Einführung in digitale Signaturen

Hannes Thalheim
Universität Leipzig

8. Januar 2018

Zusammenfassung

Für eine sichere Kommunikation im Web ist die Geheimhaltung von Nachrichten so wichtig wie das Wissen, wer sich am anderen Ende der Leitung verbirgt. Ein digitales Signaturschema kann u. a. den Gegenüber identifizieren. Diese Arbeit wird den Zusammenhang zu asynchronen Kryptosystemen sowie die Anforderungen an ein solches Signaturschema genauer beleuchten, es definieren und festlegen, wann es als *sicher* gilt. Am einfachen RSA-Signaturschema wird ein Negativbeispiel für den Sicherheitsbegriff bei Signaturen aufgezeigt werden. Insgesamt wird eine Einführung in das Konzept digitaler Signaturen gegeben und deren Vorzüge aufgeführt.

1 Einleitung

Ob beim Versenden von E-Mails mit sensiblen Informationen, beim Tätigen von Überweisungen oder Abschließen von verbindlichen Verträgen im WWW ist das Thema *Sicherheit* allgegenwärtig. Niemand möchte, dass seine Nachrichten mitgelesen und gegebenenfalls manipuliert werden, oder die eigene Identität von Dritten verwendet bzw. missbraucht wird. Ein solcher Vorfall kostet in der Regel Geld, Zeit und Nerven, um ihn aus der Welt zu schaffen. Sicherheit ist im 21. Jahrhundert bei der Kommunikation über das Internet sehr wichtig.

Da der Sicherheitsbegriff an sich weitreichend und sehr kontextsensitiv ist, beschränkt er sich in dieser Arbeit unter der Annahme eines vorhandenen asymmetrischen Kryptosystems wie einer *Public-Key*-Infrastruktur auf vier Aspekte: *Geheimhaltung*, *Echtheit*, *Nichtleugbarkeit* und *Integrität* von Nachrichten. Die drei zuletzt genannten bilden gleichzeitig die Anforderungen an ein System digitaler Signaturen.

Geheimhaltung bedeutet, dass eine Nachricht auf dem Weg vom Sender zum Empfänger nicht von Dritten gelesen werden kann. Dies wird über eine geeignete *Ende-zu-Ende*-Verschlüsselung erreicht. Sie ermöglicht die Übertragung von Nachrichten auf einem öffentlichen Kanal. Verschlüsselung in einer Public-Key-Infrastruktur erreicht dies, solange eine Verbreitung der öffentlichen Schlüssel zugesichert ist. Im Rahmen dieser Arbeit wird das Konzept von Public-Key und der dort geltende Begriff *sicher* nicht weiter beleuchtet und als bekannt vorausgesetzt.

Unter *Echtheit* bzw. *Authentizität* (engl. *authenticity*) wird hier der Nachweis der Identität des Absenders einer Nachricht verstanden. Eine Nachricht soll unterschrieben bzw. *signiert* werden. Dies bindet zum einem den Besitzer der

Unterschrift bzw. *Signatur* als Urheber an die Nachricht. Es verhindert zum anderen eine spätere Änderung des Inhalts der Nachricht durch den Urheber. Somit wird eine *Nichtleugbarkeit* bzw. *Nichtabstreitbarkeit* (engl. *non-repudiation*) der Nachricht geschaffen. Der Inhalt der Nachricht kann nach dem Signieren nicht mehr zurückgezogen oder verändert werden, er wird also mit dem Signieren festgesetzt und zugesichert. Letzteres erschafft eine *Integrität* (engl. *integrity*) der Nachricht. Sie ist somit auch gegen spätere Manipulation geschützt. Dies alles soll mittels eines *Signaturschema*s in einer als vorhanden angenommenen Public-Key-Infrastruktur ermöglicht werden. Wie dies geschieht, wird konzeptionell in Kapitel 2 erläutert und an Umsetzungsbeispielen in Kapiteln 3 und 4 dargestellt. In Kapitel 3 werden ebenso die Anforderungen an ein *sicheres* Signaturschema gestellt.

2 Digitales Signaturschema

Konzept. Die grundlegende Idee eines digitalen Signaturschemas dreht sich um die Rolle eines Unterzeichners (engl. *Signer*). Sei nun *Alice* in der Rolle eines solchen Unterzeichners. Sie muss ihren öffentlichen Schlüssel (engl. *public key*) pk etablieren bzw. verbreiten. An dieser Stelle sei abermals darauf hingewiesen, dass eine bestehende Public-Key-Infrastruktur vorausgesetzt und die Schwierigkeit des Verbreitens eines pk vorerst ignoriert und als zuverlässig bzw. überwunden angenommen wird. *Alice* besitzt ebenfalls aufgrund dieser Voraussetzung ein Geheimnis bzw. einen privaten Schlüssel (engl. *secret/private key*) sk.

In einem Public-Key-Verschlüsselungsschema ist *Alice* Besitzer eines Schlüsselpaares (pk_A, sk_A) und gleichzeitig der Empfänger verschlüsselter Nachrichten. Dazu benutzt ein Sender *Bob* pk_A um Nachrichten an *Alice* zu verschlüsseln. Nur *Alice* kann diese dann mithilfe von sk_A entschlüsseln.

Im Gegensatz dazu stellt *Alice* in einem Signaturschema den Sender einer Nachricht dar. Sie nutzt sk_A um eine Nachricht m zu signieren. Dazu erzeugt sie eine *Signatur* σ zu m mit einem geeignetem Algorithmus in Abhängigkeit von sk_A. *Alice* sendet nun das Paar (m, σ) an *Bob*, der dann mithilfe von pk_A und einem geeigneten Algorithmus aus σ eine Nachricht m' errechnen kann. Falls nun $m = m'$ gilt, ist m erfolgreich *verifiziert*, im Negativfall ist die Signatur *ungültig*. Eine Manipulation der Nachricht nach dem Signieren wird somit erkannt.

Formale Definition. Ein *Signaturschema* $\mathcal{S} := (\texttt{Gen}, \texttt{Sign}, \texttt{Vrfy})$ besteht aus drei PPT-Algorithmen:

`Gen` Ein *Schlüsselgenerator* mit:

Eingabe: Sicherheitsparameter 1^n $(n \in \mathbb{N})$

Ausgabe: Schlüsselpaar (pk, sk)

mit $|k| \geq n$, $k \in \{pk, sk\}$

`Sign` Ein *Signieralgorithmus* zu einem Geheimnis sk mit:

Eingabe: Nachricht m

Ausgabe: Signatur σ

mit $\sigma \leftarrow \texttt{Sign}_{sk}(m)$

`Vrfy` Ein *Verifikationsalgorithmus* zu einem öffentlichen Schlüssel pk mit:

Eingabe: Nachricht m, Signatur σ

Ausgabe: Wahrheitswert $b \in \{\texttt{0}, \texttt{1}\}$

$$\text{mit } b := \texttt{Vrfy}_{pk}(m, \sigma) = \begin{cases} \texttt{1} & \text{falls } \sigma \text{ valide zu } m \text{ bzgl. } pk \\ \texttt{0} & \text{sonst} \end{cases}$$

Von einem Signaturschema $\mathcal{S} = (\texttt{Gen}, \texttt{Sign}, \texttt{Vrfy})$ wird gefordert, dass

$$\texttt{Vrfy}_{pk}(m, \texttt{Sign}_{sk}(m)) = \texttt{1}$$

für jede (legale) Nachricht m aus einem geeignetem Nachrichtenbereich gilt, bis auf eine vernachlässigbar geringe Wahrscheinlichkeit bzgl. $(pk, sk) \leftarrow \texttt{Gen}(1^n)$.

Falls eine Funktion ℓ in Abhängigkeit vom Parameter n existiert, sodass jedes von $\texttt{Gen}(1^n)$ erzeugte Schlüsselpaar im Nachrichtenbereich $\{0,1\}^{\ell(n)}$ liegt, so wird $\mathcal{S}$ als *Signaturschema für Nachrichten der Länge* $\ell(n)$ bezeichnet.

Ablauf. *Alice* besitzt bereits ein Schlüsselpaar (pk_A, sk_A) oder erzeugt ein solches mithilfe eines Schlüsselgenerators $\texttt{Gen}(1^n)$ und eines Sicherheitsparameters 1^n für ein geeignetes $n \in \mathbb{N}$. *Alice* veröffentlicht nun pk_A (bspw. auf einer Website). *Bob* kann somit eine korrekte Kopie von pk_A erlangen und diese nutzen. Möchte nun *Alice* eine Nachricht m aus einem geeignetem Nachrichtenbereich versenden, so berechnet sie die Signatur $\sigma \leftarrow \texttt{Sign}_{sk_A}(m)$ und übermittelt anschließend das Tupel (m, σ) an *Bob*. Dieser kann nun mittels pk_A prüfen, ob $\texttt{Vrfy}_{pk_A}(m, \sigma) = \texttt{1}$ gilt. Ist dies der Fall, so wurde *Alice* als rechtmäßiger Sender verifiziert. Ebenfalls wurde nun sichergestellt, dass m auf dem Weg zu B unverändert blieb.

3 Sicherheit

Wie der Begriff der Sicherheit eines Signaturschema gegen das Fälschen einer Nachricht definiert ist, wird nun genauer ausgeführt. Unter einer *Fälschung* (engl. *forgery*) wird in diesem Kontext eine Nachricht m mit valider Signatur σ bezeichnet, die jedoch nicht vom Besitzer des Schlüsselpaares (pk, sk) erzeugt wurde. Das bedeutet, ein Angreifer hat ohne Kenntnis von sk eine passende Signatur σ zu m und somit eine *erfolgreiche Fälschung* erzeugt. Dieser Vorgang lässt sich anhand eines Spiels darstellen:

Erzeugen einer Fälschung. $\texttt{Spiel}_{forg}$

1. Sei ein Herausforderer beliebig, aber fest mit Schlüsseln (pk, sk).
2. Ein Angreifer kennt pk, aber sk nicht.
3. Der Angreifer darf zu beliebigen Nachrichten $m_1, \ldots, m_k$ vom Herausforderer Signaturen $\sigma_1, \ldots, \sigma_k$ anfordern.
4. Der Angreifer gewinnt, falls er ein Paar (m, σ) mit m bzw. σ paarweise verschieden zu den m_i bzw. σ_i $(i \in [1, k] \subseteq \mathbb{N})$ erfolgreich fälscht.

Definition. Ein Signaturschema $\mathcal{S} := (\texttt{Gen}, \texttt{Sign}, \texttt{Vrfy})$ heißt *existentiell fälschungssicher* für Angriffe mit adaptiv ausgewählten Nachrichten oder einfach nur *sicher*, falls es für jeden PPT-Angreifer eine vernachlässigbare (engl. *negligible*) Schranke `negl` in Abhängigkeit vom Parameter n gibt, sodass gilt:

$$P[\text{Angreifer gewinnt } \texttt{Spiel}_{forg}] \leq \texttt{negl}(n)$$

3.1 Einfaches RSA-Signaturschema

Nun wird am konkreten Beispiel von *einfachem* (engl. *plain*) RSA als Public-Key-Infrastruktur ein Signaturschema betrachtet hinsichtlich Ablauf und Sicherheit. Das Konzept ist analog dem in Kapitel 2 beschriebenen. Doch zunächst zur Modulus-Erzeugung von RSA:

Algorithmus. `GenModulus`

Eingabe: Sicherheitsparameter 1^n $(n \in \mathbb{N})$

Ausgabe: (N, p, q)
wobei p, q zwei n-bit Primzahlen sind und $N := p \cdot q$ gilt.
Außerdem gilt:

$$\phi(N) = (p-1) \cdot (q-1)$$

und

$$\phi(N) = |\mathbb{Z}_N^*| = ord(\mathbb{Z}_N^*)$$

Definition. Ein *einfaches RSA-Signaturschema* $\mathcal{S}_{\text{RSA}} := (\texttt{GenRSA}, \texttt{Sign}, \texttt{Vrfy})$ besteht aus drei PPT-Algorithmen:

`GenRSA` Ein *RSA-Schlüsselgenerator* mit:

Eingabe: Sicherheitsparameter 1^n $(n \in \mathbb{N})$

Ausgabe: Modulus N; $e, d \in \mathbb{Z}_N^*$

mit $e \cdot d \equiv_{\phi(N)} 1$ und $(N, p, q) \leftarrow \texttt{GenModulus}(1^n)$

Schlüsselpaar: $(pk, sk) := (\langle N, e\rangle, \langle N, d\rangle)$

`Sign` Ein *Signieralgorithmus* zum Geheimnis $sk = \langle N, d\rangle$ mit:

Eingabe: Nachricht $m \in \mathbb{Z}_N^*$

Ausgabe: Signatur $\sigma := m^d \in \mathbb{Z}_N^*$

`Vrfy` Ein *Verifikationsalgorithmus* zum öffentlichen Schlüssel $pk = \langle N, e\rangle$ mit:

Eingabe: Nachricht $m \in \mathbb{Z}_N^*$, Signatur $\sigma \in \mathbb{Z}_N^*$

Ausgabe: Wahrheitswert $b \in \{\texttt{0}, \texttt{1}\}$

mit $b := \texttt{Vrfy}_{pk}(m, \sigma) = \begin{cases} \texttt{1} & \text{falls } m \equiv_N \sigma^e \\ \texttt{0} & \text{sonst} \end{cases}$

Bemerkung. Offenbar gilt für legitime Nachrichten m und deren Signatur $\sigma := \texttt{Sign}_{sk}(m) = m^d$ weiterhin:

$$\texttt{Vrfy}_{pk}(m, \texttt{Sign}_{sk}(m)) = 1$$

da $e, d \in \mathbb{Z}_N^*$ invers zueinander sind und somit folgendes gilt:

$$\sigma^e = (m^d)^e = m^{e \cdot d} = m^1 = m$$

Das RSA-Signaturschema erfüllt somit diese grundlegende Anforderung. Der Ablauf des Signieren und Sendens verhält sich analog dem in Kapitel 2 beschriebenen Ablauf.

Schwächen. Sei nun $\mathcal{S}_{\text{RSA}}$ ein einfaches RSA-Signaturschema.

Es ist möglich alleine aus dem öffentlichen Schlüssel pk eine Fälschung gemäß der Beschreibung in Kapitel 3 zu erzeugen. Dazu wählt ein Angreifer eine beliebige Signatur $\sigma \in \mathbb{Z}_N^*$. Da hier ein beliebiger öffentlicher Schlüssel $pk = \langle N, e \rangle$ als bekannt angenommen wird, kann der Angreifer einfach die Nachricht $m := \sigma^e$ passend zur Signatur σ berechnen. Dies scheint auf den ersten Blick nicht weiter bedenklich, da bei der Wahl einer willkürlichen Signatur auch nur eine willkürliche Nachricht berechnet wird. Dies bietet jedoch dem Angreifer zumindest die Möglichkeit zufällig auch sinnvolle Nachrichten zu erzeugen. Nichtsdestotrotz sind Signatur und Nachricht immer valide. Somit ist laut Definition (m, σ) in jedem Fall eine erfolgreiche Fälschung. Ebenso könnte er trivialerweise $m = \sigma = 0$ wählen. In beiden Fällen gilt:

$$P[\text{Angreifer gewinnt } \texttt{Spiel}_{forg} \text{ mit RSA}] = 1$$

Eine weitaus erheblichere Schwachstelle des einfachen RSA-Signaturschemas besteht im *Faktor-Angriff*. Hierbei wählt ein Angreifer zwei Nachrichten m_1, m_2. Nun bringt er den Besitzer eines privaten Schlüssels $\langle N, d \rangle$ dazu diese Nachrichten zu signieren und erhält somit σ_1, σ_2. Der Angreifer berechnet nun eine neue Nachricht $m := m_1 \cdot m_2$ und Signatur $\sigma := \sigma_1 \cdot \sigma_2$. Offenbar gilt:

$$\sigma = \sigma_1 \cdot \sigma_2 = m_1^d \cdot m_2^d = (m_1 \cdot m_2)^d = m^d$$

und (m, σ) ist eine erfolgreiche Fälschung. Der Angreifer gewinnt $\texttt{Spiel}_{forg}$ mit RSA ebenfalls jedes mal und besitzt hierbei sogar die volle Kontrolle über der Wahl von m.

Fazit. Das einfache RSA-Signaturschema erfüllt *nicht* die Definition eines *sicheren* Signaturschemas aus Kapitel 3.

4 Das Hash-and-Sign Paradigma

Ein digitales Signaturschema ist bei der Anwendung im Allgemeinen mehrere Größenordnungen ineffizienter als ein Nachrichtenauthentifizierungscode. Das signieren sehr langer Nachrichten ist teuer, da Operationen mit privaten Schlüsseln zum Einsatz kommen. In [1] wird dies mit *asymptotischen* Kosten bezeichnet. Eine Lösung besteht darin zuerst einen *Hash* für die Nachricht zu berechnen und diesen anschließend zu signieren.

Beispiel. Angenommen es existiert ein Signaturschema für Nachrichten der Länge $\ell \in \mathbb{N}$. Um nun eine Nachricht $m \in \{0,1\}^*$ mit Länge $|m| > \ell$ signieren zu können, wird eine geeignete Hashfunktion H eingesetzt, um einen Hash $H(m)$ der fixen Länge ℓ zu berechnen.

Formale Konstruktion. Sei $\mathcal{S} = (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Vrfy})$ ein Signaturschema für Nachrichten der Länge $\ell(n)$ und $n \in \mathbb{N}$ ein Parameter. Sei $\mathcal{S}_H = (\mathtt{Gen}_H, H)$ eine Hashfunktion mit fester Ausgabelänge $\ell(n)$. Konstruiere ein *Hash-and-Sign*-Signaturschema $\mathcal{S}' = (\mathtt{Gen}', \mathtt{Sign}', \mathtt{Vrfy}')$ bestehend aus drei PPT-Algorithmen:

$\mathtt{Gen}'$ Ein *Hash-and-Sign-Schlüsselgenerator* mit:

Eingabe: Sicherheitsparameter 1^n $(n \in \mathbb{N})$

Ausgabe: $(pk', sk') := (\langle pk, s\rangle, \langle sk, s\rangle)$

mit $(pk, sk) \leftarrow \mathtt{Gen}(1^n)$ und $s \leftarrow \mathtt{Gen}_H(1^n)$

$\mathtt{Sign}'$ Ein *Signieralgorithmus* zum Geheimnis $sk' = \langle sk, s\rangle$ mit:

Eingabe: Nachricht $m \in \{0,1\}^*$

Ausgabe: Signatur σ

mit $\sigma \leftarrow \mathtt{Sign}_{sk}(H^s(m))$

$\mathtt{Vrfy}'$ Ein *Verifikationsalgorithmus* zum öffentlichen Schlüssel $pk = \langle pk, s\rangle$ mit:

Eingabe: Nachricht $m \in \{0,1\}^*$, Signatur σ

Ausgabe: Wahrheitswert $b \in \{\mathtt{0}, \mathtt{1}\}$

mit $b := \mathtt{Vrfy}_{pk}(m, \sigma) = \begin{cases} \mathtt{1} & \text{falls } \mathtt{Vrfy}_{pk}(H^s(m), \sigma) = \mathtt{1} \\ \mathtt{0} & \text{sonst} \end{cases}$

Satz. Sei $\mathcal{S}$ ein sicheres Signaturschema für Nachrichten der Länge ℓ und sei $\mathcal{S}_H$ eine kollisionsresistente Hashfunktion. Dann ist das aus der oben aufgeführten Konstruktion entstandene Hash-and-Sign-Signaturschema *sicher* für Nachrichten willkürlicher Länge.

4.1 Hash-and-Sign mit RSA

Im Folgenden wird nun abermals am Beispiel von RSA eine mögliche Umsetzung dieses Verfahrens demonstriert.

Konstruktion. Ein *Hash-and-Sign-Signaturschema mit RSA*
$\mathcal{S}_{\mathrm{RSA}_H} := (\mathtt{GenRSA}_H, \mathtt{Sign}, \mathtt{Vrfy})$ besteht aus drei PPT-Algorithmen:

$\mathtt{GenRSA}_H$ Ein *RSA-Schlüsselgenerator* mit:

Eingabe: Sicherheitsparameter 1^n $(n \in \mathbb{N})$

Ausgabe: Modulus N; $e, d \in \mathbb{Z}_N^*$ und Funktion $H : \{0,1\}^* \to \mathbb{Z}_N^*$

Schlüsselpaar: $(pk, sk) := (\langle N, e\rangle, \langle N, d\rangle)$

`Sign` Ein *Signieralgorithmus* zum Geheimnis $sk = \langle N, d \rangle$ mit:

Eingabe: Nachricht $m \in \{0,1\}^*$

Ausgabe: Signatur $\sigma := H(m)^d \in \mathbb{Z}_N^*$

`Vrfy` Ein *Verifikationsalgorithmus* zum öffen. Schlüssel $pk = \langle N, e \rangle$:

Eingabe: Nachricht m, Signatur σ

Ausgabe: Wahrheitswert $b \in \{\texttt{0}, \texttt{1}\}$

$$\text{mit } b := \texttt{Vrfy}_{pk}(m, \sigma) = \begin{cases} \texttt{1} & \text{falls } H(m) = \sigma^e \text{ gilt} \\ \texttt{0} & \text{sonst} \end{cases}$$

Nun bleibt die Frage: Ist das Hash-and-Sign-Signaturschema mit RSA sicher? Um diese Frage zu beantworten, werden nun die Anforderungen an die Funktionen H genauer beleuchtet:

Anforderungen an Funktion H

1. H muss hart bzgl. Invertieren sein, sonst
   i) wählt ein Angreifer σ und berechnet $m' = \sigma^e \in \mathbb{Z}_N^*$
   ii) sucht m, sodass $H(m) = m'$ gilt, also $m = H^{-1}(m')$
2. H muss gegen Faktorangriff geschützt sein, sodass es für einen Angreifer schwer ist drei Nachrichten m, m_1, m_2 zu finden, für die gilt:
   $$H(m) = H(m_1) \cdot H(m_2)$$
3. H muss *kollisionsresistent* sein, sodass für zwei Nachrichten m, m' eine Kollision $H(m) = H(m')$ nicht leicht zu finden ist.

Fazit: Es gibt keinen bekannten Weg H so zu bestimmen, um den Anforderungen so gerecht zu werden, dass die Konstruktion des Hash-and-Sign-Signaturschemas laut Definition *sicher* wäre.
Das Hash-and-Sign-Signaturschema mit RSA kann demnach *nicht* als *sicher* bewiesen werden.

RSA full-domain hash (RSA-FDH). Modelliere H als *random oracle*, sodass H uniform zufällig auf $\mathbb{Z}_N^*$ abbildet. In diesem Fall wird das Signaturschema als *RSA full-domain hash* bezeichnet und erfüllt die Anforderungen.

Satz zur Sicherheit. Falls das RSA-Problem *hart* bzgl. `GenRSA` und die Funktion H als *random oracle* modelliert ist, so gilt die Konstruktion des Hash-and-Sign-Signaturschema mit RSA als *sicher*.

5 Fazit

Es wurde unter der Annahme einer vorhandenen Public-Key-Infrastruktur das Signaturschema eingeführt und definiert. Zum signieren einer Nachricht verwendet der Absender seinen privaten Schlüssel und einen geeigneten Signieralgorithmus. Ein Empfänger kann darauf hin mithilfe des zugehörigen Verifikationsalgorithmus und des öffentlichen Schlüssels des Absenders überprüfen, ob

Nachricht und Signatur zusammengehören. Dabei wird die Identität des Absenders bestätigt und es kann eine Manipulation der Nachricht nach dem Signieren ausgeschlossen werden. Ebenfalls wird der Inhalt der Nachricht zugesichert, der Absender kann ihn nicht mehr zurückziehen oder abstreiten. Ein Signaturschema gilt als *sicher*, falls jeder beliebige PPT-Angreifer nur mit jeweils vernachlässigbar geringer Wahrscheinlichkeit eine erfolgreiche Fälschung eines Paars bestehend aus Nachricht und passender Signatur erzeugen kann. Das einfache RSA-Signaturschema stellte sich als *unsicher* heraus.

Im Rahmen dieser Arbeit wurde lediglich ein grober Überblick und eine knappe Einführung in das Thema digitale Signaturen gegeben. Weitere Anwendungen und Umsetzungen wie z. B. *Zertifikate* können in Literatur wie bspw. in [1] nachgelesen und studiert werden.

Abschließend ist anzumerken, dass ein digitales Signaturschema viele praktische Anwendungsfälle und Nutzen in der heutigen Zeit von Smartphones, Tablets, PCs und flächendeckendem drahtlosen Internetzugang besitzt.

Literatur

[1] KATZ, Jonathan; LINDELL, Yehuda. *Introduction to Modern Cryptography* Second Edition. CRC Press, Taylor & Francis Group. 2015.

[2] *Transkription der Vorlesung Kryptographie*, WS 2016/17. Lektor: GESER, Alfons; HTWK. Transkriptor: THALHEIM, Hannes. 2017.