Technikseminer SS 2012

Thema:

Einwegfunktionen mit und ohne Falltür

Eingereicht von: Bjarne Adam, B_Wing 8875

Neuhausweg 5

21368 Dahlenburg

Referent(FH Wedel): Prof. Dr. Michael Anders

Fachhochschule Wedel

Feldstraße 143

22880 Wedel

1. Einwegfunktion

1.1 Was sind Einwegfunktionen

Eine Einwegfunktion ist eine einfach zu berechnende Funktion, für die bisher keine in angemessener Zeit ausführbare Umkehrung bekannt ist.

Mathematische Einwegfunktion $f : X \rightarrow Y$

Für die Berechnung des Funktionswertes von $y = f(x)$ existiert ein Algorithmus, der ihn in polynomieller Zeit lösen kann. Aus Komplextheoretischer Sicht, wird dies in der Landau-Notation $\mathcal{O}(n^k)$ beschrieben, wobei k eine konstante natürliche Zahl und n die Eingabe in Bitlänge ist.

Für die Berechnung der Inversen $f(x) = y$, bei bekanntem Funktionswert y, existiert kein „schneller“ Algorithmus. Die Laufzeit zum lösen des Problems ist Exponentiell wachsend. Das Verhalten zum lösen der Funktion ist aus komplextheoretischer Sicht in Landau-Notation $\mathcal{O}(2^n)$.

1.2 Einwegfunktionen mit und ohne Falltür

Im Gegensatz zu einer Einwegfunktion ohne Falltür, deren Umkehrung nicht bzw. nur mit sehr großem Aufwand durchzuführen ist, besteht bei Einwegfunktionen mit einer Falltür (engl. One-Way-Trapdoor-Function) die Möglichkeit der effizienten Umkehrung. Die Einwegfunktion muss demnach so konzipiert werden, dass mithilfe einer bestimmten Information, die Invertierbarkeit der Funktion effizient Durchführbar ist. Diese Methode wird bei asymmetrischen Verschlüsselungsverfahren zur Ver- und Entschlüsselung von Nachrichten verwendet.

Die Funktionsweise einer Einwegfunktion mit Falltür lässt sich anhand eines Briefkastens verdeutlichen:

Jeder Mensch kann einen Brief in den Briefkasten werfen, ihn aber nur mit großem Aufwand wieder entnehmen, es sei denn, die Person ist im Besitz des Schlüssels (Falltür bzw. Zusatzinformation).

Zu den Einwegfunktionen die in heutigen Verschlüsselungsverfahren verwendet werden, zählen

- Hash Funktionen
- Bestimmung einer Primfaktorzerlegung
- Berechnung des Diskreten Logarithmus

zu den bekannten Funktionen bzw. Probleme auf denen Einwegfunktionen aufbauen. Insbesondere die Bestimmung einer Primfaktorzerlegung und die Berechnung des Diskreten Logarithmus ist aus Komplextheoretischer Sicht ein großes Problem. Es ist bis heute kein Verfahren bekannt, die diese mathematischen Probleme effizient lösen. Einen Beweis für diese Schwierigkeit konnte hingegen noch nicht erbracht werden.

2. Hash Funktionen

2.1 Was sind Hashfunktionen

Hashfunktionen sind Einwegfunktionen, die einen beliebig langen Klartext auf einen Hashwert fester Länge abbilden. Der Begriff Hash stammt von dem Verb „to hash“ ab und bedeutet übersetzt Zerhacken bzw. Zerstreuen. Hashalgorithmen zerstreuen einen Input in einen Hexadezimalen Output. Der resultierende Hashwert wird auch Fingerprint bezeichnet, da er den „gehashten“ Klartext einmalig abbilden soll und eine Wiederholung, bei unterschiedlichem Klartext, nicht eintreffen darf.

2.2 Anwendungen von Hash-Funktionen:

Hashfunktionen werden in unterschiedlichen Bereichen der Informationsverschlüsselung verwendet. Im Bereich der Datenübertragung werden Hashfunktionen für die Kontrolle von Datenmanipulation mittels Digitalen Signaturen oder Prüfsummen verwendet. Innerhalb von Peer-to-Peer Anwendungen dienen sie der Identifikation und des Auffindens einzelner Datenfragmente.

Im Bereich der Datensicherung werden Hashfunktionen häufig bei der Sicherung sensibler Passwörter verwendet. Diese werden nicht als Klartext, sondern lediglich der Hashwert im Datenbanksystem abgespeichert. Bei wiederholtem Login, werden dann die Hashwerte und nicht der Klartext miteinander verglichen.

2.3 Anforderungen an Hash-Funktionen

<u>Kollisionsresistent</u>

Bei der Generierung von Hashwerten bei unterschiedlichen Eingaben, darf keine Kollision der Hashwerte, bzw. eine Gleichheit der Hashwerte, auftreten.

<u>Kompression</u>

Der generierte Hashwert muss immer deutlich kürzer sein, als der zu „hashende“ Input.

<u>Chaos</u>

Durch kleinste Abweichungen des Eingabewertes, verursacht der Algorithmus eine starke Varianz im Hashwert.

<u>Surjektivität</u>

Hashfunktionen müssen surjektiv sein, damit jeder Hashwert aus dem definierten Wertebereich vorkommen kann.

<u>Effizienz</u>

Der Algorithmus sollte einen Hashwert ohne viel Zeit- und Speicherbedarf berechnen können.

Preimage Resistance

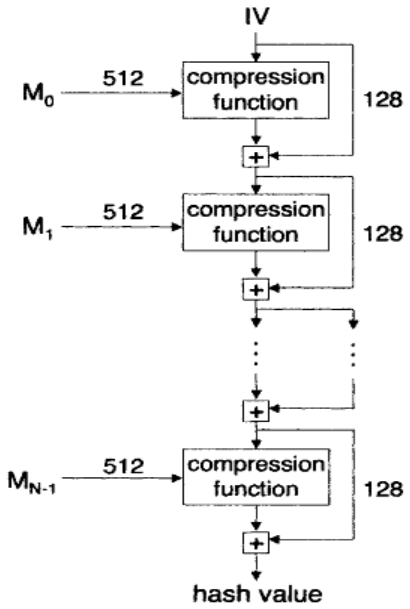
Für einen Angreifer darf die Generierung einer Nachricht, zu einem vorgegebenen Hashwert, nicht in realisierbarer Zeit möglich sein.

Second Preimage Resistant

Für einen Angreifer darf die Generierung einer zweiten Nachricht, bei gegebener Nachricht mit gleichem Hashwert, nicht in realisierbarer Zeit möglich sein.

2.4 Allgemeiner Ablauf einer Hash-Funktion

Die Überwiegende Form der Hash-Funktionen basieren auf dem Merkle-Damgard-Verfahren. Das Verfahren beruht auf dem iterativen Durchlauf einer Kompressionfunktion, die den resultierenden Hashwert generiert.



Padding

Die Länge der Nachricht wird für die Kompressionsfunktion, mittels so genannter Füllbytes, aufgestockt und angepasst.

Trennen

Die mit Füllbytes erweiterte Nachricht M wird für die Kompressionsfunktion in M_{n-1} (für n=0,1,2,3,…,n-1) gleich große Teile aufgetrennt (hier 512 Bit) .

Kompression

Der gesamte Nachrichtenblocksatz M_n wird einzeln über die Kompressionsfunktion iterativ verarbeitet. Beginnend mit dem IV (Initial Value) bis zum fertigen Hash-Value. Von jedem verarbeiteten Block geht der entstandene Hashwert in die darauf folgende Kompressionsfunktion mit ein. Daraus resultiert der Finale Hashwert aus allen Blöcken (hier 128 Bit).

Optionale zusätzliche Transformationen

Der fertige Hashwert wird in der letzten Phase, je nach Verfahren, durch weitere Operationen manipuliert, um somit keine Rückschlüsse auf den Hashalgorithmus bzw. auf die Originalnachricht führen zu können.

3. Zahlentheorie

3.1 Primzahlen

Primzahlen sind positive ganze Zahlen größer 1, die nur durch 1 und durch sich selbst teilbar sind. Jede Primzahl, außer der ersten Primzahl (2), ist ungerade.

Aus Zahlentheoretischer Sicht, sind Primzahlen der Grundbaustein der natürlichen Zahlen. Die natürlichen Zahlen bilden sich aus der Einheitszahl 1, den Primzahlen und den zusammengesetzten Zahlen (jede nicht-Primzahl > 1 kann durch das Produkt von Primzahlen gebildet werden).

4 = 2 * 2

6 = 2 * 3

60 = 2 * 2 * 3 * 5

91 = 7 * 13

Unter den ersten 100 natürlichen Zahlen existieren 25 Primzahlen:

2; 3; 5; 7; 11; 13; 17; 19; 23; 29; 31; 37; 41; 43; 47; 53; 59; 61; 67; 71; 73; 79; 83; 89; 97

3.2 Eulersche Phi-Funktion

Die Eulersche φ-Funktion berechnet die Anzahl der Teilerfremden ganzen Zahlen einer natürlichen Zahl n für alle ganzen Zahlen 1 bis n.

$$\varphi(n) \;:=\; \left|\{1 \leq a \leq n \mid \mathrm{ggT}(a,n) = 1\}\right|$$

Wenn n = prim, dann gilt: φ(n) = n – 1

Wenn n das Produkt zweier verschiedener Primzahlen ist, dann gilt:

φ(p * q) = (p – 1) * (q – 1)

Beispiel:

φ(8) = 4 : { 1, 3, 5, 7 }

φ(11) = 10 : { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }

φ(15) = 3 * 5 = (3 - 1) * (5 - 1) = 2 * 4 = 8 : { 1, 2, 4, 7, 8, 11, 13, 14 }

3.3 Größter gemeinsamer Teiler

Verglichen werden zwei ganze Zahlen a und b die nicht beide 0 sind. Es existiert mindestens eine Zahl t der natürlichen Zahlen, die a und b teilt, denn 1 ist der Teiler von jeder Zahl.

$$1 <= t < |a| \text{ oder } |b|$$

Da der gemeinesame Teiler t eine endliche Menge darstellt, muss es einen größten Teiler geben. Dieser wird größter gemeinsamer Teiler von a und b genannt.

$$ggT\ (a, b)$$

Beispiel:

ggT(8,14) = 2

ggT(11,24) = 1

3.4 Division mit Rest

Mathematisch ist die Division mit Rest wie folgt aufgebaut:

$a, b, q, r \in \mathbb{Z}$ und $a \neq 0$

$b = q*a + r$ und $0 \leq r < |a|$

Der Restoperator Modulo wird durch % dargestellt.

r = b % a

r nimmt den ganzzahligen Rest der Division von b / a an.

Beispiel:

17 % 5 = 2

3.5 Prime Restklassengruppen $\mathbb{Z}_n^*$

Prime Restklassengruppen spielen in der Kryptologie eine wichtige Rolle. Verschlüsselungsverfahren die auf dem diskreten Logarithmusproblem oder der Primfaktorzerlegung aufbauen, basieren auf einer Exponentialfunktion mit Restbildung, deren Wertebereich innerhalb der Elemente einer primen Restklassengruppe ist.

Die multiplikative Gruppe $\mathbb{Z}_n^*$ wird aus den zu n teilerfremden Elementen gebildet. Bei primen Restklassengruppen ist die Ordnung stets n-1. Jedes Ergebnis einer Multiplikation innerhalb dieser Gruppe, weist auf ein Element dieser Gruppe, der Wertebereich ist endlich und kann, aufgrund der Restoperation nicht überschritten werden.

Die Gruppenelemente sind wie folgt berechnet:

$ZW_i * SW_k \% n$

$i,k = \{1, 2, \ldots, n-1\}$

ZW = Zeilenwert

SW = Spaltenwert

Beispiel: $\mathbb{Z}_7^*$

$\varphi(7) = 7 - 1 = 6$

	1	**2**	**3**	**4**	**5**	**6**
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

3.6 Primitivwurzeln:

Primitivwurzeln sind bestimmte Elemente von primen Restklassengruppen $\mathbb{Z}_n$*. Eine primitivwurzeleigenschaft liegt vor, wenn ein Element mit allen Elementen der Gruppe exponiert und anschließend über die Restbildung mit n, jedes Element der Restklassengruppe darstellt.

Die Anzahl vorhandener Primitivwurzeln in einer primitiven Restklassengruppe $\mathbb{Z}_n$* wird über die zweifach angewandte Phi-Funktion von n ermittelt.

φ(φ(n)) = Anzahl der Primitivwurzeln

Die Gruppenelemente sind wie folgt berechnet:

ZW_i ^ SW_k % n

i,k = {1, 2, …, n-1}

ZW = Zeilenwert

SW = Spaltenwert

Beispiel: $\mathbb{Z}_7$*

φ(φ(7))= φ(7 – 1) = φ(6) = 2

Es existieren 2 Elemente, die eine primitivwurzeleigenschaft besitzen.

	1	**2**	**3**	**4**	**5**	**6**
1	1	1	1	1	1	1
2	2	4	1	2	4	1
3	3	2	6	4	5	1
4	4	2	1	4	2	1
5	5	4	6	2	3	1
6	6	1	6	1	6	1

Da die Elemente 3 und 5 nehmen jeden möglichen Wert an und sind daher Primitivwurzeln.

4. Diffie-Hellman-Schlüsselaustauschverfahren

4.1 Einleitung

Begründung des ersten Public-Key-Verfahrens 1976 von Whitefield Diffie und Martin Hellman.

Der Diffie-Hellman Algorithmus wird zur Erzeugung einer sicheren Schlüsselvereinbarung zwischen zwei oder mehr Teilnehmern über einen unsicheren Kanal verwendet, um den darauf folgenden geheimen Informationsaustausch zu ermöglichen.

Die Sicherheit des Verfahrens beruht auf dem diskreten Logarithmus Problems.

4.2 Diskreter Logarithmus

Der diskrete Logarithmus ist vergleichbar mit dem natürlichen Logarithmus, wobei sich der diskrete Logarithmus auf den ganzzahligen Wertebereich innerhalb einer zyklischen Gruppe beschränkt. Die Umkehrfunktion ist die diskrete Exponentiation.

Das Diffie-Hellman-Verfahren basiert auf folgender diskreten Exponentialfunktion mit Restbildung:

$$\mathbf{y = g^x \bmod p}$$

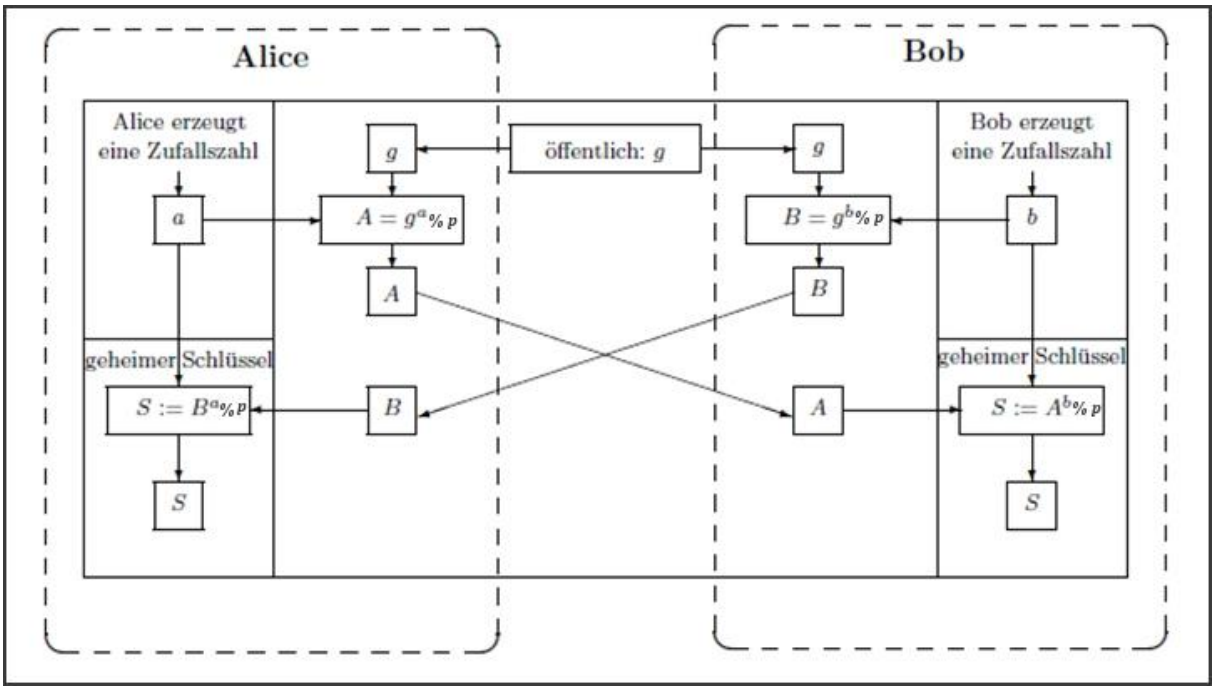
Die Umkehrung (bei gegebenen y, g, p $\in \mathbb{Z}$) bzw. das logarithmieren zur Bestimmung des gesuchten x ist nur mit hohem Aufwand durchzuführen und es existiert bis heute kein Algorithmus, der dies in akzeptabler Zeit durchführt. Die Exponentiation hingegen, lässt sich aus komplextheoretischer Sicht effizient berechnen.

4.3 Einwegfunktionsaufbau $y = g^x \bmod p$

Die Funktionsparameter g und p sind öffentlich und für jeden Kommunikationspartner bekannt. Der Parameter p ist eine Primzahl. Der Exponent x ist eine positive ganze Zahl, die von jedem Teilnehmer frei gewählt wird. Der Parameter g bzw. die Basis g ist eine Primitivwurzel der Primzahl p. Hierdurch nimmt das errechnete y jeden Wert zwischen 1 und p-1 an. Diese Maximalzahl verschiedener y-Werte erzwingen ein größtmögliches probieren um den richtigen x-Wert zu erhalten, da jedes Element dieser primen Restklassengruppe den Wertebereich darstellt.

Auch wenn eine unbekannte dritte Person die Parameter g, p oder die Zwischenergebnisse A oder B abfängt, ist aufgrund der schwierigen Berechnung des diskreten Logarithmus die Sitzung nicht in Gefahr.

4.4 Ablauf des Diffie-Hellman Verfahrens



Vereinbarung der öffentlichen Parameter **g** und **p**

Alice wählt eine Zufallszahl **a** und hält sie geheim.

Bob wählt eine Zufallszahl **b** und hält sie geheim.

Alice berechnet $\mathbf{A = g^a \bmod p}$

Bob berechnet $\mathbf{B = g^b \bmod p}$

Alice sendet Bob das errechnete **A**

Bob sendet Alice das errechnete **B**

Alice berechnet den geheimen Schlüssel $\mathbf{S = B^a}$

Bob berechnet den geheimen Schlüssel $\mathbf{S = A^b}$

Der ermittelte Sitzungs-Schlüssel S kann jetzt, von alle Kommunikationspartnern, für den weiteren Informationsaustausch über ein asymmetrisches Verschlüsselungsverfahren verwendet werden.

5. RSA

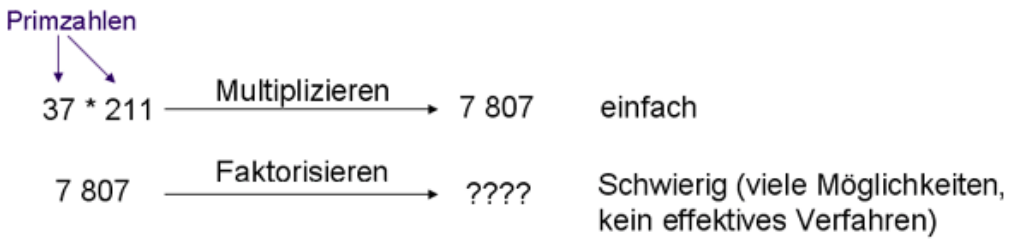
5.1 Einleitung

Der RSA Algorithmus wurde 1978 von den Erfindern Linn Rivest, Adi Shamir und Leonard Adleman aufgestellt. Dieses Verfahren basiert auf einer Einwegfunktion mit Falltür und gehört zu den asymmetrischen Verschlüsselungsverfahren.

Bei asymmetrischer Verschlüsselung wird je ein Schlüssel für die Verschlüsselung sowie der Entschlüsselung benötigt. Die dafür notwendige Einwegfunktion wird mittels der schwer zu berechnenden Primfaktorzerlegung einer definierten Zahl realisiert. Beim RSA Verfahren veröffentlicht der Empfänger ein Schlüsselpaar mit dem andere Nachrichten Verschlüsseln und diese dem Empfänger zusenden können. Die Verschlüsselte Nachricht kann mit Hilfe der Zusatzinformation (privater Schlüssel), die nur der Empfänger besitzt, entschlüsselt werden.

5.2 Primfaktorzerlegung

Wie in **KapitelXX** beschrieben, lässt sich jede Zahl > 3, die keine Primzahl ist, in ihre Primfaktoren zerlegen. Die Primfaktorzerlegung großer Zahlen ist aus komplextheoretischer Sicht ein Rechen- bzw Zeitintensives Problem, für das bis heute kein effizienter Algorithmus existiert. Im Gegensatz zur Zerlegung, ist die Multiplikation von zwei Primzahlen leicht zu berechnen.



Aufgrund der leichten Multiplikation und der ineffizienten Umkehrung wird diese Einwegfunktion bei Verschlüsselungsverfahren wie z.B. beim RSA verwendet.

5.3 Verschlüsselungsfunktion $y = x^d \bmod n$

Der Funktionsparameter **x** ist der zu verschlüsselnde Klartext bzw. die in ASCII-Code umgewandelte Nachricht.

Der Parameter **n** ist die vom Empfänger errechnete Primzahlmultiplikation und stellt den ersten Teil des öffentlichen Schlüssels dar **(n, d)**.

Der Exponent **d** ist der zweite Teil des öffentlichen Schlüssels **(n, d)** und für jeden bekannt. Der Wert von **d** ist ein vielfaches der Zahl **e.** Desweiteren müssen **e*d** und **φ(n)** teilerfremd sein.

Das Funktionsergebnis **y** ist die fertig Verschlüsselte Nachricht.

5.4 Entschlüsselungsfunktion $y = x^e \bmod n$

Der Funktionsparameter **x** ist die zu entschlüsselnde Nachricht.

Der Parameter **n** ist die vom Empfänger errechnete Primzahlmultiplikation und stellt den ersten Teil des privaten Schlüssels dar **(n, e)**.

Der Exponent **e** ist der zweite Teil des privaten Schlüssels **(n, e)** und nur für den Empfänger bekannt. Der Wert von **e** muss größer als 1 und kleiner als das Ergebnis der Phi-Funktion von **n** sein. Darüber hinaus müssen **e** und **φ(n)** teilerfremd sein.

Das Funktionsergebnis **y** ist die fertig Entschlüsselte Nachricht.

5.5 Ablauf des RSA Verfahrens

Schlüsselgenerierung (privat und öffentlich)

Der (künftige) Empfänger wählt zwei große Primzahlen p und q und generiert den öffentlichen Schlüssel:

$$n = p*q$$

$$e = 1 < e < \varphi(n) \text{ und } ggT(e, \varphi(n)) = 1$$

Der (künftige) Empfänger generiert den privaten Schlüssel:

$$d = e*d \text{ und } ggT (e*d, \varphi(n)) = 1$$

Verschlüsselung mit öffentlichem Schlüssel

Der Sender erfragt den öffentlichen Schlüssel des Empfängers.

Die Nachricht wird mit dem öffentlichen Schlüsselpaar codiert: geheim = $x^d \bmod n$

Der Sender verschickt die Nachricht „geheim“ an den Empfänger.

Entschlüsselung mit privatem Schlüssel

Nach erfolgreichem Empfang der Nachricht „geheim“, kann diese mit dem privaten Schlüsselpaar decodiert werden: Nachricht = geheim ^{d} mod n.

6. Quellen

http://www.cryptool.org/images/ctp/documents/CrypToolScript-de.pdf

http://www.r-krell.de/if-java-k.htm

http://www2.informatik.hu-berlin.de/Forschung_Lehre/algorithmenII/Lehre/WS2002-2003/Perlen/Variationen.pdf

http://www.hh.schule.de/julius-leber-schule/melatob/referat1.html

http://de.wikipedia.org/wiki/Eulersche_%CF%86-Funktion

http://de.wikipedia.org/wiki/Hashfunktion

http://de.wikipedia.org/wiki/Prime_Restklassengruppe

http://de.wikipedia.org/wiki/Primitivwurzel

Albrecht Beutelsbacher, Marc-Alexander Zschiegner

Diskrete Mathematik für Einsteiger, 2. Auflage

Paar, C. , Pelzl J.: Understanding Cryptography,

Verlag: Springer-Verlag Berlin Heidelberg ; 2010