendes Modell die Sicherheit gegen interne Angreifer untersucht und eine Modellerweiterung vorgeschlagen.

Für die Modellierung von STEINER [117] gibt es zumindest eine Untersuchung von ausfallenden Teilnehmern von CACHIN UND STROBL [40]. KATZ UND SHIN [83] haben interne Angreifer im sehr ähnlichen UC-Modell betrachtet. Sie geben auch einen Protokollcompiler (Katz-Shin-Compiler) an, der ein Protokoll, das gegen aktive Angriffe sicher ist, in ein Protokoll transformiert, das auch Sicherheit gegen interne Angreifer bietet. Dieser Protokollcompiler muss dazu dem Protokoll eine weitere Runde hinzufügen und eignet sich daher nicht zur Konstruktion effizienter Protokolle. Da jede Kommunikationsrunde erfordert, dass die Protokollteilnehmer auf ankommende Nachrichten warten müssen, sind Protokolle mit niedriger Rundenzahl in der Regel effizienter. In diesem Kapitel wird gezeigt, dass es möglich ist, eine hohe Sicherheit gegen interne Angreifer auch in zwei Runden mit einem sehr effizienten Protokoll zu erzielen.

Protokolle, die resistent gegen interne Angreifer sind, sind bisher nur in einem Modell mit authentifiziertem Broadcastkanal bekannt. In einem solchen Modell kann der Angreifer Nachrichten, die über den Broadcastkanal gesendet werden, weder verzögern noch verändern. Beispiele für solche Protokolle finden sich in [86, 89, 123, 122, 121]. Dieses Modell erscheint in einer Umgebung wie dem Internet jedoch nicht gerechtfertigt. Es passt auch nicht mit den oben erwähnten Sicherheitsmodellen zusammen, die dem Angreifer volle Kontrolle über das Netzwerk erlauben. Als Beispiel für ein solches Protokoll wird hier das Protokoll von TSENG [121] betrachtet. Es wird gezeigt, dass der Katz-Yung-Compiler das Protokoll in einem asynchronen Netzwerk zwar gegen aktive Angreifer sichern kann, jedoch die Sicherheit gegen interne Angreifer nicht erhalten kann. Es ist jedoch möglich das Protokoll so zu transformieren, dass Sicherheit

gegen interne Angreifer erhalten bleibt und zusätzlich die Rundenanzahl konstant gehalten werden kann. Dazu soll in Abschnitt 4.4 ein Rahmenwerk präsentiert werden.

4.1.1 Sicherheitsmodell

Dieser Abschnitt beschreibt das Sicherheitsmodell für Gruppenschlüsselaustausch aus Bresson u. a. [32], das auch in Katz und Yung [84] zugrunde gelegt wurde. Abweichungen des hier beschriebenen Modells werden im darauf folgenden Abschnitt diskutiert.

Teilnehmer

Es ist eine Menge von m Benutzern $\mathcal{U} = \{U_1, \ldots, U_m\}$ gegeben, wobei m im Sicherheitsparameter ℓ polynomiell wachsen darf. Die Benutzer $U_i \in \mathcal{U}$ stellen die potentiellen Protokollteilnehmer dar. Die Menge der tatsächlichen Teilnehmer an einem Protokoll wird in dieser Arbeit vereinfachend immer mit $U_1, \ldots, U_n$ bezeichnet. Dabei ist n variabel und $U_1, \ldots, U_n$ können *beliebige* Benutzer aus $\mathcal{U}$ sein. Jeder Benutzer $U_i \in \mathcal{U}$ kann eine polynomielle Anzahl von Protokollausführungen simultan durchführen. Dazu hat jeder Benutzer eine unbeschränkte Anzahl an *Instanzen*, von denen jede das Protokoll einmalig ausführen kann. Instanz s des Benutzers U_i wird mit Π_i^s für $s \in \mathbb{N}$ bezeichnet. Jede solche Instanz Π_i^s ist ein Prozess des Benutzers U_i, der die sieben Variablen state_i^s, sid_i^s, pid_i^s, sk_i^s, term_i^s, used_i^s und acc_i^s benutzt.[3] Die Variablen term_i^s, used_i^s und acc_i^s sind boolesche Variablen, die mit false initialisiert werden, state_i^s, sk_i^s, sid_i^s und pid_i^s können Zeichenketten speichern und werden mit einem eindeutigen Wert NULL initialisiert. Die Bedeutung der Variablen soll im Folgenden erklärt werden:

used_i^s gibt an, ob die Instanz schon für eine Protokollausführung initialisiert wurde oder noch im Anfangszustand ist. Die Variable wird auf true gesetzt, sobald die Instanz eine spezielle Nachricht empfängt, die eine Protokollausführung startet.

term_i^s gibt an, ob die Instanz die Protokollausführung beendet hat, d. h. keine weiteren Nachrichten annimmt und keine Berechnungen mehr durchführt. Die Variable wird auf true gesetzt, sobald die Instanz ihre letzte Aktion in einer Protokollausführung beendet hat.

acc_i^s gibt an, ob die beendete Protokollausführung erfolgreich war, d. h., ob die Instanz bereit ist, den erhaltenen Sitzungsschlüssel zur Kommunikation zu verwenden. Die Variable wird bevor die Instanz terminiert auf true gesetzt, wenn alle Überprüfungen, die das Protokoll vorschreibt, erfolgreich waren. Es kann

[3] Der Übersichtlichkeit wegen, werden die Indizes der Instanzen und Variablen gegebenenfalls weggelassen und beispielsweise nur sid_i für eine Variable von U_i oder sid für die Variable im Allgemeinen geschrieben. Falls mehrere Instanzen vorhanden sind, wird die Nummer s dem jeweiligen Benutzer zugeordnet, indem s_i oder s_j geschrieben wird.

vorkommen, dass eine Instanz terminiert, jedoch den Sitzungsschlüssel nicht akzeptiert, weil in der Protokollausführung Fehler auftraten.

state_i^s speichert die Statusinformation während der Protokollausführung.

sk_i^s speichert letztendlich den Sitzungsschlüssel. Die Variable wird belegt, wenn die Instanz Π_i^s eine Protokollausführung akzeptiert.

sid_i^s speichert die Sitzungsnummer, die zugleich ein Name für den Sitzungsschlüssel sk_i^s bildet. Die Variable wird während der Protokollausführung belegt.

pid_i^s speichert die Menge der beabsichtigten Sitzungsteilnehmer, inklusive dem Benutzer U_i selbst. Der Wert pid_i^s wird zu Beginn der Protokollausführung festgelegt.

Initialisierungsphase

In einer Initialisierungsphase müssen die Benutzer einmalig vorbereitet werden, bevor sie an Protokollausführungen teilnehmen können. Dazu wird für jeden Benutzer $U_i \in \mathcal{U}$ ein Schlüsselpaar (SK_i, PK_i) für ein EUF-CMA-sicheres Signaturverfahren erzeugt. Der Benutzer U_i erhält den geheimen Schlüssel SK_i, den zugehörigen öffentlichen Schlüssel PK_i erhalten alle Benutzer. Alle Instanzen eines Benutzers haben Zugriff auf die Schlüssel, die der jeweilige Benutzer besitzt. Es wird angenommen, dass diese Schlüssel *ehrlich* generiert und an die Teilnehmer verteilt werden.

Netzwerkmodell und Angreifer

Das Netzwerk ermöglicht beliebige Punkt-zu-Punkt-Verbindungen zwischen den Benutzern. Die Verbindungen sind jedoch unsicher und das Netzwerk ist vollständig asynchron. Das wird durch einen aktiven Angreifer modelliert, der die Nachrichten ausliefert. Der Angreifer $\mathcal{A}$ ist ein PPT-Algorithmus, der mit den Benutzerinstanzen mittels der Anfragen Send, Reveal, Corrupt und Test interagiert. Dadurch kann der Angreifer Nachrichten lesen, unterdrücken, einfügen, ändern oder verzögern, alte Sitzungsschlüssel lernen und Benutzer korrumpieren, um sie zu personifizieren. Die Anfragen werden im Folgenden genauer erklärt:

$\mathsf{Send}(U_i, s, M)$ Diese Anfrage sendet die Nachricht M an die Instanz Π_i^s. Falls Π_i^s mit einer Nachricht antwortet, wird diese Nachricht zurückgegeben. Send ermöglicht es auch, eine neue Protokollausführung zu starten. Dazu wird die spezielle Nachricht $M = \{U_1, \ldots, U_n\}$ gesendet, wobei U_i selbst in der Menge enthalten sein muss. Die Instanz Π_i^s wird dann die Variablen $\mathsf{pid}_i^s := \{U_1, \ldots, U_n\}$ und $\mathsf{used}_i^s := \mathsf{true}$ setzen und mit der Protokollausführung beginnen.

$\mathsf{Reveal}(U_i, s)$ Diese Anfrage wird an eine Instanz Π_i^s gerichtet, die eine Protokollausführung akzeptiert hat. Dann wird der Sitzungsschlüssel sk_i^s der Instanz zurückgegeben. Falls die adressierte Instanz nicht akzeptiert hat, gibt Reveal den Wert NULL zurück.

$\mathsf{Corrupt}(U_i)$ Diese Anfrage ermöglicht dem Angreifer, die Kontrolle über eine Partei zu übernehmen. Dazu wird der geheime Schlüssel SK_i von U_i zurückgegeben. Benutzer U_i wird *ehrlich* genannt, falls keine Anfrage $\mathsf{Corrupt}(U_i)$ gestellt wurde, sonst *unehrlich* oder *korrumpiert*.

$\mathsf{Test}(U_i, s)$ Der Angreifer darf nur einen Test-Aufruf durchführen. Falls die adressierte Instanz einen Sitzungsschlüssel $\mathsf{sk}_i^s \neq \text{NULL}$ akzeptiert hat, wird ein Zufallsbit b bestimmt und abhängig von b, also jeweils mit Wahrscheinlichkeit $1/2$, der Sitzungsschlüssel sk_i^s oder ein zufällig[4] gezogenes Element aus dem Raum der möglichen Sitzungsschlüssel zurückgegeben.

Der Angreifer muss weitere Bedingungen erfüllen: Die adressierte Instanz Π_i^s muss *frisch* (siehe Definition 4.2) sein und die Frische darf im weiteren Verlauf des Experiments nicht aufgehoben werden. Was das bedeutet wird in den folgenden Absätzen erläutert.

Korrektheit

Um triviale Protokolle auszuschließen, werden nur *korrekte* Protokolle zugelassen. Ein Gruppenschlüsselaustauschprotokoll $\mathcal{P}$ heißt *korrekt*, falls eine Ausführung von $\mathcal{P}$ zwischen beliebigen teilnehmenden Benutzern $U_1, \ldots, U_n$, bzw. ihren Instanzen $\Pi_1^{s_1}, \ldots, \Pi_n^{s_n}$ dazu führt, dass mit überwältigender Wahrscheinlichkeit alle Instanzen denselben Sitzungsschlüssel sk mit einer global eindeutigen Sitzungsnummer sid und einer vollständigen Liste der Teilnehmer in pid akzeptieren. Formal ausgedrückt, sollen die Variablen nach der Protokollausführung die folgende Belegung haben:

- $\mathsf{acc}_1^{s_1} = \cdots = \mathsf{acc}_n^{s_n} = \mathsf{true}$;
- $\mathsf{sk} = \mathsf{sk}_1^{s_1} = \cdots = \mathsf{sk}_n^{s_n}$;
- $\mathsf{sid} = \mathsf{sid}_1^{s_1} = \cdots = \mathsf{sid}_n^{s_n}$ und sid ist global eindeutig;
- $\mathsf{pid} = \mathsf{pid}_1^{s_1} = \cdots = \mathsf{pid}_n^{s_n} = \{U_1, \ldots, U_n\}$.

Frische

Für die Definition der Sicherheit muss festgelegt werden, welche Schlüssel der Angreifer nicht kennen darf, denn natürlich ist es kein Angriff, falls der Angreifer einen Schlüssel kennt, an dessen Protokollausführung er als Teilnehmer mitgewirkt hat. Als

[4] gemäß der Wahrscheinlichkeitsverteilung bei einem ehrlichen Schlüsselaustausch

ein erster Schritt ist es nötig zu definieren, welche Instanzen potentiell miteinander einen gemeinsamen Schlüssel haben und damit *Partner* sind.

Definition 4.1 (Partner). *Zwei Instanzen* $\Pi_i^{s_i}$ *und* $\Pi_j^{s_j}$ *sind* Partner, *falls* $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ *und* $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.

Die Instanzen, die einen Sitzungsschlüssel berechnet haben, der dem Angreifer unbekannt sein soll, werden nun *frisch* genannt. Es sind die Instanzen, die an einer Sitzung ohne korrumpierte Teilnehmer teilgenommen haben, und deren Sitzungsschlüssel nicht vom Angreifer abgefragt wurde.

Definition 4.2 (frisch). *Eine Benutzer-Instanz* $\Pi_i^{s_i}$, *die einen Protokolldurchlauf akzeptiert hat, heißt* frisch, *falls* keine *der folgenden Bedingungen zutrifft:*

- *Für einen Benutzer* $U_j \in \mathsf{pid}_i^{s_i}$ *wurde* $\mathsf{Corrupt}(U_j)$ *aufgerufen und anschließend wurde* $\mathsf{Send}(U_k, s_k, M)$ *mit einem Benutzer* $U_k \in \mathsf{pid}_i^{s_i}$ *aufgerufen.*
- *Ein Aufruf* $\mathsf{Reveal}(U_j, s_j)$ *wurde ausgeführt, wobei die Instanzen* $\Pi_i^{s_i}$ *und* $\Pi_j^{s_j}$ *Partner sind.*

Diese frischen Instanzen sind diejenigen, auf die sich der Test-Aufruf beziehen darf.

Sicherheit

Ein Protokoll kann nur dann sicher sein, wenn der Angreifer keine Möglichkeit hat, einen frischen Schlüssel zu kennen. Dieser Aspekt der Sicherheit wird *implizite Schlüsselauthentifizierung* genannt. Der Schlüssel ist insofern authentifiziert, dass, wenn eine Instanz den Schlüssel akzeptiert hat, der Schlüssel höchstens den Benutzern, die in pid aufgezählt sind, bekannt sein darf.

Der Angreifer soll einen frischen Schlüssel nicht einmal von einem zufälligen Wert aus dem Schlüsselraum unterscheiden können. Dazu wird der Erfolg des Angreifers bestimmt, das geheime Bit der Test-Anfrage zu erraten. Ein Angreifer $\mathcal{A}$, der nach Interaktion mit den Instanzen über die Anfragen – inklusive einem Aufruf $\mathsf{Test}(\Pi_i^s)$ mit einer frischen Instanz Π_i^s – ein Bit d ausgibt, ist erfolgreich, wenn $d = b$ gilt, wobei b das geheime Bit der Test-Anfrage ist und die Frische der Instanz Π_i^s erhalten wurde. Die Wahrscheinlichkeit, dass der Angreifer Erfolg hat, wird mit $\mathsf{Succ}_{\mathcal{A}}^{\mathcal{P}}$ bezeichnet und der Vorteil, den der Angreifer gegenüber zufälligem Raten hat, ergibt sich als

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{P}} := |2 \cdot \mathsf{Succ}_{\mathcal{A}}^{\mathcal{P}} - 1|.$$

Definition 4.3 (Sicherheit bei ehrlichen Teilnehmern). *Ein Gruppenschlüsselaustauschprotokoll* $\mathcal{P}$ *heißt* sicher bei ehrlichen Teilnehmern, *falls für alle PPT-Angreifer* $\mathcal{A}$ *die Funktion* $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{P}}$ *vernachlässigbar im Sicherheitsparameter* ℓ *ist.*

Von den vorgestellten Sicherheitseigenschaften beinhaltet diese Definition implizite Schlüsselauthentifizierung und bei der hier angegebenen Definition von Frische auch Forward-Secrecy, weil ein Corrupt-Aufruf nicht generell die Frische aufhebt, sondern erst ein darauf folgender Send-Aufruf.

4.1.2 Diskussion und Einordnung des Sicherheitsmodells

Das Modell von Bresson u. a. [32] umfasste keine *Forward-Secrecy*. Die Definition der *Frische* bei Bresson u. a. [32] und Katz und Yung [84] sollte Forward-Secrecy in die Sicherheitsdefinition integrieren. Sie wurde von dem Zwei-Parteien-Modell [9] übernommen, ist aber für Gruppenschlüsselaustausch ungenügend. Der Angreifer kann gemäß dieser Definition die Frische von Instanzen erhalten, obwohl sie intuitiv verletzt sein sollte. Dadurch lassen sich Protokolle, die intuitiv sicher sein sollten, in diesem Modell nicht als sicher beweisen. Der „Angriff" läuft folgendermaßen ab: Der Angreifer führt den Schlüsselaustausch solange ehrlich durch, bis ein Teilnehmer U_1 terminiert. Jetzt kann $\mathcal{A}$ einen weiteren Teilnehmer U_2 korrumpieren und die letzte Nachricht von U_2 vor der Auslieferung an einen weiteren Teilnehmer U_3 so modifizieren, dass die Instanz von U_3 eine falsche Sitzungsnummer erhält, also nicht Partner der Instanz von U_1 ist, aber denselben Schlüssel wie U_1 berechnet. U_1 bleibt also frisch, wenn der Angreifer den Sitzungsschlüssel von U_3 lernt, und kann als Test-Instanz benutzt werden. Das erklärt die in dieser Arbeit gegebene Definition der Frische, die drei verschiedene Benutzer betrachtet.

Eine Sitzungsnummer sid zur Definition der Partnerschaft wird erstmals in [9] benutzt. Die Idee entstand demzufolge jedoch schon 1995 in einem Gespräch zwischen Bellare, Petrank, Rackoff und Rogaway. In früheren Versionen des Modells wurde Partnerschaft durch einen „übereinstimmenden Nachrichtenfluss" definiert. Ein übereinstimmender Nachrichtenfluss zwischen zwei Instanzen liegt dann vor, wenn alle Nachrichten, welche eine Instanz gesendet hat, jeweils von der anderen Instanz empfangen wurden. Jedoch ist das Vorhandensein eines übereinstimmenden Nachrichtenflusses eine sehr strenge Forderung, da der Angreifer schon durch eine kleine Modifikation an den versendeten Nachrichten die Partnerschaft unterbrechen kann. Je nachdem, wie weit der Nachrichtenfluss gefasst wird, genügt es beispielsweise schon, dass das Datenpaket auf der Netzwerkschicht verändert wird. Letztendlich wird eine explizite Sitzungsnummer auch in real eingesetzten Protokollen wie beispielsweise SSL/TLS [54] verwendet. Gerade in den Modellen für Gruppenschlüsselaustausch erscheint es viel passender, Sitzungen durch *Sitzungsnummern* zu unterscheiden, weil in diesem Fall nicht davon ausgegangen werden kann, dass alle Teilnehmer von jedem anderen Teilnehmer eine Nachricht erhalten haben, ein Nachrichtenfluss also prinzipiell nicht stattfinden muss. Die Modelle von Bresson u. a. definierten die Sitzung komplizierter über eine Kette von Instanzen, von denen jeweils benachbarte Instanzen Nachrichten ausgetauscht haben. Der hier verwendete Ansatz einer einheitlichen Sitzungsnummer für alle Teilnehmer der Gruppe stammt aus Katz und Yung [84].

4.2 Angriffe interner Protokollteilnehmer

Es wurde bereits erwähnt, dass Angriffe, die nicht die Geheimhaltung des Schlüssels betreffen, in dem bisher beschriebenen Modell nicht ausgeschlossen sind. Nun sollen

relevante Angriffe auf zwei etablierte Gruppenschlüsselaustauschprotokolle vorgestellt werden, die beide in einem Sicherheitsmodell wie dem hier vorgestellten als sicher bewiesen sind. Zum einen wird das Protokoll von KATZ UND YUNG [84] betrachtet. Das Protokoll entspricht dem Protokoll von BURMESTER UND DESMEDT [37], das mittels des Compilers, der in [84] entwickelt wurde, transformiert wird, so dass es sicher im Sinne des obigen Modells ist. Das zweite Protokoll ist das Protokoll von KIM U. A. [85]. Das Protokoll basiert auf ähnlichen Ideen, ist aber durch die Verwendung von Hashfunktionen insgesamt effizienter, benötigt jedoch das Random-Oracle-Modell für den Beweis.

4.2.1 Angriffe auf das Protokoll von Katz und Yung

Das Protokoll von KATZ UND YUNG [84] wird im folgenden mit $\mathcal{KY}$ bezeichnet. Es benötigt eine endliche zyklische Gruppe $\mathbb{G}$ von Primordnung q mit einem Erzeuger g, so dass die DDH-Annahme in der Gruppe $\mathbb{G}$ zulässig ist, und ein SEUF-CMA-sicheres Signaturverfahren. Die Teilnehmer einer Protokollausführung $U_1, \ldots, U_n$ werden in einem Kreis angeordnet, so dass U_1 und U_n benachbart sind. Das Protokoll nutzt Punkt-zu-Punkt-Verbindungen zwischen allen Teilnehmern. Wenn von einem *Broadcast* gesprochen wird, ist das als vielfaches Versenden von Nachrichten auf den entsprechenden Punkt-zu-Punkt-Verbindungen an die beabsichtigten Protokollteilnehmer zu verstehen.

Das Protokoll für einen Schlüsselaustausch zwischen Benutzern $U_1, \ldots, U_n$ läuft in folgender Weise ab: Die Teilnehmer tauschen in der ersten Runde Zufallszahlen aus, um eine eindeutige Kennung für die Sitzung zu erhalten. In der folgenden Runde senden die Teilnehmer $z_i = g^{r_i}$ per Broadcast und benutzen diese Werte, um Diffie-Hellman-Schlüssel mit ihren Nachbarn zu etablieren. In der dritten Runde berechnen die Teilnehmer den Quotienten der Schlüssel mit ihren beiden Nachbarn $X_i = (z_{i+1}/z_{i-1})^{r_i}$ und senden diesen Wert per Broadcast. Dadurch ist es für alle Teilnehmer, die einen Diffie-Hellman-Schlüssel kennen, möglich, alle weiteren Diffie-Hellman-Schlüssel auszurechnen. Den Sitzungsschlüssel berechnet U_i mittels $\mathsf{sk}_i^{s_i} = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i+n-2}$, was einem Produkt aller Diffie-Hellman-Schlüssel entspricht. Einen detaillierteren Überblick über die ausgetauschten Nachrichten gibt Bild 4.1. Das Protokoll $\mathcal{KY}$ wurde in dem in Abschnitt 4.1.1 beschriebenen Modell als sicher bewiesen.

Wenn die Sitzungsnummer als Verkettung aller Nachrichten inklusive ihrer Signaturen gewählt wird, muss das verwendete Signaturverfahren SEUF-CMA sein, was dem Angreifer auch verbietet eine neue Signatur zu einer bereits signierten Nachricht auszugeben. Wenn der Angreifer für eine signierte Nachricht eine weitere Signatur findet, würde das in dem Protokoll dazu führen, dass der Angreifer die Signatur in einer Nachricht ändern kann, so dass der Empfänger dieser Nachricht eine andere Sitzungsnummer berechnet, jedoch den gleichen Sitzungsschlüssel. Da nun zwei Instanzen mit demselben Schlüssel keine Partner sind und damit ein Reveal-Aufruf an den einen, die

	Teilnehmer U_i
Runde 1:	
Vorberechnung	$t_i \xleftarrow{R} \{0,1\}^\ell$
Broadcast	$(U_i \| 0 \| t_i)$
Nachberechnung	$t := t_1 \| \ldots \| t_n$
Runde 2:	
Vorberechnung	$r_i \xleftarrow{R} \mathbb{Z}_q$, $z_i = g^{r_i}$, $\sigma_i^1 = Sig_{SK_i}(1 \| z_i \| \mathsf{pid}_i \| t)$
Broadcast	$(U_i \| 1 \| z_i \| \sigma_i^1)$
Nachberechnung	Verifiziere σ_j^1 $(j = 1, \ldots, n)$
Runde 3:	
Vorberechnung	$X_i = (z_{i+1}/z_{i-1})^{r_i}$, $\sigma_i^2 = Sig_{SK_i}(2 \| X_i \| \mathsf{pid}_i \| t)$
Broadcast	$(U_i \| 2 \| X_i \| \sigma_i^2)$
Nachberechnung	Verifiziere σ_j^2 $(j = 1, \ldots, n)$
Sitzungsschlüssel	$\mathsf{sk}_i = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i+n-2}$, sid_i ist die Konkatenation aller Nachrichten.

Bild 4.1: Das Schlüsselaustauschprotokoll $\mathcal{KY}$ [84].

Frische bezüglich Test des anderen nicht verletzen würde, kann der Angreifer das Bit des Test-Orakels einfach erfahren.

Ein Angriff auf die Integrität einer Sitzung. Es wird ein Angriff auf das Protokoll $\mathcal{KY}$ vorgestellt, bei dem das Ziel ist, dass ehrliche Teilnehmer in derselben Sitzung unterschiedliche Schlüssel erhalten, davon jedoch nichts bemerken. Ein Angriff dieser Art soll Angriff auf die *Integrität* genannt werden. Der Angriff liegt außerhalb des Modells von Abschnitt 4.1.1, später soll das Modell aber so erweitert werden, dass es auch diesen Angriff abdeckt.

Bei dem folgenden Angriff sind interne Angreifer involviert, grundsätzlich ist ein solcher Angriff auch ohne korrumpierte Teilnehmer denkbar. Es wird angenommen, dass vier Benutzer teilnehmen. Der Angreifer $\mathcal{A}$ kann dann einen Angriff durch die folgenden Schritte durchführen:

1. $\mathcal{A}$ korrumpiert U_1 und U_3. Dann startet $\mathcal{A}$ einen Schlüsselaustausch zwischen U_1, U_2, U_3 und U_4, an dem $\mathcal{A}$ im Namen von U_1 und U_3 zunächst ehrlich teilnimmt.

2. In der dritten Runde wird $\mathcal{A}$ die Werte X_1 und X_3 zunächst wie angegeben berechnen, aber anschließend mittels $\widetilde{X_1} := X_3, \widetilde{X_3} := X_1$ vertauschen. Dann signiert $\mathcal{A}$ die Nachricht $(U_1 \| 2 \| \widetilde{X_1} \| t)$ mit dem Signaturschlüssel von U_1 und $(U_3 \| 2 \| \widetilde{X_3} \| t)$ mit dem Signaturschlüssel von U_3. Der Angreifer sendet beide

Nachrichten $(U_1\|2\|\widetilde{X_1}\|\sigma_1^2)$ und $(U_3\|2\|\widetilde{X_3}\|\sigma_3^2)$ an alle Teilnehmer. $\mathcal{A}$ hat in dieser Runde also die Werte X_i von U_1 und U_3 vertauscht.

Bei diesem Angriff erhalten alle Teilnehmer identische Nachrichten und berechnen daher auch dieselbe Sitzungsnummer. Die Teilnehmer nutzen jedoch unterschiedliche Teile der Nachrichten zur Berechnung des Sitzungsschlüssels. Das führt dazu, dass U_2 und U_4 mit überwältigender Wahrscheinlichkeit unterschiedliche Sitzungsschlüssel berechnen: Bei einer ehrlichen Protokollausführung berechnen U_2 und U_4 dieselben Schlüssel sk_2 bzw. sk_4. Als Quotient der Schlüssel ergibt sich $\mathsf{sk}_2/\mathsf{sk}_4 = X_3^4 \cdot (z_2/z_4)^{r_3 4} = 1$. Der Quotient der Schlüssel $\widetilde{\mathsf{sk}_2}$ und $\widetilde{\mathsf{sk}_4}$ aus der Protokollausführung, wenn U_1 und U_3 ihre Werte X_i in den Nachrichten der dritten Runde vertauschen, ist $\widetilde{\mathsf{sk}_2}/\widetilde{\mathsf{sk}_4} = X_1^4 \cdot (z_2/z_4)^{r_3 4}$. In diesem Fall stimmen die Schlüssel von U_2 und U_4 nur mit vernachlässigbarer Wahrscheinlichkeit überein, nämlich dann, wenn $X_1 = X_3$. Dieser Angriff verletzt genau genommen die Korrektheit aus [84], bei der gefordert wird, dass Teilnehmer, welche dieselbe Sitzungsnummer haben, auch denselben Sitzungsschlüssel errechnen. Da eine solche Forderung mit aktiven und internen Angreifern nicht-trivial ist, wird in dieser Arbeit die Korrektheit wie üblich explizit auf passive Angreifer beschränkt und die weitergehende Forderung für aktive und interne Angreifer wird in Abschnitt 4.3 als eine eigene Eigenschaft, die *Integrität* genannt wird, modelliert.

Das Protokoll $\mathcal{KY}$ ist kein Key-Agreement, weil nicht zwingend alle Teilnehmer zur Sitzungsschlüsselberechnung beitragen. Zwei benachbarte unehrliche Teilnehmer U_i und U_{i+1} können einen beliebigen Sitzungsschlüssel erzwingen, da sie den Diffie-Hellman-Schlüssel zwischen sich – und damit das Produkt aller Diffie-Hellman-Schlüssel – vollständig unter Kontrolle haben.

4.2.2 Angriffe auf das Protokoll von Kim, Lee und Lee

Kim u. a. [85] haben ein effizientes authentifiziertes Gruppenschlüsselaustauschprotokoll vorgestellt, das hier mit $\mathcal{KLL}$ bezeichnet wird. Es wird behauptet, dass das Protokoll Vorkehrungen gegen illegale Teilnehmer oder Systemfehler trifft, also eine gewisse Sicherheit gegen interne Angreifer bietet. Allerdings findet sich werder in der Sicherheitsdefinition noch im Sicherheitsbeweis eine Formalisierung in dieser Richtung und dieser Abschnitt wird zeigen, dass das Protokoll keinen Widerstand gegen interne Angreifer bietet – schon ein einziger korrumpierter Teilnehmer genügt, um einen Impersonations-Angriff durchzuführen.

Ähnlich wie für das Protokoll $\mathcal{KY}$, wird auch hier eine zyklische Gruppe $\mathbb{G}$ von Primordnung q mit einem Erzeuger g gewählt, so dass die CDH-Annahme zulässig ist. Benötigt werden zusätzlich ein Signaturverfahren, das EUF-CMA-sicher ist, und eine kryptographische Hashfunktion $H(\cdot)$, die als Random-Oracle modelliert wird. Die Protokollteilnehmer sind wieder in einem Kreis angeordnet und auch hier steht *Broadcast* für das vielfache Versenden einer Nachricht über Punkt-zu-Punkt-Kanäle. Das Protokoll beginnt damit, dass alle Teilnehmer $y_i = g^{x_i}$ per Broadcast senden.

	Teilnehmer U_i, $i \neq n$	Teilnehmer U_n
Runde 1:		
Vorberechnung	$k_i \xleftarrow{R} \{0,1\}^\ell$, $x_i \xleftarrow{R} \mathbb{Z}_q$, $y_i = g^{x_i}$	
	$M_i^1 = y_i$	$M_n^1 = H(k_n\|0)\|y_n$
	$\sigma_i^1 = Sig_{SK_i}(M_i^1\|\mathsf{pid}_i\|0)$	
Broadcast	$(M_i^1\|\sigma_i^1)$	
Nachberechnung	Verifiziere σ_j^1 $(j = 1, \ldots, n)$	
Runde 2:		
Vorberechnung	$t_i^L = H(y_{i-1}^{x_i}\|\mathsf{pid}_i\|0)$, $t_i^R = H(y_{i+1}^{x_i}\|\mathsf{pid}_i\|0)$, $T_i = t_i^L \oplus t_i^R$	
	$M_i^2 = k_i\|T_i$	$M_n^2 = k_n \oplus t_n^R\|T_n$
	$\sigma_i^2 = Sig_{SK_i}(M_i^2\|\mathsf{pid}_i\|0)$	
Broadcast	$(M_i^2\|\sigma_i^2)$	
Nachberechnung	Verifiziere σ_j^2 $(j = 1, \ldots, n)$, $T_1 \oplus \cdots \oplus T_n = 0$, „entschlüssele“ k_n, prüfe $H(k_n\|0)$ aus Runde 1 für k_n	
Sitzungsschlüssel	$\mathsf{sk}_i = H(k_1\| \ldots \|k_n\|0)$	

Bild 4.2: Das Gruppenschlüsselaustauschprotokoll $\mathcal{KLL}$ [85].

Das dient wieder dazu Diffie-Hellman-Schlüssel t_i^L, t_i^R zwischen Instanzen benachbarter Teilnehmer zu etablieren. In der zweiten Runde senden die Teilnehmer die XOR-Summe $T_i = t_i^L \oplus t_i^R$ ihrer beiden Schlüssel per Broadcast. Alle Benutzer, die einen Diffie-Hellman-Schlüssel kennen, können dadurch die übrigen Schlüssel ausrechnen. Zusätzlich wird eine Zufallszahl k_i als Beitrag zum Sitzungsschlüssel an alle gesendet. Ein ausgezeichneter[5] Teilnehmer U_n wird jedoch k_n nur verschlüsselt als $k_n \oplus t_n^R$ übertragen. Da alle Teilnehmer die Diffie-Hellman-Schlüssel kennen, können sie die Zufallszahl k_n entschlüsseln und als Sitzungsschlüssel $\mathsf{sk} = H(k_1\| \ldots \|k_n\|0)$ berechnen. Bild 4.2 fasst das Protokoll $\mathcal{KLL}$ für die Teilnehmer $U_1, \ldots, U_n$ zusammen.

In Kim u. a.[85] wird die Sitzungsnummer sid nicht festgelegt. Damit ist das Beweismodell nicht anwendbar und der Beweis in [85] kann nicht als gültig akzeptiert werden. Selbst das Standardverfahren, die Sitzungsnummer aus der Konkatenation aller Protokollnachrichten zu berechnen, genügt nicht, um das Protokoll zu sichern: Seien $n > 3$ Teilnehmer gegeben. Dann kann ein aktiver Angreifer $\mathcal{A}$ folgendermaßen vorgehen, um eine Situation zu produzieren, in der U_1 und U_3 mit unterschiedlichen Sitzungsnummern enden, aber einen identischen Sitzungsschlüssel $\mathsf{sk}_1^{s_1} = \mathsf{sk}_3^{s_3}$ berechnen:

1. Zunächst führt $\mathcal{A}$ einen vollständigen Protokolldurchlauf aus und fängt die Nachricht $M_1^1\|\sigma_1^1$ ab, die von U_1 in Runde 1 gesendet wurde.

[5] Der ausgezeichnete Teilnehmer U_n kann beispielsweise derjenige mit dem lexikographisch größten öffentlichen Schlüssel sein.

2. Dann kann $\mathcal{A}$ eine weitere Protokollausführung initiieren und in Runde 1 die neue Nachricht von U_1 nur an alle Teilnehmer außer U_3 ausliefern, aber an U_3 die alte Nachricht $M_1^1 \| \sigma_1^1$ aus dem vorhergehenden Protokolldurchlauf senden.

Da U_3 nicht mit U_1 benachbart ist, wird diese Nachricht von U_3 nicht für das Protokoll benötigt, sondern dient jetzt nur der Berechnung der Sitzungsnummer. Da sich die Nachricht von der Nachricht, welche die übrigen Protokollteilnehmer bekommen haben, mit überwältigender Wahrscheinlichkeit unterscheidet, wird U_3 eine andere Sitzungsnummer ausrechnen. Jedoch beeinflusst diese Nachricht nicht die Berechnung des Sitzungsschlüssels durch U_3. Damit sind U_1 und U_3 letztendlich keine Partner und der Angreifer darf Reveal auf U_3 anwenden, ohne die Frische von U_1 zu verletzen.

Auch eine Protokollausführung, die mit verschiedenen Sitzungsschlüsseln unter derselben Sitzungsnummer endet, kann analog wie für das Protokoll $\mathcal{KY}$ herbeigeführt werden, falls als Sitzungsnummer die Konkatenation aller Protokollnachrichten verwendet wird.

Dieser „triviale“ Angriff lässt sich verhindern, indem, etwas komplizierter, die Sitzungsnummer sid als

$$\mathsf{sid} = H(k_1 \| \ldots \| k_{n-1} \| H(k_n \| 0))$$

berechnet wird. Das führt zusätzlich dazu, dass gleiche Sitzungsschlüssel auch mit überwältigender Wahrscheinlichkeit zu gleichen Sitzungsnummern führen. Ein Angriff auf die Integrität wie anhand des Protokolls $\mathcal{KY}$ gezeigt wurde, ist daher nicht möglich. Diese Konstruktion der Sitzungsnummer wird in Abschnitt 4.4 genauer untersucht.

Ein Impersonations-Angriff. Der folgende Angriff ist unabhängig von der Wahl der Sitzungsnummer möglich. Es ist bei dem Protokoll $\mathcal{KLL}$ nicht garantiert, dass tatsächlich alle beabsichtigten Teilnehmer an der Sitzung überhaupt beteiligt sind. Seien $n > 2$ Teilnehmer gegeben, dann kann der Angreifer $\mathcal{A}$ folgendermaßen eine Impersonation herbeiführen:

1. Zuerst benötigt $\mathcal{A}$ eine Protokollmitschrift einer erfolgreichen Protokollausführung zwischen Benutzern $U_1, \ldots, U_n$. Das erhält er, indem er sich passiv verhält und Nachrichten ehrlich ausliefert.

2. $\mathcal{A}$ wird als nächstes Corrupt(U_1) aufrufen, um den geheimen Schlüssel von Benutzer U_1 zu lernen und damit U_1 kontrollieren zu können.

3. $\mathcal{A}$ initialisiert unbenutzte Instanzen von $U_3, \ldots, U_n$ mit $\mathsf{pid} = \{U_1, \ldots, U_n\}$, um eine Protokollausführung zu beginnen. $\mathcal{A}$ wird als Benutzer U_1 teilnehmen und versuchen Benutzer U_2 zu personifizieren.

4. In Runde 1 sendet $\mathcal{A}$ für U_2 erneut die Nachrichten, die U_2 im vorherigen Protokolldurchlauf gesendet hat, und führt im Namen von U_1 das Protokoll ehrlich aus.

5. In Runde 2 wird $\mathcal{A}$ wieder die alten Nachrichten von U_2 verwenden. Im Namen von U_1 muss der Angreifer jetzt

$$T_1 := T_2 \oplus \cdots \oplus T_n$$

berechnen. Dadurch gleicht er den Fehler, den U_2 bei der „Nicht-Berechnung“ von T_2 macht, aus, so dass alle Teilnehmer $T_1 \oplus \cdots \oplus T_n = 0$ überprüfen können. Im Namen von U_1 sendet er die Nachricht $M_1^2 := k_1 \| T_1 \| \sigma_1^2$.

Alle Teilnehmer berechnen wie gewohnt den Sitzungsschlüssel und akzeptieren die Protokollausführung, da keine Möglichkeit besteht festzustellen, dass U_2 nicht beteiligt war.

4.3 Erweiterte Sicherheitseigenschaften

Es wurde in diesem Kapitel gezeigt, dass Protokolle, die im Modell aus Abschnitt 4.1.1 sicher sind, immer noch Angriffe mit unehrlichen Protokollteilnehmern zulassen. In Präsenz korrumpierter Teilnehmer kann es natürlich nicht verhindert werden, dass der Angreifer den Sitzungsschlüssel lernt. Für ehrliche Teilnehmer kann es jenseits der Geheimhaltung des Schlüssels jedoch weitere wünschenswerte Eigenschaften geben. Es kann für bestimmte Anwendungen – beispielsweise eine Art „sicherer Notruf“ – wichtig sein, den Fall auszuschließen, dass Parteien unterschiedliche Schlüssel erhalten und daher nicht kommunizieren können – auch wenn andere Teile des Notrufsystems ausfallen oder sich fehlerhaft verhalten. In diesem Abschnitt werden daher in Ergänzung des Modells aus Abschnitt 4.1.1 weitere Sicherheitsbegriffe formalisiert, die das Modell für sicheren Gruppenschlüsselaustausch erweitern und die in den Abschnitten 4.2.1 und 4.2.2 beschriebenen Angriffe abdecken.

Integrität

Diese Eigenschaft erweitert die Korrektheit des Verfahrens für den Fall, dass der Angreifer aktiv ist und Teilnehmer des Protokolls korrumpieren darf. Damit wird effektiv gefordert, dass Teilnehmer derselben durch sid identifizierten Sitzung denselben Sitzungsschlüssel sk berechnen und über den potentiellen Teilnehmerkreis pid übereinstimmen. Der Angriff auf das Protokoll $\mathcal{KY}$ aus Abschnitt 4.2.1, bei dem zwei ehrliche Protokollteilnehmer in derselben Sitzung unterschiedliche Sitzungsschlüssel berechnen, wird dadurch erfasst.

Die Bedingung, dass die Menge pid unter den Teilnehmern einer Sitzung selbst unter Angriffen übereinstimmt, ist wichtig. Eine Situation mit abweichender pid wäre beispielsweise gegeben, wenn durch Einfluss des aktiven Angreifers verursacht wird, dass ein Teilnehmer U_2 die Menge $\mathsf{pid}_2 = \{U_1, U_2, U_3\}$ für seine Partner hält, und ein weiterer Teilnehmer U_3 von Partnern $\mathsf{pid}_3 = \{U_1, U_2, U_3, U_4\}$ ausgeht. Falls tatsächlich nur U_1, U_2 und U_3 einen gemeinsamen Sitzungsschlüssel sk berechnet haben, ist die

implizite Schlüsselauthentifizierung nicht verletzt. Jedoch stimmen U_2 und U_3 nicht überein, ob die Sitzung gegenüber dem Benutzer U_4 vertraulich ist oder nicht, so dass U_3 nicht bereit ist, eine gegenüber U_4 geheimzuhaltende Information zu senden. Um eine solche Situation zu vermeiden, wird die Integrität verlangen, dass eine übereinstimmende Sitzung in einer übereinstimmenden Menge von Partnern resultiert. Die Integrität kann nun formal in der folgenden Weise definiert werden:

Definition 4.4 (Integrität)**.** *Ein Gruppenschlüsselaustauschprotokoll bietet* Integrität*, wenn mit überwältigender Wahrscheinlichkeit alle Instanzen von ehrlichen Teilnehmern U_j $(j = 1, \ldots, n)$, die einen Protokolldurchlauf mit derselben Sitzungsnummer* sid *akzeptieren*

- *einen identischen Sitzungsschlüssel* sk *berechnen, und*
- *die Sitzung einer identischen Menge* pid *der beabsichtigten Partner zuordnen.*

Entity-Authentication

Entity-Authentication verhindert Impersonationsangriffe. Ein Beispiel für diese Art von Angriff auf das Protokoll $\mathcal{KLL}$ zeigt Abschnitt 4.2.2. In den Modellen für Zwei-Parteien-Schlüsselaustausch [10, 47] und in dem Modell für Passwort-basierten Schlüsselaustausch von Bellare u. a. [9] wird Entity-Authentication betrachtet. Eine andere Definition für Entity-Authentication wird in Jiang und Gong [80] vorgestellt. Die hier vorgestellte Definition kombiniert die Formulierung aus [80] mit der Identifikation der Sitzung durch die Sitzungsnummer wie in [9]. Die formale Definition für den Fall eines Gruppenschlüsselaustauschs lautet dann folgendermaßen:

Definition 4.5 (Entity-Authentication)**.** Entity-Authentication *für eine Instanz* $\Pi_i^{s_i}$ *ist gegeben, wenn sowohl* $\mathsf{acc}_i^{s_i} = \mathsf{true}$ *als auch für alle ehrlichen* $U_j \in \mathsf{pid}_i^{s_i}$ *mit überwältigender Wahrscheinlichkeit eine Instanz* $\Pi_j^{s_j}$ *mit* $\mathsf{sid}_j^{s_j} = \mathsf{sid}_i^{s_i}$ *und* $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ *existiert.*

Key-Agreement. Auch wenn der Sitzungsschlüssel bei Anwesenheit von unehrlichen Parteien nicht als geheim angenommen werden kann, sind die unehrlichen Parteien nicht zwangsläufig auch in der Lage, den Schlüssel vorherbestimmen zu können – also im Protokoll auf einen eventuell schon vorher vereinbarten Schlüssel hinzuarbeiten. Ein Verfahren, bei dem jeder Teilnehmer zur Berechnung des Schlüssels beiträgt, wird *Key-Agreement* genannt. Idealerweise sollte ein Key-Agreement dann, wenn wenigstens ein Teilnehmer ehrlich dem Protokoll folgt, einen Sitzungsschlüssel berechnen, der vor Beginn des Key-Agreement-Protokolls unbekannt war. Jedoch ist der Begriff Key-Agreement nicht klar definiert und wird unterschiedlich ausgelegt.

Weil die Agreement-Eigenschaft nicht häufig berücksichtigt wird, wird zunächst ein motivierendes Szenario für diese Eigenschaft beschrieben: Die Protokollteilnehmer sind in eine Umgebung eingebettet, die andere Kommunikation als die Protokollnachrichten unterbindet. Ein gegebener Angreifer ist auch nicht in der Lage, die

Protokollnachrichten des Schlüsselaustauschs abzuhören. In dieser Situation hat kein Protokollteilnehmer die Möglichkeit den Schlüssel nach außen zu kommunizieren. Der Schlüssel kann also nur dann dem (externen) Angreifer bekannt werden, wenn ein oder mehrere Protokollteilnehmer es schaffen, einen vorher vereinbarten Schlüssel zu erzielen. Ein *Key-Agreement* garantiert in diesem Szenario, dass der Sitzungsschlüssel gegenüber Angreifern, welche die Nachrichten des Schlüsselaustauschprotokolls nicht abhören können, geheim bleibt.

Die Agreement-Eigenschaft soll nun formal gefasst werden. Die Definition begrenzt den Einfluss, den die Teilnehmer auf den Sitzungsschlüssel haben so, dass solange auch nur ein ehrlicher Teilnehmer an einem Key-Agreement teilnimmt, der Sitzungsschlüssel nur mit vernachlässigbarer Wahrscheinlichkeit in einem zuvor festgelegten vernachlässigbaren Anteil des Schlüsselraums liegt.

Definition 4.6 (Key-Agreement). *Seien Benutzer $U_1, \ldots, U_m$ gegeben und sei $\mathcal{P}$ ein Schlüsselaustauschprotokoll. Für ein Paar von PPT-Algorithmen $(\mathcal{A}_1, \mathcal{A}_2)$ sei das folgende Experiment definiert:*

1. *Die nötige Initialisierung für das Protokoll $\mathcal{P}$ wird ausgeführt, um die langfristigen Schlüssel zu etablieren.*

2. *Der Angreifer $\mathcal{A}_1$ bekommt Zugriff auf die öffentlichen Schlüssel der Benutzer, die Anfragen* Send *und* Reveal *und darf bis zu $m-1$-mal* Corrupt *aufrufen. $\mathcal{A}_1$ muss schließlich ein Quadrupel (i, s, χ_κ, a) ausgeben, wobei i und s eine Instanz auswählen und a eine beliebige Zustandsinformation ist. Weiterhin muss gelten, dass*

   - *U_i ein ehrlicher Benutzer ist und die Instanz Π_i^s unbenutzt ist, d. h. $\mathsf{used}_i^s = \mathsf{false}$;*
   - *χ_κ ein PPT-Algorithmus mit der Eigenschaft ist, dass die Menge $\kappa := \{\mathsf{sk} \in \mathcal{K} : \chi_\kappa(\mathsf{sk}) = \mathsf{true}\}$ nur einen vernachlässigbaren Anteil des Schlüsselraums umfasst, d. h. $|\kappa|/|\mathcal{K}| = \mathrm{negl}(\ell)$.*

3. *Der Angreiferalgorithmus $\mathcal{A}_2$ bekommt a als Eingabe und versucht zu erreichen, dass die Instanz Π_i^s einen Sitzungsschlüssel $\mathsf{sk}_i^s \in \kappa$ akzeptiert. Dazu darf $\mathcal{A}_2$ die Anfragen* Send *und* Reveal *beliebig benutzen, aber* Corrupt *nur für Benutzer ungleich U_i aufrufen und insgesamt darf die Anzahl der* Corrupt*-Aufrufe von $\mathcal{A}_1$ und $\mathcal{A}_2$ nicht $m-1$ übersteigen.*

Wenn für alle Paare von PPT-Algorithmen $(\mathcal{A}_1, \mathcal{A}_2)$ die Wahrscheinlichkeit, dass $\mathcal{A}_2$ im obigen Experiment Erfolg hat, vernachlässigbar ist, dann wird das Protokoll $\mathcal{P}$ mit Key-Agreement *bezeichnet.*

Zusammengefasst wird ein Gruppenschlüsselaustauschprotokoll *sicher* genannt, falls es korrekt ist, Sicherheit bei ehrlichen Teilnehmern bietet und die Eigenschaften Integrität sowie Entity-Authentication erfüllt. Wenn das Verfahren zusätzlich ein Key-Agreement ist, heißt das Protokoll sicheres Group-Key-Agreement.

Definition 4.7 (Sicherheit). *Ein Gruppenschlüsselaustauschprotokoll ist* sicher*, wenn es korrekt ist, sicher im Sinne von Definition 4.3, Integrität nach Definition 4.4 und Entity-Authentication nach Definition 4.5 bietet. Das Protokoll heißt* sicheres Group-Key-Agreement*, falls es zusätzlich ein Key-Agreement im Sinne von Definition 4.6 ist.*

4.4 Ein Rahmenwerk für sichere Protokolle

In diesem Abschnitt wird ein Rahmenwerk beschrieben, das es erlaubt sichere Zwei-Runden-Gruppenschlüsselaustauschprotokolle zu konstruieren. Zunächst wird ein Protokoll von TSENG [121] betrachtet, das Sicherheit gegen passive Angreifer und unehrliche Protokollteilnehmer bietet, jedoch keine Sicherheit gegen aktive Angreifer in einem Modell ohne authentifizierte Kanäle. Die Sicherheit von Gruppenschlüsselaustauschprotokollen gegen aktive Angreifer lässt sich üblicherweise durch den Protokoll-Compiler von Katz und Yung erzielen. Wenn man ihn auf das Protokoll von Tseng anwendet, verliert man aber die Sicherheit gegen interne Angreifer. Das wird durch einen Angriff demonstriert. Um das Rahmenwerk zu entwickeln wird anschließend die Key-Confirmation genauer betrachtet und eine abgeschwächte Fassung motiviert. Es folgt eine Diskussion der Rolle der Sitzungsnummer im Hinblick auf aktive Angreifer, bevor aus den Erkenntnissen das Rahmenwerk hergeleitet und formalisiert wird.

4.4.1 Ein gegen interne Angreifer sicheres Protokoll

In TSENG [121] wird eine Variante des Protokolls von BURMESTER UND DESMEDT [37] vorgestellt, die in einem authentifizierten Broadcast-Netzwerk sicher gegen interne Angreifer ist. Das Protokoll erlaubt sogar die Identifizierung der internen Angreifer. Der authentifizierte Broadcastkanal garantiert, dass, wenn eine Nachricht gesendet wird, immer alle Teilnehmer genau diese Nachricht erhalten. Der Angreifer oder der Sender haben keine Möglichkeit, selektiv Nachrichten an einzelne Teilnehmer zu blockieren bzw. unterschiedliche Nachrichten zu senden. Eine solche starke Annahme scheint jedoch in der Regel nicht gerechtfertigt und entspricht nicht dem Netzwerkmodell des in diesem Kapitel beschriebenen Sicherheitsmodells.

Das Verfahren benötigt wie das Protokoll $\mathcal{KY}$ eine zyklische Gruppe $\mathbb{G}$, erzeugt von g mit Ordnung einer Primzahl q, so dass die DDH-Annahme zulässig ist. Der Sitzungsschlüssel sk wird ein Gruppenelement aus $\mathbb{G}$ sein. Das Protokoll kann wie folgt zusammengefasst werden: In der ersten Runde wählen alle Protokollteilnehmer U_i ein zufälliges $x_i \in \mathbb{Z}_q$ und senden $y_i = g^{x_i}$ per Broadcast. In der zweiten Runde wartet U_i auf die y_j und berechnet $z_i = (y_{i+1}/y_{i-1})^{x_i}$ und einen nicht-interaktiven Zero-Knowledge-Beweis, dass der allen Teilnehmern bekannte Quotient (y_{i+1}/y_{i-1}) tatsächlich mit dem geheimen Exponenten x_i des Wertes y_i potenziert wurde. Jeder Benutzer U_i sendet z_i zusammen mit dem Beweis. Schließlich werden die Nachrichten

	Teilnehmer U_i
Runde 1:	
Vorberechnung	$x_i \xleftarrow{R} \mathbb{Z}_q,\ y_i = g^{x_i},\ M_i^1 = y_i$
Broadcast	M_i^1
Runde 2:	
Vorberechnung	$r_i \xleftarrow{R} \mathbb{Z}_q,$ $z_i = (y_{i+1}/y_{i-1})^{x_i}, \quad \alpha_i = g^{r_i},$ $\beta_i = (y_{i+1}/y_{i-1})^{r_i}, \quad \gamma_i = r_i + H(z_i, \alpha_i, \beta_i) \cdot x_i \bmod q,$ $M_i^2 = (z_i, \alpha_i, \beta_i, \delta_i)$
Broadcast	M_i^2
Nachberechnung	Verifiziere $(y_{i+1}/y_{i-1})^{\delta_j} = \beta_j z_j^{H(z_j,\alpha_j,\beta_j)}$ und $g^{\delta_j} = \alpha_j y_j^C$ für $j = 1 \ldots n, j \neq i$
Sitzungsschlüssel	$\mathsf{sk}_i = (y_{i-1})^{nx_i} \cdot z_i^{n-1} \cdot z_{i+1}^{n-2} \cdots z_{i-2}$

Bild 4.3: Das Protokoll von Tseng [121]

und Beweise überprüft und der Sitzungsschlüssel als $\mathsf{sk}_i = (y_{i-1})^{nx_i} \cdot z_i^{n-1} \cdot z_{i+1}^{n-2} \cdots z_{i-2}$ berechnet. Ein detaillierterer Ablauf findet sich in Bild 4.3.

Tseng [121] beweist, dass dieses Protokoll sicher gegen passive Angreifer und betrügende Teilnehmer ist, unter der Annahme, dass ein authentifizierter Broadcast-Kanal vorhanden ist. Aufgrund dieser Annahme ist das Protokoll in einem Netzwerk-Modell wie es in dieser Arbeit betrachtet wurde nicht sicher gegen aktive Angreifer. Ein solcher aktiver Angreifer wäre in der Lage einzelne Nachrichten des Broadcast aufzuhalten oder zu ersetzen. Ebenso wären betrügende Teilnehmer in der Lage, statt dieselbe Nachricht per Broadcast an alle zu senden, unterschiedliche Nachrichten an verschiedene Teilnehmer zu senden. In den Protokollbeschreibungen wie beispielsweise bei $\mathcal{KY}$ [84] bedeutet "Broadcast" lediglich, dass die Nachricht an alle Teilnehmer gesendet wird, es wird jedoch keine Garantie des zugrundeliegenden Netzwerks erwartet. Anhand von Tsengs Protokoll soll bewiesen werden, dass in einem Punkt-zu-Punkt-Netzwerk die ursprüngliche Sicherheit gegen betrügende Teilnehmer nicht besteht. Eine Anwendung des Katz-Yung-Compilers kann zwar die Sicherheit gegen aktive Angreifer garantieren, den folgenden Angriff, der nur einen unehrlichen Teilnehmer benötigt, aber nicht verhindern.

Ein Angriff mit einem unehrlichen Teilnehmer

Es wird eine Protokollausführung zwischen vier Teilnehmern $U_1, \ldots, U_4$ angenommen. Sei U_1 ein korrumpierter Teilnehmer. In der ersten Runde wird U_1 zusätzlich zu dem Zufallswert x_1 auch einen Zufallswert $\tilde{x}_1$ wählen. Dann sendet er den falschen Wert $\tilde{y}_1 = g^{\tilde{x}_1}$ an U_3 und den ehrlich berechneten Wert $y_1 = g^{x_1}$ an U_2 und U_4. Da die Benutzer U_2 und U_4 die einzigen Teilnehmer sind, die den Wert y_1 in Runde 2 überhaupt

benutzen, werden die Nachrichten in Runde 2 nicht beeinflusst, auch nicht die eines ehrlichen U_3. Jedoch kann U_1 in Runde 2 an U_3 einen falschen Wert $\tilde{z}_1 = (y_2/y_4)^{\tilde{x}_1}$ senden und einen für U_3 gültigen Beweis für diesen Wert produzieren. An U_2 und U_4 wird der korrekte Wert $z_1 = (y_2/y_4)^{x_1}$ mit dem zugehörigen Beweis gesendet. Da U_3 den falschen Wert $\tilde{z}_1$ jetzt zur Berechnung des Sitzungsschlüssels verwendet ohne den Fehler zu erkennen wird er einen anderen Sitzungsschlüssel als U_2 und U_4 berechnen und akzeptieren. Falls die Sitzungsnummern der Teilnehmer trotzdem gleich sind, ist die Integrität verletzt, andernfalls die Entity-Authentication – in jedem Fall hat also ein Angriff auf die zusätzlichen Eigenschaften stattgefunden.

Das Protokoll kann durch Standardverfahren in ein sicheres Gruppenschlüsselaustauschprotokoll transformiert werden. Dazu kann nach dem Katz-Yung-Compiler [84] auch der Katz-Shin-Compiler [83] angewendet werden. Das resultierende Protokoll wird dann jedoch vier Runden haben, da beide Compiler dem Protokoll eine Runde hinzufügen. Im nächsten Abschnitt soll daher ein Rahmenwerk eingeführt werden, das es erlaubt, die Rundenanzahl des Ursprungsprotokolls beizubehalten, und damit eine Konstruktionsbeschreibung für effiziente und sichere Gruppenschlüsselaustauschprotokolle liefert.

4.4.2 Key-Confirmation

Eine Key-Confirmation ist eine Protokollrunde, in der die Teilnehmer sich gegenseitig beweisen, im Besitz des Sitzungsschlüssels zu sein. Das Protokoll erreicht dadurch *explizite Schlüsselauthentifizierung*. Hier wird die Key-Confirmation in einem etwas weiteren Sinne aufgefasst: Die Teilnehmer beweisen nicht nur den Besitz des Sitzungsschlüssels, sondern auch die zugehörige Sitzungnummer und die Menge der Teilnehmer. Eine Key-Confirmation ohne den Bezug zur Sitzung scheint wenig sinnvoll zu sein: ein falsch zugeordneter Sitzungsschlüssel führt ebenso zu Kommunikationsproblemen.

Eine Key-Confirmation mit Sitzungsbestätigung kann ein Schlüsselaustauschprotokoll gegen interne Angreifer sichern:

- **Integrität** bedeutet, dass keine zwei Teilnehmer an derselben Sitzung – identifiziert durch die Sitzungsnummer – unterschiedliche Sitzungsschlüssel berechnen oder von einer unterschiedlichen Teilnehmermenge ausgehen. Unter der Annahme, dass die Sitzung durch die Sitzungsnummer *eindeutig* bestimmt ist, genügt die Key-Confirmation, um Integrität zu erreichen: Jeder Teilnehmer wird nur bei erfolgreicher Key-Confirmation den Schlüssel akzeptieren, d. h. nur wenn alle seine Partner bewiesen haben, dass sie denselben Schlüssel in derselben Sitzung mit denselben beabsichtigten Teilnehmern berechnet haben. Damit ist ausgeschlossen, dass zwei Teilnehmer unterschiedliche Schlüssel akzeptieren.

- **Entity-Authentication** erfordert, dass eine Partei nur dann akzeptiert, wenn alle ihre beabsichtigten Partner auch eine Sitzung mit derselben Sitzungsnummer haben. Durch die erweiterte Bedeutung der Key-Confirmation, wird be-

wiesen, in welcher Sitzung der Schlüssel erhalten wurde. Für eine Partei, die erst dann akzeptiert, wenn alle anderen Teilnehmer einen Schlüssel in derselben Sitzung erhalten haben, muss die Entity-Authentication daher erfüllt sein.

Der Katz-Shin-Compiler erreicht die Sicherheit gegen interne Angreifer im Wesentlichen durch Hinzufügen einer Key-Confirmation, ein Beweis dazu findet sich in [83]. Die folgende Diskussion soll zeigen, dass eine Key-Confirmation nicht nötig ist, um Sicherheit gegen interne Angreifer zu erreichen. Das ermöglicht die Konstruktion effizienterer Protokolle.

4.4.3 Die Sitzungsnummer

Als nächstes soll die Rolle der Sitzungsnummer genauer untersucht werden. Wenn nur passive Angreifer betrachtet werden, benötigen die Benutzer kein langfristiges Geheimnis zur Authentifizierung, wie beispielsweise das Schlüsselpaar für ein Signaturverfahren. Die Benutzer können also durch den Angreifer simuliert werden. Um die Sicherheit eines Schlüsselaustauschprotokolls, das keinen Zustand zwischen verschiedenen Ausführungen speichert, zu beweisen, genügt es dann, eine einzelne Sitzung des Protokolls zu betrachten – der Angreifer kann weitere Ausführungen simulieren.

Sobald jedoch auch aktive Angreifer zugelassen sind, ist für die Authentifizierung ein langfristiges Geheimnis notwendig und der Angreifer ist nicht in der Lage, Benutzer, deren Geheimnisse er nicht kennt, zu simulieren. Daher müssen mehrere Instanzen des Schlüsselaustauschs zugelassen werden, deren Nachrichten der Angreifer vertauschen oder verändern kann. Für das Sicherheitsmodell muss klar sein, welche Instanzen gemeinsam eine Sitzung haben – dies zeigt sich in den Definitionen für Partner und Frische. Ein wichtiges Werkzeug, um die gemeinsame Sitzung zu identifizieren, ist die Sitzungsnummer. Die Sitzungsnummer ist jedoch kein Artefakt des Sicherheitsmodells. Auch für den realen Einsatz der Protokolle werden Sitzungsnummern benötigt, um auf einzelne Sitzungsschlüssel zugreifen zu können.

Leider wird die Sitzungsnummer nicht immer mit der nötigen Sorgfalt behandelt. Wie bereits diskutiert, wird in dem Protokoll $\mathcal{KLL}$ keine Sitzungsnummer angegeben. Auch bei anderen Protokollen stellt sich heraus, dass es sogar unmöglich ist eine konsistente Sitzungsnummer zu generieren. Beispielsweise zeigen Choo u. a. [49], dass es für das Protokoll aus [12] nicht möglich ist, aus den Nachrichten der Benutzer eine Sitzungsnummer zu berechnen. Das führt nicht nur zu einem ungültigen Beweis, sondern erschwert auch den praktischen Einsatz des Protokolls. Choo und Hitchcock [50] zeigen Angriffe auf Zwei-Parteien-Protokolle, die ebenfalls durch eine ungenügende Spezifikation der Sitzungsnummer möglich werden.

Zwei Konstruktionen der Sitzungsnummer werden häufig verwendet. Das ist zum einen eine zuvor vereinbarte Sitzungsnummer, und zum anderen eine Berechnung der Sitzungsnummer als Verkettung aller Nachrichten des Protokolls. Beide Methoden der Sitzungsnummergenerierung haben unterschiedliche Vorteile im Hinblick auf die Sicherheit des Protokolls gegen interne Angreifer:

1. Mit der zuvor vereinbarten Sitzungsnummer ist Entity-Authentication einfach zu verwirklichen: Wenn die Sitzungsnummer in einer signierten Nachricht verwendet wird, ist der Empfänger davon überzeugt, dass der Sender eine Sitzung mit eben dieser Nummer hat. Das ist aber nicht möglich, falls die Sitzungsnummer durch Verkettung aller Nachrichten entsteht. In diesem Fall ist die Sitzungsnummer erst nach dem Protokolldurchlauf definiert und kann folglich nicht in einer Nachricht verwendet werden.

2. Mit der Sitzungsnummer als Verkettung aller Protokollnachrichten kann die Integrität des Schlüsselaustauschs häufig einfach erzielt werden: Es muss gewährleistet sein, dass alle Teilnehmer aus denselben Nachrichten auch denselben Schlüssel berechnen. Dann ist die Menge der Nachrichten gewissermaßen ein Commitment auf den Schlüssel. Das ist in dem Protokoll $\mathcal{KY}$ nicht der Fall, wie der Angriff im Abschnitt 4.2.1 zeigt. Falls jedoch gleiche Nachrichten zu gleichen Sitzungsschlüsseln führen, wie es häufig bei Zwei-Parteien-Protokollen der Fall ist, die Menge der Nachrichten also ein Commitment auf den Sitzungsschlüssel darstellt, ist klar, dass alle Teilnehmer, die diese Nachrichten bekommen, den gleichen Sitzungsschlüssel berechnen. Wenn die Nachrichten zugleich die Sitzungsnummer bilden, dann ist die Integrität des Protokolls erfüllt. Auf diese Art kann die Integrität nicht erzielt werden, falls die Sitzungsnummer schon vorab festgelegt wird. Eine solche Sitzungsnummer kann unmöglich ein Commitment auf einen frischen Schlüssel sein.

Um beides, Entity-Authentication und Integrität, in einfacher Weise zu erfüllen, sollte die Sitzungsnummer also

- schon vor der letzten Runde festliegen, damit in der letzten Runde überprüfbar ist, dass alle Teilnehmer eine identische Sitzungsnummer haben;

- vom Sitzungsschlüssel in einer Weise abhängen, so dass verschiedene Sitzungsschlüssel zu verschiedene Sitzungsnummern führen.

Die folgende Konstruktion der Sitzungsnummer bietet beide oben genannten Eigenschaften und ist möglich, wenn das Protokoll wenigstens zwei Runden hat:

- In der ersten Runde legen alle Teilnehmer die Zufallswerte, die den Sitzungsschlüssel beeinflussen, in einem Commitment fest.

- Die Sitzungsnummer kann dann unmittelbar nach der ersten Runde durch hashen der Verkettung aller Commitments mittels einer kollisionsresistenten Hashfunktion H gebildet werden.

- In der zweiten Runde können die Teilnehmer die Sitzungsnummer in eine signierte Nachricht aufnehmen, um sicherzustellen, dass alle die gleiche Sitzungsnummer erhalten haben.

Auf diese Weise wird erreicht, dass die Sitzungsnummer schon nach der ersten Protokollrunde allen Teilnehmern bekannt ist und zusätzlich mit sehr hoher Wahrscheinlichkeit eindeutig ist. Die so erhaltene Sitzungsnummer ist mit überwältigender Wahrscheinlichkeit eindeutig:

Lemma 4.8. *Mit der oben beschriebenen Konstruktion der Sitzungsnummer werden keine zwei Instanzen eines ehrlichen Protokollteilnehmers U_i, der seinen Zufallswert gleichverteilt zufällig aus einer Menge super-polynomieller Größe zieht, mit signifikanter Wahrscheinlichkeit eine übereinstimmende Sitzungsnummer berechnen.*

Beweis. Die Sitzungsnummer hängt ausschließlich von den Nachrichten der ersten Runde ab. Gegeben sei eine Instanz Π_i^s des Benutzers U_i. Diese Instanz wird einen Zufallswert input_i^s wählen, ein Commitment commit_i^s auf diesen Wert berechnen und eine kollisionsresistente Hashfunktion benutzen, um die Sitzungsnummer zu berechnen. Sei Π_i^t eine weitere Instanz, welche dieselbe Sitzungsnummer basierend auf einem eigenen Commitment commit_i^t berechnet. Dann muss einer der folgenden Fälle eingetreten sein:

- eine Kollision von H ist aufgetreten;
- die Commitments der beiden Instanzen stimmen überein, d. h. $\mathsf{commit}_i^s = \mathsf{commit}_i^t$, jedoch nicht die Zufallswerte, also $\mathsf{input}_i^s \neq \mathsf{input}_i^t$;
- schon die Zufallswerte stimmen überein, d. h. $\mathsf{input}_i^s = \mathsf{input}_i^t$.

Durch die Kollisionsresistenz der Hashfunktion, die Festlegungseigenschaft des Commitmentverfahrens und die gleichverteilte Zufallswahl aus einer super-polynomiell großen Menge ist die Wahrscheinlichkeit aller drei Fälle vernachlässigbar im Sicherheitsparameter. □

4.4.4 Beschreibung des Rahmenwerks

Es stellt sich die Frage, wie das Rahmenwerk für die Konstruktion von Zwei-Runden-Protokollen genutzt werden kann. Es sei ein Zwei-Runden-Protokoll $\mathcal{P}$ gegeben, das sicher gegen passive Angreifer ist. Daraus soll das Protokoll $\mathcal{P}_F$ gebildet werden, das sicher gegen aktive Angreifer und unehrliche Teilnehmer ist. Dazu wird die Sitzungsnummer wie oben beschrieben konstruiert. Zudem müssen alle Protokollteilnehmer sich in Runde 1 auf ihre Zufallseingaben festlegen, indem sie ein Commitment senden. In der zweiten Runde muss bewiesen werden, dass alle Nachrichten des ursprünglichen Protokolls $\mathcal{P}$ ehrlich bezüglich des Commitments berechnet wurden. Das sollte im Allgemeinen mit einem Zero-Knowledge-Beweis geschehen, jedoch mag es in Einzelfällen auch möglich sein, das Commitment gegenüber den restlichen Teilnehmern aufzudecken. Alle Nachrichten des Protokolls $\mathcal{P}_F$ werden mit einem EUF-CMA-sicheren Signaturverfahren (siehe Definition 2.2) signiert. Es folgt eine detailliertere Übersicht über das Protokollrahmenwerk für ein passiv sicheres Zwei-Runden-Protokoll $\mathcal{P}$ aus Sicht von Teilnehmer U_i:

1. Es werden alle Zufallswerte input_i gewählt, die U_i für die Ausführung von $\mathcal{P}$ benötigt.
2. Ein Commitment commit_i auf input_i wird berechnet.
3. Die erste Protokollnachricht P_i^1 des Protokolls $\mathcal{P}$ wird berechnet, wobei die Zufallswerte aus input_i benutzt werden.
4. Eine Signatur σ_i^1 für die Nachricht $M_i^1 = \mathsf{commit}_i \| P_i^1$ wird berechnet.
5. **Broadcast** von $M_i^1 \| \sigma_i^1$.
6. Signaturen eingehender Nachrichten $M_j^1, (j = 1, \ldots, n)$ werden überprüft.
7. Die Sitzungsnummer $\mathsf{sid}_i = H(\mathsf{pid}_i \| \mathsf{commit}_1, \ldots, \mathsf{commit}_n)$ wird berechnet.
8. Die zweite Protokollnachricht P_i^2 des Protokolls $\mathcal{P}$ wird unter Verwendung der eingegangenen Nachrichten berechnet, wobei die Zufallswerte aus input_i benutzt werden.
9. Ein nicht-interaktiver Zero-Knowledge-Beweis proof_i, dass beide Protokollnachrichten P_i^1 und P_i^2 ehrlich unter Verwendung von input_i berechnet wurden, wird erstellt.
10. Eine Signatur σ_i^2 für die Nachricht $M_i^2 = \mathsf{sid}_i \| \mathsf{proof}_i \| P_i^2$ wird berechnet.
11. **Broadcast** von $M_i^2 \| \sigma_i^2$.
12. Signaturen eingehender Nachrichten $M_j^2, (j = 1, \ldots, n)$ werden überprüft.
13. Übereinstimmung der Sitzungsnummern $\mathsf{sid}_i = \mathsf{sid}_j$ und die Korrektheit der Beweise proof_j wird für alle $j = 1, \ldots, n$ überprüft.
14. U_i berechnet den Sitzungsschlüssel wie in $\mathcal{P}$ und akzeptiert.

4.4.5 Sicherheit

Ein Zwei-Runden-Schlüsselaustauschprotokoll $\mathcal{P}$ kann durch Anwendung des Rahmenwerks, wie im letzten Abschnitt beschrieben, so verändert werden, dass ein Zwei-Runden-Protokoll $\mathcal{P}_F$ entsteht, das sicher gegen aktive Angreifer und betrügende Teilnehmer ist. Die Key-Agreement-Eigenschaft wird von dem Rahmenwerk nicht berührt – das Protokoll $\mathcal{P}$ muss also schon ein Key-Agreement sein, damit das Protokoll $\mathcal{P}_F$ ein sicheres Key-Agreement nach Definition 4.7 ist.

Für den Beweis, dass Protokoll $\mathcal{P}_F$ sicher gegen interne Angreifer ist, wird das folgende Lemma benötigt. Das Lemma drückt aus, dass es dem Angreifer nicht mit signifikanter Wahrscheinlichkeit gelingt, in eine aktuelle Protokollausführung eine Nachricht aus einer anderen Protokollausführung einzuschleusen.

Lemma 4.9. *Falls U_i ehrlich ist, dann gibt es mit überwältigender Wahrscheinlichkeit* keine *Instanz $\Pi_j^{s_j}$ mit $U_i \in \mathsf{pid}_j^{s_j}$, die eine Protokollausführung eines Protokolls $\mathcal{P}_F$ akzeptiert, falls nicht beide Nachrichten, die von U_i signiert sind, auch tatsächlich von einer einzigen Instanz $\Pi_i^{s_i}$ des Benutzers U_i generiert wurden.*

Beweis. Falls nicht beide Nachrichten von derselben Instanz $\Pi_i^{s_i}$ gesendet wurden, dann akzeptiert $\Pi_j^{s_j}$ den Protokolldurchlauf nur, falls eine andere Nachricht als Nachricht von $\Pi_i^{s_i}$ aufgefasst wurde, also:

1. Eine Nachricht wurde von einer anderen Instanz von U_i ehrlich generiert und durch den Angreifer vertauscht (ein Replay-Angriff).

2. Eine Nachricht wurde nicht von einer Instanz von U_i generiert.

Um im ersten Fall erfolgreich zu sein, müssen beide U_i-Instanzen dieselbe Sitzungsnummer berechnen. Das wird jedoch durch Lemma 4.8 mit überwältigender Wahrscheinlichkeit ausgeschlossen. Der zweite Fall tritt nur mit vernachlässigbarer Wahrscheinlichkeit auf, da beide Nachrichten eine Signatur tragen und das Signaturverfahren als EUF-CMA-sicher angenommen wurde. □

Mithilfe dieses Lemmas kann bewiesen werden, dass das Protokoll $\mathcal{P}_F$ sicher gegen betrügende Teilnehmer und aktive Angreifer ist. Das wird in den folgenden Propositionen formuliert.

Proposition 4.10. *Ein Protokoll $\mathcal{P}_F$, das wie im Rahmenwerk beschrieben aufgebaut wurde, bietet Entity-Authentication nach Definition 4.5 und Integrität nach Definition 4.4.*

Beweis. Für den Beweis wird ohne Einschränkung ein Schlüsselaustausch zwischen den Benutzern $U_1, \ldots, U_n$ angenommen.
Beweis der Entity-Authentication. Sei Π_i^s eine beliebige Instanz eines ehrlichen Teilnehmers U_i, der eine Protokollsitzung des Protokolls $\mathcal{P}_F$ mit Sitzungsnummer sid_i *akzeptiert* hat. Sei $U_j \in \mathsf{pid}_i$ ein weiterer ehrlicher Teilnehmer, von dem die Instanz Π_i^s glaubt, dass er an der Protokollausführung teilnahm. Lemma 4.9 garantiert, dass Π_i^s mit überwältigender Wahrscheinlichkeit zwei Nachrichten von *einer* U_j-Instanz empfangen hat. Da die Nachricht akzeptiert wurde, muss die Instanz von U_j dieselbe Sitzungsnummer $\mathsf{sid}_j = \mathsf{sid}_i$ erhalten haben, was mit überwältigender Wahrscheinlichkeit auch $\mathsf{pid}_j = \mathsf{pid}_i$ impliziert (sonst trat eine Kollision der Hashfunktion auf).

Beweis der Integrität. Gegeben seien zwei Instanzen $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ von ehrlichen Teilnehmern U_i und U_j, die beide eine Sitzung mit der Sitzungsnummer sid akzeptiert haben. Lemma 4.9 garantiert, dass sie in der zweiten Runde des Protokolls mit überwältigender Wahrscheinlichkeit signierte Nachricht von der jeweils anderen Instanz erhalten haben. Da diese Nachrichten die Sitzungsnummer enthalten, wissen beide Instanzen, dass sie an derselben Sitzung teilnahmen $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j} = \mathsf{sid}$.

Die Konstruktion von sid garantiert, wie schon zuvor argumentiert wurde, dass beide Instanzen über die Menge der beabsichtigten Teilnehmer pid übereinstimmen, und, sofern keine Kollision der Hashfunktion H gefunden wurde, müssen sie dieselben Werte commit_k für jeden Teilnehmer $U_k \in \mathsf{pid}$ berechnet haben. Beide können prüfen, dass alle Werte, die für die Berechnung des Sitzungsschlüssels benötigt werden, mit überwältigender Wahrscheinlichkeit ehrlich auf Basis der Commitments $\mathsf{commit}_1, \ldots, \mathsf{commit}_n$ berechnet wurden. Es folgt, dass sie mit überwältigender Wahrscheinlichkeit konsistente Werte für die Berechnung des Sitzungsschlüssels verwenden und damit auch denselben Sitzungsschlüssel sk berechnen. □

Bevor die Sicherheit des Rahmenwerks gegen aktive Angreifer bewiesen werden kann, muss geklärt werden, was die passive Sicherheit eines Protokolls bedeutet. Ein passiver Angreifer verändert, vertauscht oder löscht keine Nachrichten – alle Nachrichten zwischen den Protokollteilnehmern werden ausgeliefert. Ein solcher Angreifer kann nur ehrliche Protokolldurchläufe erzeugen und eine Protokollmitschrift davon erhalten. Es wurde schon begründet, dass eine Instanz pro Benutzer genügt, um Sicherheit gegen passive Angreifer zu zeigen, da der Angreifer alles außer der Test-Anfrage selber simulieren kann. Ein Protokoll ist passiv sicher, falls der Angreifer eine Protokollmitschrift aller Nachrichten einer ehrlichen Ausführung des Protokolls bekommt, und nicht in der Lage ist, den in dieser Ausführung berechneten Sitzungsschlüssel von einem zufällig gewählten Element aus dem Schlüsselraum zu unterscheiden.

Proposition 4.11. *Ein Protokoll $\mathcal{P}_F$, das wie im Rahmenwerk beschrieben aufgebaut wurde, ist für eine konstante Anzahl von Benutzern sicher nach Definition 4.3.*

Beweis. Gegeben sei ein Angreifer $\mathcal{A}$, der im Modell aus Abschnitt 4.1.1 für das Protokoll $\mathcal{P}_F$ das Bit des Test-Aufrufs signifikant häufiger als durch zufälliges Raten vorhersagt. Aus $\mathcal{A}$ kann ein Unterscheider D konstruiert werden, der die passive Sicherheit des Protokolls $\mathcal{P}$ bricht. D simuliert die Instanzen und Anfragen für $\mathcal{A}$. Dazu erzeugt D zunächst eine Protokollmitschrift, den er in einem geeigneten Moment dem Angreifer $\mathcal{A}$ vorspielen wird.

Sei $\Pi_i^{s_i}$ die Instanz, die in dem Test-Aufruf des Angreifers spezifiziert wurde. Die Instanz muss die Sitzung akzeptiert haben. Frische kann erhalten bleiben, wenn ein Partner korrumpiert wird, jedoch wird die Instanz $\Pi_i^{s_i}$ des Test-Aufrufs nur dann frisch sein, falls $\Pi_i^{s_i}$ akzeptiert und kein beabsichtigter Partner $U_j \in \mathsf{pid}_i$ eine Nachricht durch $\mathsf{Send}(U_j, \cdot)$ erhalten hat, nachdem ein Teilnehmer $U_k \in \mathsf{pid}_i$ durch $\mathsf{Corrupt}(U_k)$ korrumpiert wurde. Also müssen alle angeblichen Absender der Nachrichten zu dem Zeitpunkt der Nachrichtenauslieferung an Partnerinstanzen von $\Pi_i^{s_i}$ unkorrumpiert sein. Lemma 4.9 garantiert dann, dass es mit überwältigender Wahrscheinlichkeit für alle Partner $U_j \in \mathsf{pid}_i$ jeweils eine Instanz $\Pi_j^{s_j}$ gibt, von denen alle Nachrichten, die $\Pi_i^{s_i}$ empfangen hat, stammen. Es ist $\mathcal{A}$ also nur mit vernachlässigbarer Wahrscheinlichkeit möglich, vom Protokoll abzuweichen.

D wählt m Indizes s_1 bis s_m jeweils aus der Menge $\{1, \ldots, q_s\}$. Anschließend wählt D zufällig $\tau \in \{1, \ldots, n\}$ und produziert eine Protokollmitschrift einer Protokollaus-

führung mit τ Benutzern. In einer Sitzung, die eine Teilmenge von τ der Instanzen $\Pi_i^{s_i}$, $(i = 1, \ldots, n)$ bilden, verwendet D die Protokollmitschrift, in der Hoffnung, dass diese Sitzung die Test-Sitzung sein wird. Mit einer Wahrscheinlichkeit von $1/n$ wird $\mathcal{A}$ genau τ Teilnehmer in seiner Test-Sitzung verwenden und mit einer Wahrscheinlichkeit von $q_s^\tau \leq q_s^m$ wird er genau die vorhergesagten Instanzen dazu verwenden. In diesem Fall liefert die Antwort des Angreifers $\mathcal{A}$ auch die Antwort, die der Unterscheider D für das passive Modell benötigt. Für ein konstantes m ist q_s^m vernachlässigbar, daher ist das Protokoll $\mathcal{P}_F$ für eine konstante Menge von Benutzern sicher. □

Mittels des Rahmenwerks kann also aus einem Schlüsselaustauschprotokoll, das sicher gegen passive Angreifer ist, ein Protokoll erzeugt werden, das sicher gegen aktive Angreifer und unehrliche Teilnehmer ist. Falls das passiv sichere Protokoll schon ein Key-Agreement ist, ist das resultierende Protokoll nach Definition 4.7 ein sicheres Key-Agreement, da die Berechnung des Sitzungsschlüssels nicht verändert wird.

4.5 Anwendungen des Rahmenwerks

Die Konstruktion des Beweises ehrlicher Protokollausführung in Schritt 9 des Rahmenwerks wird im Allgemeinen nicht effizient sein. Aber für spezielle Berechnungen lässt sich ein solcher Beweis effizient erbringen. Darauf sollte schon bei der Konstruktion des passiv sicheren Protokolls geachtet werden.

Im Protokoll von Tseng (siehe Bild 4.3) legt die erste Protokollnachricht $M_i^1 = y_i = g^{x_i}$ den Zufallswert x_i fest. Hier kann daher von einem Commitmentverfahren abgesehen werden. In Runde 2 konstruiert das Protokoll einen Beweis ehrlicher Berechnung der Protokollnachricht M_i^2, um die Sicherheit gegen betrügende Teilnehmer zu erreichen, die schon in [121] bewiesen wurde. Dem Protokoll fehlt jedoch die Definition verschiedener Sitzungen mittels der Sitzungsnummer, wie es im Rahmenwerk vorgeschlagen wird. Wenn diese Sitzungsnummerbehandlung hinzugefügt wird, so entsteht ein Zwei-Runden-Protokoll, das sicher gegen betrügende Teilnehmer und aktive Angreifer in einem asynchronen Punkt-zu-Punkt-Netzwerk ist. Das Protokoll hat die Zusatzeigenschaft, dass ein betrügender Teilnehmer identifiziert werden kann – sofern der Angreifer alle Nachrichten ausliefert – eine nicht ausgelieferte Nachricht ist von einer nicht gesendeten Nachricht natürlich nicht zu unterscheiden. Auf diese Eigenschaft „*cheater-identification*“ soll in dieser Arbeit aber nicht weiter eingegangen werden (vgl. dazu auch Bohli, Röhrich und Müller-Quade [21]).

Der nächste Abschnitt soll ein besonders effizientes Protokoll vorstellen, das im Random-Oracle-Modell als sicher bewiesen wird und ebenfalls nach dem Konstruktionsprinzips des Rahmenwerks aufgebaut ist.

4.6 Ein effizientes sicheres Protokoll

Wie schon in Abschnitt 4.2.2 gezeigt wurde, erfüllt das Protokoll $\mathcal{KLL}$ die Sicherheitsanforderungen gegen interne Angreifer nicht. Ausgehend von dem Rahmenwerk aus Abschnitt 4.4 wird ein sicheres Protokoll basierend auf $\mathcal{KLL}$ konstruiert. Das Protokoll wird nicht exakt ins Rahmenwerk passen, sondern, aufgrund der speziellen Struktur, gegenüber einem durch das Rahmenwerk erhaltenen Protokoll vereinfacht sein. Es soll weiterhin effizient sein und weiterhin nur zwei Runden benötigen. In dem Protokoll $\mathcal{KLL}$ trägt jeder Teilnehmer U_i eine Zufallszahl k_i zur Berechnung des Sitzungsschlüssels als $\mathsf{sk} = H(k_1, \ldots, k_n)$ bei. Das Rahmenwerk verlangt in der ersten Runde ein Commitment auf die Beiträge zur Schlüsselberechnung. Im Protokoll $\mathcal{KLL}$ werden die Beiträge $k_1, \ldots, k_{n-1}$ ohnehin im Klartext versendet. Dann kann das auch schon in der ersten Runde geschehen, so dass für diese Werte kein Commitment benötigt wird. Ein ausgezeichneter Teilnehmer U_n sendet seinen Beitrag k_n verschlüsselt und in Runde 1 den Hashwert $H(k_n)$, um k_n festzulegen. Von einem weiteren Commitment kann auch für den Teilnehmer U_n abgesehen werden. Die Sitzungsnummer berechnet sich nach dem Rahmenwerk dann als $H(\mathsf{pid}\|k_1\|\ldots\|k_{n-1}\|H(k_n))$. In Runde 2 muss bewiesen werden, dass tatsächlich der Zufall in den Commitments in die Schlüsselberechnung eingebracht wurde. Da nur die Zufallszahl k_n von U_n in der ersten Runde noch unbekannt war, im Verlauf des Protokolls aber ohnehin allen Teilnehmern bekannt wird, ist hier weder ein Beweis noch ein Aufdecken des Commitments nötig. Jeder Teilnehmer kann $H(k_n)$ berechnen und es mit dem in der ersten Runde von U_n gesendeten Wert vergleichen. Abweichend von der allgemeinen Beschreibung des Rahmenwerks wird kein Beweis benötigt, dass die Berechnung bezüglich der Commitments korrekt war.

Das Protokoll $\mathcal{KLL}$ wie auch die hier beschriebene Variante ist ein Key-Agreement wie es in dieser Arbeit definiert wurde. Das Protokoll kann aber auch einfach so umgewandelt werden, dass der Angreifer, bzw. eine Teilmenge von betrügenden Teilnehmern, nicht ein einzelnes Bit des Sitzungsschlüssel vorhersagen kann („perfektes“ Key-Agreement, vgl. [93]). Dazu müssen alle Teilnehmer in Runde 1 statt ihrer k_i-Werte genau wie U_n einen Hashwert $H(k_i)$ als Commitment senden und erst in der zweiten Runde k_i als Klartext verschicken. Die Verschlüsselung des Wertes ist weiterhin nur für einen Teilnehmer U_n nötig.

Das Protokoll, das sich als eine Variante von $\mathcal{KLL}$ ergibt, ist schematisch in Bild 4.4 dargestellt. Das Protokoll ist gemäß Definition 4.7 sicher. Aufgrund einiger Abweichungen von dem vorgestellten Rahmenwerk und wegen des unvollständigen Sicherheitsbeweises des Originalprotokolls in [85], findet sich im folgenden Abschnitt ein detaillierter Sicherheitsbeweis für das Protokoll. Das Protokoll ist sogar für eine polynomielle Anzahl potentieller Teilnehmer sicher, weil die DDH-Annahme nur zwei Parteien involviert.

	Teilnehmer U_i, $i \neq n$	Teilnehmer U_n
Runde 1:		
Vorberechnung	$k_i \xleftarrow{R} \{0,1\}^\ell$, $x_i \xleftarrow{R} \mathbb{Z}_q$, $y_i = g^{x_i}$	
	$M_i^1 = k_i \| y_i$	$M_n^1 = H(k_n) \| y_n$
	$\sigma_i^1 = Sig_{SK_i}(M_i^1)$	
Broadcast	$(M_i^1 \| \sigma_i^1)$	
Nachberechnung	Verifiziere σ_j^1 $(j = 1, \ldots, n)$	
Runde 2:		
Vorberechnung	$t_i^L = H(y_{i-1}^{x_i})$, $t_i^R = H(y_{i+1}^{x_i})$, $T_i = t_i^L \oplus t_i^R$	
	$\mathsf{sid}_i = H(\mathsf{pid}_i \| k_1 \| \ldots \| k_{n-1} \| H(k_n))$	
	$M_i^2 = \mathsf{sid}_i \| T_i$	$M_n^2 = k_n \oplus t_n^R \| \mathsf{sid}_n \| T_n$
	$\sigma_i^2 = Sig_{SK_i}(M_i^2)$	
Broadcast	$(M_i^2 \| \sigma_i^2)$	
Nachberechnung	Verifiziere σ_j^2 $(j = 1, \ldots, n)$, $T_1 \oplus \cdots \oplus T_n = 0$,	
	$\mathsf{sid}_i = \mathsf{sid}_j$ $(j = 1, \ldots, n)$	
	„entschlüssele“ k_n, vergleiche $H(k_n)$ mit M_n^1	
Sitzungsschlüssel	$\mathsf{sk}_i = H(\mathsf{pid}_i \| k_1 \| \ldots \| k_n)$	

Bild 4.4: Eine sichere Protokollvariante von $\mathcal{KLL}$.

Proposition 4.12. *Falls die CDH-Annahme für $(\mathbb{G}, g)$ gilt und $H(\cdot)$ als Random-Oracle modelliert wird, dann ist das Protokoll aus Bild 4.4 ein sicheres Gruppenschlüsselaustauschprotokoll im Sinne der Definition 4.7.*

Beweis. Seien m die Anzahl der Benutzer und q_s und q_{ro} die polynomiellen Schranken für die Anzahl der Send- bzw. Random-Oracle-Aufrufe des Angreifers. Der Beweis beginnt mit der Definition dreier Ereignisse, die im Laufe des Beweises mehrmals auftreten werden. Für die Eintrittswahrscheinlichkeiten der Ereignisse werden obere Schranken angegeben.

Forge bezeichne das Ereignis, dass der Angreifer erfolgreich eine authentifizierte Nachricht $M_i||\sigma_i$ für einen Benutzer U_i fälscht, ohne dass zuvor U_i durch $\mathsf{Corrupt}(U_i)$ korrumpiert wurde oder eine Instanz von U_i die Nachricht M_i ausgegeben hat. Ein Angreifer $\mathcal{A}$, der Forge herbeiführen kann, kann benutzt werden um eine Signatur für einen gegebenen Schlüssel zu fälschen, also EUF-CMA zu brechen. Es kann ein Angreifer D konstruiert werden, der zunächst einen Challenge-Schlüssel des EUF-CMA-Experiments einem der m potentiellen Protokollteilnehmer zuweist und dann gegenüber $\mathcal{A}$ die Instanzen und Anfragen des Schlüsselaustauschmodells simuliert. Um Nachrichten für den Benutzer, dem der Challenge-Schlüssel zugeordnet wurde, zu erhalten, benutzt D das Signaturorakel. Falls $\mathcal{A}$ erfolgreich eine signierte Nachricht für diesen Benutzer fälscht, kann

D die Eigenschaft EUF-CMA des benutzten Signaturverfahrens brechen, also $\mathsf{Succ}_S^{\mathrm{cma}} \geq \frac{1}{m} \cdot \Pr(\mathsf{Forge})$. Für die Wahrscheinlichkeit von Forge ergibt sich dann:

$$\Pr(\mathsf{Forge}) \leq m \cdot \mathsf{Succ}_S^{\mathrm{cma}}.$$

Weil diese Wahrscheinlichkeit $\mathsf{Succ}_S^{\mathrm{cma}}$, dass D das Experiment aus Definition 2.2 gewinnt, bei einem EUF-CMA-sicheren Signaturverfahren vernachlässigbar sein muss, kann auch das Ereignis Forge nur mit vernachlässigbarer Wahrscheinlichkeit auftreten.

Collision bezeichne das Ereignis, dass das Random-Oracle eine Kollision ausgibt. Ein Send-Aufruf führt maximal zu drei Aufrufen des Random-Oracle. Die Gesamtzahl der Random-Oracle-Hashwerte ist dann durch $3q_s + q_{ro}$ beschränkt und die Wahrscheinlichkeit, dass unter den Aufrufen eine Kollision auftritt, ist beschränkt durch

$$\Pr(\mathsf{Collision}) \leq \frac{(3q_s + q_{ro})^2}{2^\ell},$$

was vernachlässigbar in ℓ ist.

Repeat bezeichne schließlich das Ereignis, dass ein ehrlicher Teilnehmer eine Zufallszahl k_i auswählt, die zuvor schon bei einer beliebigen anderen Instanz ausgewählt wurde. Es kann in der Simulation höchstens q_s benutzte Instanzen geben, weil ein Send-Aufruf des Angreifers benötigt wird, um eine Instanz zu initialisieren. Die benutzten Instanzen können eine Zufallszahl k_i gewählt haben und die Wahrscheinlichkeit, dass für eine gleichverteilt gezogene Zufallszahl das Ereignis Repeat eintritt, ist folgendermaßen beschränkt:

$$\Pr(\mathsf{Repeat}) \leq \frac{q_s^2}{2^\ell}.$$

Diese Wahrscheinlichkeit ist ebenfalls vernachlässigbar in ℓ.

Die drei Ereignisse Collision, Repeat und Forge treten also nur mit vernachlässigbarer Wahrscheinlichkeit in ℓ ein.

Beweis der Sicherheit bei ehrlichen Teilnehmern. Um die Sicherheit im Sinne der Definition 4.3 zu beweisen, wird eine Folge von Spielen (Spiel 0 bis Spiel 3) betrachtet. In diesen Spielen muss der Angreifer $\mathcal{A}$ mit einem Simulator interagieren. Der Simulator simuliert die Instanzen und beantwortet die Anfragen des Angreifers in Spiel 0 genau wie von dem Modell vorgeschrieben. In den folgenden Spielen wird das Verhalten des Simulators sich in kleinen Schritten ändern, ohne die Erfolgswahrscheinlichkeit des Angreifers signifikant zu verändern. Aus der Erfolgswahrscheinlichkeit des Angreifers im letzten Spiel und den Differenzen zwischen den Spielen lässt sich schließlich der Vorteil $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{P}}$ des Angreifers beim Angreifen des Protokolls beschränken.

Spiel 0: In diesem Spiel werden die Instanzen der Protokollteilnehmer für den Angreifer ehrlich simuliert. Für den Angreifer ist die Situation mit dem realen Modell identisch, also

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 0} = \mathsf{Adv}_{\mathcal{A}}^{\mathcal{P}}.$$

Spiel 1: Dieses Spiel wird abgebrochen, falls eines der Ereignisse Forge, Collision oder Repeat eintritt. Andernfalls verhält sich das Spiel identisch zu Spiel 0. Der Angreifer stellt den Unterschied nur fest, falls eines der Ereignisse eintritt, also gilt für den Vorteil des Angreifers

$$|\mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 1} - \mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 0}| \leq \Pr(\mathsf{Forge}) + \Pr(\mathsf{Collision}) + \Pr(\mathsf{Repeat}).$$

Spiel 2: Dieses Spiel unterscheidet sich von Spiel 1 in der Antwort des Simulators auf eine Send-Anfrage in Runde 2. Wenn der Simulator eine Nachricht für eine Instanz $\Pi_i^{s_i}$ berechnen und ausgeben muss, überprüft er, ob alle beabsichtigten Partner $U_k \in \mathsf{pid}_i^{s_i}$ von U_i unkorrumpiert sind. Sofern nicht schon durch Bearbeiten einer benachbarten Instanz geschehen, wählt der Simulator in diesem Fall $t_i^L = t_{i-1}^R$ und $t_i^R = t_{i+1}^L$ jeweils gleichverteilt zufällig aus $\{0,1\}^\ell$, anstatt das Random-Oracle zu befragen. Um konsistent zu sein, muss der Simulator dieselben Werte auch für die benachbarten Instanzen benutzen.

Ein Angreifer, der zwischen Spiel 1 und Spiel 2 unterscheiden kann, kann als Orakel benutzt werden, um eine CDH-Instanz zu lösen. Dazu werden zwei Instanzen $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ zufällig ausgewählt, indem zunächst zwei unterschiedliche Benutzer $U_i, U_j \in \mathcal{U}$ gleichverteilt zufällig gewählt werden und anschließend zwei ganze Zahlen $s_i, s_j \in \{1, \ldots, q_s\}$. Spiel 2 unterscheidet sich von Spiel 1 nur dann, wenn wenigstens eine Sitzung nur aus unkorrumpierten Teilnehmern zusammengesetzt ist. Aufgrund der Random-Oracle-Annahme kann der Angreifer den Unterschied nur entdecken, falls er für ein Paar benachbarter Instanzen das Random-Oracle nach dem Hashwert des Diffie-Hellman-Schlüssels befragt. Die Wahrscheinlichkeit, dass das die beiden zufällig ausgewählten Instanzen sind, ist mindestens $1/(m \cdot q_s)^2$.

Eine gegebene CDH-Instanz (g^a, g^b) wird den ausgewählten Instanzen $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ zugeordnet, so dass diese Instanzen g^a bzw. g^b als ihre Nachricht in der ersten Runde benutzen werden. Nach der ersten Runde werden alle Instanzen über die pid übereinstimmen, da die pid bei der Signaturberechnung mit einbezogen wurde. Der Simulator bricht den Versuch ab, falls der Angreifer

- U_i oder U_j korrumpiert,
- $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ nicht in einer Sitzung benachbart sind,
- oder ein korrumpierter Teilnehmer an der Sitzung mit $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ teilnimmt, d. h. ein Benutzer $U_k \in \mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ wurde korrumpiert.

In diesen Fällen kommen $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ ohnehin nicht in Frage die benachbarten Teilnehmer zu sein, anhand derer der Angreifer die Spiele unterscheidet.

Der Simulator wählt außerdem noch eine Zahl $z \in \{1, \ldots, q_{ro}\}$ gleichverteilt zufällig, in der Hoffnung, dass der Angreifer den Diffie-Hellman-Schlüssel $y_{i-1}^{x_i} = y_i^{x_{i-1}}$ zwischen den beiden Instanzen in der $z$ten Anfrage an das Random-Oracle benutzt. Der Wert aus der $z$ten Anfrage wird daher als Antwort auf die CDH-Challenge gegeben. Die Antwort ist korrekt, falls $\mathcal{A}$ die ausgewählten Instanzen zur Unterscheidung verwendet und auch der Index z richtig geraten wurde. Also:

$$|\mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 2} - \mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 1}| \leq \mathsf{Succ}_{(\mathbb{G},g)}^{\text{CDH}} \cdot q_{ro} \cdot m^2 \cdot q_s^2.$$

Unter der CDH-Annahme ist die Wahrscheinlichkeit $\mathsf{Succ}_{(\mathbb{G},g)}^{CDH}$, die CDH-Challenge richtig zu beantworten, vernachlässigbar. Damit ist auch die Schranke für die Erfolgswahrscheinlichkeit des Angreifers, die Spiele zu unterscheiden, vernachlässigbar.

Spiel 3: In diesem Spiel ändert der Simulator die Berechnung des Sitzungsschlüssels. Nachdem eine Instanz $\Pi_i^{s_i}$ alle Nachrichten der zweiten Runde empfangen hat, prüft der Simulator, ob alle beabsichtigten Partner $U_j \in \mathsf{pid}_i^{s_i}$ unkorrumpiert sind. Falls das der Fall ist, wählt der Simulator den Sitzungsschlüssel $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ gleichverteilt zufällig, anstatt das Random-Oracle zu befragen. Um die Konsistenz zu wahren, muss der Schlüssel auch allen Partner-Instanzen zugeordnet werden.

Der Angreifer kann diesen Unterschied nur feststellen, falls er an das Random-Oracle die Anfrage nach dem Hashwert $H(\mathsf{pid}_i^{s_i} \| k_1 \| \ldots \| k_n)$ stellt. Jedoch ist dem Angreifer keine Information über k_n bekannt. Es gibt 2^ℓ mögliche Werte für k_n, dem Angreifer stehen aber höchstens q_{ro} Anfragen an das Random-Oracle zur Verfügung. Das resultiert in

$$|\mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 3} - \mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 2}| \leq \frac{q_{ro}}{2^\ell}.$$

In der Sitzung mit der Instanz, die der Angreifer für seine Test-Anfrage ausgewählt hat, darf aufgrund der Frische-Bedingung (s. Definition 4.3) kein Teilnehmer korrumpiert sein und der Schlüssel darf auch nicht durch Reveal aufgedeckt worden sein. Die Test-Sitzung wurde also von der Modifikation in Spiel 3 erfasst und alle Instanzen haben einen Zufallswert als Sitzungsschlüssel, den der Angreifer natürlich nicht von einem Zufallswert der Test-Anfrage unterscheiden kann. Die Erfolgswahrscheinlichkeit des Angreifers in Spiel 3 das Bit der Test-Anfrage richtig zu raten, ist also $\frac{1}{2}$. Das ergibt den Vorteil des Angreifers $\mathsf{Adv}_{\mathcal{A}}^{\text{Spiel } 3} = 0$. Wenn nun alle Differenzen zwischen den Spielen berücksichtigt werden, ergibt sich ein vernachlässigbarer Vorteil des Angreifers im realen Modell:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{P}} \leq \Pr(\mathsf{Forge}) + \Pr(\mathsf{Collision}) + \Pr(\mathsf{Repeat}) + \mathsf{Succ}_{(\mathbb{G},g)}^{\text{CDH}} \cdot q_{ro} \cdot m^2 \cdot q_s^2 + \frac{q_{ro}}{2^\ell}$$

Beweis der Integrität. Bezeichne NoIntegrity das Ereignis, dass einige Instanzen die Integritätsbedingung von Definition 4.4 verletzen. Um die Wahrscheinlichkeit des Eintretens von NoIntegrity zu bestimmen, seien U_i und U_j zwei ehrliche Teilnehmer einer Protokollausführung, bei der ihre Instanzen $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ mit einer gemeinsamen Sitzungsnummer $\mathsf{sid} := \mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ akzeptiert ($\mathsf{acc} = \mathsf{true}$) haben. Die Sitzungsnummer sid ist eindeutig, falls wenigstens ein unkorrumpierter Teilnehmer eine frische Zufallszahl k_i beigetragen hat (sonst trat Repeat ein) und keine Kollision der als Random-Oracle modellierten Hashfunktion auftrat (sonst trat Collision ein). Darüber hinaus kann die Nachricht dieses unkorrumpierten Teilnehmers nicht vom Angreifer gefälscht worden sein (sonst trat Forge ein). Daher müssen die Instanzen $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ die Nachricht $\mathsf{sid}_i^{s_i} \| T_i \| \sigma_i^2$ bzw. $\mathsf{sid}_j^{s_j} \| T_j \| \sigma_j^2$ voneinander empfangen haben, wobei aufgrund des Tests in der Nachberechnung notwendigerweise $\mathsf{sid} := \mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ übereinstimmen.

Die Konstruktion von sid garantiert, dass $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ den Schlüsselaustausch mit derselben Benutzergruppe $\mathsf{pid} := \mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ planen und dass sie dieselben Zufallszahlen $k_1, \ldots, k_{n-1}$ und $H(k_n)$ kennen. Solange nicht Repeat eintritt, akzeptieren $\Pi_i^{s_i}$ und $\Pi_j^{s_j}$ nur, wenn sie dieselbe Zufallszahl k_n erhalten. In dem Fall werden beide Instanzen denselben Sitzungsschlüssel $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j}$ berechnen. Zusammengenommen ist eine obere Schranke für die Wahrscheinlichkeit

$$\Pr(\mathsf{NoIntegrity}) \leq \Pr(\mathsf{Collision}) + \Pr(\mathsf{Repeat}) + \Pr(\mathsf{Forge}).$$

Beweis der Entity-Authentication. Das Ereignis, dass Entity-Authentication gemäß Definition 4.5 verletzt ist, werde durch EntAuthFail bezeichnet. Es muss die Wahrscheinlichkeit, dass EntAuthFail eintritt, berechnet werden. Sei U_i ein ehrlicher Benutzer, dessen Instanz $\Pi_i^{s_i}$ eine Protokollausführung akzeptiert hat. Sofern keines der Ereignisse Collision, Repeat und Forge eintrat ist Entity-Authentication für $\Pi_i^{s_i}$ gegeben: $\Pi_i^{s_i}$ muss in Runde 2 Nachrichten von allen Teilnehmern $U \in \mathsf{pid}_i^{s_i}$ erhalten haben, in denen die Sitzungsnummer sid enthalten ist (sonst ist das Ereignis Forge eingetreten). Wie oben kann argumentiert werden, dass die Sitzungsnummer eindeutig ist, sofern keines der Ereignisse Collision, Repeat und Forge auftritt. Also müssen ehrliche Teilnehmer auch über die beabsichtigte Partnermenge $\mathsf{pid}_i^{s_i}$ übereinstimmen. Entity-Authentication ist also höchstens mit der Wahrscheinlichkeit

$$\Pr(\mathsf{EntAuthFail}) \leq \Pr(\mathsf{Collision}) + \Pr(\mathsf{Repeat}) + \Pr(\mathsf{Forge})$$

verletzt.

Beweis der Key-Agreement-Eigenschaft. Die Werte, die relevant sind, um den Sitzungsschlüssel zu berechnen, sind die Zufallszahlen k_i der Teilnehmer $U_i, (i = 1, \ldots, n)$. Wenn nicht Repeat eintritt, wählt ein ehrlicher Teilnehmer einen frischen Zufallswert k_i. Der Angreifer lernt k_i erst während des Protokolldurchlaufs, und kann den endgültigen Schlüssel nur durch Anfragen an das Random-Oracle herausfinden, also aus einer Menge von q_{ro} Sitzungsschlüsseln wählen. Eine Antwort des Random-Oracle

ist zufällig über den Schlüsselraum verteilt und liegt daher nur mit vernachlässigbarer Wahrscheinlichkeit in der vom Angreifer nach Definition 4.6 vor Protokollbeginn festzulegenden Menge κ, die nur einen vernachlässigbaren Teil des Schlüsselraums umfasst. Die Wahrscheinlichkeit des Angreifers, das Experiment aus Definition 4.6 zu gewinnen, ist

$$\Pr(\text{„}\mathsf{sk} \in \kappa\text{“}) = \Pr(\mathsf{Repeat}) + q_{ro} \cdot \mathrm{negl}(\ell),$$

also vernachlässigbar in ℓ. Das Protokoll ist ein Key-Agreement.

Die Korrektheit des Protokolls ergibt sich unmittelbar und damit folgt die Aussage der Proposition. □

4.7 Zusammenfassung und offene Fragen

Dieses Kapitel hat gezeigt, dass die Konstruktion von Gruppenschlüsselaustauschprotokollen qualitativ neue Herausforderungen gegenüber Zwei-Parteien-Schlüsselaustauschprotokollen stellt. Sicherheit gegenüber betrügenden Teilnehmern wurde in den bisher verwendeten Modellen nicht betrachtet und dementsprechend erfüllen etablierte Protokolle keine Sicherheitseigenschaften in dieser Richtung. Bisherige Ansätze zu betrügenden Teilnehmern an einem Schlüsselaustausch gab es nur in einem stark idealisierten Netzwerkmodell. Es wurde gezeigt, dass in diesem Modell bestehende Protokolle sich nicht in das gängige Modell mit einem Punkt-zu-Punkt-Netzwerk übertragen lassen.

Diese Arbeit hat mit der Formalisierung der Eigenschaften „Integrität“, „Entity-Authentication“ und „Key-Agreement“ Werkzeuge vorgestellt, die es erlauben, die Sicherheit eines Protokolls bezüglich betrügender Teilnehmer zu analysieren. Die Analyse der Eigenschaft „Key-Confirmation“ und eine genaue Untersuchung der Sitzungsnummer führte schließlich zu einem Rahmenwerk, nach dem sichere und rundeneffiziente Protokolle konstruiert werden können. Es konnte gezeigt werden, dass es nahezu ohne Effizienzverlust möglich ist, ein auch für betrügende Teilnehmer sicheres Gruppenschlüsselaustauschprotokoll zu entwerfen.

Offene Fragen ergeben sich aus der Modellierung der eingeführten Eigenschaften in weiteren Sicherheitsmodellen für Schlüsselaustausch. Auch ein modularer Aufbau von Schlüsselaustauschprotokollen, der über das hier vorgestellte Rahmenwerk hinaus geht, kann das Design von sicheren Group-Key-Agreement-Protokollen weiter vereinfachen.

5. Zusammenfassung

In dieser Arbeit wurden Signaturverfahren und Schlüsselaustauschprotokolle im Hinblick auf unbekannte, von Innentätern ausgehende Angriffe untersucht. Es wurde gezeigt, dass der für Signaturverfahren derzeit verwendete Sicherheitsbegriff EUF-CMA die erforderliche Bindung der Signatur an eine Person im Allgemeinen nicht zufriedenstellend garantieren kann. Dadurch wird letztendlich auch die Nichtabstreitbarkeit von Signaturen in Frage gestellt. Als neuer Angriff, der auf EUF-CMA-sichere Verfahren möglich ist, wurden *Schlüsselersetzungsangriffe* mit unehrlichem Unterzeichner untersucht. Mit unehrlicher Schlüsselerzeugung, unehrlicher Signaturberechnung oder Veröffentlichen des geheimen Schlüssels stehen dem Angreifer zusätzliche Angriffsmethoden zur Verfügung. Für etablierte Signaturverfahren hat das neue Szenario die folgenden Auswirkungen: Die Verfahren EC-DSA und EC-GDSA stellen sich als anfällig gegen diese Schlüsselersetzungsangriffe heraus, während EC-KCDSA sowie auch DSA beweisbar nicht betroffen sind. Es wurde eine verbesserte Parameterwahl vorgeschlagen, die Schlüsselersetzungsangriffe auch bei EC-DSA erschwert. Zusammengefasst müssen beim Einsatz von EUF-CMA-sicheren Signaturverfahren in einer Anwendung die nötigen Anforderungen an die Signatur überprüft werden und gegebenenfalls ein Signaturverfahren mit weiterreichenden Sicherheitsgarantien gewählt werden. Verfahren, die neben EUF-CMA auch sicher gegen Schlüsselersetzungsangriffe sind, reichen aus derzeitiger Sicht für einen Großteil der Anwendungen aus.

Subliminale Kanäle in Signaturverfahren bieten dem Unterzeichner die Möglichkeit, das Signaturverfahren zur verborgenen Kommunikation zu missbrauchen. Erstmals wurden in dieser Arbeit Signaturverfahren mit subliminalem Kanal formal gefasst. Es wurde gezeigt, dass auch in deterministischen Signaturverfahren ein subliminaler Breitbandkanal vorhanden sein kann: In den deterministischen Verfahren ESIGN-D und SFLASH^{v3} wurden neue subliminale Kanäle entdeckt. Auch die Frage nach *subliminalfreien Signaturverfahren* konnte beantwortet werden. Es wurde eine Formalisierung von Subliminalfreiheit gefunden, so dass für Signaturverfahren bewiesen werden kann, dass subliminale Kommunikation durch die Signaturen unmöglich

ist. Für das praktisch eingesetzte Signaturverfahren RSA-PSS konnte gezeigt werden, dass eine deterministische Variante, die beweisbar sicher im Sinne von EUF-CMA ist, subliminalfrei ist. In vielen Fällen muss das Verfahren jedoch nicht in diesem strengen Sinne subliminalfrei sein; es genügt, wenn ein Wächter davon überzeugt ist, dass keine subliminale Kommunikation stattfindet. Für diesen Fall wurde die Verallgemeinerung *subliminalfrei mit Wächter* eingeführt und formal gefasst. Eine in diesem Sinne subliminalfreie Variante des Signaturverfahrens EC-DSA wurde präsentiert.

Ein weiteres Thema der Arbeit waren *Gruppenschlüsselaustauschprotokolle*. Die Konstruktion von Gruppenschlüsselaustauschprotokollen stellt qualitativ neue Herausforderungen gegenüber der Konstruktion von Zwei-Parteien-Schlüsselaustauschprotokollen. Auch die Entwicklung von Modellen für Gruppenschlüsselaustausch ist noch nicht abgeschlossen. Sicherheit gegenüber *unehrlichen Teilnehmern* wurde in den bisher verwendeten Modellen nicht betrachtet und dementsprechend erfüllen etablierte Protokolle keine Sicherheitseigenschaften in dieser Richtung. Diese Arbeit hat mit der Formalisierung der Eigenschaften „Integrität“, „Entity-Authentication“ und „Key-Agreement“ Werkzeuge vorgestellt, die es erlauben, die Sicherheit eines Protokolls bezüglich betrügender Teilnehmer zu analysieren. Es konnte gezeigt werden, dass es nahezu ohne Effizienzverlust möglich ist, ein Gruppenschlüsselaustauschprotokoll zu entwerfen, das zusätzlich zu bisher schon erfüllten Sicherheitseigenschaften auch sicher gegen Angriffe unehrlicher Teilnehmer ist.

6. Ausblick

Dieses Kapitel soll über die schon angesprochenen offenen Fragen hinaus einen Ausblick auf wichtige an diese Arbeit anknüpfende Fragestellungen der Kryptographie geben. Insgesamt zeigte sich, dass je nach Anwendung die Anforderungen an die kryptographischen Bausteine sehr verschiedenartig sein können. Neben den in dieser Arbeit betrachteten Bausteinen *digitale Signatur* und *Schlüsselaustausch* lohnt es sich daher auch für weitere kryptographische Verfahren, die bestehenden Anforderungen und Modelle im Hinblick auf interne Angreifer zu überdenken.

Die Untersuchung der Signaturen zeigte, dass der derzeitige Sicherheitsbegriff EUF-CMA nicht für alle Anwendungen von digitalen Signaturen genügt. Eine Vielzahl von Angriffen wird nicht von EUF-CMA erfasst. Es besteht weiterer Forschungsbedarf, welche Verfahren welche Ziele erreichen können, welche Ziele für welche Anwendungen nötig sind und wie digitale Signaturen umfassend modelliert werden können.

Zum Thema Gruppenschlüsselaustausch besteht ebenfalls weiterer Forschungsbedarf. Dazu konnten in eigenen Arbeiten schon Beiträge geliefert werden. Eine Schnittstelle zu neuen mathematischen Primitiven wird in Bohli, Glas und Steinwandt [17] untersucht. Bohli, González Vasco und Steinwandt [19] präsentieren ein durch ein Passwort geringer Entropie authentifiziertes Protokoll, dessen Sicherheit kein Random-Oracle benötigt. Passworte werden auch in Zukunft eine wichtige Rolle spielen, in einer heterogenen Umgebung können Protokolle interessant werden, bei denen sich ein Teil der Parteien mit Passworten, andere mit Geheimnissen hoher Entropie authentifizieren. In diesem Zusammenhang sind auch Protokolle wichtig, welche die nötige Rechenkomplexität asymmetrisch auf die Teilnehmer verteilen, beispielsweise um die Integration von leistungsschwachen Sensoren zu ermöglichen.

Es stellt sich heraus, dass die Beweise zunehmend komplexer werden. Daher sind automatisierte formale Beweismethoden oder Beweisverifikation und modularer Aufbau von Protokollen in Zukunft sehr wichtig – nicht nur für Schlüsselaustauschprotokolle. Ein erster Beitrag in dieser Richtung ist ein Compiler in Abdalla, Bohli, González Vasco und Steinwandt [1], der ein Zwei-Parteien-Protokoll in ein

Gruppenschlüsselaustauschprotokoll transformieren kann, ohne erneute Authentifizierung zu benötigen. Auch Anonymität und Datenschutz wird in zukünftigen Anwendungen eine wichtige Rolle spielen. Wie eine Abstreitbarkeit der Teilnahme an einem Gruppenschlüsselaustausch realisiert werden kann, wird in BOHLI UND STEINWANDT [24] untersucht, die Umsetzungen weiterer Eigenschaften in dieser Richtung bleibt ebenfalls eine offene Frage. Es kann beispielsweise erwünscht sein, dass einige Teilnehmer an einem Schlüsselaustausch gegenüber anderen anonym bleiben.

A. Definitionen

In der Kryptographie spielen asymptotische Aussagen eine wichtige Rolle. Viele Sicherheitseigenschaften sind asymptotisch in einer natürlichen Zahl definiert, dem sogenannten Sicherheitsparameter ℓ. Durch die ganze Arbeit wird ℓ diesen Sicherheitsparameter bezeichnen. Sicherheitsdefinitionen fordern häufig, dass die Erfolgswahrscheinlichkeit eines Angriffs *vernachlässigbar* im Sicherheitsparameter ℓ ist:

Definition A.1 (Vernachlässigbar)**.** *Eine Funktion $f : \mathbb{N} \to [0,1]$ heißt* vernachlässigbar*, wenn es für jede natürliche Zahl $c \in \mathbb{N}$ ein $\ell_0 \in \mathbb{N}$ gibt, so dass für alle $\ell \geq \ell_0$ gilt, dass $f(\ell) \leq \ell^{-c}$ ist.*

Eine Funktion, die nicht vernachlässigbar ist, wird *signifikant* genannt. Für Aussagen über die Zuverlässigkeit wird gefordert, dass Algorithmen mit *überwältigender* Wahrscheinlichkeit erfolgreich sind. Eine Funktion f wird *überwältigend* genannt, falls Funktion $1 - f$ vernachlässigbar ist.

Die Algorithmen und der Angreifer sind in der Regel PPT-Algorithmen:

Definition A.2 (PPT-Algorithmus)**.** *Ein Algorithmus heißt* PPT-Algorithmus*, falls der Algorithmus probabilistisch ist und durch ein festes Polynom in der Länge seiner Eingaben laufzeitbeschränkt ist.*

Damit ein PPT-Algorithmus laufzeitbeschränkt im Sicherheitsparameter ℓ ist, bekommt der Algorithmus ℓ in unärer Codierung 1^ℓ als Eingabe.

Definition A.3 (Ununterscheidbar)**.** *Zwei Familien $\{X_\ell\}_{\ell\in\mathbb{N}}$ und $\{Y_\ell\}_{\ell\in\mathbb{N}}$ von Wahrscheinlichkeitsverteilungen heißen* ununterscheidbar*, wenn es eine vernachlässigbare Funktion f gibt, so dass für alle PPT-Algorithmen A gilt:*

$$| \Pr[A(X_\ell)] - \Pr[A(Y_\ell)] | \leq f(\ell).$$

In einigen Beweisen werden die folgenden Annahmen verwendet:

Definition A.4. *Die* CDH-Annahme *(„computational Diffie-Hellman assumption“) für eine endliche zyklische Gruppe $\mathbb{G}$ mit Erzeuger g von der Ordnung q besagt, dass jeder PPT-Algorithmus zu einer gegeben CDH-Challenge (g, g^a, g^b) mit gleichverteilt zufällig gewählten $a, b \in \mathbb{Z}_q$ nur mit vernachlässigbarer Wahrscheinlichkeit g^{ab} ausgeben kann.*

Die DDH-Annahme *(„decisional Diffie-Hellman assumption“) besagt, dass die beiden Quadrupel (g, g^a, g^b, g^{ab}) und (g, g^a, g^b, g^c) mit gleichverteilt zufällig gewählten $a, b, c \in \mathbb{Z}_q$ ununterscheidbar sind.*

Neben dem Random-Oracle wird auch die folgende Modellierung einer Hashfunktion verwendet:

Definition A.5 (Universelle Familie von Hashfunktionen)**.** *Eine Familie $\mathcal{H}$ von Funktionen aus einer Menge U in eine Menge V heißt* universelle Familie von Hashfunktionen*, falls für jedes $x, y \in U$, $x \neq y$, die Wahrscheinlichkeit für $h(x) = h(y)$ für eine zufällig aus $\mathcal{H}$ gezogene Funktion h genau $1/|V|$ ist. Die Familie heißt* fast universell*, falls die Wahrscheinlichkeit höchstens $1/|V| + 1/|U|$ ist.*

Literaturverzeichnis

[1] Abdalla, Michel, Jens-Matthias Bohli, María Isabel González Vasco und Rainer Steinwandt: *(Password) Authenticated Key Establishment: From 2-Party to Group*. In: Vadhan, Salil P. (Herausgeber): *Theory of Cryptography Conference – TCC 2007*, Band 4392 der Reihe *Lecture Notes in Computer Science*, Seiten 499–514. Springer, 2007.

[2] Abe, Masayuki und Tatsuaki Okamoto: *A Signature Scheme with Message Recovery as Secure as Discrete Logarithm*. In: Lam, Kwok Yan, Eiji Okamoto und Chaoping Xing (Herausgeber): *Advances in Cryptology – ASIACRYPT'99*, Band 1716 der Reihe *Lecture Notes in Computer Science*, Seiten 378–389. Springer, 1999.

[3] Anderson, Ross, Serge Vaudenay, Bart Preneel und Kaisa Nyberg: *The Newton Channel*. In: Anderson, Ross (Herausgeber): *Information Hiding*, Band 1174 der Reihe *Lecture Notes in Computer Science*, Seiten 151–156. Springer, 1996.

[4] Backes, Michael und Dennis Hofheinz: *How to Break and Repair a Universally Composable Signature Functionality*. In: Zhang, Kan und Yuliang Zheng (Herausgeber): *Information Security – ISC 2004*, Band 3225 der Reihe *Lecture Notes in Computer Science*, Seiten 61–72. Springer, 2004.

[5] Baek, Joonsang und Kwangjo Kim: *Remarks on the Unknown Key-Share Attacks*. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E83-A(12):2766–2769, 2000.

[6] Baier, Harald: *Efficient Algorithms for Generating Elliptic Curves over Finite Fields Suitable for Use in Cryptography*. Doktorarbeit, Technische Universität Darmstadt, 2002.

[7] Bao, Feng und Xinkai Wang: *Steganography of Short Messages through Accessories*. In: *Pacific Rim Workshop on Digital Steganography 2002 (STEG'02)*, 2002.

[8] Bellare, Mihir, Ran Canetti und Hugo Krawczyk: *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols*. In:

Proceedings of the 30th ACM Symposium on Theory of Computing (STOC'98), Seiten 419–428. ACM Press, 1998.

[9] Bellare, Mihir, David Pointcheval und Phillip Rogaway: *Authenticated Key Exchange Secure Against Dictionary Attacks.* In: Preneel, Bart (Herausgeber): *Advances in Cryptology – EUROCRYPT 2000*, Band 1807 der Reihe *Lecture Notes in Computer Science*, Seiten 139–155. Springer, 2000.

[10] Bellare, Mihir und Phillip Rogaway: *Entity Authentication and Key Distribution.* In: Stinson, Douglas R. (Herausgeber): *Advances in Cryptology – CRYPTO '93*, Band 773 der Reihe *Lecture Notes in Computer Science*, Seiten 232–249. Springer, 1993.

[11] Bellare, Mihir und Phillip Rogaway: *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.* In: *ACM Conference on Computer and Communications Security*, Seiten 62–73. ACM Press, 1993.

[12] Bellare, Mihir und Phillip Rogaway: *Provably Secure Session Key Distribution – The Three Party Case.* In: *Proceedings of the 27th ACM Symposium on Theory of Computing (STOC'95)*, Seiten 57–66. ACM Press, 1995.

[13] Bellare, Mihir und Phillip Rogaway: *The Exact Security of Digital Signatures – How to Sign with RSA and Rabin.* In: Maurer, Ueli (Herausgeber): *Advances in Cryptology – EUROCRYPT '96*, Band 1070 der Reihe *Lecture Notes in Computer Science*, Seiten 399–416. Springer, 1996.

[14] Blake-Wilson, Simon und Alfred Menezes: *Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol.* In: Imai, Hideki und Yuliang Zheng (Herausgeber): *International Workshop on Practice and Theory in Public Key Cryptography, PKC'99*, Band 1560 der Reihe *Lecture Notes in Computer Science*, Seiten 154–170. Springer, 1999.

[15] *Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen).* Bundesanzeiger Nr. 58, 2006. `http://www.bundesnetzagentur.de/media/archive/5951.pdf`.

[16] Bohli, Jens-Matthias: *A Framework for Robust Group Key Agreement.* In: Gavrilova, Marina, Osvaldo Gervasi, Vipin Kumar, C. J.Kenneth Tan, David Taniar, Antonio Laganà, Youngsong Mun und Hyunseung Choo (Herausgeber): *ACIS'06 in Computational Science and Its Applications – ICCSA 2006 (3)*, Band 3982 der Reihe *Lecture Notes in Computer Science*, Seiten 355–364. Springer, 2006.

[17] Bohli, Jens-Matthias, Benjamin Glas und Rainer Steinwandt: *Towards Provably Secure Group Key Agreement Building on Group Theory.* In: Nguyen, Phong Q. (Herausgeber): *International Conference on Cryptology*

in Vietnam – VietCrypt 2006, Band 4341 der Reihe *Lecture Notes in Computer Science*, Seiten 322–336. Springer, 2006.

[18] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *Secure Group Key Establishment Revisited.* Cryptology ePrint Archive, Report 2005/395, 2005. `http://eprint.iacr.org/2005/395/`.

[19] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *Password-Authenticated Constant-Round Group Key Establishment with a Common Reference String*, 2006. `http://eprint.iacr.org/2006/214/`.

[20] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *A Subliminal-free Variant of ECDSA.* In: *Information Hiding – IH2006*, Lecture Notes in Computer Science. Springer, erscheint.

[21] Bohli, Jens-Matthias, Jörn Müller-Quade und Stefan Röhrich: *On Group Key Agreement with Cheater Identification.* Western European Workshop on Research in Cryptology, WEWoRC 2005, 2005.

[22] Bohli, Jens-Matthias, Stefan Röhrich und Rainer Steinwandt: *Key substitution attacks revisited: Taking into account malicious signers.* International Journal of Information Security, 5(1):30–36, 2006.

[23] Bohli, Jens-Matthias und Rainer Steinwandt: *On Subliminal Channels in Deterministic Signature Schemes.* In: Park, Choonsik und Seongtaek Chee (Herausgeber): *Information Security and Cryptology – ICISC 2004*, Band 3506 der Reihe *Lecture Notes in Computer Science*, Seiten 182–194. Springer, 2005.

[24] Bohli, Jens-Matthias und Rainer Steinwandt: *Deniable Group Key Agreement.* In: Nguyen, Phong Q. (Herausgeber): *International Conference on Cryptology in Vietnam – VietCrypt 2006*, Band 4341 der Reihe *Lecture Notes in Computer Science*. Springer, 2006.

[25] Boneh, Dan und Xavier Boyen: *Short Signatures Without Random Oracles.* In: Cachin, Christian und Jan Camenisch (Herausgeber): *Advances in Cryptology – EUROCRYPT 2004*, Band 3027 der Reihe *Lecture Notes in Computer Science*, Seiten 56–73. Springer, 2004.

[26] Bosma, Wieb, John Cannon und Catherine Playoust: *The Magma Algebra System I: The User Language.* Journal of Symbolic Computation, 24(3/4):235–265, 1997.

[27] Boudot, Fabrice: *Efficient Proofs that a Committed Number Lies in an Interval.* In: Preneel, Bart (Herausgeber): *Advances in Cryptology – EUROCRYPT 2000*, Band 1807 der Reihe *Lecture Notes in Computer Science*, Seiten 431–444. Springer, 2000.

[28] Boyd, Colin und Juan Manuel González Nieto: *Round-optimal Contributory Conference Key Agreement.* In: Desmedt, Yvo (Herausgeber): *Public Key Cryptography – PKC 2003*, Band 2567 der Reihe *Lecture Notes in Computer Science*, Seiten 161–174. Springer, 2003.

[29] Boyd, Colin und Anish Mathuria: *Protocols for Authentication and Key Establishment.* Springer, 2004.

[30] Bresson, Emmanuel, Olivier Chevassut und David Pointcheval: *Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case.* In: Boyd, Colin (Herausgeber): *Advances in Cryptology – ASIACRYPT 2001*, Band 2248 der Reihe *Lecture Notes in Computer Science*, Seiten 290–309. Springer, 2001.

[31] Bresson, Emmanuel, Olivier Chevassut und David Pointcheval: *Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions.* In: Knudsen, Lars (Herausgeber): *Advances in Cryptology – EUROCRYPT 2002*, Band 2332 der Reihe *Lecture Notes in Computer Science*, Seiten 321–336. Springer, 2002.

[32] Bresson, Emmanuel, Olivier Chevassut, David Pointcheval und Jean-Jacques Quisquater: *Provably Authenticated Group Diffie-Hellman Key Exchange.* In: Samarati, Pierangela (Herausgeber): *Proceedings of the 8th ACM Conference on Computer and Communications Security*, Seiten 255–264. ACM Press, 2001.

[33] Brickell, Ernest, David Pointcheval, Serge Vaudenay und Moti Yung: *Design Validations for Discrete Logarithm Based Signature Schemes.* In: Imai, Hideki und Yuliang Zheng (Herausgeber): *Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000*, Band 1751 der Reihe *Lecture Notes in Computer Science*, Seiten 276–292. Springer, 2000.

[34] Brickell, Ernest F., David Chaum, Ivan B. Damgård und Jeroen van de Graaf: *Gradual and verifiable release of a secret.* In: Pomerance, Carl (Herausgeber): *Advances in Cryptology – CRYPTO '87*, Band 293 der Reihe *Lecture Notes in Computer Science*, Seiten 156–166. Springer, 1988.

[35] Bundesamt für Sicherheit in der Informationstechnik: *Die Lage der IT-Sicherheit in Deutschland 2005.* `http://www.bsi.de/literat/lagebericht/lagebericht2005.pdf`, 2005.

[36] Bundesamt für Sicherheit in der Informationstechnik: *Digitale Sicherheitsmerkmale im elektronischen Reisepass.* `http://www.bsi.de/fachthem/epass/Sicherheitsmerkmale.pdf`, 2005.

[37] Burmester, Mike und Yvo Desmedt: *A Secure and Efficient Conference Key Distribution System.* In: Santis, Alfredo De (Herausgeber): *Advances in Cryptology – EUROCRYPT '94*, Band 950 der Reihe *Lecture Notes in Computer Science*, Seiten 275–286. Springer, 1995.

[38] Burmester, Mike und Yvo Desmedt: *A secure and scalable Group Key Exchange System.* Information Processing Letters, 94:137–143, 2005.

[39] Burmester, Mike, Yvo Desmedt, Toshiya Itoh, Kouichi Sakurai und Hiroko Shizuya: *Divertible and Subliminal-Free Zero-Knowledge Proofs for Languages.* Journal of Cryptology, 12(3):197–223, 1999.

[40] Cachin, Christian und Reto Strobl: *Asynchronous Group Key Exchange with Failures.* In: *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, Seiten 357–366. ACM Press, 2004.

[41] Camenisch, Jan und Markus Michels: *Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes.* In: Stern, Jacques (Herausgeber): *Advances in Cryptology – EUROCRYPT '99*, Band 1592 der Reihe *Lecture Notes in Computer Science*, Seiten 107–122. Springer, 1999.

[42] Camenisch, Jan und Markus Stadler: *Efficient Group Signature Schemes for Large Groups (Extended Abstract).* In: Jr., Burton S. Kaliski (Herausgeber): *Advances in Cryptology – CRYPTO '97*, Band 1294 der Reihe *Lecture Notes in Computer Science*. Springer, 1997.

[43] Camenisch, Jan und Markus Stadler: *Proof Systems for General Statements about Discrete Logarithms.* Technical report TR 260, Department of Computer Science, ETH Zürich, 1997. `ftp://ftp.inf.ethz.ch/pub/crypto/publications/CamSta97b.pdf`.

[44] Canetti, Ran: *Universally Composable Security: A New Paradigm for Cryptographic Protocols.* In: *Proceedings of the 42th Annual Symposium on Foundations of Computer Science (FOCS 2001)*, Seiten 136–145. IEEE Computer Society, 2001. `http://eprint.iacr.org/2000/067/`.

[45] Canetti, Ran: *Universally Composable Signature, Certification, and Authentication.* In: *17th IEEE Computer Security Foundations Workshop (CSFW'04)*, Seiten 219–233. IEEE, 2004.

[46] Canetti, Ran, Oded Goldreich und Shai Halevi: *The Random Oracle Methodology Revisited.* In: *Proceedings of the 30th ACM Symposium on Theory of Computing, STOC 98*, Seiten 209–218. ACM Press, 1998.

[47] Canetti, Ran und Hugo Krawczyk: *Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels.* In: Pfitzmann, Birgit (Herausgeber): *Advances in Cryptology – EUROCRYPT 2001*, Band 2045 der Reihe *Lecture Notes in Computer Science*, Seiten 453–474. Springer, 2001.

[48] Choo, Kim-Kwang Raymond, Colin Boyd und Yvonne Hitchcock: *Examining Indistinguishability-Based Proof Models for Key Establishment Protocols.* In: Roy, Bimal (Herausgeber): *Advances in Cryptology – ASIACRYPT 2005*, Band 3788 der Reihe *Lecture Notes in Computer Science*, Seiten 585–604. Springer, 2005.

[49] Choo, Kim-Kwang Raymond, Colin Boyd, Yvonne Hitchcock und Greg Maitland: *On Session Identifiers in Provably Secure Protocols: The Bellare-Rogaway Three-Party Key Distribution Protocol Revisited.* In: Blundo, Carlo und Stelvio Cimato (Herausgeber): *Security in Communication Networks, SCN 2004*, Band 3352 der Reihe *Lecture Notes in Computer Science*, Seiten 351–366. Springer, 2005.

[50] Choo, Kim-Kwang Raymond und Yvonne Hitchcock: *Security Requirements for Key Establishment Proof Models: Revisiting Bellare-Rogaway and Jeong-Katz-Lee Protocols.* In: Boyd, Colin und Juan M. González Nieto (Herausgeber): *Information Security and Privacy, ACISP 2005*, Band 3574 der Reihe *Lecture Notes in Computer Science*, Seiten 429–442. Springer, 2005.

[51] Courtois, Nicolas, Louis Goubin und Jacques Patarin: *SFLASH^{v3}, a fast asymmetric signature scheme.* Cryptology ePrint Archive, 2003. `http://eprint.iacr.org/2003/211/`.

[52] CSO magazine, U.S. Secret Service und CERT Coordination Center: *2004 E-Crime Watch Survey.* `http://www.cert.org/archive/pdf/2004eCrimeWatchSummary.pdf`, 2004.

[53] Desmedt, Yvo: *Simmons' Protocol is Not Free of Subliminal Channels.* In: *Proceedings of the 9th IEEE Computer Security Foundations Workshop (CSFW '96)*, Seiten 170–175. IEEE Computer Society Press, 1996.

[54] Dierks, Tim und Christopher Allen: *The TLS Protocol Version 1.0*, 1999. `http://ietf.org/rfc/rfc2246.txt`.

[55] Diffie, Whitfield und Martin Hellman: *New Directions in Cryptography.* IEEE Transactions on Information Theory, 22(6):644–652, 1976.

[56] Ding, Jintai und Dieter Schmidt: *Cryptanalysis of SFlash^{v3}.* Cryptology ePrint Archive, 2004. `http://eprint.iacr.org/2004/103/`.

[57] ElGamal, Taher: *A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms.* IEEE Transactions on Information Theory, 31(4):469–472, 1985.

[58] Fiat, Amos und Adi Shamir: *How to Prove Yourself: Practical Solutions to Identification and Signature Problems.* In: Odlyzko, Andrew M. (Herausgeber): *Advances in cryptology – CRYPTO '86*, Band 263 der Reihe *Lecture Notes in Computer Science*, Seiten 186–194. Springer, 1987.

[59] *FIPS PUB 186-2 Digital Signature Standard (DSS) + Change Notice 1 (October 2001)*, Jan 2000. `http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf`.

[60] Fischlin, Marc: *Completely Non-malleable Schemes.* In: Caires, Luís, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi und Moti Yung (Herausgeber): *Automata, Languages and Programming, ICALP 2005*, Band 3580 der Reihe *Lecture Notes in Computer Science*, Seiten 779–790. Springer, 2005.

[61] Fujisaki, Eiichiro, Tetsutaro Kobayashi, Hikaru Morita, Hiroaki Oguro, Tatsuaki Okamoto und Satomi Okazaki.: *ESIGN: Efficient Digital Signature Scheme.* `https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions.html`, 2000.

[62] Fujisaki, Eiichiro und Tatsuaki Okamoto: *Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations.* In: Jr., Burton S. Kaliski (Herausgeber): *Advances in Cryptology – CRYPTO '97*, Band 1294 der Reihe *Lecture Notes in Computer Science*, Seiten 16–30. Springer, 1997.

[63] Gathen, Joachim von zur und Jürgen Gerhard: *Modern Computer Algebra.* Cambridge University Press, 1999.

[64] Geiselmann, Willi und Rainer Steinwandt: *A Key Substitution Attack on SFLASH^{v3}.* Journal of Discrete Mathematical Sciences & Cryptography, 8(2):137–141, 2005.

[65] Goldwasser, Shafi, Silvio Micali und Ronald L. Rivest: *A "paradoxical" solution to the signature problem.* In: *Proceedings of the IEEE 25th Annual Symposium on Foundations of Computer Science*, Seiten 441–448, 1984.

[66] Goldwasser, Shafi, Silvio Micali und Ronald L. Rivest: *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.* SIAM Journal on Computing, 17(2):281–308, 1988.

[67] Goldwasser, Shafi und Rafail Ostrovsky: *Invariant Signatures and Non-interactive Zero-Knowledge Proofs Are Equivalent.* In: Brickell, Ernest F.

(Herausgeber): *Advances in Cryptology – CRYPTO '92*, Band 740 der Reihe *Lecture Notes in Computer Science*, Seiten 228–245. Springer, 1993.

[68] Gollmann, Dieter: *Insider Fraud (Position Paper).* In: Christianson, Bruce, Bruno Crispo, William S. Harbison und Michael Roe (Herausgeber): *Security Protocols, 6th International Workshop*, Band 1550 der Reihe *Lecture Notes in Computer Science*, Seiten 213–219. Springer, 1998.

[69] Granboulan, Louis: *How to repair ESIGN.* In: Cimato, Stelvio, Clemente Galdi und Giuseppe Persiano (Herausgeber): *Security in Communication Networks, SCN 2002*, Band 2576 der Reihe *Lecture Notes in Computer Science*, Seiten 234–240. Springer, 2002.

[70] Güttinger, Michael: *Universell komponierbare Sicherheitsmodellierung digitaler Signaturverfahren*, 2005. Diplomarbeit am Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe.

[71] Harn, Lein und Guang Gong: *Digital signature with a subliminal channel.* IEE Proceedings Computers and Digital Techniques, 144(6):387–389, 1997.

[72] Hoffstein, Jeffrey, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman und William Whyte: *NTRUSign: Digital Signatures Using the NTRU Lattice.* In: Joye, Marc (Herausgeber): *Topics in Cryptology – CT-RSA 2003*, Band 2612 der Reihe *Lecture Notes in Computer Science*, Seiten 122–140. Springer, 2003.

[73] Hofheinz, Dennis, Jörn Müller-Quade und Rainer Steinwandt: *Initiator-Resilient Universally Composable Key Exchange.* In: Snekkenes, Einar und Dieter Gollmann (Herausgeber): *Computer Security – ESORICS 2003*, Band 2808 der Reihe *Lecture Notes in Computer Science*, Seiten 61–84. Springer, 2003.

[74] *IEEE P1363 Working Group for Public-Key Cryptography Standards; Meeting Minutes (unapproved)*, May 22nd 2001. `http://grouper.ieee.org/groups/1363/WorkingGroup/minutes/010522.txt`.

[75] Impagliazzo, Russel, Leonid A. Levin und Michael Luby: *Pseudo-random generation from one-way functions.* In: *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC'89)*, Seiten 12–24. ACM Press, 1989.

[76] Impagliazzo, Russell und David Zuckerman: *How to recycle random bits.* In: *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS'89)*, Seiten 248–253. IEEE, 1989.

[77] Ingemarsson, Ingemar, Donald T. Tang und C.K. Wong: *A Conference Key Distribution System.* IEEE Transactions on Information Theory, 28(5):714–720, 1982.

[78] *ISO/IEC 15946-1: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General*, 2002.

[79] *ISO/IEC 15946-2: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: Digital Signatures*, 2002.

[80] Jiang, Shaoquan und Guang Gong: *Password Based Key Exchange with Mutual Authentication.* In: Handschuh, Helena und M. Anwar Hasan (Herausgeber): *Selected Areas in Cryptography: 11th International Workshop, SAC 2004*, Band 3357 der Reihe *Lecture Notes in Computer Science*, Seiten 267–279. Springer, 2004.

[81] Juels, Ari und Jorge Guajardo: *RSA Key Generation with Verifiable Randomness.* In: Naccache, David und Pascal Paillier (Herausgeber): *Public Key Cryptography – PKC2002*, Band 2274 der Reihe *Lecture Notes in Computer Science*, Seiten 357–374. Springer, 2002.

[82] Juels, Ari, Michael Luby und Rafail Ostrovsky: *Security of Blind Digital Signatures.* In: Kaliski Jr., Burton S. (Herausgeber): *Advances in Cryptology – CRYPTO '97*, Band 1294 der Reihe *Lecture Notes in Computer Science*, Seiten 150–164. Springer, 1997.

[83] Katz, Jonathan und Ji Sun Shin: *Modeling Insider Attacks on Group Key-Exchange Protocols.* In: *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005.

[84] Katz, Jonathan und Moti Yung: *Scalable Protocols for Authenticated Group Key Exchange.* In: Boneh, Dan (Herausgeber): *Advances in Cryptology – CRYPTO 2003*, Band 2729 der Reihe *Lecture Notes in Computer Science*, Seiten 110–125. Springer, 2003.

[85] Kim, Hyun-Jeong, Su-Mi Lee und Dong Hoon Lee: *Constant-Round Authenticated Group Key Exchange for Dynamic Groups.* In: Lee, Pil Joong (Herausgeber): *Advances in Cryptology – ASIACRYPT 2004*, Band 3329 der Reihe *Lecture Notes in Computer Science*, Seiten 245–259. Springer, 2004.

[86] Klein, Birgit, Marcus Otten und Thomas Beth: *Conference Key Distribution Protocols in Distributed Systems.* In: Farrell, Patrick G. (Herausgeber): *Codes and Ciphers – Cryptography and Coding IV*, Seiten 225–241. IMA, 1993.

[87] Ko, Ki Hyoung, Koo Ho Choi, Mi Sung Cho und Jang Won Lee: *New Signature Scheme Using Conjugacy Problem.* Cryptology ePrint Archive, 2002. http://eprint.iacr.org/2002/168/.

[88] Kuwakado, Hidenori und Hatsukazu Tanaka: *New Subliminal Channel Embedded in the ESIGN.* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E82-A(10):2167–2171, 1999.

[89] Li, Chih-Hung und Josef Pieprzyk: *Conference Key Agreement from Secret Sharing.* In: Pieprzyk, Josef, Rei Safavi-Naini und Jennifer Seberry (Herausgeber): *Information Security and Privacy, ACISP'99*, Band 1587 der Reihe *Lecture Notes in Computer Science*, Seiten 64–76. Springer, 1999.

[90] Lysyanskaya, Anna: *Unique Signatures and Verifiable Random Functions from the DH-DDH Separation.* In: Yung, Moti (Herausgeber): *Advances in Cryptology – CRYPTO 2002*, Band 2442 der Reihe *Lecture Notes in Computer Science*, Seiten 597–612. Springer, 2002.

[91] Menezes, Alfred und Nigel Smart: *Security of Signature Schemes in a Multi-User Setting.* Designs, Codes and Cryptography, 33(3):261–274, 2004.

[92] Micali, Silvio, Michael Rabin und Salil Vadhan: *Verifiable Random Functions.* In: *Proceedings of the 40th Annual Symposium on the Foundations of Computer Science*, Seiten 120–130. IEEE, October 1999.

[93] Mitchell, Chris J., Mike Ward und Piers Wilson: *Key control in key agreement protocols.* IEE Electronics Letters, 34(10):980–981, 1998.

[94] Möller, Bodo: *A Public-Key Encryption Scheme with Pseudo-random Ciphertexts.* In: Samarati, Pierangela, Peter Ryan, Dieter Gollmann und Refik Molva (Herausgeber): *Computer Security – ESORICS 2004*, Band 3193 der Reihe *Lecture Notes in Computer Science*, Seiten 335–351. Springer, 2004.

[95] M'Raïhi, David, David Naccache, David Pointcheval und Serge Vaudenay: *Computational Alternatives to Random Number Generators.* In: Tavares, Stafford und Henk Meijer (Herausgeber): *Fifth Annual Workshop on Selected Areas in Cryptography, SAC'98*, Band 1556 der Reihe *Lecture Notes in Computer Science*, Seiten 72–80. Springer, 1999.

[96] Naor, Moni und Omer Reingold: *Number-Theoretic Constructions of Efficient Pseudo-Random Functions.* Journal of the ACM, 51(2):231–262, 2004.

[97] *NESSIE Portfolio of recommended cryptographic primitives.* https://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/decision-final.pdf, 2003.

[98] Nguyen, Phong Q.: *A Note on the Security of NTRUSign*. Cryptology ePrint Archive, Report 2006/387, 2006. http://eprint.iacr.org/2006/387.

[99] NTT Information Sharing Platform Laboratories, NTT Corporation: *ESIGN-D Specification*. https://www.cosic.esat.kuleuven.be/nessie/updatedPhase2Specs/esign/esignd-spec.pdf, 2002.

[100] NTT Information Sharing Platform Laboratories, NTT Corporation: *Self-evaluation of ESIGN-D*, 2002. https://www.cosic.esat.kuleuven.be/nessie/updatedPhase2Specs/esign/esignd-eval.pdf.

[101] Pedersen, Torben Pryds: *Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing*. In: Feigenbaum, Joan (Herausgeber): *Advances in Cryptology – CRYPTO '91*, Band 576 der Reihe *Lecture Notes in Computer Science*, Seiten 129–140. Springer, 1992.

[102] Pfitzmann, Birgit und Michael Waidner: *A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission*. In: *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, Seiten 184–200. IEEE Computer Society, 2001. http://eprint.iacr.org/2000/066.ps.

[103] *PKCS #1 v.2.1: RSA Cryptography Standard*, 2002. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf.

[104] Pointcheval, David und Jacques Stern: *Security Proofs for Signature Schemes*. In: Maurer, Ueli (Herausgeber): *Advances in Cryptology – EUROCRYPT '96*, Band 1070 der Reihe *Lecture Notes in Computer Science*, Seiten 387–399. Springer, 1996.

[105] Rosa, Tomáš: *Key-collisions in (EC)DSA: Attacking Non-repudiation*. Cryptology ePrint Archive, 2002. http://eprint.iacr.org/2002/129/.

[106] Shoup, Victor: *On Formal Models for Secure Key Exchange*. Cryptology ePrint Archive, 1999. http://eprint.iacr.org/1999/012.

[107] *Gesetz über Rahmenbedingungen für elektronische Signaturen*. Bundesgesetzblatt I vom 21. Mai 2001.

[108] Simmons, Gustavus J.: *Symmetric and Asymmetric Encryption*. ACM Computing Surveys, 11(4):305–330, 1979.

[109] Simmons, Gustavus J.: *Message authentication without secrecy*. In: Simmons, Gustavus J. (Herausgeber): *Secure Communications and Asymmetric Cryptosystems*, Band 69 der Reihe *AAAS Selected Symposia Series*. Westview Press, 1982.

[110] Simmons, Gustavus J.: *Verification of Treaty Compliance Revisited.* In: *IEEE Symposium on Security and Privacy*, Seiten 61–66. IEEE Computer Society Press, 1983.

[111] Simmons, Gustavus J.: *The Subliminal Channel and Digital Signatures.* In: Beth, Thomas, Norbert Cot und Ingemar Ingemarsson (Herausgeber): *Advances in Cryptology – EUROCRYPT '84*, Band 209 der Reihe *Lecture Notes in Computer Science*, Seiten 364–378. Springer, 1984.

[112] Simmons, Gustavus J.: *An Introduction to the Mathematics of Trust in Security Protocols.* In: *Proceedings of the Computer Security Foundations Workshop VI*, Seiten 121–127. IEEE Computer Society Press, 1993.

[113] Simmons, Gustavus J.: *Subliminal Channels; Past and Present.* European Transactions on Telecommunications, 5(4):459–473, 1994.

[114] Simmons, Gustavus J.: *Subliminal Communication Is Easy Using the DSA.* In: Helleseth, Tor (Herausgeber): *Advances in Cryptology – EUROCRYPT '93*, Band 765 der Reihe *Lecture Notes in Computer Science*, Seiten 218–232. Springer, 1994.

[115] Simmons, Gustavus J.: *Results Concerning the Bandwidth of Subliminal Channels.* IEEE Journal on Selected Areas in Communications, 16(4):463–473, 1998.

[116] Simmons, Gustavus J.: *The History of Subliminal Channels.* IEEE Journal on Selected Areas in Communication, 16(4):452–462, 1998.

[117] Steiner, Michael: *Secure Group Key Agreement.* Doktorarbeit, Universität des Saarlandes, 2002. `http://www.semper.org/sirene/publ/Stei_02.thesis-final.pdf`.

[118] Stern, Jacques, David Pointcheval, John Malone-Lee und Nigel P. Smart: *Flaws in Applying Proof Methodologies to Signature Schemes.* In: Yung, Moti (Herausgeber): *Advances in Cryptology – CRYPTO 2002*, Band 2442 der Reihe *Lecture Notes in Computer Science*, Seiten 93–110. Springer, 2002.

[119] Stinson, Douglas R.: *Universal hash families and the leftover hash lemma, and applications to cryptography and computing.* Journal of Combinatorial Mathematics and Combinatorial Computing, 42:3–31, 2002.

[120] Tan, Chik-How: *Key Substitution Attacks on Some Provably Secure Signature Schemes.* IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E87–A(1):1–2, 2004.

[121] Tseng, Yuh-Min: *A Robust Multi-Party Key Agreement Protocol Resistant to Malicious Participants.* The Computer Journal, 48(4):480–487, 2005.

[122] Tzeng, Wen-Guey: *A Practical and Secure Fault-Tolerant Conference-Key Agreement Protocol.* In: Imai, Hideki und Yuliang Zheng (Herausgeber): *Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000*, Band 1751 der Reihe *Lecture Notes in Computer Science*, Seiten 1–13. Springer, 2000.

[123] Tzeng, Wen-Guey und Zhi-Jia Tzeng: *Round-Efficient Conference Key Agreement Protocols with Provable Security.* In: Okamoto, Tatsuaki (Herausgeber): *Advances in Cryptology – ASIACRYPT 2000*, Band 1976 der Reihe *Lecture Notes in Computer Science*, Seiten 614–627. Springer, 2000.

[124] Vaudenay, Serge: *Hidden Collisions on DSS.* In: Koblitz, Neal (Herausgeber): *Advances in Cryptology – CRYPTO '96*, Band 1109 der Reihe *Lecture Notes in Computer Science*, Seiten 83–88. Springer, 1996.

[125] Vaudenay, Serge: *The Security of DSA and ECDSA.* In: Desmedt, Yvo G. (Herausgeber): *Public Key Cryptography – PKC 2003*, Band 2567 der Reihe *Lecture Notes in Computer Science*, Seiten 309–323. Springer, 2003.

[126] Vaudenay, Serge: *Digital Signature Schemes with Domain Parameters.* In: Wang, Huaxiong, Josef Pieprzyk und Vijay Varadharajan (Herausgeber): *Information Security and Privacy, ACISP 2004*, Band 3108 der Reihe *Lecture Notes in Computer Science*, Seiten 188–199. Springer, 2004.

[127] Young, Adam und Moti Yung: *The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?* In: Koblitz, Neal (Herausgeber): *Advances in Cryptology – CRYPTO '96*, Band 1109 der Reihe *Lecture Notes in Computer Science*, Seiten 89–103. Springer, 1996.

[128] Young, Adam und Moti Yung: *Malicious Cryptography: Exposing Cryptovirology.* Wiley, 2004.

[129] Young, Adam und Moti Yung: *An Elliptic Curve Backdoor Algorithm for RSASSA.* In: *Information Hiding – IH2006*, Lecture Notes in Computer Science. Springer, erscheint.

[130] Zhang, Fangguo, Byoungcheon Lee und Kwangjo Kim: *Exploring Signature Schemes with Subliminal Channel.* In: *The 2003 Symposium on Cryptography and Information Security; SCIS 2003*, 2003.

[131] Zhang, Fangguo, Reihaneh Safavi-Naini und Willy Susilo: *An Efficient Signature Scheme from Bilinear Pairings and Its Applications.* In: Bao, Feng, Robert Deng und Jianying Zhou (Herausgeber): *Public Key Cryptography*

– *PKC 2004*, Band 2947 der Reihe *Lecture Notes in Computer Science*, Seiten 277–290. Springer, 2004.

Eigene Veröffentlichungen

Beiträge in Zeitschriften

[1] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *Weak Keys in MST1.* Designs, Codes and Cryptography, 37(3):509–524, 2005.

[2] Bohli, Jens-Matthias, Stefan Röhrich und Rainer Steinwandt: *Key substitution attacks revisited: Taking into account malicious signers.* International Journal of Information Security, 5(1):30–36, 2006.

[3] Bohli, Jens-Matthias, Benjamin Glas und Rainer Steinwandt: *Algebraic Cryptosystems and Side Channel Attacks: Braid Groups and DPA.* Zur Veröffentlichung in Congressus Numerantium angenommen.

Beiträge zu Konferenz-Proceedings

[4] Bohli, Jens-Matthias und Rainer Steinwandt: *On Subliminal Channels in Deterministic Signature Schemes.* In: Park, Choonsik und Seongtaek Chee (Herausgeber): *Information Security and Cryptology – ICISC 2004*, Band 3506 der Reihe *Lecture Notes in Computer Science*, Seiten 182–194. Springer, 2005.

[5] Bohli, Jens-Matthias, Jörn Müller-Quade und Stefan Röhrich: *Fairness and Correctness in Case of a Premature Abort.* In: Maitra, Subhamoy, C. E.Veni Madhavan und Ramarathnam Venkatesan (Herausgeber): *Progress in Cryptology – INDOCRYPT 2005*, Band 3797 der Reihe *Lecture Notes in Computer Science*, Seiten 322–331. Springer, 2005.

[6] Bohli, Jens-Matthias: *A Framework for Robust Group Key Agreement.* In: Gavrilova, Marina, Osvaldo Gervasi, Vipin Kumar, C. J.Kenneth Tan, David Taniar, Antonio Laganà, Youngsong Mun und Hyunseung Choo (Herausgeber): *ACIS'06 in Computational Science and Its Applications – ICCSA 2006 (3)*, Band 3982 der Reihe *Lecture Notes in Computer Science*, Seiten 355–364. Springer, 2006.

[7] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *A Subliminal-free Variant of ECDSA*. In: *Information Hiding – IH2006*, Lecture Notes in Computer Science. Springer, erscheint.

[8] Bohli, Jens-Matthias und Rainer Steinwandt: *Deniable Group Key Agreement*. In *Progress in Cryptology – VIETCRYPT 2006*, Band 4341 der *Lecture Notes in Computer Science*, S. 298–311. Springer, 2006.

[9] Bohli, Jens-Matthias, Benjamin Glas und Rainer Steinwandt: *Towards Provably Secure Group Key Agreement Building on Group Theory*. In *Progress in Cryptology – VIETCRYPT 2006*, Band 4341 der *Lecture Notes in Computer Science*, S. 322–336. Springer, 2006.

[10] Abdalla, Michel, Jens-Matthias Bohli, María Isabel González Vasco und Rainer Steinwandt: *(Password) Authenticated Key Estabmisment: From 2-Party to Group*. In *Theory of Cryptography Conference – TCC 2007*, Band 4392 der *Lecture Notes in Computer Science*, S. 499–514. Springer, 2007.

Sonstiges

[10] Bohli, Jens-Matthias: *Schwache Schlüssel des Public-Key-Systems MST_1*, 2001. Studienarbeit am Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe.

[11] Bohli, Jens-Matthias: *Algorithmen für iterative Entscheidungen in der Signalverarbeitung*, 2003. Diplomarbeit am Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe.

[12] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *Burmester-Desmedt Tree-Based Key Transport Revisited: Provable Security*. Cryptology ePrint Archive, Report 2005/360, 2005. `http://eprint.iacr.org/2005/360`.

[13] Bohli, Jens-Matthias, Jörn Müller-Quade und Stefan Röhrich: *On Group Key Agreement with Cheater Identification*. Western European Workshop on Research in Cryptology, WEWoRC 2005, 2005.

[14] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *Secure Group Key Establishment Revisited*. Cryptology ePrint Archive, Report 2005/ 395, 2005. `http://eprint.iacr.org/2005/395/`.

[15] Bohli, Jens-Matthias, María Isabel González Vasco und Rainer Steinwandt: *Password-Authenticated Constant-Round Group Key Establishment with a Common Reference String*, 2006. `http://eprint.iacr.org/2006/214/`.

Index

Ausweis, 28

Breitbandkanal, 32

CDH-Annahme, 98
Commitment-Verfahren, 50

DDH-Annahme, 98
DSA, 16

EC-DSA, 17, 47
EC-KCDSA, 20
Entity-Authentication, 74
EUF-CMA, 5

festlegen, 50
Forgery, 4
fortgeschrittene Signatur, 7
Forward-Secrecy, 60, 66
Frische, 60, 66

Gruppenschlüsselaustausch, 59

Integrität, 74

Kapazität, 32
Key-Agreement, 60, 75
Key-Confirmation, 60
korrumpiert, 65

Leftover-Hashlemma, 48

MKS-Angriff, 9

NTRUSign, 12

Partner, 66
passiver Angreifer, 60, 79, 84
PPT-Algorithmus, 97

qualifizierte Signatur, 7

Random Oracle, 34
Reisepass, 28
RSA-PSS, 15

Schlüsselauthentifizierung
    explizite, 60
    implizite, 60, 66
Schlüsselersetzungsangriff, 8–10
Schlüsselverifikation, 6
Schmalbandkanal, 32
SEUF-CMA, 5
sicherer Schlüsselaustausch
    bei ehrlichen Teilnehmern, 66
sicherer Schlüsselaustausch, 76
Sicherheitsmodell für
        Schlüsselaustausch, 63
Sicherheitsparameter, 97
Signaturverfahren, 3
signifikant, 97
Sitzungsnummer, 64, 79
Sitzungsschlüssel, 64
Station-to-Station-Protokoll, 8
Steganographie, 25
subliminaler Kanal, 30
subliminalfrei, 42
subliminalfrei mit Wächter, 17, 44

uberwältigend, 97
UC-Modell, 7, 61
universelle Hashfunktion, 48, 98
ununterscheidbar, 97

vernachlässigbar, 97

Zertifikat, 5, 21