

# Lektion 2

## Symmetrische Kryptosysteme

#### LERNZIELE

Nach der Bearbeitung dieser Lektion werden Sie gelernt haben, …

… welche grundlegenden symmetrischen kryptografischen Algorithmen es gibt: Substitution und Transposition.

… welche die einzige kryptografisch vollkommen sichere Chiffre ist: das One-Time-Pad.

… was die modernen Algorithmen Data Encryption System und Advanced Encryption System ausmacht.

… welche vielfältigen Einsatzmöglichkeiten es für (kryptografische) Hashfunktionen gibt.

DL-E-DLMCSEAITSC02-U02

1. Symmetrische Kryptosysteme

### Einführung

Bis zum Ende der 1970er Jahre, vor den Veröffentlichungen von Diffie und Hellman (1976) und Rivest et al. (1978), waren alle (bekannten) kryptografischen Algorithmen symmetrisch (oder Ein-Schlüssel-Verfahren), d. h. sie verwendeten denselben Schlüssel zum Ver- und Entschlüsseln. Jeder historische Algorithmus, so ausgeklügelt er auch gewesen sein mag, sei es die Cäsar-Chiffre, die Skytale oder die Enigma, war symmetrisch.

Symmetrische Algorithmen verarbeiten die Eingabe als Zeichenkette (aus Bits oder Buchstaben) durch (iterierte) Substitutionen und Transpositionen. Im Gegensatz dazu beruhen asymmetrische Algorithmen auf einem rechnerisch schwierigen Problem, z. B. der Faktorisierung einer zusammengesetzten Zahl in ihre Primfaktoren, und betrachten die Eingabe als eine natürliche Zahl. Die einzige vollkommen sichere Chiffre ist das One-Time-Pad, bei dem der Schlüssel so lang ist wie der Klartext, und der Geheimtext wird durch buchstabenweises Hinzufügen jedes Buchstabens des Schlüssels zu dem entsprechenden (d. h. an der gleichen Stelle befindlichen) Buchstaben des Klartextes entschlüsselt. Für komplexere Nachrichten, z. B. Text-, Bild- oder Videonachrichten, ist ein großer Schlüssel jedoch unpraktisch. Heute bedeutet das, dass zum Verschlüsseln einer Festplatte eine weitere Festplatte benötigt wird, auf der sich der Schlüssel befindet. Um die kürzere Schlüssellänge zu kompensieren, erzeugen moderne Algorithmen im Idealfall so viele Verflechtungen, dass sie eine nahezu perfekte Diffusion erreichen. Mit anderen Worten: Die Änderung eines einzigen Bits der Eingabe oder des Schlüssels bewirkt die Änderung von etwa der Hälfte der Ausgabebits. Moderne Algorithmen wie der Data Encryption Standard (DES) oder der Advanced Encryption Standard (AES) sind Blockchiffren mit Substitutions- und Permutationsverfahren, d. h., sie verschlüsseln jeweils ein Datenpaket durch wiederholte Transpositionen und Substitutionen.

### Substitution und Transposition

Die beiden grundlegenden Verfahren zur Verschlüsselung sind Transposition und Substitution:

* Eine Transposition verändert die Reihenfolge der Zeichen im Text (d. h. transponiert oder permutiert diese), nicht aber die Zeichen selbst.
* Bei einer Substitution wird jedes Zeichen im Text durch ein anderes (eine andere Gruppe von) Zeichen ersetzt, jedoch nicht die Reihenfolge der Zeichen geändert.

Die historisch gesehen prototypischen Algorithmen für diese beiden Verfahren sind:

* Die Cäsar-Substitutions-Chiffre, die jeden Buchstaben im Klartext um drei Positionen vorrückt und z. B. A als D, B als E usw. verschlüsselt, sowie
* die Permutation des Klartextes durch die Skytale (Stab von Licurgo), bei der das Pergamentband um den Stab gewickelt und der Text horizontal darauf geschrieben wird.

Symmetrische Kryptosysteme

Wir werden sehen, dass ein Algorithmus selbst bei vielen möglichen Schlüsseln (wie z. B. der durch eine beliebige Permutation des Alphabets gegebene, der fast 280 Schlüssel hat) leicht geknackt werden kann, wenn er Regelmäßigkeiten, wie die Häufigkeit der Buchstaben, beibehält. Substitutions- und Permutationsnetze verbinden und iterieren diese beiden komplementären prototypischen Algorithmen, um dieses Ziel zu erreichen.

###### Substitutions-Chiffren

Bei einer Substitutions-Chiffre bestimmt der Schlüssel die Substitutionen des Klartext-alphabets (betrachtet als eine Menge von Einheiten von Zeichen wie einzelne Buchstaben oder Buchstabenpaare) durch das Geheimtextalphabet. Wenn zum Beispiel sowohl die Einheiten des Klartextes als auch die des Geheimtextes Buchstaben des lateinischen Alphabets sind, dann vertauscht eine Substitution die Buchstaben des lateinischen Alphabets. Handelt es sich bei der Substitutionschiffre um eine monoalphabetische Chiffre (z. B. die Cäsar-Chiffre), so wird auf jeden Buchstaben des Klartextes dieselbe Substitution angewendet, unabhängig von seiner Position. Wenn die Substitutions-Chiffre polyalphabetisch ist (wie die Enigma), variiert die Substitution mit der Position des Buchstabens im Klartext. Zum Verschlüsseln wird jede alphabetische Einheit des Klartextes durch die substituierte alphabetische Einheit ersetzt und zum Entschlüsseln entsprechend invertiert. Die monoalphabetische Substitution gilt als unsicher, weil die Häufigkeit der Zeichen, die im Klartext und im Geheimtext zu finden sind, gleichbleibt. Im Englischen zum Beispiel genügen etwa 25 Buchstaben des Geheimtextes für eine Kryptoanalyse. Der vorwiegende Ansatz, um diese Erhaltung der Häufigkeit von Einzelbuchstaben im Geheimtext zu reduzieren, ist die so genannte polyalphabetische Substitution – die Verwendung mehrerer Geheimalphabete.

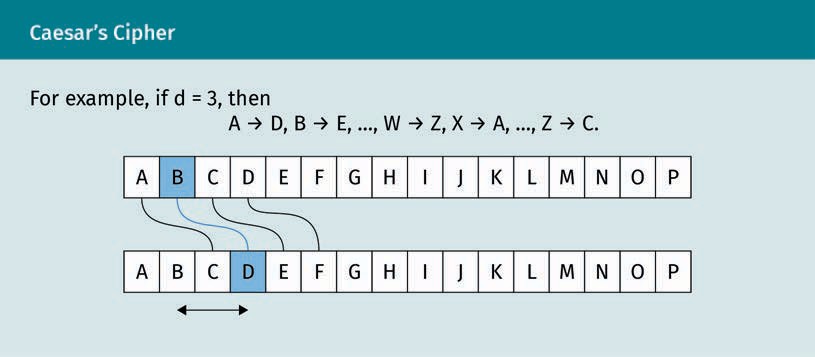
Verschiebung um einen festen Abstand

Die einfachste Substitutions-Chiffre ist eine zyklische Verschiebung des Klartextalphabets, die so genannte Cäsar-Chiffre. Die römischen Kaiser Cäsar (100 bis 44 v. Chr.) und Augustus (63 v. Chr. bis 14 n. Chr.) bedienten sich dieser Methode (Bauer, 2016). Wir legen einen Abstand d zwischen den Buchstaben in alphabetischer Reihenfolge fest, d. h. eine Zahl zwischen 0 und 25, und verschieben dann jeden Buchstaben des (lateinischen) Alphabets um diesen Abstand d (nach vorne). Wir stellen uns dabei vor, dass Alphabet sei kreisförmig, d. h. die Buchstaben sind in einem Ring angeordnet, so dass die Verschiebung eines Buchstabens am Ende des Alphabets zu einem Buchstaben am Anfang des Alphabets führt.

Substitutions-Chiffre Bei dieser Chiffre wird jede alphabetische Einheit des Klartextes durch eine entsprechende alphabetische Einheit ersetzt.

Cäsar-Chiffre

Diese Substitutions-Chiffre verschiebt die alphabetische Position jedes Klartextbuchstabens um den gleichen Abstand.



Es gibt 26 Schlüssel (einschließlich des trivialen Schlüssels d = 0). Bei der Cäsar-Chiffre wird jeder Buchstabe des Alphabets um den gleichen Abstand verschoben. Zur Entschlüsselung wird jeder Buchstabe um den negativen Abstand –d verschoben, d. h. um d Positionen nach hinten.

Substitution durch Permutation der Buchstaben des Alphabets

Anstatt jeden Buchstaben durch einen um den gleichen Abstand d verschobenen Buchstaben zu ersetzen, können wir jeden Buchstaben durch einen beliebigen Buchstaben ersetzen, wie im folgenden Beispiel.

A B … Y Z

…

E Z … G A

Wir mischen die Buchstaben untereinander. Auf diese Weise erhalten wir 26 · 25 · 1 = 26! > 1026

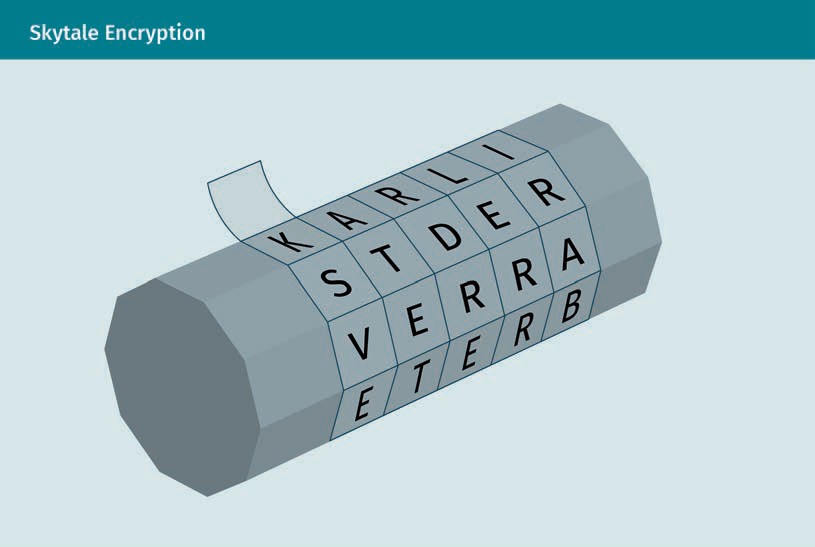
Schlüssel (was in etwa der Anzahl von Passwörtern entspricht, die mit 80 Bit möglich sind).

###### Transpositions- bzw. Permutations-Chiffren

Eine Transpositions-Chiffre verschlüsselt den Klartext durch Permutation seiner Einheiten (und entschlüsselt ihn durch die umgekehrte Permutation). Jede alphabetische Einheit bleibt gleich; die Verschlüsselung hängt lediglich von der Position der Einheiten ab.

Die Skytale ist ein Stab, mit dem die Spartaner wie folgt verschlüsselt haben: (1) das Band wird um einen schmalen Stab gewickelt, (2) dieses Band wird horizontal, also entlang der längeren Seite beschrieben und (3) das Band wird abgewickelt (Bauer, 2016). Die auf dem Band transponierten Buchstaben konnten nur mit einem Stab gleichen Umfangs (dem Schlüssel) auf die gleiche Weise entziffert werden, wie der Text verschlüsselt wurde: Das Band wird wieder um den Stab gewickelt und der Text darauf horizontal entlang der längeren Seite abgelesen.

Symmetrische Kryptosysteme



In diesem Fall ist der Schlüssel die Zahl, die sich aus der Anzahl der Buchstaben ergibt, die um den Umfang des Stabes herum angeordnet sind.

###### Sicherheit historischer Beispiele

Wenden wir nun die festgelegten Sicherheitskriterien auf die Substitutions-Chiffre an.

Cäsar-Chiffre

Diese einfache Substitutions-Chiffre verstößt gegen alle wünschenswerten Eigenschaften. Das Kerckhoffs‘sche Prinzip besagt zum Beispiel, dass der Algorithmus öffentlich ist, sobald die Methode bekannt ist. In Anbetracht der geringen Anzahl von 25 Schlüsseln lässt sich der Geheimtext in kurzer Zeit durch einen Brute-Force-Angriff (Durchprobieren aller Möglichkeiten) knacken.

Substitution durch willkürliche Permutation der Buchstaben des Alphabets

Eine Ersetzung durch eine beliebige Permutation der Buchstaben des Alphabets hat 26 · 25 · … · 1 = 26! > 1026 Schlüssel, so dass ein Brute-Force-Angriff rechnerisch nicht durchführbar ist.

Dies verstößt jedoch gegen die Ziele der Diffusion und der Konfusion. Wenn der Schlüssel (Permutation des Alphabets) den Buchstaben α gegen den Buchstaben β austauscht, dann ergibt sich:

* eine geringe Konfusion, da die Substitution von β im Schlüssel nur die Substitution jedes Buchstabens β im Geheimtext impliziert,
* eine geringe Diffusion, da die Substitution eines Buchstabens α im Klartext nur die Ersetzung des entsprechenden Buchstabens β im Geheimtext impliziert.

Brute-Force-Angriff

Diese erschöpfende Schlüsselsuche überprüft jeden möglichen Schlüssel.

###### Produktchiffre

Produktchiffre Eine Zusammensetzung von Chiffren, bei der die Ausgabe einer Chiffre als Eingabe für die nächste dient.

Eine Produktchiffre setzt Chiffren zusammen, d. h. bei einem zweifachen Produkt ist die Ausgabe der einen Chiffre die Eingabe der anderen. Der Geheimtext der Produktchiffre ist der Geheimtext der endgültigen Chiffre. Kombinieren wir Transpositionen nur mit Transpositionen oder Substitutionen nur mit Substitutionen, ist die erhaltene Chiffre wieder eine Transposition oder Substitution und kaum sicherer. Eine Mischung aus beiden, eine Transposition mit Substitutionen, kann die Chiffre jedoch tatsächlich sicherer machen.

Eine Fraktionierungs-Chiffre ist eine Produktchiffre, bei der jedes Zeichen im Klartext durch eine Gruppe von Zeichen (in der Regel Paare) ersetzt und der erhaltene Geheimtext anschließend transponiert wird. Die bekannteste Fraktionierungschiffre war die ADFGVX-Chiffre, die von den deutschen Streitkräften während des Ersten Weltkriegs verwendet wurde (Bauer, 2016). Bei dieser Chiffre wurden die 26 Buchstaben des lateinischen Alphabets und zehn Ziffern in einer 6 × 6-Tabelle angeordnet und durch die Buchstabenpaare A, D, F, G, V und X ersetzt, welche die Zeile und Spalte des Buchstabens oder der Ziffer angaben. Der resultierende Text wurde wie üblich von links nach rechts in die Zeilen einer Tabelle geschrieben und dann jede Spalte in der durch ein Schlüsselwort angegebenen Reihenfolge gelesen.

Wir werden uns nun ansehen, wie moderne Chiffren diese Idee einer Produktchiffre weiterentwickeln, um eine hohe Diffusion zu erreichen.

### Blockchiffren

Bei den klassischen Chiffren wurden in der Regel einzelne Buchstaben und manchmal auch Buchstabenpaare ersetzt. Systeme, die mit Dreiergruppen (Trigrammen) oder größeren Buchstabengruppen arbeiteten, wurden als zu umständlich angesehen und fanden keine breite Anwendung.

Stattdessen ist es sicherer, einen ganzen Block von Buchstaben anstelle eines einzelnen Buchstabens entsprechend dem Schlüssel zu substituieren. Das Alphabet dieses Austauschverfahrens wäre allerdings riesengroß, so dass dieses Ideal praktisch unerreichbar ist, insbesondere bei einer so beschränkten Hardware wie einer Chipkarte mit einem 8-Bit-Prozessor. Für einen Block von 16 Byte hätte diese Substitutionstabelle zum Beispiel die horrende Größe von 2128 • 16 Byte. In der modernen Kryptografie mit einem Schlüssel beträgt die Informationseinheit jedoch in der Regel 128 Bit, d. h. etwa 27 Buchstaben, während in der auf dem RSA-Algorithmus basierenden Kryptografie mit zwei Schlüsseln üblicherweise Einheiten von 2048 Bit, also etwa 620 Buchstaben, verwendet werden.

So ersetzt AES zum Beispiel nur jedes Byte, jeden Eintrag in einem Block – eine Ersetzungstabelle von 28 = 256 Einträgen von 1 Byte (und transponiert anschließend die Einträge). Wir werden sehen, dass sich diese Verfahren so gut ergänzen, dass sie praktisch so sicher sind wie eine Substitution des gesamten Blocks.

Symmetrische Kryptosysteme

###### Block- und Stromchiffren

Bei einer Blockchiffre wird der Klartext in Blöcke gleicher Größe unterteilt und jeder Block mit einem gemeinsamen Schlüssel verschlüsselt; ein Block könnte zwar aus einem einzigen Zeichen bestehen, ist aber normalerweise größer. Bei DES beträgt die Blockgröße beispielsweise 64 Bit, bei AES 128 Bit.

Eine Stromchiffre unterteilt den Klartext in Einheiten, die normalerweise aus einem einzigen Zeichen bestehen und verschlüsselt dann die i-te Einheit des Klartextes mit der i-ten Einheit eines Schlüsselstroms. Beispiele sind das One-Time-Pad, Rotor-Chiffriermaschinen (wie die Enigma) und Triple DES (bei dem die Ausgabe einer Verschlüsselung die Eingabe der nächsten Verschlüsselung ist). Bei einer Stromchiffre muss derselbe Abschnitt des Schlüsselstroms, der zur Verschlüsselung verwendet wurde, auch zur Entschlüsselung verwendet werden. Daher müssen die Schlüsselströme von Absender und Empfänger zu Beginn und danach ständig synchronisiert werden.

###### Feistelchiffren

Eine Feistelchiffre oder ein Substitutions- und Permutationsnetzwerk (SPN) gliedert den Text (Bytefolge) in n-Byte-Blöcke (z. B. n = 16 für AES) und verschlüsselt jeden Block durch Iteration (z. B. zehnmal bei AES und fünfmal in unserem prototypischen Modell) der folgenden drei Schritte:

* + 1. Addieren (XOR) des Schlüssels,
    2. Substituieren des Alphabets (das in Teilblöcken des Blocks arbeitet, z. B. auf jedem Byte), und
    3. Permutieren aller Teilblöcke eines Blocks.

Nach der Addition (XOR) des Schlüssels, der Ersetzung des Alphabets, z. B. im AES-Algorithmus (jedes Byte, hexadezimales Buchstabenpaar, durch ein anderes), und der Permutation des Textes (aus dem aktuellen Schritt, genannt „Zustand“), z. B. im AES, werden die Einträge in jeder Zeile (und den Spalten) permutiert.

Diese beiden einfachen Verfahren, die Substitution des Alphabets und die Permutation des Textes, ergänzen sich gut. Sie führen nach wenigen Iterationen zu starker Konfusion und Diffusion. In der ersten und letzten Runde werden die Schritte vor und nach dem Hinzufügen des Schlüssels weggelassen, da sie die kryptografische Sicherheit nicht erhöhen. Da der Algorithmus öffentlich ist (Kerckhoffs‘sches Prinzip), ist jeder Angreifer in der Lage, alle Schritte rückgängig zu machen, die keine Kenntnis des Schlüssels erfordern. Obwohl sich eine Feistelchiffre also von klassischen Kryptosystemen zu unterscheiden scheint, handelt es sich tatsächlich um eine Produktchiffre aus Transpositionen und Substitutionen.

Stromchiffre und Blockchiffre im Vergleich

Eine Stromchiffre arbeitet mit einzelnen Zeichen (z. B. einzelnen Bytes), während eine Blockchiffre mit Gruppen von Zeichen (z. B. jeweils 16 Bytes groß) arbeitet.

Substitutions- und Permutationsnetzwerk

Bei dieser Chiffre wird jeder Block nach dem Hinzufügen eines Schlüssels iterativ substituiert und permutiert.

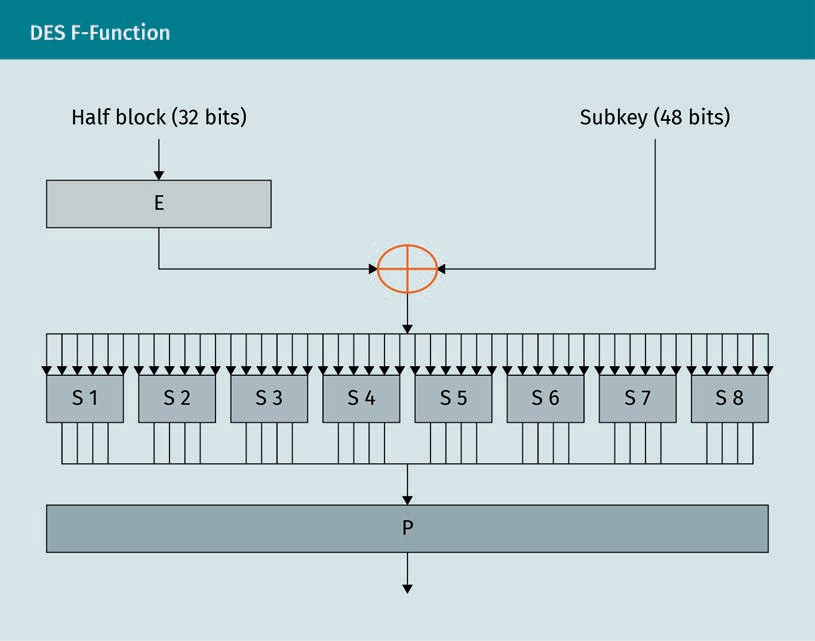
### Data Encryption Standard (DES)

Data Encryption  
Standard  
Diese Blockchiffre hat eine Schlüssellänge von 56 Bit. Sie wurde von Horst Feistel von IBM entwickelt.

Der Data Encryption Standard (DES) wurde 1977 zu einem öffentlichen Standard, nachdem mit ihm ein vom US National Bureau of Standards (heute National Institute of Standards and Technology, NIST) ausgeschriebener öffentlicher Wettbewerb gewonnen wurde. Der DES umfasst 16 Iterationen, bei denen Substitution und Permutation (Transposition) durchgeführt werden. Die Blockgröße beträgt 64 Bit, der Schlüssel besteht aus 64 Bit. Allerdings können nur 56 Bit vom Benutzer ausgewählt werden und bilden den Schlüssel: die restlichen acht sind redundante Paritätsprüfbits.

Wie der Name vermuten lässt, handelt es sich um eine Feistelchiffre oder ein Substitutions- und Permutationsnetzwerk, ähnlich dem oben beschriebenen Prototyp. Der Algorithmus gruppiert den Text (eine Bytefolge) in 32-Bit-Blöcke mit Teilblöcken von vier Bit und verschlüsselt jeden Block in 16 Iterationen der folgenden drei Schritte, die als Feistelfunktion oder F-Funktion bezeichnet werden, in der nachstehender Reihenfolge:

1. Addieren (XOR) des Schlüssels,
2. Substituieren jedes 4-Bit-Teilblocks des Blocks durch die S-Box (in hexadezimaler Schreibweise),
3. Permutieren aller Teilblöcke.



In jeder Runde i wird die Ausgabe der vorangegangenen Runde in die 32 am weitesten links liegenden Bits L(i) und die 32 am weitesten rechts liegenden Bits R(i) aufgeteilt. R(i) wird zu L(i + 1), während R(i + 1) die Ausgabe einer komplexen Funktion, L(i) + f(R(i)*,* K(i + 1)) ist, deren Eingabe der i + 1-te Block der Schlüsselbits, K(i + 1) und die gesamte vorangegangene Zwischenchiffre ist. Dieser Vorgang wird 16-mal wiederholt.

Entscheidend für die Sicherheit von DES ist die nichtlineare S-Box der vom NIST spezifizierten F-Funktion f. Sie ist nichtlinear, d. h. f(A) + f(B) ≠ f(A + B), und maximiert zugleich die Konfusion und Diffusion im Sinne der Shannon-Kriterien für eine sichere Chiffre.

###### Schlüsselgröße und die Geburt von 3DES

Die Sicherheit des ist wie bei jedem Algorithmus nicht größer als der Aufwand für die Suche nach 256 Schlüsseln. Was 1977 noch als undurchführbare Aufgabe für einen Computer galt, schaffte eine spezielle DES-Suchmaschine zwei Jahrzehnte später in drei Tagen. Ein Workaround, Triple DES oder 3DES genannt, wurde entwickelt, welcher dem DES mit einem 112-Bit-Schlüssel die doppelte Schlüsselgröße gegenüber den DES-Schlüsseln verleiht, indem einfach zwei normale DES-Schlüssel verwendet werden.

Die Verschlüsselung lautet dann V(1) (E(2) (V(1), also: (1) verschlüsseln mit dem ersten Schlüssel, (2) entschlüsseln mit dem zweiten Schlüssel und (3) verschlüsseln mit dem ersten Schlüssel.

Die Entschlüsselung lautet E(1) (V(2) (E(1) bzw. (1) entschlüsseln mit dem ersten Schlüssel, (2) verschlüsseln mit dem zweiten Schlüssel und (3) entschlüsseln mit dem ersten Schlüssel.

Wenn die beiden Schlüssel übereinstimmen, wird dieses Kryptosystem zu einem gewöhnlichen DES mit nur einem Schlüssel. Somit ist Triple-DES rückwärtskompatibel mit Anlagen, die für (Single-)DES implementiert wurden. DES war der erste kryptografische Algorithmus, der das Kerckhoffs‘sche Prinzip erfüllte.

### Advanced Encryption Standard (AES)

Im Januar 1997 kündigte das NIST einen öffentlichen Wettbewerb für einen sogenannten Advanced Encryption Standard (AES) an, der den früheren symmetrischen Verschlüsselungsstandard, den Data Encryption Standard (DES), ersetzen sollte. Nachdem der 56-Bit-Schlüssel des DES mit einer Brute-Force-Attacke gefunden wurde, verlangten die NIST-Spezifikationen für den AES eine vergrößerte Schlüssellänge. Der AES-Algorithmus war der Gewinner dieses Wettbewerbs (Bauer, 2016).

###### Beurteilung der Sicherheit

Die Erfinder von AES konnten nachweisen, dass sich diese beiden Verfahren so gut ergänzen, dass sie nach mehreren Iterationen das fehlende Ersetzen des gesamten Blocks durch einen anderen nahezu ausgleichen (Daemen & Rijmen, 1999; 2002). Wie es schon bei DES der Fall war, so ist auch AES Jahrzehnte nach seiner Einführung noch immer gegen alle Angriffe durch Kryptoanalyse gewappnet. Dennoch ist aufgrund seiner anpassbaren Schlüsselgröße nicht zu erwarten, dass er durch Entwicklungen in der Computertechnik geknackt wird, wie es bei DES der Fall war.

3DES

Dieser Begriff bezeichnet die dreifache Anwendung von DES mit der doppelten Schlüsselgröße des DES-Algorithmus.

Advanced Encryption Standard

Dieses Substitutions- und Permutationsnetzwerk mit einer (variablen) Schlüssellänge von in der Regel 128 Bit wurde im Jahr 2000 als Nachfolger von DES zu einem kryptografischen Standard.

###### Anwendungsfälle

Rijndael zeichnete sich durch seine Einfachheit und seine besonders rechenökonomische Implementierung aus. Es war nicht nur ein sicheres Verschlüsselungsverfahren, sondern dank seines eleganten und einfachen Designs auch klein genug, um auf Chipkarten implementiert zu werden (weniger als 10.000 Byte Code). Bis heute gilt er als der sicherste Algorithmus. Es besteht kein Bedarf an einem weiteren symmetrischen kryptografischen Standardalgorithmus, und tatsächlich wird er überall eingesetzt. Zur Verschlüsselung eines drahtlosen Netzwerks wird beispielsweise ein Schlüssel verwendet, so dass die Verschlüsselung symmetrisch ist. Die sicherste und daher empfehlenswerteste Option ist die Verschlüsselung mit AES.

###### Algorithmus

AES ist eine Blockchiffre, d. h., sie arbeitet mit Byte-Blöcken, in der Regel mit einer Größe von 16 Byte (AES-128).

|  |  |
| --- | --- |
| SubBytes | Ersetzen jedes Bytes des Blocks gemäß einer Substitutionstabelle (S-Box) mit 28 = 256 Einträgen von 1 Byte. |
| ShiftRows | Zyklisches Verschieben der Einträge der letzten drei Zeilen |
| MixColumns | Hinzufügen der Spalten dazwischen. |

###### Blockweise Verschlüsselung

Der AES-Algorithmus gruppiert den Klartext (und die Schlüssel) in 4×B Byte-Rechtecke, wobei B die Anzahl der Spalten des Blocks ist: 4, 6 oder 8. Üblicherweise, und im Folgenden gilt das auch für uns, ist B = 4, d. h. das Rechteck ist ein Quadrat. Auf hexadezimaler Basis hat ein solches Quadrat beispielsweise diese Form:

A1 13 B1 4A A3 AF 04 1E 3D 13 C1 55

B1 92 83 72

Symmetrische Kryptosysteme

###### Der Rijndael-Binärkörper

Um den AES zu verstehen, sind einige mathematische Definitionen von Gruppen und Feldern sowie Grundkenntnisse im Rechnen mit Polynomen nötig. Die nachfolgenden mathematischen Darstellungen erläutern die AES-Funktionen SubBytes, ShiftRows und MixColumns im Detail.

Gruppen und Felder

Eine Gruppe ist eine Menge G mit

* einer Operation • : G × G → G, die das Assoziativgesetz erfüllt,
* einem neutralen Element e, sodass gilt x • e = x = e • x für alle x in G, und
* einem inversen Element y für jedes Element x, sodass gilt x • y = e = y • x.

Allgemein wird die Operation mit • bezeichnet, das neutrale Element mit 1 und das inverse Element von x in G mit x-1.

Beispiel

Die Menge der rationalen Zahlen ungleich Null \*\* mit der Multiplikationsoperation • ist eine Gruppe. Ist die Gruppe G kommutativ (die Operation erfüllt das Kommutativgesetz), so wird die Operation mit +, das neutrale Element mit Null und das inverse Element von x in G mit –x bezeichnet.

Beispiel

Die Menge der rationalen Zahlen \* mit der Additionsoperation + ist eine kommutative Gruppe. Ein „Körper“ ist eine Menge F mit einer Additions- und Multiplikationsoperation + und • so dass gilt:

* die Menge F mit + ist eine kommutative Gruppe,
* die Menge F\* = F - { 0 } mit • ist eine kommutative Gruppe, und
* das Distributivgesetz ist erfüllt.

Beispiel

Die Menge der rationalen Zahlen \* mit Addition + und Multiplikation • ist ein Körper.

Bytes als Polynome mit binären Koeffizienten vom Grad 7

Ein Byte, also eine Folge b7 ... b1, b0 von acht Bits aus {0, 1}, wird als Polynom mit binären Koeffizienten interpretiert:

b7 . . . b1b0 b7X7 + ! + b1X+ b0

So entspricht beispielsweise die Hexadezimalzahl 0 x 57 oder die Binärzahl 0101 0111 dem Polynom x6 + x4 + x2 + x + 1.

Alle Additionen und Multiplikationen im AES finden im binären Körper IF28 mit 28 = 256 Elementen statt, einer Zahlenmenge mit Addition und Multiplikation, für die das Assoziativ-, Kommutativ- und Distributivgesetz (z. B. \*) gilt, und die wie folgt definiert ist. Es sei

IF2 = 0,1

der „Körper mit zwei Elementen“ sowie mit der Addition 1 + 0 = 0 + 1 = 1 und 0 + 0 = 1 + 1 = 0 (entspricht der ⊕-Addition durch XOR), und der (natürlichen) Multiplikation 1 • 0 = 0 • 1 = 0 • 0 = 0 und 1 • 1 = 1.

Es sei weiter

IF2 X = +/2+ = die Polynome über IF2,

d. h. die endlichen Summen a0 + a1 X + a2 X2 + ·+ an Xn bis a0, a1, …, an bis IF2, sowie

IF28: = IF2 X / X8 + X4 + X3 + X + 1 .

Das heißt, das Ergebnis der beiden Operationen + und • in IF2[X] ist der Rest der Division durch X8 + X4 + X3 + X + 1.

Addition

Die Plus-Addition (+) von zwei Polynomen ist die Addition in IF2 Koeffizient zu Koeffizient. Das heißt, dass die Plus-Addition als Bytes durch die XOR-Addition gegeben ist.

Multiplikation

Die Multiplikation • ist gegeben durch die natürliche Multiplikation, gefolgt von der Division mit Rest durch das Polynom

m x = x8 + x4 + x3 + x + 1 .

In hexadezimaler Notation ist beispielsweise 57• 83 = C1 , da

x6 + x4 + x2 + x + 1 x7 + x + 1 = x13 + x11 + x9 + x8 + x6 + x5 + x4 + x3 + 1

und

x13 + x11 + x9 + x8 + x6 + x5 + x4 + x3 + 1 = x5 + x3 + 1 x8 + x4 + x3 + x + 1 + x7

+ x6 + 1

Die Multiplikation mit dem Polynom 1 verändert nichts: Es ist das neutrale Element. Für ein beliebiges Polynom b(x) berechnet Euklids erweiterter Algorithmus die Polynome a(x) und c(x) , so dass gilt:

b x a x + m x c x = 1 .

Das heißt, bei der Division mit dem verbleibenden a(x) b(x) für m(x) bleibt der Rest 1. Dies bedeutet, dass a das multiplikative Inverse in IF 8 ist,

2

b−1 x = a x in IF28 .

Wenn wir ein Byte b in IF2 8 invertieren, meinen wir damit das Byte a = b–1.

###### Runden

Der AES verschlüsselt jeden Block iterativ in Runden (Durchgängen). R sei die Anzahl der Runden, die von B abhängt. Es gibt R = 10 Runden für B = 4 Spalten, R = 12 Runden für B = 6 Spalten und R = 14 Runden B = 8 Spalten.

Für uns gilt R = 10. In diesen Runden werden Schlüssel erzeugt und der Klartext wird durch folgende Operationen ersetzt und transponiert:

|  |  |  |
| --- | --- | --- |
| 1. | Runde r = 0 | AddRoundKey zum Hinzufügen (durch XOR) des Schlüssels zum Klartextblock |
| 2. | Runde r = 1,  …, R – 1 | Zum Verschlüsseln sind folgende Funktionen anzuwenden:   1. SubBytes zum Ersetzen jedes Bytes (= Folge von acht Bits) durch eine besser verteilte Folge von Bits 2. ShiftRows zum Permutieren der Einträge in jeder Zeile des Quadrats 3. MixColumn zum Hinzufügen der Spaltenboxen im Quadrat zwischen den Spalten 4. AddRoundKey zur Erzeugung eines Schlüssels aus dem Schlüssel der vorherigen Runde und dessen Hinzufügen (durch XOR) zum Quadrat |
| 3. | Runde r = R | Zum Verschlüsseln sind folgende Funktionen anzuwenden:   1. SubBytes 2. ShiftRows 3. AddRoundKey |

Im Vergleich zu früheren Runden entfällt die Funktion MixColumn. Es zeigt sich, dass MixColumn und AddRoundKey nach einer geringfügigen Änderung von AddRoundkey die Plätze tauschen können, ohne das Ergebnis der beiden Operationen zu verändern. In dieser äquivalenten Reihenfolge erhöht die Operation MixColumn nicht die kryptografische Sicherheit, da die letzte Operation ohne Kenntnis des Schlüssels invertierbar ist. Somit kann die letzte MixColumn-Operation entfallen.

Das CrypTool 1 bietet in seinem Menü „Individuelle Verfahren > Visualisierung von Algorithmen > AES“ einen Eintrag „Animation“ der Runden und einen Eintrag „Inspektor“ zum Experimentieren mit den Werten von Klartext und Schlüssel. Die verwendeten Funktionen werden im Folgenden näher beschrieben.

SubBytes

SubBytes ersetzt jedes Byte des Blocks durch ein anderes Byte, das in der Substitutionstabelle der S-Box angegeben ist.

Berechnung des Werts des Eintrags, durch den die S-Box die einzelnen Bytes ersetzt:

* + 1. Berechnung des Kehrwerts B in IF 8,

2

* + 1. Berechnung von ai = b1 + bi+4 + bi+5 + bi+6 + bi+7 + ci

mit i = 0, 1, …, 7 als Index jedes Bits eines Bytes, und

* + B = b7 b6 b5 b4 b3 b2 b1 b0 ist das Eingangsbyte,
  + A = a7 a6 a5 a4 a3 a2 a1 a0 ist das Ausgangsbyte der Operation und
  + c ist das konstante Byte 01100011.

In Matrixform:

a0 b0

1 0 0 0 1 1 1 1 1

a1 1 1 0 0 0 1 1 1 b1 1

a2 1 1 1 0 0 0 1 1 b2 0

a3 1 1 1 1 0 0 0 1 b3 0

= +

a4 1 1 1 1 1 0 0 0 b4 0

a5 0 1 1 1 1 1 0 0 b5 1

a6 0 0 1 1 1 1 1 0 b6 1

0 0 0 1 1 1 1 1 0

b

a7 7

ShiftRows

ShiftRows verschiebt die Zeilennummer l des Quadrats l Positionen nach links (wobei die Zeilennummern von Null an gezählt werden, d. h. l durchläuft 0, 1, 2 und 3, und die Verschiebung erfolgt zyklisch). *Anmerkung:* Die erste Zeile wird nicht verschoben.

MixColumn

MixColumn multipliziert jede Spalte des Blocks mit einer festen Matrix:

Symmetrische Kryptosysteme

* wenn Bj (mit den Koeffizienten b0,j, b1,j, b2,j und b3,j) der Spalte j des Eingangsblocks entspricht, und
* wenn Aj (mit den Koeffizienten a0,j, a1,j, a2,j und a3,j) der Spalte j des Ausgangsblocks der Operation entspricht,

dann gilt:

a0,j

a1, j

=

a2, j

a3, j

02 03 01 01

01 02 03 01

01 01 02 03

03 01 01 02

b0,j

b1, j

.

b2, j

b3, j

Das Byte a0,j wird zum Beispiel wie folgt berechnet:

a0,j = 2 · b0, j + 3 · b1, j + b2, j + b3, j .

AddRoundKey

AddRoundKey fügt durch die XOR-Operation den Schlüssel W(r) der aktuellen Runde r zum B-Quadrat des Geheimtextes hinzu, das heißt:

B ⊕ W r .

Der Schlüssel wird spaltenweise erzeugt. Wir bezeichnen die Spalten mit W(r)0, W(r)1, W(r)2 und W(r)3, d. h.:

W r = W r 0 ∣ W r 1 ∣ W r 2 ∣ W r 3 .

Da der Schlüssel 16 Byte hat, hat jede Spalte vier Byte. Das heißt, dass die letzte Spalte des vorherigen Rundenschlüssels plus das Ergebnis von ScheduleCore auf die erste Spalte des vorherigen Rundenschlüssels (den wir mit T bezeichnen) angewendet wird. Hier ist ScheduleCore die Zusammensetzung von Transformationen:

* SubWord ersetzt jedes der 4 Byte von T entsprechend der S-Box von SubBytes.
* RotWord verschiebt T ein Byte nach links (zirkulär, d. h. das letzte Byte wird das erste).
* Rcon(r) addiert (durch XOR) zu T den konstanten Wert, in hexadezimaler Schreibweise [(02)r–1 00 00 00] (wobei die Potenz, d. h. das iterierte Produkt, im Rijndael-Körper IF 8 berechnet wird). Das einzige Byte, das sich ändert, ist das erste, indem entweder der Wert 2r–1 (für r ≤ 8) oder der Wert 2r–1 in F 8 für r = 9, 10 addiert wird.

2

2

1. Der Schlüssel der ersten Runde W(0) ist durch den Anfangsschlüssel W gegeben.
2. Für r = 1, ..., R (wobei R die Gesamtzahl der Runden ist, für uns R = 10), werden die vier Spalten W(r)0, W(r)1, W(r)2 und W(r)3 des neuen Schlüssels wie folgt aus den Spalten des alten Schlüssels W(r-1) erzeugt:
   1. Die erste Spalte W(r)0 ist gegeben durch:

W r 0 = W r − 1 0 ⊕ ScheduleCore W r − 1 3 ;

* 1. Die nächsten Spalten W(r)1, W(r)2 und W(r)3 sind für i = 1, 2 und 3 gegeben durch:

W r i = W r i − 1 ⊕ W r − 1 i;

* 1. Das heißt, die vorherige Spalte des aktuellen Rundenschlüssels plus die aktuelle Spalte des vorherigen Rundenschlüssels.

###### Diffusion

Wir stellen fest, dass die einzige nicht affine Transformation die Inversion im Körper IF 8 in der Operation SubBytes ist. Bei der Operation SubBytes werden durchgeführt: die Inversion in IF 8, eine lineare Operation, und die Übersetzung durch einen konstanten Vektor. (a) ShiftRows ist eine lineare Permutation, (b) MixColumn ist eine lineare Addition, und (c) AddRoundKey ist die Translation durch den Rundenschlüssel.

2

2

Im Hinblick auf die Ziele der idealen „Diffusion“ und „Konfusion“ ist zu beachten, dass in jedem Schritt etwa die Hälfte der Bits (in SubBytes) oder Bytes (in MixColumn und ShiftRows) ersetzt und transponiert werden. Die Komplementarität der einfachen Operationen für hohe Sicherheit, d. h. ihre Erzeugung von hoher Konfusion und Diffusion nach einigen Iterationen, wird durch die Substitution des Alphabets und die Permutation von Text deutlich. Bei Letzteren ist dies insbesondere bei der Permutation zwischen den Einträgen jeder Zeile und der Permutation zwischen den Spalten zu beobachten. Das Cryptool von Forma Estudio (2007) ist ein Beispiel für diese Eigenschaft.

Wir sehen, wie sich dieser kleine anfängliche Unterschied ausbreitet und nach nur vier Runden zu sehr unterschiedlichen Ergebnissen führt! Dies plausibilisiert die Immunität von AES gegen differenzielle Kryptoanalyse.

Für den Fall, dass alle Schlüssel- und Klartexteinträge gleich 00 sind, verstehen wir auch die Auswirkung des Addierens der Konstante Rcon(r) zu dem Schlüssel in jeder Runde: Daher kommt die gesamte Konfusion!

### Kryptografische Hashfunktionen

Eine Hashfunktion wandelt beliebige Daten (z. B. eine Datei), d. h. eine Zeichenkette mit variabler Länge, in eine Zeichenkette mit fester Länge (in der Regel 16 oder 32 Byte) um. Sie wandelt eine große Datenmenge in eine kleine Informationsmenge um.

Symmetrische Kryptosysteme

###### Hash als ID

Eine Hashfunktion sollte auf alle möglichen Sequenzen mit fester Länge anwendbar sein und einer gleichförmigen Zufallsvariablen ähneln, d. h. die Wahrscheinlichkeit für jeden Funktionswert ist gleich groß.

Wenn die Ausgabe beispielsweise aus 256 Bits besteht, dann sollte im Idealfall jeder Wert die gleiche Wahrscheinlichkeit 2–256 haben. Also identifiziert die Ausgabe die Eingabe praktisch eindeutig (mit einer Kollisionswahrscheinlichkeit von idealerweise 2–256). So könnten wir uns beispielsweise einen Datenhash einer Datei als deren Personalausweis vorstellen. Ein Hash identifiziert viele Daten mit wenigen Informationen.

Da die Länge der Hash-Sequenz begrenzt ist (selten ≥ 512 Bits) und die Länge der Eingabesequenz unbegrenzt ist, kommt es zu Kollisionen, d. h. zu gleichen Hashes aus verschiedenen Dateien. Der Algorithmus minimiert jedoch die Wahrscheinlichkeit von Kollisionen, indem er die Werte so gleichmäßig wie möglich verteilt – oder intuitiv ausgedrückt: so zufällig wie möglich macht. Genauer gesagt ist jede mögliche Sequenz mit fester Länge ein Wert, und die Wahrscheinlichkeit für jeden der Werte ist die gleiche.

###### Kryptografische Hashfunktionen

Für eine kryptografische oder Einweg-Hashfunktion sollte der Algorithmus resistent sein gegenüber

1. der Erstellung eines Urbildes, d. h. die Funktion ist unidirektional. Wenn eine Ausgabe gegeben ist, ist der schnellste Weg, eine Eingabe mit dieser Ausgabe zu finden, ein Brute-Force-Verfahren, d. h. die Prüfung aller möglichen Eingaben.
2. der Erstellung einer zweiten umgekehrten Abbildung, d. h. bei einer Eingabe ist der schnellste Weg, eine andere Eingabe mit demselben Ergebnis zu finden, ein Brute-Force-Verfahren.
3. der Erstellung von Kollisionen, d. h. der schnellste Weg, zwei (beliebige) Einträge mit demselben Ergebnis zu finden, ist ein Brute-Force-Verfahren.

Nach dem Kerckhoffs‘schen Prinzip muss der Algorithmus zudem öffentlich sein. In der Praxis ist die wichtigste Eigenschaft die Resistenz gegen den Angriff über die Erstellung einer zweiten, umgekehrten Abbildung, und die am wenigsten wichtige ist die gegen Kollisionsangriffe. Es gibt verschiedene Algorithmen (z. B. MD4, MD5 und SHA-1), die nicht gegen Kollisionsangriffe resistent sind, aber immer noch verwendet werden. Der CRC-Algorithmus zum Beispiel ist eine (nicht kryptografische) Hashfunktion. Andere gängige kryptografische Hashfunktionen sind SHA-256 und SHA-3.

Die Ausgabe der Hashfunktion SHA-256 von IUBH lautet zum Beispiel 0a65cfb2c33b58749182d3eb965155336a52219793ff633016cb3942b2809a3f. Die Ausgabe einer Hashfunktion wird zum Beispiel für die Integrität und Authentifizierung von Nachrichten verwendet, je nach Eingabe aber auch für einen Message Digest oder digitalen Fingerabdruck.

###### Gängige kryptografische Hashfunktionen und Anwendungen

Die am häufigsten verwendeten (kryptografischen) Hash-Algorithmen sind auch heute noch 16 Byte groß wie MD4 und MD5, oder SHA-1, das 20 Byte verwendet. Obwohl keine von ihnen Kollisionsangriffen standhält, sind sie nach wie vor beliebt. Die Einzelheiten ihrer Implementierung sind im Request for Comments (RFC) beschrieben. Ein RFC spezifiziert öffentlich die Einzelheiten eines geplanten Internetstandards oder einer neuen Version eines bestehenden Standards und wird in der Regel von Forschern an Universitäten und in Unternehmen verfasst, um Feedback einzuholen. Ein RFC wird online und in offiziellen Sitzungen der von der Internet Engineering Task Force (IETF) beauftragten Arbeitsgruppe diskutiert. So wurden beispielsweise Netzwerkstandards wie IP und Ethernet in RFCs dokumentiert.

Der Message-Digest-Algorithmus (MD4) wurde 1991 von Ron Rivest entwickelt, einem der drei Erfinder des RSA-Algorithmus. Er wird in RFC 1320 beschrieben, ist schnell, aber anfällig für Urbild-Angriffe.

MD5 wurde von RSA Data Security entwickelt. Das in RFC 132 beschriebene Verfahren ist anfällig für Kollisionen, aber nicht für die Erstellung eines zweiten Urbildes. Es wird häufig für Integritätsprüfungen, Software mit Peer-to-Peer-Protokoll (P2P) und als Passwortspeicher verwendet.

Der Secure-Hash-Algorithmus (SHA-1) wurde vom NIST entwickelt. Er ist anfällig für Kollisionen, aber nicht für die Erstellung eines zweiten Urbildes. Die neueren Versionen, wie SHA-256 und SHA-3, sind gegenüber den alten SHA-1 oder MD4 und MD5 vorzuziehen.

Digitale Signaturen

Sind die Rollen des öffentlichen und des privaten Schlüssels vertauscht, so handelt es sich bei der Verschlüsselung um eine digitale Signatur. Während die verschlüsselte Nachricht nicht mehr geheim ist, kann jeder Besitzer des öffentlichen Schlüssels überprüfen, ob der private Schlüssel die ursprüngliche Nachricht verschlüsselt hat. Theoretisch würde jedoch das Signieren einer Datei (unter Verwendung des RSA-Algorithmus), die mit dem RSA-Algorithmus verschlüsselt wurde, diese entschlüsseln. In der Praxis wird ein kryptografischer Hash in der Regel mit dem privaten Schlüssel verschlüsselt, genauso wie eine Signatur ausreicht, um die signierte Datei eindeutig zu identifizieren (ihr Inhalt ist jedoch oft geheim, z. B. beim Signieren eines geheimen Schlüssels).

(H)MAC

Zur Nachrichtenauthentifizierung wird ein Message Authentication Code (MAC) verwendet. Wenn dieser MAC durch eine Hashfunktion gesichert ist, wird er als Keyed-Hash Message Authentication Code (HMAC) bezeichnet. Das American National Standards Institute (ANSI) hat einen Message Digest entwickelt, einen Hash einer Nachricht, mit einem Passwort als Hash-Schlüssel und unter Verwendung des DES-Algorithmus als neuen Standard der Kryptografie. Ein HMAC wird also aus einer Nachricht und einem geheimen Schlüssel erzeugt, der in einer kryptografischen Hashfunktion wie SHA-1 oder MD5 verwendet wird. Der MAC-Wert schützt die Integrität und Authentizität einer Nachricht, indem er es den vorgesehenen Empfängern ermöglicht, Änderungen am Inhalt der Nachricht zu erkennen. Der HMAC liefert dem Benutzer außerdem sowohl eine Integritäts- als auch eine Authentizitätsprüfung (Krawczyk et al., 1997).

Symmetrische Kryptosysteme

Datenspeicherung und Datenintegrität

Hashfunktionen (sowohl kryptografische als auch nicht kryptografische) werden für die schnelle Datenabfrage, d. h. mit konstanter Dauer, in einer Hashtabelle und in einem Merkle-Baum (auch Hash-Baum) verwendet. Sie werden außerdem dazu eingesetzt, die Integrität einer Datei angesichts versehentlicher Änderungen zu gewährleisten, d. h. um Unterschiede zwischen der Datei und einer Referenzversion zu erkennen.

Passwörter

Kryptografische Hashfunktionen werden verwendet, um Werte gleichmäßig zu verteilen (Schlüsselstreckung, engl. Key Stretching), was sie intuitiv weniger vorhersehbar macht, bekannt als Key Derivation Function (KDF), und um Passwörter zu erzeugen und zu speichern, als PBKDF (Password Based Key Derivation Function). Kryptografische Hashfunktionen werden auch verwendet, um die Integrität einer Datei vor Manipulationen zu schützen, d. h. um Unterschiede zwischen der Datei und einer Referenzversion (in der Regel die Version, bevor die Datei übertragen wird) zu erkennen. Insbesondere, um die Echtheit einer Datei zu gewährleisten und Unterschiede zwischen der Datei und einer Version zu erkennen, die unter der Kontrolle einer bestimmten Person stand.

Beachten Sie den Unterschied zwischen Authentizität und Authentifizierung. Erstere garantiert die Gleichheit der von einer Person empfangenen und gesendeten Daten, z. B. bei der digitalen Signatur, Letztere die Identität dieser Person, z. B. bei einem sicheren Website-Zugang.

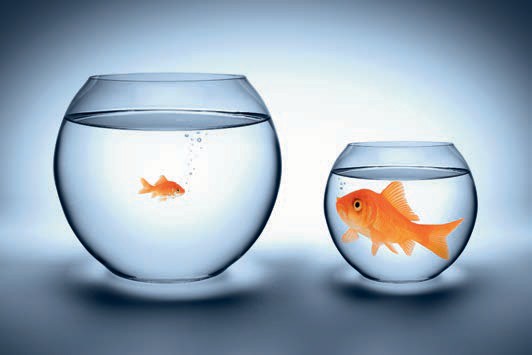
Die oben aufgeführten kryptografischen Hash-Algorithmen verteilen die Werte gleichmäßig, sind aber schnell und eignen sich daher nicht für die Erstellung von Passwörtern, da sie anfällig für Brute-Force-Angriffe sind. Um diese zu verhindern, sind die PBKDF-Algorithmen (a) absichtlich langsam, wie z. B. bcrypt, (b) erfordern absichtlich viel Speicher für die Berechnung, wie z. B. der scrypt-Algorithmus, oder (c) werden nur einmal für jeden Eintrag verwendet, garantiert durch ein Salt (englisch für Salz), ein zusätzliches, einzigartiges, normalerweise zufälliges Argument. Ohne Salt ist der Algorithmus anfällig für Angriffe durch „Rainbow Tables“, Tabellen mit den Hashes der gängigsten Passwörter.

Zusammenfassung

Beim Chiffrieren gibt es zwei grundlegende Verfahren: Transposition und Substitution. Transpositionen ordnen den Text neu an, ohne die Zeichen zu verändern; Substitutionen ersetzen die Zeichen des Textes, ohne die Reihenfolge zu verändern. Feistelchiffren sind unsicher, weil sie die statistischen Daten des Klartextes erhalten. Bei modernen Chiffren werden im Gegensatz dazu Substitutions- und Permutations-Chiffren kombiniert, die als Substitutions- und Permutationsnetzwerke bezeichnet werden.

Obwohl es keine Methode der Kryptoanalyse gibt, die schneller ist als eine erschöpfende Schlüsselsuche, erreichen moderne Chiffren wie DES (1976) oder AES (2000) in der Praxis dasselbe. Mit anderen Worten: Es ist keine kryptoanalytische Methode bekannt, die schneller als die erschöpfende Schlüsselsuche ist. Das entscheidende Kriterium ist eine hohe Diffusion im Sinne von Shannon.

Hashfunktionen wandeln beliebige Daten mit variabler Länge in eine Ausgabe mit fester Länge um. Der ausgegebene Hashwert sollte eine gute Diffusion aufweisen: die Umkehrung eines Eingabebits führt zu Änderungen bei etwa der Hälfte der Ausgabe; er sollte für alle möglichen Sequenzen mit fester Länge anwendbar sein und einer gleichförmigen Zufallsvariablen ähneln, d. h. die Wahrscheinlichkeit für jeden der Funktionswerte ist die gleiche. Bei einer Ausgabe mit fester Länge von 256 Bits sollte im Idealfall jeder Wert die gleiche Wahrscheinlichkeit 2-256 haben. Es handelt sich um eine kryptografische (oder Einweg-) Hashfunktion, wenn eine Umkehrung rechnerisch undurchführbar ist und wenn ähnliche Daten unterschiedliche Hashwerte ergeben. Empfohlen werden die neueren Versionen SHA-256 und SHA-3.



# Lektion 3

## Asymmetrische Kryptosysteme

#### LERNZIELE

Nach der Bearbeitung dieser Lektion kennen Sie …

… die Vorteile und Grenzen der asymmetrischen Kryptografie.

… Falltürfunktionen und modulare Arithmetik.

… die gängigsten asymmetrischen kryptografischen Algorithmen und die ihnen zugrunde liegenden Falltürfunktionen.

DL-E-DLMCSEAITSC02-U03

1. Asymmetrische Kryptosysteme

### Einführung

Das große praktische Problem der „Single-Key“-Kryptografie ist die Schlüsselverteilung, d. h. die geheime Weitergabe desselben Schlüssels an alle Kommunikationsteilnehmer, die normalerweise weit voneinander entfernt sind, bevor diese sicher kommunizieren können. Problematisch ist zudem die große Anzahl von Schlüsseln, die eine Gruppe von Kommunikationsteilnehmern für eine sichere Kommunikation benötigt.

1976 fanden Diffie und Hellman heraus, dass das Problem der Schlüsselverteilung durch einen Algorithmus gelöst werden kann, der (rechnerisch) folgende Bedingungen erfüllt:

* + einfache Erstellung eines passenden Schlüsselpaares für die Ver- und Entschlüsselung,
  + einfache Ver- und Entschlüsselung,
  + praktisch unmögliche Wiederherstellung eines der Schlüssel trotz Kenntnis des Algorithmus, des anderen Schlüssels und einer beliebigen Anzahl übereinstimmender Klartext- und Geheimtext-Paare, und
  + praktisch unmögliche Wiederherstellung des Klartextes für nahezu alle Schlüssel k und Nachrichten x.

Als Benutzerin dieses Algorithmus hält Andrea ihren Entschlüsselungsschlüssel geheim, macht aber ihren Verschlüsselungsschlüssel öffentlich, zum Beispiel in einem öffentlichen Verzeichnis. Jeder, der sicher mit Andrea kommunizieren möchte, braucht nur ihren öffentlichen Schlüssel nachzuschlagen, um ihr einen Geheimtext zu senden, den jedoch nur Andrea entschlüsseln kann. Eine Nachricht kann also verschlüsselt werden, ohne dass eine Geheimhaltung erforderlich ist. Jeder, der den entsprechenden öffentlichen Schlüssel verwendet, kann einen mit dem geheimen Schlüssel von Andrea verschlüsselten Geheimtext entschlüsseln, d. h. ein Absender kann identifiziert werden, ohne dass eine Geheimhaltung erforderlich ist.

Die Sicherheit kryptografischer Algorithmen mit zwei Schlüsseln hängt von der Schwierigkeit der Berechnung eines mathematischen Problems ab, z. B. der Faktorisierung einer Zahl, die das Produkt aus zwei großen Primzahlen ist. Im Idealfall ist die Berechnung des geheimen Schlüssels gleichbedeutend mit der Lösung des schwierigen Problems, so dass der Algorithmus mindestens so sicher ist, wie das zugrunde liegende mathematische Problem schwierig ist. Dies wurde für keinen der Standardalgorithmen bewiesen, obgleich angenommen wird, dass es für jeden gilt.

###### Arithmetische Falltüren

Um die Nützlichkeit der (modularen) Arithmetik in der Kryptografie zu erkennen, erinnern wir uns daran, dass die asymmetrische Kryptografie auf einer Falltürfunktion basiert, die leicht berechenbar sein muss, deren Umkehrung ohne Kenntnis des Schlüssels jedoch praktisch unberechenbar sein muss.

Die Einfachheit der Berechnung der Funktion entspricht der Einfachheit der Verschlüsselung, während die Schwierigkeit der Berechnung der Umkehrung der Schwierigkeit der Entschlüsselung, d. h. der Umkehrung der Verschlüsselung, entspricht. RSA verwendet zum Beispiel als Verschlüsselungsfunktion die Berechnung der n-ten Potenz und als Entschlüsselungsfunktion deren Umkehrung, das Wurzelziehen.

Während sowohl die Funktion selbst als auch ihre Umkehrung leicht mit der üblichen Multiplikation von Zahlen berechnet werden können, verwenden kryptografische Algorithmen (wie RSA) modulare Arithmetik, um die Berechnung der Umkehrfunktion ohne Kenntnis des Schlüssels komplexer zu gestalten.

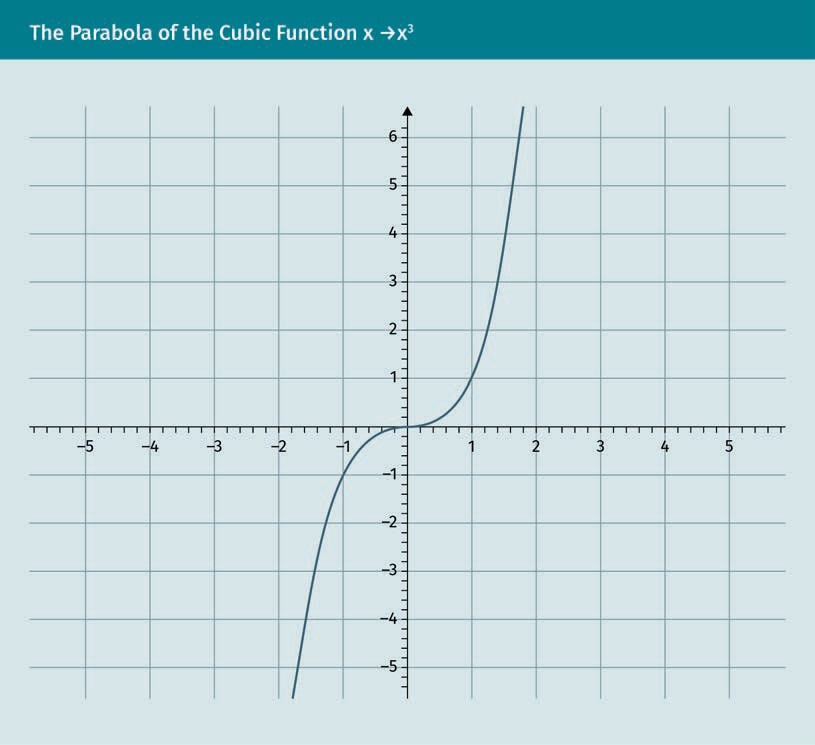
Die modulare oder zirkuläre Arithmetik kennen wir bereits aus der Arithmetik der Uhr, bei der m = 12 als gleich 0 angesehen wird. Da der Zeiger nach jeder Umdrehung wieder bei 0 beginnt zu zählen, ist es zum Beispiel drei Stunden nach 11 Uhr zwei Uhr, 11 + 3 = 14 = 12 + 2 = 2. In diesen Definitionsbereichen, die als endliche Ringe bezeichnet werden, werden (die Graphen der) Funktionen unregelmäßig und praktisch unberechenbar, zumindest ohne Kenntnis einer Abkürzung (des Schlüssels).

Modulare Arithmetik als Randomisierung

Die Schwierigkeit, die Umkehrung zu berechnen, entspricht der Schwierigkeit der Entschlüsselung, d. h. der Umkehrung der Verschlüsselung. Bei einem asymmetrischen kryptografischen Algorithmus basieren die Einfachheit der Verschlüsselung einer Zahl und die Schwierigkeit der Entschlüsselung einer Zahl auf einer umkehrbaren Funktion, die selbst zwar leicht berechenbar ist, ihre Umkehrfunktion jedoch nur äußerst schwierig. Zum Beispiel sind die Umkehrungen der Falltürfunktionen Potenzierung x /xe (im RSA-Algorithmus) und Exponentiation x /gx (im Diffie-Hellman-Algorithmus) durch das Wurzelziehen x /x1/e und den Logarithmus log*g* gegeben. Wir können sie im Definitionsbereich der reellen Zahlen 0 leicht berechnen (z. B. mit dem Intervallhalbierungsverfahren (Bisektion) für stetige Funktionen in 0), aber in ihren endlichen kryptografischen Definitionsbereichen sind sie nahezu unberechenbar. Wir wollen diese endlichen Definitionsbereiche nun einführen.

Funktionen auf diskreten Mengen

Ihr Definitionsbereich ist nicht die Menge der ganzen Zahlen + (oder die ihrer Obermenge, der reellen Zahlen 0), weil beide Funktionen, Exponentiation und Potenzierung, stetig in 0 sind.



Wenn der Definitionsbereich dieser Funktionen 0 wäre, könnten ihre Umkehrungen über 0 angenähert werden, zum Beispiel durch iterative Intervallhalbierung (Bisektion), bei welcher der Umkehrpunkt in Intervallen eingeschlossen wird, die bei jeder Iteration halbiert werden.

Endlicher Ring

Dies ist eine endliche Menge, die 0 und 1 enthält und über die eine Summe erklärt ist, die dem Assoziativ- und dem Kommutativgesetz gehorcht.

Endliche Ringe

Um die iterative Annäherung an die Null zu vermeiden und damit die Berechnung der Umkehrfunktion zu erschweren (neben der Erleichterung der Berechnung der eigentlichen Funktion), ist der Definitionsbereich einer Falltürfunktion ein endlicher Ring, angegeben durch:

+/m+ = 0,1, . . . , m − 1

für eine natürliche Zahl m.

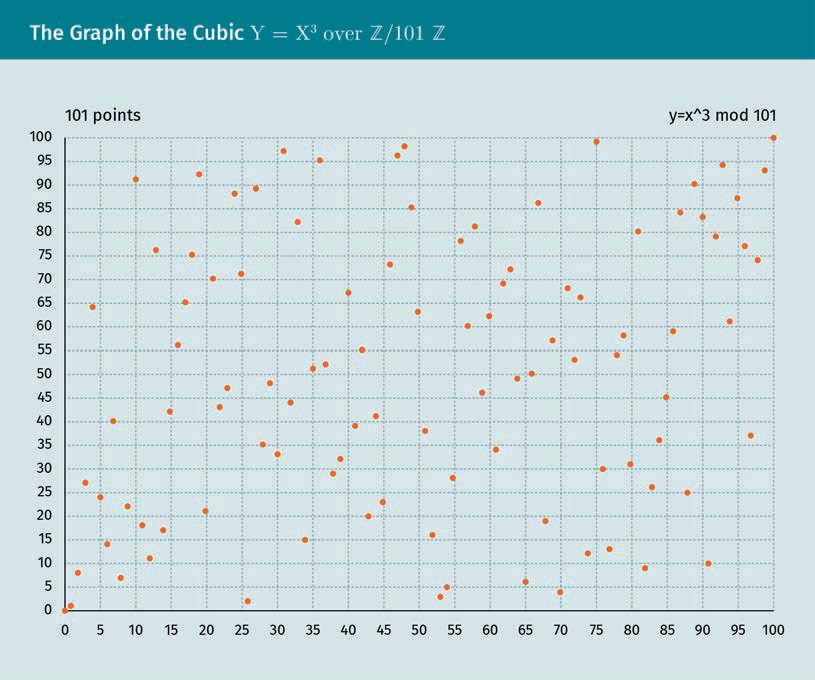
In solch einem endlichen Ring gilt:

m = 1 + ! + 1 = 0

Notwendigerweise hat jede Addition (und damit jede Multiplikation und jede Potenzierung) ein Ergebnis < m. Also ist + / m + seine Addition anders als die von + (oder 0). Für m = 7 erhalten wir z. B.

22 = 2 · 2 = 4 und 32 = 3 · 3 = 7 + 2 = 0 + 2 = 2 .

Wir werden diese endlichen Ringe zunächst anhand der Beispiele + / 12 + und + / 7 + (den Ringen, die durch die Stundenzahlen auf einer Uhr und der Wochentage gegeben sind) einführen, dann für jedes m.



Wenn wir uns die Graphen der Funktionen ansehen, die auf 0 so regelmäßig sind, stellen wir fest, dass die Parabel über dem endlichen Ring + /101 + zunächst über + / 101 + genauso regelmäßig ist wie über +. Ab x = 11 (wegen 112 = 121 > 100) beginnt sie sich jedoch unregelmäßig zu verhalten (mit Ausnahme der Symmetrie der Parabel auf der Mittelachse x = 50,5 aufgrund der Gleichheit ((–x)2 = x2).

Zwei ganze Zahlen a und b sind kongruent modulo m

a ≡ b mod m

wenn m | a – b, das heißt, wenn ihre Differenz a – b durch m teilbar ist.

Kongruenz

Zwei ganze Zahlen a und b sind kongruent modulo m, wenn sie nach der Division durch m den gleichen Rest ergeben.

Die Zahl m ist der Modul. Oder anders formuliert, a ≡ b mod m , wenn a und b geteilt durch m den gleichen Rest ergeben.

### Der Diffie-Hellman-Schlüsselaustausch

Diffie-Hellman-  
Schlüsselaustausch

Dieses Protokoll war eine offene Vereinbarung über einen gemeinsamen geheimen Schlüssel, dessen Sicherheit darauf beruhte, dass es praktisch unmöglich ist, den Logarithmus modulo einer großen Zahl zu berechnen.

Das erste veröffentlichte Protokoll, in dem ein gegenseitiger geheimer Schlüssel offen vereinbart wurde, ist der Diffie-Hellman-Schlüsselaustausch (Diffie & Hellman, 1976). Dies stellt noch keine Zwei-Schlüssel-Kryptografie dar, weil beide Korrespondenten den einzigen geheimen Schlüssel kennen. Die asymmetrischen kryptografischen Algorithmen, die auf diesem Protokoll aufbauen (z. B. Elgmal und ECC), erzeugen für jede Nachricht einen eindeutigen Schlüssel.

###### Schlüsselaustauschprotokoll

Bezeichnen wir in jeder asymmetrischen Verschlüsselung eine öffentliche Zahl mit einem Großbuchstaben und eine geheime Zahl mit einem Kleinbuchstaben. Damit sich zwei Parteien, Andrea und Robert, offen auf einen geheimen Schlüssel einigen können, kombinieren sie zunächst eine geeignete Primzahl p (der Modul) und eine geeignete natürliche Zahl g (die Basis).

Zunächst erzeugt Andrea eine Hälfte des Schlüssels, indem sie (1) eine Zahl a wählt, (2) A ≡ga mod p berechnet und (3) A an Robert überträgt. Danach generiert Robert die andere Hälfte des Schlüssels, indem er (1) eine Zahl b wählt, (2) B ≡gb mod p berechnet und (3) B an Andrea übermittelt. Der gemeinsame geheime Schlüssel zwischen Andrea und Robert lautet:

b a

b a aab ba b

c ( A mod p = g mod p mod p = g mod p = g mod p = g mod p mod p = B mod p .

Dieses Protokoll zeigt, wie ein gemeinsamer geheimer Schlüssel offen erstellt wird. Dieser Schlüssel kann dann zur Verschlüsselung der gesamten weiteren Kommunikation verwendet werden, so wie mit AES. Er zeigt jedoch weder wie eine Nachricht verschlüsselt, noch wie sie signiert wird.

###### Sicherheit

Die Sicherheit des Diffie-Hellman-Schlüsselaustauschs beruht auf der Schwierigkeit, den Logarithmus modulo p zu berechnen. Ein Abhörender würde den geheimen Schlüssel Ab = Ba aus A und B erhalten, wenn er folgendes berechnen könnte:

a = loggA oder b = loggB mod p .

Während eine Potenz leicht berechenbar ist, erst recht in der modularen Arithmetik, ist ihre Umkehrung, der Logarithmus, also der Exponent für eine bestimmte Potenz, bei geeigneter Wahl von p und g praktisch nicht berechenbar.

###### Logarithmus

Da zu Beginn (für x < logg p) die Werte gx über + / p + gleich den Werten gx über + sind, sollten die geheimen Zahlen a und b ausreichend groß sein, d. h. > logg p. Um dies sicherzustellen, werden diese Zahlen in der Praxis künstlich erhöht, die Nachricht wird also „ausgestopft“.

Der derzeit schnellste Algorithmus zur Berechnung des Logarithmus x aus gx ist eine Adaption des allgemeinen Zahlenfeldsiebs (Gordon, 1993). Die Anzahl der Operationen zur Berechnung des Logarithmus einer ganzen Zahl mit n Bits entspricht grob

exp logn1/3 .

### Der RSA-Algorithmus

Der bekannteste Public-Key-Algorithmus ist der Rivest-Shamir-Adleman-Kryptoalgorithmus (kurz RSA-Algorithmus, Rivest et al., 1978). Ein Benutzer wählt im Geheimen ein Paar Primzahlen p und q, die so groß sind, dass die Faktorisierung des Produkts N = p q die geschätzte Rechenleistung für die gesamte Nutzungsdauer der Chiffre übersteigt. Die Zahl N wird der Modul sein, d. h. unsere Falltürfunktion basiert auf + / N + = { 0, 1, …, N-1 }.

N ist öffentlich, p und q jedoch nicht. Wenn die Faktoren p und q von N bekannt wären, dann könnte der geheime Schlüssel leicht berechnet werden. Damit RSA sicher ist, darf die Faktorisierung rechnerisch nicht durchführbar sein, was derzeit einer Größe von 2048 Bit entspricht. Der Schwierigkeitsgrad der Faktorisierung verdoppelt sich ungefähr für jeweils drei zusätzliche Ziffern von N.

Der RSA-Algorithmus erzeugt einen öffentlichen Schlüssel zum Verschlüsseln und einen privaten Schlüssel zum Entschlüsseln. Im Vergleich zum Diffie-Hellman-Protokoll hat dieser Algorithmus den Vorteil, dass er vollständig asymmetrisch ist. Es besteht keine Notwendigkeit, einen gemeinsamen geheimen Schlüssel zu teilen (welcher nur an einem einzigen Ort aufbewahrt würde). Stattdessen hat ein einziger Kommunikationspartner Zugriff auf den geheimen Schlüssel. In diesem Fall wird die Kommunikation jedoch nur gegenüber dem Besitzer des geheimen Schlüssels verschlüsselt. Um in beide Richtungen zu verschlüsseln, erstellt entweder jeder Kommunikationspartner einen asymmetrischen RSA-Schlüssel, oder der andere Kommunikationspartner verschlüsselt und sendet einen symmetrischen Schlüssel (bekannt als hybride Verschlüsselung, da sie ein asymmetrisches mit einem symmetrischen Kryptosystem kombiniert).

###### Zahlentheorie

Für die Eulersche Formel werden zwei verschiedene Primzahlen p und q benötigt. Ist a nun weder durch p noch durch q teilbar, dann gilt:

a p − 1 q − 1 ≡ 1 mod pq .

RSA

Bei diesem Algorithmus werden Zahlen (Nachrichten) durch Potenzierung verschlüsselt. Seine Sicherheit beruht darauf, dass es rechnerisch nahezu unmöglich ist, ein Produkt aus Primzahlen zu faktorisieren.

Wurzeln ziehen

Seien p und q verschiedene Primzahlen, N = p q und φ(N) = (p – 1)(q – 1). Für jeden Exponenten n mit

n ≡ 1 mod 3 N

gilt:

an ≡ a mod N für jede ganze Zahl a .

Wenn m ≡1 mod φ(N), dann gilt nach der Eulerschen Formel am ≡a mod N, d. h. die Potenzierung ist die Identitätsfunktion,

m· ≡ id mod N .

Insbesondere, wenn m = E d das Produkt zweier ganzer Zahlen E und d ist, d. h.

Ed ≡ 1 mod 3 N ,

dann gilt:

a = am = aEd = aE d .

Das heißt • d = • 1/E mod N. Eine Potenz zu berechnen ist wesentlich einfacher als eine Wurzel!

Beispiel

Wenn p = 3 und q = 11 gilt, dann ist

N = pq = 33 und 3 N = p − 1 q − 1 = 20 .

Wenn E = 7 und d = 3 gilt, dann ist n = Ed = 21 ≡1 mod 20. Für die Basis 2 prüfen wir zum Beispiel

2E = 27 = 128 = 29 + 3 · 33 ≡ 29 mod N

und

29d = 293 = − 4 3 ≡ − 64 = 2 − 2 · 33 ≡ 2 mod N.

Somit gilt:

E 29 = 2 = 29d mod N .

Euklidischer Algorithmus

Ein gemeinsamer Teiler von zwei ganzen Zahlen a und b ist eine natürliche Zahl, die diese beiden Zahlen teilt. Der größte gemeinsame Teiler von zwei ganzen Zahlen ist die größte natürliche Zahl, die diese beiden teilt. Der größte gemeinsame Teiler von 12 und 18 ist zum Beispiel 6. Bezeichnen wir mit mdc(a,b) den größten gemeinsamen Teiler von a und b, dann ist:

mdc a, b = die größte natürliche Zahl, die a und b teilt.

Die wiederholte Division mit Rest ergibt einen effizienten Algorithmus zur Berechnung des größten gemeinsamen Teilers: den euklidischen Algorithmus.

Erweiterter Euklidischer Algorithmus

Für die Berechnung des Exponenten der Entschlüsselungsfunktion benötigen wir mehr Informationen als den größten gemeinsamen Teiler (berechnet über den euklidischen Algorithmus). Im erweiterten Algorithmus stellen wir fest, dass in jedem Schritt des euklidischen Algorithmus der größte gemeinsame Teiler ggT(x, m) von x und m eine Linearkombination (oder Summe von Vielfachen) von x und m ist, d. h.

ggT x, m = λx + µm für die ganzen Zahlen x und m .

Als Beispiel nehmen wir ggT(528, 220) = 44 und tatsächlich

44 = 5 · 748 − 7 · 528 .

###### Verschlüsselungsalgorithmus

Wir erinnern uns, dass ein Großbuchstabe jeweils für eine öffentliche Zahl steht, während ein Kleinbuchstabe für eine geheime Zahl steht. Wir betrachten zwei Parteien: Andrea sendet die Nachricht m über einen unsicheren Kanal heimlich an Robert.

1. Für die Erzeugung des Schlüssels wählt Robert zwei Primzahlen p und q sowie einen Exponenten E,

der teilerfremd ist zu φ(N) := (p – 1)(q – 1).

1. Zur Übermittlung des Schlüssels sendet Robert Andrea das Produkt N = pq (den Modul) und den Exponenten E (den öffentlichen Schlüssel).
2. Zur Verschlüsselung berechnet Andrea M = mE mod N und übermittelt M an Robert.
3. Zur Entschlüsselung berechnet Robert (mit dem erweiterten euklidischen Algorithmus) d so, dass Ed ≡ 1 mod (p – 1)(q – 1) (da E teilerfremd ist zu φ(N)), und berechnet Md = mEd = m mod N (mit der Eulerschen Formel).

Zusammenfassend lässt sich sagen, dass über die Potenzierung y = xE mod N verschlüsselt wird, wobei der Exponent E der öffentliche Schlüssel ist. Dementsprechend entschlüsseln wir mit der Umkehrung, dem Ziehen der E-ten Wurzel x = y1/E mod N. Sie ist praktisch nicht berechenbar. Aber über modulo N und mit der Eulerschen Formel gibt es ein d, so dass gilt:

y1/E = yd mod N

Die Zahl d wir über den euklidischen Algorithmus sowohl aus E als auch aus p und q berechnet. Der geheime Schlüssel ist also d, oder auch die Kenntnis der Primfaktoren p und q von N.

###### Sicherheit

Da der Modul N, der Exponent E und die verschlüsselte Nachricht M (= mE) alle öffentlich sind, basiert die rechnerische Sicherheit von RSA allein auf der Schwierigkeit, eine Wurzel aus einer sehr großen Zahl zu finden:

m ≡ E M = M1/E mod N.

### Elliptische Kurven

Wir schreiben IFp := + / p +. Unter allen Kurven besteht der entscheidende Vorteil der elliptischen Kurven (gegeben durch die Gleichung y2 = x3 + ax + b) darin, dass wir auf ihnen Punkte hinzufügen können: p + q + r = 0 wenn eine Linie durch p*,* q und r verläuft. By restricting the solutions to points (x, y) in (IFp × IFp) for a large prime number p and fixing a point P on the curve while it is easy to compute the exponential, i.e., for n compute

Diffie-Hellman über elliptischen Kurven

Ein Analogon des Diffie-Hellman-Protokolls, in dem die iterative Multiplikation einer Zahl modulo p durch die iterative Addition eines Punktes auf einer elliptischen Kurve ersetzt wird. Das Diffie-Hellman-Protokoll (über IFp) hat ein Analogon über elliptischen Kurven. Anstatt wiederholt (n-mal) die Basis g in IF \* zu *multiplizieren*, d. h. zu berechnen\_\_\_\_\_\_\_\_\_

p

Es ist von Vorteil, den Logarithmus über einer endlichen elliptischen Kurve zu verwenden (die Funktion, die für einen gegebenen Punkt G und Y den Skalar x in 5 so bestimmt, dass Y = x G) anstelle des Logarithmus über IFp (die Funktion, die bei gegebenen Zahlen g und y den Exponenten x so bestimmt, dass y ≡gx mod p). Abhängig von der Anzahl der Bits n von p (in Bezug auf die schnellsten derzeit bekannten Algorithmen) steigt die Zeit zur Berechnung des Logarithmus über einer elliptischen Kurve linear an und benötigt etwa n/2 Operationen, während die Zeit zur Berechnung des multiplikativen Logarithmus sublinear ansteigt und etwa n1/3 Operationen benötigt.

Zum Beispiel entspricht die von einem 2048-Bit-Schlüssel erreichte Sicherheit für den multiplikativen Logarithmus ungefähr der eines 224-Bit-Schlüssels für den Logarithmus über einer elliptischen Kurve. Ein 512-Bit-Schlüssel für eine elliptische Kurve entspricht einer Länge von 15360 Bit eines RSA-Schlüssels.

###### Elliptische Kurven

Eine elliptische Kurve E über einem endlichen Körper (in dem 0 ≠2, 3) ist eine Gleichung

y2 = x3 + ax + b

Asymmetrische Kryptosysteme

für die Koeffizienten a und b, sodass die Kurve nicht singulär ist, d. h. ihre Diskriminante ist ungleich Null: 4 a3 + 27 b2 ≠ 0.

Die Gleichung y2 = x3 + ax + b ist die Weierstraß-Normalform, es gibt jedoch mehrere andere, die sich als rechnerisch effizienter erwiesen haben, wie z. B. Montgomery:

By2 = x3 + Ax2 + x mit B A2 − 4 ≠ 0 .

Ist die Charakteristik 2, also IFq mit q = 2n, dann lautet die Gleichung y2 + cxy + dy = x3

+ ax + b.

Nach der Wahl eines Definitionsbereichs (z. B. +, \*, 0, 6 oder IFp für eine Primzahl p) bilden die Punkte (x, y) , die diese Gleichung lösen, E(x,y) = 0, eine Kurve in der Ebene. Diese Ebene, 0 steht für die übliche kartesische Ebene, + steht für ein Gitter von Punkten und + / m + steht für das endliche Gitter von Punkten innerhalb des Quadrats der Länge m , dessen untere linke Ecke im Ursprung liegt.

Zusätzlich zu den Punkten in der Ebene gibt es auch den Punkt in der Unendlichkeit (oder Idealpunkt), der mit 0 bezeichnet wird. Die Punkte der elliptischen Kurve sind also gegeben durch

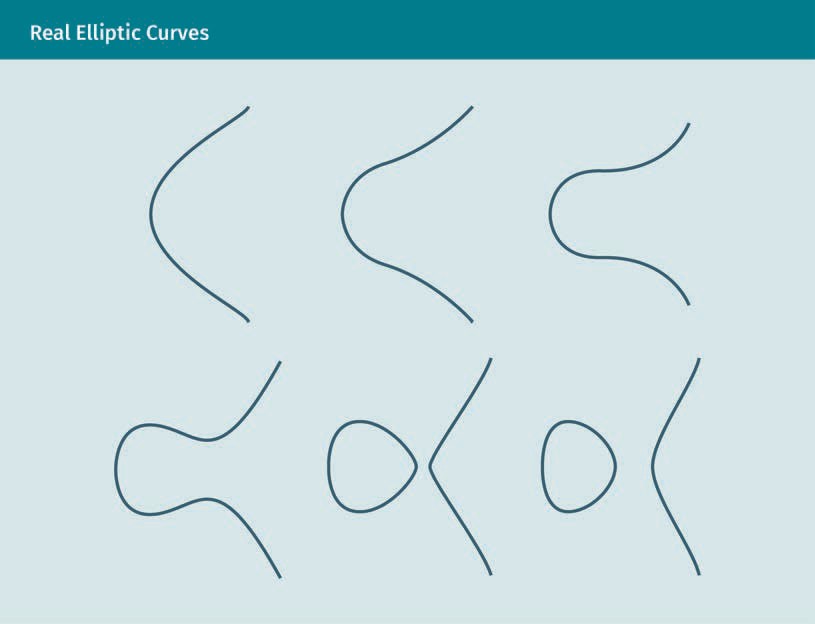
E: = alle Punkte x, y mit E x, y = 0 ∪ 0 ,

wobei der Begriff des Punktes vom Definitionsbereich abhängt. Auf einem endlichen Körper IFq ist die Anzahl der Punkte # E begrenzt durch q + 1 – t mit t ≤ 2 √{q}, also asymptotisch gleich # IF \* = q –1. Der Algorithmus von Schoof (1999) kann dies in etwa n5 Operationen für n *=* log2 q die Anzahl der binären Ziffern von q berechnen.

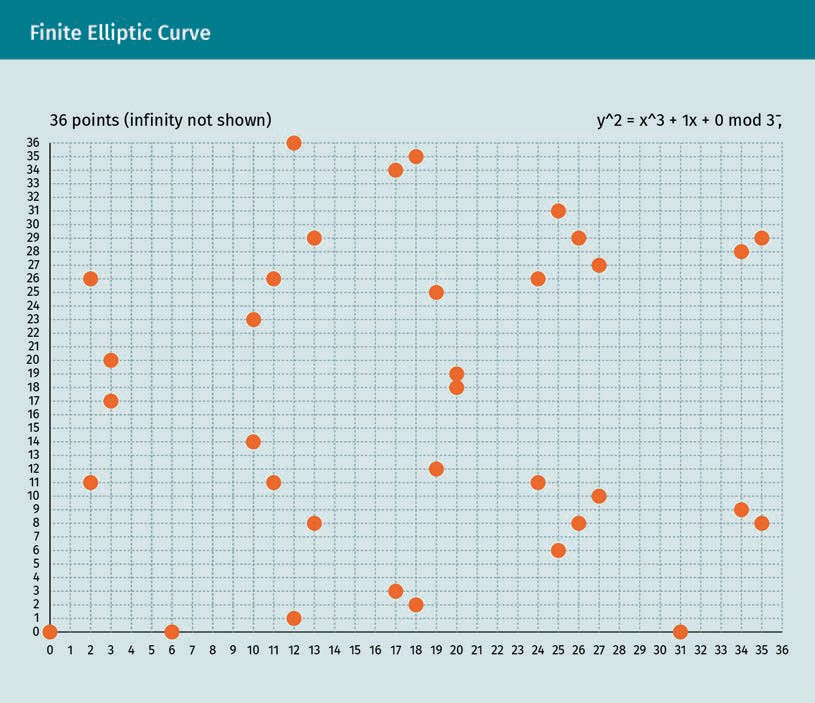
q

Stetige und diskrete endliche Kurven

Für den Definitionsbereich 0 nehmen die Kurven in der Ebene der reellen Zahlen für verschiedene Parameter a und b die folgenden Formen an:



Auf endlichen Körpern erhalten wir hingegen eine diskrete Menge von Punkten (symmetrisch um die mittlere horizontale Achse).



Kurven in der Kryptografie

Damit der kryptografische Algorithmus auf dieser Kurve sicher ist, d. h. die Berechnung des Logarithmus auf ihr viel Zeit benötigt, gibt es Einschränkungen bei der Wahl von q = pn und der elliptischen Kurve (bei ihren definierenden Koeffizienten a und b). Als Richtwert gilt zum Beispiel, dass q ≥ 2224, um bei einer Schlüssellänge von 2048 Bit genauso sicher zu sein wie RSA. Eine sichere Wahl ist zum Beispiel Curve25519, die durch

y2 = x3 + 486662x2 + x

über IFq mit q = p2 und p = 2255 – 19 (daher der Name) gegeben ist. Die Anzahl ihrer Punkte beträgt # E = 2252 + 277423177773585193790883648493. Diese Kurve wurde als neutrale Alternative zu den empfohlenen, aber bald schon bezweifelten Kurven des NIST populär.

###### Schlüsselaustausch unter Benutzung elliptischer Kurven

ECC verwendet den Diffie-Hellman-Schlüsselaustausch, um einen geheimen Schlüssel zu erstellen, ihn in einen kryptografischen Hash umzuwandeln und ihn zur Verschlüsselung der Kommunikation durch einen symmetrischen kryptografischen Algorithmus zu verwenden.

Die Verschlüsselung durch ECC ist standardisiert durch das ECIES (Elliptic Curve Integrated Encryption Scheme), ein hybrides Verfahren (asymmetrische Kryptografie mit symmetrischer Kryptografie).

Sobald der gemeinsame geheime c-Schlüssel (ein Punkt auf der endlichen elliptischen Kurve) vereinbart ist, leiten Andrea und Robert einen Schlüssel für eine symmetrische Chiffre wie AES oder 3DES ab. Die Ableitungsfunktion, die eine geheime Information in das entsprechende Format umwandelt, wird Schlüsselableitungsfunktion (Key Deviation Function KDF) genannt. Eine solche standardisierte Funktion ist ANSI-X9.63-KDF mit der Option SHA-1. Das TLS-Protokoll verwendet zum Beispiel die x-Koordinate des Punktes c, verknüpft sie mit Zahlen, die sich auf die Verbindung beziehen, und berechnet einen kryptografischen Hash dieser verknüpften Zahl.

Wir wollen nun das Diffie-Hellman-Protokoll von der Multiplikation auf einem endlichen Körper auf die Addition auf einer endlichen elliptischen Kurve übertragen. G bezeichne einen Punkt auf der Kurve, und

xG = G + ! + G

die x-fach iterierte Addition über die endliche elliptische Kurve (anstelle von g und gx = g ·g für einen endlichen Körper).

Ansatz

Die kritische kryptografische Zahl ist die Ordnung n des Basispunktes G, die groß genug sein sollte.

Hier ein Beispiel für einen Basispunkt. Die elliptische Kurve Curve25519 mit

y2 = x3 + 486662x2 + x

über IFq mit q = p2, mit p = 2255 – 19, legt als Basispunkt G = (xG, yG) fest, der eindeutig bestimmt durch folgenden Ablauf:

Schritte

1. Im ECDH-Protokoll (Elliptic Curve Diffie-Hellman) können Andrea und Robert offen einen geheimen Schlüssel erstellen, indem sie eine Potenz q aus einer geeigneten Primzahl p, eine geeignete elliptische Kurve E über IFq und einen geeigneten Punkt G in E kombinieren.
2. Um eine Hälfte des Schlüssels zu erzeugen, wählt Andrea eine Zahl a, berechnet A = a G und übermittelt A an Robert.
3. Um eine weitere Hälfte des Schlüssels zu generieren, wählt Robert eine Zahl b, berechnet B ≡b G und sendet B an Andrea.

Der gemeinsame geheime Schlüssel zwischen Andrea und Robert lautet

c: = bA = baG = abG = aB .

Damit beide denselben Schlüssel c berechnen können, muss die Addition das Assoziativ- und Kommutativgesetz erfüllen, d. h. E muss eine Gruppe bilden.

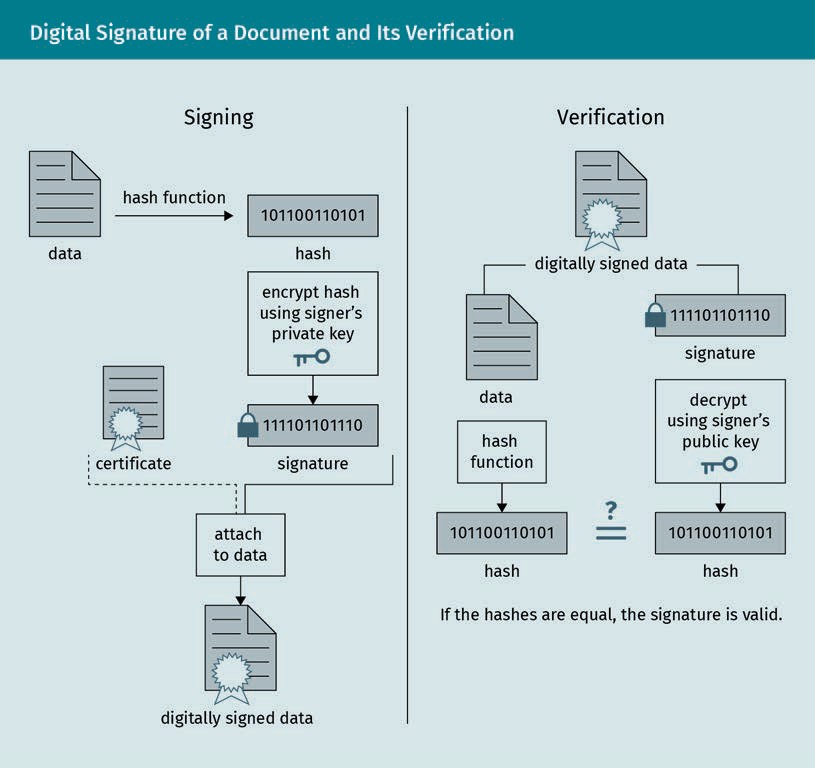
Das ECDHE-Protokoll, bei dem das zusätzliche E für „ephemeral“ (flüchtig, kurzlebig) steht, verwendet den gleichen Schlüsselaustausch wie das ECDH-Protokoll, verwirft aber direkt nach der Sitzung die Schlüssel. Diese werden notwendigerweise durch permanente Schlüssel signiert, um die Identität zu bezeugen (Corbellini, 2015b).

### Signaturen

Bei der Public-Key-Kryptografie werden, wie oben erwähnt, zwei Schlüssel genutzt: ein öffentlicher und ein privater Schlüssel. Normalerweise wird der öffentliche Schlüssel zum Verschlüsseln und der private Schlüssel zum Entschlüsseln genutzt. So kann ein Text von der verschlüsselnden Person (Andrea) nur an eine Person sinnvoll übertragen werden, nämlich an die entschlüsselnde Person (Robert). Die Rollen des öffentlichen und des privaten Schlüssels können vertauscht werden. Der private Schlüssel wird dann zum Verschlüsseln verwendet, während der öffentliche Schlüssel zum Entschlüsseln verwendet wird. Auf diese Weise kann der Verschlüsselnde allen Entschlüsselnden (denjenigen, die den öffentlichen Schlüssel besitzen) beweisen, dass er im Besitz des privaten Schlüssels ist: der digitalen Signatur.

Die Theorie hinter der Verschlüsselung mit dem öffentlichen Schlüssel (digitale Nachrichten) oder dem privaten Schlüssel (digitale Signatur) ist beinahe dieselbe – lediglich die Rollen der Argumente der Falltürfunktion sind vertauscht. Beim RSA-Algorithmus zum Beispiel ist dieser Austausch von Variablen tatsächlich alles, was passiert. In der Praxis verschlüsseln der private Schlüssel und ein kryptografischer Hash in der Regel jedoch die Auffüllungen des Klartextes. So können Schwachstellen vermieden werden, die den Schlüssel bei zu kurzen Texten offenbaren könnten. Bei der Verschlüsselung mit öffentlichen Schlüsseln ist zwar die Funktion, mit welcher der Text (die Auffüllung) zunächst umgewandelt wird, leicht umkehrbar; andererseits ist jedoch bei der Verschlüsselung die Funktion, mit welcher der Text zunächst umgewandelt wird (der Hash), mit privaten Schlüsseln kaum umkehrbar.

Digitale Signatur Auf diese Verschlüsselung einer Nachricht mit dem privaten Schlüssel folgt die Entschlüsselung mit dem öffentlichen Schlüssel, um zu überprüfen, ob die ursprüngliche Nachricht mit dem privaten Schlüssel verschlüsselt wurde.



###### RSA-Signaturalgorithmus

Beim RSA-Signaturalgorithmus besteht der einzige Unterschied darin, dass die Exponenten E und d ihre Rollen tauschen, d. h. die signierte Nachricht ist M = md (statt mE).

Sowohl Signieren als auch Dechiffrieren sind durch • d für den privaten Schlüssel d gegeben. Das Signieren eines verschlüsselten Dokuments (für den öffentlichen Schlüssel E, der d entspricht) ist also gleichbedeutend mit dessen Entschlüsselung! In der Praxis werden verschiedene Schlüsselpaare zum Verschlüsseln/Entschlüsseln, sowie zum Signieren/Verifizieren verwendet und ein kryptografischer Hash h(d) des Dokuments d wird signiert, d. h. eine kleine Zahl, die das Dokument identifiziert.

Hierbei wird also anstelle der Originalnachricht ein kryptografischer Hash (z. B. mit dem Algorithmus MD5) der Originalnachricht signiert, und zwar mit zusätzlichen Informationen wie dem Namen des Signierenden und den Algorithmen, die zur Verschlüsselung und Berechnung des Hash verwendet wurden.

###### Digital Signature Algorithm (DSA)

Elgamal (1985) zeigte als Erster, wie ein Verschlüsselungs- und Signaturalgorithmus auf der Grundlage des Diffie-Hellman-Protokolls entwickelt werden konnte. Zwar wird der Verschlüsselungsalgorithmus nur selten verwendet (das Standard-Kryptografie-Befehlszeilentool GnuPG bietet ihn als erste Alternative zu RSA an), aber der Signaturalgorithmus steckt hinter dem des Digital Signature Algorithm (DSA), der im 1994 vom NIST herausgegebenen Digital Signature Standard (DSS) der US-Regierung verwendet wird. Der Elliptic Curve DSA (ECDSA) ist eine Variante des DSA, die Punkte auf endlichen (elliptischen) Kurven anstelle von ganzen Zahlen verwendet.

### Schlüsselaustausch und Public-Key-Infrastrukturen

Im Vergleich zur symmetrischen Kryptografie entfällt bei der asymmetrischen Verschlüsselung das Risiko der Aufdeckung des Entschlüsselungs-Schlüssels, das mit dem Schlüsseltausch verbunden ist. Diese sichere Kommunikation mit jedermann über einen unsicheren Kanal ist ein großer Vorteil gegenüber der symmetrischen Kryptografie. Wir wollen uns zunächst die klassischen Methoden zum Austausch eines symmetrischen Schlüssels ins Gedächtnis rufen, bevor wir das asymmetrische Gegenstück betrachten. Die asymmetrische Kryptografie hat es zwar ermöglicht, einen geheimen Schlüssel offen auszutauschen, aber diese Annehmlichkeit verschleiert die Identität des Schlüsselbesitzers und macht sie anfällig für einen „Man-in-the-Middle“-Angriff. Durch die Nutzung von Zertifikaten (digitale Signaturen öffentlicher Schlüssel durch Dritte) unterbindet die Public-Key-Infrastruktur dieses Risiko.

###### Symmetrische Kryptosysteme

Ein symmetrischer Schlüssel muss heimlich weitergegeben werden. Mögliche Methoden sind:

* Die Ableitung von einem Basisschlüssel unter Verwendung einer Schlüsselableitungsfunktion (Key Derivation Function, KDF), einer kryptografischen Hashfunktion, die einen geheimen Schlüssel aus geheimen – und möglicherweise anderen öffentlichen – Informationen ableitet, z. B. einer eindeutigen Zahl,
* Die Erstellung eines Schlüssels aus Schlüsselteilen, die sich im Besitz verschiedener Personen befinden, z. B. als Analogie zum One-Time-Pad. Wenn *s* die geheime (binäre) Zahl ist, dann ist s = s1 ⊕s2 ⊕…⊕sn für die Teilgeheimnisse s1, s2, … . Die Rekonstruktion von s ist nur möglich, wenn alle s1, s2, … kombiniert werden.
* Die Übermittlung über einen anderen Kanal, z. B. persönlich, einen versiegelten Brief, per Telefon oder durch Quantenverschränkung, bei der Quantenpartikel so miteinander verbunden sind, dass sich der Zustand des einen Teilchens unabhängig von der räumlichen Trennung sofort auf das andere überträgt, obwohl sich Informationen nicht schneller als mit Lichtgeschwindigkeit bewegen können.

###### Man-In-The-Middle-Angriff (MITM)

Der große Vorteil der asymmetrischen Kommunikation besteht darin, dass die Geheimhaltung von der Authentifizierung getrennt werden kann, d. h. für einen erstellten Geheimtext wird nur der jeweilige öffentliche Schlüssel zur Entschlüsselung benötigt. Hinzu kommt, dass es keinerlei Informationen über den Entschlüsselnden gibt. Für die Kommunikation zwischen mehreren Parteien, z. B. Andrea, Robert und Eva, müssen die öffentlichen Schlüssel authentifiziert werden, d. h. die Entschlüsselnden, Robert und Eva, müssen im Verzeichnis der Verschlüsselnden, also Andrea, authentifiziert werden. Andernfalls könnte Eva Andrea vorgaukeln, dass diese mit Robert kommuniziert, indem sie ihren öffentlichen Schlüssel durch den von Robert ersetzt.

Bei einem MITM platziert sich der Angreifer zwischen den Kommunikationspartnern und nimmt gegenüber jedem von ihnen die Identität des anderen an, um Nachrichten abzufangen. Ein Beispiel:

1. Robert sendet seinen öffentlichen Schlüssel an Andrea. Eva fängt ihn ab und sendet Andrea ihren eigenen öffentlichen Schlüssel, gibt jedoch Robert als Eigentümer an. Wenn Andrea eine Nachricht an Robert sendet, dann verwendet sie, ohne es zu wissen, den öffentlichen Schlüssel von Eva!
2. Andrea verschlüsselt eine Nachricht mit dem öffentlichen Schlüssel von Eva und sendet sie an Robert.
3. Eva fängt die Nachricht ab und entschlüsselt sie mit ihrem privaten Schlüssel. Sie kann die Nachricht lesen und sie verändern.
4. Eva verschlüsselt die Nachricht mit Roberts öffentlichem Schlüssel.
5. Robert entschlüsselt die Nachricht mit seinem privaten Schlüssel und schöpft keinen Verdacht.

Sowohl Andrea als auch Robert sind davon überzeugt, dass sie den öffentlichen Schlüssel des jeweils anderen benutzt haben, aber in Wirklichkeit benutzen sie den von Eva!

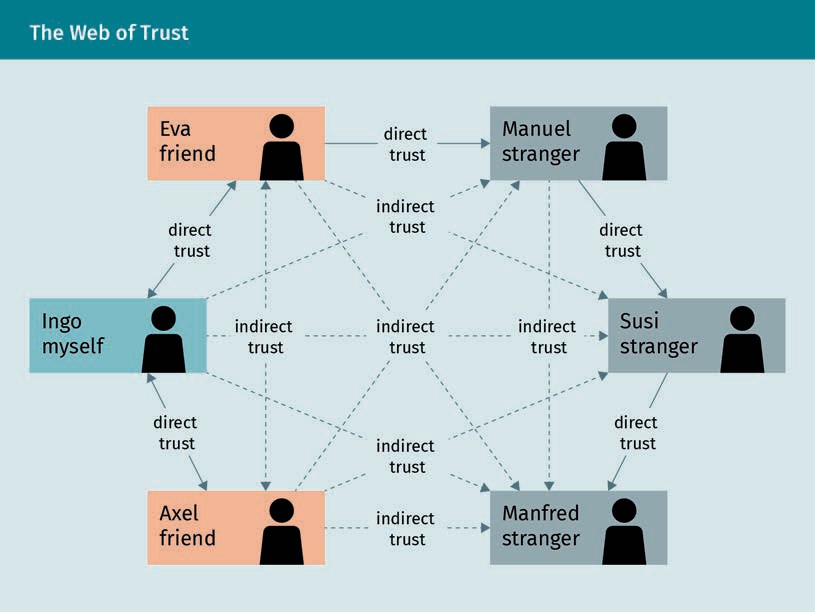
###### PKI

Eine Public-Key-Infrastruktur (PKI) eines Netzwerks stellt das Vertrauen zwischen den räumlich getrennten Benutzern her, indem sie diese zunächst authentifiziert und dann ihre öffentlichen Schlüssel durch Signieren (digitale Zertifikate) autorisiert und verteilt. In Institutionen und Unternehmen wird eine PKI oft als „Vertrauenshierarchie“ von Zertifizierungsstellen implementiert, während sie in weniger organisierten Gemeinschaften dezentralisiert sein kann und das Vertrauen von den Benutzern selbst aufgebaut wird. Eine PKI umfasst:

* + (digitale) Zertifikate. Es handelt sich dabei um öffentliche Schlüssel, die signiert sind, um ihre Benutzer zu authentifizieren. Neben dem Namen und dem Schlüssel enthalten sie weitere persönliche Daten, wie z. B. eine E-Mail-Adresse und in der Regel ein Ablaufdatum.
  + Zertifikatssperrliste (Certificate Revocation List, CRL). Dies ist eine Liste von Zertifikaten, die vor Ablauf ihrer Gültigkeit widerrufen wurden, weil der Schlüssel kompromittiert wurde oder der Schlüsselinhaber aufgrund seines Verschwindens nicht mehr vertrauenswürdig ist.
  + Verzeichnisdienst. Dies ist eine durchsuchbare Datenbank der emittierten Zertifikate. Beispiele hierfür sind eine Vertrauenshierarchie, ein LDAP-Server (Lightweight Directory Access Protocol, ein Standard, der von großen Unternehmen zur Verwaltung des Zugriffs von Benutzern auf Geräte, Drucker, Server und Anwendungsdaten verwendet wird) und das Web of Trust, ein Server, der eine Datenbank hostet, die über ein Webformular durchsucht werden kann.

###### Philosophie der Lösungen

Dritte, d. h. andere Identitäten mit privaten Schlüsseln, die mit ihren digitalen Signaturen bestätigen können, bestätigen die Identität des Schlüsselinhabers. Es kommt jedoch erneut das Problem der Identität des öffentlichen Schlüssels auf: Wie können wir die Identitäten der Besitzer privater Schlüssel sicherstellen? Es gibt zwei Lösungen: hierarchische Autoritäten und das Web of Trust.



Während beim Web of Trust die durch Vertrauen aufgebauten Verbindungen einen Graphen bilden, bilden sie beim Ansatz der hierarchischen Autoritäten einen Baum.

Hierarchische Autoritäten

Beim Ansatz der hierarchischen Autoritäten werden die Besitzer privater Schlüssel durch hierarchische Ebenen unterschieden. Auf der höchsten Ebene befinden sich die Stammautoritäten, denen wir bedingungslos vertrauen. VeriSign, GeoTrust und Commode sind zum Beispiel wichtige US-Zertifizierungsunternehmen. Ein Blick in den Ordner /etc/ssl/certs in der Linux-Distribution openSUSE zeigt eine deutsche (TeleSec der Deutschen Telekom AG, dem ehemaligen staatlichen Telekommunikationsunternehmen), drei spanische (Firmaprofesional, ACCVRAIZ1-Agencia de Tecnología y Certificación Electrónica und ACC RAIZ FNMT-Fábrica Nacional de Moneda y Timbre) sowie viele amerikanische Autoritäten.

Web of Trust

Im Web of Trust sind die Besitzenden privater Schlüssel nicht voneinander zu unterscheiden. Das Fehlen von Stammautoritäten, d. h. von bedingungslos vertrauenswürdigen Entitäten, wird durch das Vertrauen kompensiert, das ursprünglich dadurch aufgebaut wurde, dass eine Person den öffentlichen Schlüssel persönlich (z. B. bei einer Key-Sign-Party) oder über einen anderen Kanal erhalten hat (Website, E-Mail usw.) und die Prüfsumme über einen weiteren Kanal (Telefon, SMS, Instant Messenger usw.) übermittelt hat. Vertrauen wird im aufgebauten Netzwerk transitiv weitergegeben.

###### Standardisierung von Philosophien im Internet

Im Internet wurde das System des Vertrauens durch hierarchische Autoritäten durch das Schema X.509 standardisiert, das hauptsächlich zur Verschlüsselung der Kommunikation zwischen einem Benutzer und einer (kommerziellen) Website (aber auch zwischen Benutzern in Unternehmensumgebungen, wie z. B. die S/MIME-E-Mail-Verschlüsselung) verwendet wird, sowie durch das OpenPGP-Schema (wie es durch das GnuPG-Programm implementiert wurde), dessen Hauptanwendung die Verschlüsselung von E-Mails ist. Dieses Schema lehnt jede Hierarchie radikal ab. Benutzer können einen öffentlichen Schlüssel mit einer E-Mail-Adresse auf einem Public-Key-Server veröffentlichen, ohne auch nur zu bestätigen, dass sie Zugriff auf dieses E-Mail-Konto haben.

###### DANE

Die Internet Engineering Task Force (IETF) hat (in RFC 63941: DANE Use Cases und RFC 66982: DANE Protocol) das DANE-Protokoll vorgeschlagen, das darauf abzielt, die Protokolle TLS, DTLS, SMTP und S/MIME mit DNSSEC kryptografisch abzusichern. Mit DNSSEC können DNS-Resolver eine DNS-Auflösung authentifizieren, d. h. feststellen, ob sie mit derjenigen auf dem maßgeblichen DNS-Server identisch ist, indem sie dessen Signatur (des verbindlichen DNS-Servers) überprüft. Anstatt sich wie bei diesen Protokollen vollständig auf Zertifizierungsstellen (CAs) zu verlassen, können die Inhaber von Definitionsbereichen die CAs einschränken, die die Zertifikate der Domain validieren, und können selbst Zertifikate ausgeben, ohne auf CAs zu verweisen.

Bei Verwendung von CAs gibt es keine Einschränkung bei den ausgestellten Zertifikaten. Wenn Angreifende die Kontrolle über eine einzige der vielen CAs erlangen, denen der Client vertraut, können sie gefälschte Zertifikate für jede Domain emittieren. DANE ermöglicht es Clients, die DNS-Server zu fragen, welche Zertifikate vertrauenswürdig sind, so dass die Domain-Inhaber den Geltungsbereich einer CA einschränken können. Wenn den Benutzern ein Domain-Name-Zertifikat übergeben wird (als Teil des anfänglichen TLS-Handshakes), kann der Client das Zertifikat mit einem TLSA-Ressourcendatensatz (TLSA RR) abgleichen, der im DNS für den vom maßgeblichen DNS-Server authentifizierten Dienstnamen veröffentlicht wurde.

Web of Trust

Inhaber von privaten Schlüsseln bestätigen die Identität der anderen, indem sie sich unter Gleichen gegenseitig Vertrauen schenken.

Key-Sign-Partys  
Das sind Treffen, bei denen die Teilnehmer ihre öffentlichen Schlüssel austauschen und gegenseitig signieren.

Der am weitesten verbreitete Standard für PKI ist die Hierarchie der X.509-Zertifizierungsstellen. X.509 wurde erstmals 1998 veröffentlicht und wird von der Standardisierungsabteilung der Internationalen Fernmeldeunion (ITU-T) entwickelt. X.509 legt insbesondere ein Standardformat für elektronische Zertifikate und einen Algorithmus für die Validierung des Zertifizierungspfads fest. Von der IETF wurde das wichtigste Profil, PKIX-Zertifikat und CRL-Profil (PKIX), als Teil von RFC 3280, derzeit RFC 5280, entwickelt. Es wird von allen gängigen Webbrowsern wie Chrome und Firefox unterstützt, die über eine Liste vertrauenswürdiger X.509-Zertifizierungsstellen verfügen.

Im Einzelnen enthält die TLSA RR den Eintrag Certificate Usage, dessen Wert die Stelle einschränkt, die das Zertifikat für die Benutzer validieren darf. Je niedriger die Ebene, desto restriktiver ist sie.

* + Ebene 0: PKIX-TA (Zertifizierungsstellenzwang). Das Vertrauen des Clients beruht auf einer PKIX-Stelle.
  + Ebene 1: PKIX-EE (Servicezertifikatszwang). Das Vertrauen des Clients beruht auf einem PKIX-Zertifikat.
  + Ebene 2: DANE-TA (Trust-Anchor-Zusicherung). Das Vertrauen des Clients beruht auf einer Stelle, bei der es sich im Gegensatz zu PKIX-TA nicht um eine PKIX-Zertifizierungsstelle handeln muss.
  + Ebene 3: DANE-EE (Von Domain ausgestelltes Zertifikat). Das Vertrauen des Clients beruht auf einem Zertifikat, bei dem es sich, anders als bei PKIX-EE, nicht um ein PKIX-Zertifikat handeln muss.

Die DANE-Prüfung dient zur Bestätigung von Zertifikaten, die von öffentlichen Zertifizierungsstellen ausgestellt wurden. DANE-Werte (zwei und drei) geben Domain-Inhabern die Möglichkeit, ihre eigenen, selbst signierten Zertifikate für ihre TLS-gesicherten Dienste zu erstellen, ohne eine dem Client bekannte Zertifizierungsstelle einbeziehen zu müssen. Durch die Wahl zwischen „Trust Anchor“ (TA) und „End Entity“ (EE) können Domain-Inhaber selbst entscheiden, ob sie die DANE-Sicherheit an einem Zertifizierungsstellen- oder Server-Zertifikat verankern möchten.

###### Hybride Kryptosysteme

Hybride   
Verschlüsselung

Mit einem Zwei-Schlüssel-Algorithmus werden die Kommunikationsteilnehmer authentifiziert, indem die Nachrichten digital signiert werden, oder es wird ein Schlüssel für die Ein-Schlüssel-Kryptografie ausgetauscht, und so eine effiziente Kommunikation ermöglicht.

Zwischen zwei Parteien ist es üblich, einen Hash, z. B. einer Nachricht, als Verschlüsselungsschlüssel zu erstellen. MITM wird durch die Authentifizierung der entsprechenden öffentlichen Schlüssel bei einer Zertifizierungsstelle verhindert. Da kryptografische Algorithmen mit einem Schlüssel um einen beträchtlichen Faktor effizienter sind als kryptografische Algorithmen mit zwei Schlüsseln, wird die Zwei-Schlüssel-Verschlüsselung hauptsächlich für die hybride Verschlüsselung verwendet. Hierbei wird der Zwei-Schlüssel-Algorithmus entweder genutzt, um die Kommunikationspartner durch digitales Signieren der Nachrichten zu authentifizieren, oder um einen Schlüssel für die Ein-Schlüssel-Kryptografie auszutauschen, sodass anschließend eine effiziente, sichere Kommunikation gewährleistet ist.

Mit dem TLS-Protokoll (Transport Layer Security; früher SSL) werden sichere Websites im World Wide Web verschlüsselt. Ein darin eingebettete kryptografische Paket wie TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (Identifizierungscode 0x00 0x0a) nutzt RSA zur Authentifizierung und zum Austausch der Schlüssel, 3DES im CBC-Modus zur Verschlüsselung der Verbindung und SHA als kryptografischen Hash.

Zusammenfassung

Die symmetrische Kryptografie leidet unter dem Problem der Schlüsselverteilung. Die asymmetrische Kryptografie löst dieses Problem scheinbar auf Anhieb, indem sie die Verwendung unterschiedlicher Schlüssel zum Ver- und Entschlüsseln ermöglicht. Allerdings muss dazu die Identität des Schlüsselinhabers bestätigt werden. Dies kann persönlich oder durch Dritte geschehen, z. B. durch Identitäten mit privaten Schlüsseln, die mit ihren digitalen Signaturen die Eigentümerschaft bestätigen. Es stellt sich jedoch erneut das Problem der Identität des privaten Schlüssels: Wie können wir die Identitäten dieser Besitzer privater Schlüssel sicherstellen? Dafür gibt es zwei Lösungen. Bei dem Ansatz über hierarchische Autoritäten werden die Besitzer privater Schlüssel durch hierarchische Ebenen unterschieden. Auf der höchsten Ebene befinden sich die Stammautoritäten, denen wir bedingungslos vertrauen. Im Web of Trust wird das Vertrauen von einem zum anderen übertragen.

Asymmetrische Kryptografie beruht auf einer Falltürfunktion, die einfach sein muss, deren Umkehrung aber ohne Kenntnis einer Abkürzung – des Schlüssels – praktisch nicht möglich sein darf.

Die Schwierigkeit, die Umkehrung zu berechnen, entspricht der Schwierigkeit der Entschlüsselung, d. h. der Umkehrung der Verschlüsselung. Die Berechnung der Umkehrfunktion wird durch modulare (oder zirkuläre) Arithmetik erschwert.

Die derzeit am weitesten verbreitete Kryptografie nutzt elliptische Kurven. Das Difﬁe-Hellman-Protokoll (über IFp) hat ein Analogon über elliptischen Kurven. Der Vorteil der Verwendung elliptischer Kurven sind kürzere Schlüssel. Kleine Schlüssel für ECC erreichen das gleiche Maß an Sicherheit wie große Schlüssel für RSA oder DH. So entspricht beispielsweise die Sicherheit eines 224-Bit-Schlüssels bei ECC der eines 2048-Bit-Schlüssels bei RSA oder DH. Dieser Faktor bei der Reduzierung der Schlüsselgröße entspricht einem ähnlichen Faktor bei der Reduzierung des Rechenaufwands.



# Lektion 4

## Authentifizierung

#### LERNZIELE

Nach der Bearbeitung dieser Lektion wissen Sie …

… über die Benutzerauthentifizierung durch Passwort, persönliche Identifikationsnummer (PIN), Chipkarte und biometrische Identifikatoren Bescheid.

… wie die Authentifizierung aus der Ferne am sichersten erreicht wird.

… wie der geheime Schlüssel durch Challenge- und Response-Protokolle niemals preisgegeben wird.

… wie Informationen durch Zero-Knowledge-Beweise nicht durchsickern.

… wie Kerberos zwischen Benutzern und Servern vermittelt, ohne dass einer der beiden vor dem jeweils anderen sein Passwort preisgibt.

DL-E-DLMCSEAITSC02-U04

1. Authentifizierung

### Einführung

Authentifizierung ist die Identifikation einer Person oder von Daten, oder die Bestätigung, dass die Benutzer wirklich diejenigen sind, die sie vorgeben zu sein, z. B. bei der Anmeldung an einem Server durch Eingabe eines Benutzernamens und eines Passworts. Die Authentifizierung prüft zudem, ob eine Nachricht, z. B. eine per E-Mail an eine Bank gesendete Anweisung, authentisch ist, d. h. ob sie zwischen dem Zeitpunkt des Versands und dem Zeitpunkt ihres Eintreffens unverändert geblieben ist.

Diese Lektion behandelt ausschließlich die erste Art der Authentifizierung, die Identifikation von Personen, der Benutzer, die im Internet besonders wichtig ist, da Personen sehr weit entfernt sein können. In diesem Sinne zeigt die Identifikation einem Computer oder einem Netzwerk, wer die Benutzer sind, in der Regel durch deren Benutzernamen (oder Kontonamen). Darauf folgt die Authentifizierung, die einen Computer oder ein Netzwerk davon überzeugt, dass eine Person diejenige ist, die sie vorgibt zu sein.

Zur Authentifizierung können Benutzer Informationen, die nur sie kennen, als Identitätsnachweis verwenden. Die Benutzer können ihr Passwort oder eine persönliche Identifikationsnummer (PIN) angeben oder ein Objekt vorweisen, wie z. B. ein Software-Zertifikat mit öffentlichen oder privaten Schlüsseln, ein Hardware-Zertifikat, wie z. B. ein Token oder eine Chipkarte, die einen Mikroprozessor enthält, der einen Schlüssel speichert und kryptografische Algorithmen ausführt, oder ein mobiles Endgerät oder ein E-Mail-Konto, das einen Code empfängt. Es gibt auch biometrische Identifikatoren, wie Fingerabdrücke, Gesichtserkennung oder Iris-Scanner.

Die Authentifizierung ist nicht zu verwechseln mit der Autorisierung, der abschließenden Bestätigung der Authentifizierung, mit der die Berechtigung der Benutzer zum Zugriff auf bestimmte Inhalte festgelegt wird.

Das Authentifizierungsprotokoll kann ein einfaches Protokoll, ein Zwei-Faktor-Protokoll, einseitig oder wechselseitig sein. Wenn ein einfacher Beweis ausreicht, wie zum Beispiel eine PIN oder ein Passwort, ist es einfach. Es ist ein Zwei-Faktor-Protokoll, wenn mehr als ein Beweis erforderlich ist, zum Beispiel eine PIN und eine Chipkarte. Es handelt sich um eine einseitige Authentifizierung, wenn die Authentifizierung nur in eine Richtung erfolgt, z. B. wenn sich Partei A gegenüber Partei B authentifiziert, oder um eine wechselseitige Authentifizierung, wenn die Authentifizierung in beide Richtungen erfolgt, z. B. wenn sich Partei B auch gegenüber Partei A authentifiziert. Die meisten Betriebssysteme (wie Linux) und Anwendungen speichern einen Hash der Authentifizierungsdaten und nicht die Authentifizierungsdaten selbst. Bei der Authentifizierung muss der Hash der eingegebenen Authentifizierungsdaten mit dem gespeicherten Hash übereinstimmen, damit sie verifiziert werden können. Selbst wenn Angreifer die Hashes kennen, ist es praktisch unmöglich, festzustellen, welche Authentifizierungsdaten zu einem bestimmten Hash passen.

### Passwörter

Passwörter werden in den Systemdaten der autorisierten Benutzer gespeichert. Dies ist die gängigste Methode zur Authentifizierung: kostenlos, praktisch und privat. Es sollte leicht zu merken, aber schwer zu erraten sein.

###### Vorteile

Die Authentifizierung durch etwas, das wir *wissen* (z. B. ein Passwort), hat gegenüber der Authentifizierung durch etwas, das wir *sind* (biometrische Daten wie ein Fingerabdruck), den Vorteil, dass keine komplizierte Hardware erforderlich ist, dass sie sicher gespeichert wird und dass sie nicht gefälscht werden kann.

Die Authentifizierung durch etwas, das wir *wissen* (z. B. ein Passwort), hat gegenüber der Authentifizierung durch etwas, das wir *haben* (z. B. eine Chipkarte), den Vorteil, dass wir es nicht mit uns herumtragen müssen, dass es transparent gespeichert ist und dass es nicht verloren gehen, gestohlen oder erpresst werden kann.

###### Nachteile

Je aussagekräftiger ein Passwort ist, desto leichter ist es zu erraten (z. B. ein Wort in der Sprache des Benutzers), doch je weniger ein Muster vorhanden ist, desto schwieriger ist es, es sich zu merken. Ein Kompromiss ist eine Passphrase, d. h. ein vollständiger Satz anstelle eines einzelnen Wortes. Dieser ist zwar länger, aber sein Inhalt ist aussagekräftiger und daher leichter zu merken als eine unstrukturierte Zeichenfolge. Zur Verkürzung wird der erste Buchstabe eines jeden Wortes genommen. So kann zum Beispiel aus „Es ist besser, eine Kerze anzuzünden, als die Dunkelheit zu verfluchen“ „EibeKaadDzF“ werden.

Eine weitere Möglichkeit ist die Verwendung eines Passwortmanagers. Dieses Programm speichert alle Passwörter in einer Datei, die mit einem Master-Passwort verschlüsselt wird. Dank dieses Programms brauchen wir uns die Passwörter nicht mehr zu merken, und deshalb können sie auch schwieriger sein. Dieser Komfort ist nicht ohne Risiko: Das Master-Passwort ist immer noch ein Passwort, das bei Nutzung für ein weniger wichtiges Konto die Sicherheit aller anderen Konten gefährden kann, einschließlich der wichtigsten Konten.

###### Angriffe

Mögliche Angriffe während der Eingabe des Passworts einer anderen Person umfassen:

* Spionage bei der Eingabe des Passworts. Abhilfemaßnahmen sind lästig, z. B. das Abdecken der getippten Buchstaben oder der Tastatur oder die Verwendung einer Bildschirmtastatur.
* Keylogging (Abfangen der Tastatureingaben)
* Login-Spoofing, bei dem das Konto eines Benutzers ein Login-Eingabeformular fälscht, so dass die eingegebenen Login-Daten des nächsten Benutzers gespeichert werden, eine Fehlermeldung angezeigt wird und der erste Benutzer ausgeloggt wird.
* Das Erfragen des Passwortes bei Benutzern, entweder per E-Mail oder am Telefon als Systemadministrator.
* Systemadministratoren nach dem Passwort fragen, indem wir uns als angebliche Benutzer ausgeben, die ihr Passwort vergessen haben.
* Benutzer auffordern, ihr Passwort über ein angebliches Eingabeformular zu ändern.

Angriffe auf das gespeicherte Passwort einer anderen Person nutzen vor allem die folgende Schwäche aus: Da wir uns Passwörter merken müssen, tendieren diese dazu, einem Muster zu folgen. Sie können zum Beispiel aus (Geburts-)Datumsangaben und bestimmten Namen aufgebaut sein. Übliche Vorgehensweisen sind die Umkehrung der Schreibweise oder die Änderung der Großschreibung der folgenden Wörter:

* + Benutzernamen,

Passwort

###### Eine geheime Zeichenfolge, die mit einer Benutzeridentität verknüpft ist und Zugang zu einem System (z. B. einem Computer) gewährt, wird als Passwort bezeichnet.

* + Vor- oder Nachnamen,
  + Namen von Ehepartnern, Kindern, Freunden oder Haustieren,
  + Telefonnummern oder Adressen

Diese wahrscheinlicheren Kandidaten können zuerst ausprobiert werden. Alternativ können Angreifer auch bereits bekannt gewordenen Passwörter verwenden.

### Challenge-Response- und Zero-Knowledge-Protokolle

Challenge-Response-Protokoll

Dieses Protokoll stellt eine Aufgabe, die nur von Benutzern gelöst werden kann, die über die Authentifizierungsdaten verfügen.

Zero-Knowledge-  
Protokoll

Dieses Protokoll (ursprünglich vorgestellt von Goldwasser, Micali und Rackoff) weist die Kenntnis eines Geheimnisses nach, gibt aber keine anderen Informationen preis.

Nonce

In einem kryptografischen Protokoll (z. B. zur Authentifizierung) steht dies für eine (zufällig erzeugte) Zahl, die nur ein einziges Mal verwendet wird.

Challenge-Response-Protokolle bestehen in der Regel aus einer von zwei Aufgaben. Erstens kann es sich um eine kryptografische Hashfunktion oder eine Verschlüsselungsfunktion eines symmetrischen kryptografischen Algorithmus handeln, dessen geheimer Schlüssel von Beweisenden und Verifizierenden gemeinsam genutzt wird. Die Verifizierenden generieren eine Zufallszahl, und die Beweisenden antworten mit dem Ergebnis der Anwendung der Hashfunktion auf diese Zahl. Alternativ kann es sich um einen digitalen Signaturalgorithmus handeln, bei dem die Beweisenden mit ihrem privaten Schlüssel eine von Verifizierenden generierte Nachricht unterzeichnet, die die Verifizierenden mit dem öffentlichen Schlüssel überprüfen.

Ein Zero-Knowledge-Protokoll geht – theoretisch – noch weiter, da es zeigt, wie Beweisende Verifizierenden die Kenntnis eines Geheimnisses beweisen kann, ohne dass andere Informationen (als dieser Wissensnachweis) preisgegeben werden (Goldwasser et al., 1985).

###### Challenge-Response

Ein Challenge-Response-Protokoll stellt eine Aufgabe, die nur von Benutzern mit zusätzlichen Authentifizierungsdaten gelöst werden können. Ein Beispiel für eine Aufgabe wäre ein (zufällig) generierter Wert, der mit dem Passwort für den Verschlüsselungscode verschlüsselt wird. Als Beispielantwort wird ein ähnlich verschlüsselter Wert angegeben, der von dem Originalwert abhängt und damit beweist, dass die Benutzer den Originalwert entschlüsseln konnten. Dieses beispielhafte Protokoll ist auf Chipkarten sehr verbreitet.

Dieser (zufällig) generierte Wert ist eine Nonce und verhindert einen Replay-Angriff, bei dem die ausgetauschten Daten aufgezeichnet und später erneut gesendet werden.

Bei CRAM-MD5 oder DIGEST-MD5 zum Beispiel ist die Aufgabe der (iterierte) Hash des Passworts. Die Antwort wird nach dem Zufallsprinzip generiert und besteht aus dem Hashwert des Passworts und einem Wert, der vom Originalwert abhängt. Ein mögliches Szenario könnte sein:

1. Ein Server sendet eine eindeutige Aufgabe an den Client.
2. Der Client berechnet die Antwort als Hash (Aufgabe + Geheimnis) und sendet sie an den Server.
3. Der Server berechnet den erwarteten Wert der Antwort und verifiziert, dass er mit der Antwort des Clients übereinstimmt.

Solche verschlüsselten oder gehashten Informationen geben das Passwort selbst nicht preis, können aber genug Informationen liefern, um es mit einem Wörterbuchangriff (oder einer Rainbow-Table) zu ermitteln, d. h. durch das Ausprobieren vieler Werte. In jeden Austausch geht eine zufällig generierte Information ein, ein Salt, wie z. B. die aktuelle Zeit.

Nonce, Salt und IV (Initialisierungsvektor) sind Zahlen, die (in der Regel) nach dem Zufallsprinzip generiert, offengelegt und einmalig in einem kryptografischen Prozess verwendet werden, um dessen Sicherheit zu verbessern, indem sie ihn einzigartig machen. Eine Nonce ist eine Zahl, die ein einziges Mal in einem kryptografischen Protokoll verwendet wird, um den Austausch eindeutig zu machen. Ein Salt wird als zusätzliche Eingabe für eine Hashfunktion verwendet, um die Eingabe einzigartig zu machen (so dass dieselbe ursprüngliche Eingabe, die gehasht wird, eine andere Ausgabe ergibt). Ein Initialisierungsvektor ist eine Zahl, die als zusätzliche Eingabe für eine Verschlüsselung (einer Blockchiffre) verwendet wird, um deren Eingabe einzigartig zu machen, sodass dieselbe ursprüngliche Eingabe, die mit demselben Schlüssel verschlüsselt wird, eine unterschiedliche Ausgabe ergibt.

Challenge-Response-Protokolle, wie sie im Folgenden vorgestellt werden, werden z. B. in objektrelationalen Datenbanken wie PostgreSQL oder E-Mail-Clients wie Mozilla Thunderbird verwendet.

###### Digest-MD5

Digest-MD5 war ein gängiges Challenge-Response-Protokoll, das die in RFC2831 spezifizierte MD5-Hashfunktion verwendete. Es basierte auf der HTTP-Digest-Authentifi­zierung (wie in RFC2617 spezifiziert) und wurde von Melnikov (2011) obsolet gemacht.

###### Challenge-Response Authentication Mechanism (CRAM)

CRAM-MD5 ist ein Challenge-Response-Protokoll, das auf HMAC-MD5 basiert und zur Authentifizierung von Nachrichten die MD5-Hashfunktion verwendet (Krawczyk et al., 1997). Der RFC-Entwurf von Zeilenga (2008) empfiehlt, dies durch CRAM zu ersetzen. Die CRAM-Schritte sind: (1) der Server sendet dem Client eine Nonce, (2) der Client soll mit HMAC (Geheimnis, Nonce) antworten, und (3) der Server berechnet erneut HMAC (Geheimnis, Nonce) und prüft, ob es mit der Antwort des Clients übereinstimmt, um sich davon zu überzeugen, dass der Client das Geheimnis kannte.

Zu den Schwächen von CRAM gehört, dass keine gegenseitige Authentifizierung stattfindet, also die Identität des Servers nicht verifiziert wird, und dass die verwendete Hashfunktion MD5 schnell zu berechnen ist, womit Wörterbuchangriffe erleichtert werden. Stattdessen ist das Key-Stretching, d. h. die Verwendung einer Hashfunktion, die absichtlich rechenintensiv ist, vorzuziehen. Die letzte Schwachstelle ist die Passwortspeicherung, bei der manche Implementierungen das unverschlüsselte Passwort der Benutzer speichern, während andere (wie Dovecot) einen vorläufigen Hashwert des Passworts speichern. Dies verhindert zwar die Speicherung des unverschlüsselten Passworts, aber für die Authentifizierung mit CRAM-MD5 ist die Kenntnis des Hashwerts gleichbedeutend mit der Kenntnis des Passworts selbst.

Salt (englisch für Salz)

Eine (zufällig) generierte Zahl, die zur Eingabe einer Hashfunktion hinzugefügt wird, um die Ausgabe einzigartig zu machen, insbesondere wenn es sich bei der zusätzlichen Eingabe um geheime Informationen wie ein Passwort handelt.

###### Salted Challenge-Response Authentication Mechanism (SCRAM)

Salted Challenge-Response Authentication Mechanism (SCRAM) ist ein Challenge-Response-Protokoll zur gegenseitigen Authentifizierung (Menon-Sen et al., 2010), das CRAM-MD5 (Zeilenga, 2008) ablöst.

Während bei CRAM das Passwort des Clients als Hash auf dem Server gespeichert wird, reicht nun die Kenntnis des Hashs (anstelle des Passworts) aus, um den Client bei weiteren Authentifizierungen zu imitieren. Die Verantwortung wurde somit einfach vom Schutz des Passworts auf den entsprechenden Hash verlagert. SCRAM verhindert dies, indem es zusätzlich zu den auf dem Server gespeicherten Authentifizierungsinformationen (StoredKey), die ursprünglich aus dem Passwort des Clients (ClientKey) abgeleitet wurden, weitere Informationen verlangt.

Die Vorteile von SCRAM im Vergleich zu älteren Challenge-Response-Protokollen sind, dass die auf dem Server gespeicherten Authentifizierungsinformationen nicht ausreichen, um den Client zu imitieren (Menon-Sen et al., 2010). Insbesondere wird (a) ein Wörterbuchangriff (Rainbow-Tables) nach einem Leck in der Authentifizierungsdatenbank durch das Salt verhindert, (b) der Server kann sich gegenüber anderen Servern nicht als der Client ausgeben, da er nur teilweise Authentifizierungsinformationen speichert, und (c) die Wiederverwendung von Passwörtern nach einem Datenleck wird durch die Bindung des Hashwertes an einen einzigen Server verhindert. Nur die mit Salt versehene und gehashte Version eines Passworts wird bei der Anmeldung verwendet und Salt auf dem Server ist unveränderlich.

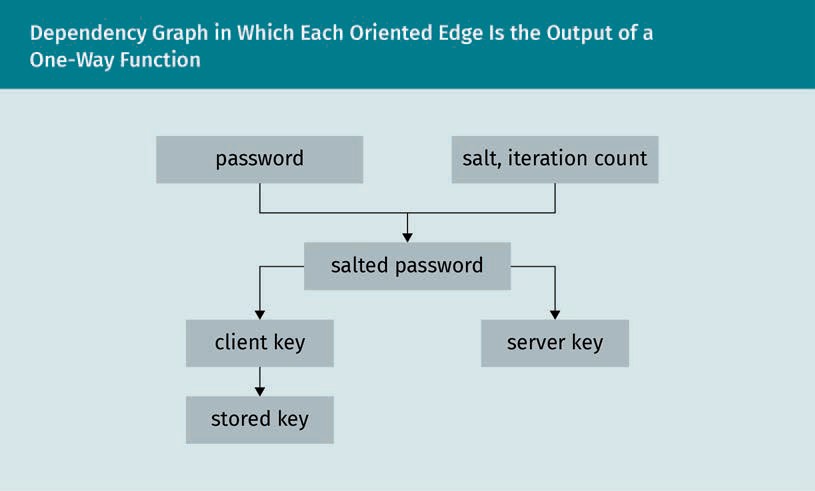
SCRAM unterstützt auch die gegenseitige Authentifizierung (durch den Client und den Server).

Erstellung, Übertragung und Speicherung von Schlüsseln

Wenn der Client ein Passwort erstellt, speichert der Server die abgeleiteten Schlüssel StoredKey und ServerKey zusammen mit den Parametern, die für ihre Ableitung verwendet wurden, wie folgt.

* 1. Der Client berechnet SaltedPassword, indem er die Passwort-Hashfunktion PBKDF2 (standardmäßig ist es PBKDF2, aber mittlerweile empfiehlt sich bcrypt) mit IterationCount viele Male auf die Eingabe anwendet. Die Eingabe besteht aus dem Passwort und dem Salt, d. h. SaltedPassword := PBKDF2( Passwort, Salt, IterationCount)
  2. Der Client berechnet ClientKey bzw. ServerKey durch Anwendung der HMAC-Funktion auf SaltedPassword mit den öffentlichen konstanten Strings „Client Key“ bzw. „Server Key“, d. h. ServerKey := HMAC(SaltedPassword, „Server Key“) und ClientKey := HMAC(SaltedPassword, „Client Key“)
  3. Der Client berechnet StoredKey durch Hashing von ClientKey, d. h. StoredKey := H(ClientKey) und sendet ServerKey und StoredKey an den Server (aber nicht ClientKey).

Der Server speichert StoredKey, ServerKey, Salt und IterationCount, um später Beweise von Clients zu prüfen und Beweise an Clients auszugeben. ClientKey wird zuerst verwendet, um den Client gegenüber dem Server zu authentifizieren, und ServerKey wird später vom Server verwendet, um sich gegenüber dem Client zu authentifizieren.



Der Server speichert nur den öffentlichen Teil der Wurzel (Salt und IterationCount) und die Blätter (StoredKey und ServerKey) dieses Baums, d. h. das Passwort wird nie an den Server gesendet. Es werden nur die folgenden Daten versandt: Salt, Iterationsanzahl, ServerKey und StoredKey, d. h. HMAC(SaltedPassword, „Server Key“) und H(HMAC(SaltedPassword, „Client Key“)).

Nach einem Einbruch in die Datenbank, d. h. nachdem Angreifer einen ServerKey gestohlen haben, muss das Passwort der Kunden nicht ersetzt werden, sondern nur das Salt und die Iterationsanzahl müssen geändert und ClientKey und ServerKey ersetzt werden.

Client-Authentifizierung gegenüber dem Server

Zur Authentifizierung durch den Server sendet der Client einen Authentifikator (der seinen Client-Namen und eine Client-Nonce enthält) an den Server. Der Server sendet dem Client ein Salt (salt), eine Iterationsanzahl (ic) und eine Server-Nonce (server-nonce). Daher kennen sowohl der Client als auch der Server AuthMessage := client-name, client-nonce, salt, ic, server-nonce. Der Client beweist Kenntnis des StoredKeys, indem er Folgendes berechnet:

1. ClientSignature := HMAC(StoredKey, AuthMessage) ClientProof := ClientKey ⊕ ClientSignature

Der Server stellt den ClientKey wieder her, indem er die ClientSignature berechnet (indem er den StoredKey aus dem Speicher und die AuthMessage aus diesem Austausch kennt) und den ClientKey' aus dem One-Time-Pad ClientProof „entschlüsselt“, indem er ClientKey' = ClientProof ⊕ ClientSignature berechnet.

Er berechnet StoredKey' durch H(ClientKey) und überprüft, ob der berechnete StoredKey' mit dem gespeicherten StoredKey übereinstimmt. Ist dies der Fall, wird der Client erfolgreich authentifiziert.

Wenn nur die ClientSignature gesendet würde, könnten sich Angreifer, die den StoredKey kennen, als der Client ausgeben. Stattdessen verlangt ClientProof zusätzlich, dass der Client den ClientKey kennt. Daher sollte der auf dem Server berechnete Wert von ClientKey' nach der Verifizierung sofort und unwiderruflich gelöscht werden.

Vorbehalt

Wenn Angreifer den StoredKey vom Server sowie die AuthMessage und den ClientProof aus einem Authentifizierungsaustausch kennen, können sie die ClientSignature und damit den ClientKey berechnen und sich gegenüber dem Server als Client ausgeben.

###### Zero-Knowledge-Beweise

Ein Zero-Knowledge-Protokoll zeigt, wie Beweisende die Kenntnis eines Geheimnisses gegenüber Verifizierenden beweisen können, so dass keine anderen Informationen als die Gültigkeit des Beweises (gegenüber jedermann, einschließlich der Verifizierenden) offengelegt werden.

Beweis ist hier probabilistisch zu verstehen, d. h. die Wahrscheinlichkeit, dass die Behauptung der Beweisenden wahr ist, ist über jeden vernünftigen Zweifel erhaben. Da die Beweise unabhängig voneinander sind, kann die Wahrscheinlichkeit durch eine Erhöhung ihrer Anzahl beliebig gesteigert werden. Die Unmöglichkeit, aus diesem Austausch Informationen über das Geheimnis zu gewinnen, hängt mit der rechnerischen Schwierigkeit zusammen, ein mathematisches Problem zu lösen.

Diese Eigenschaft hat zwei Vorteile. Erstens können die Verifizierenden keine Informationen erhalten, wenn sie sich nicht an das Protokoll halten. Jeder Beweis ist unabhängig von den anderen. Zweitens können sich die Verifizierenden gegenüber Dritten nicht als die Beweisenden ausgeben. Eine Aufzeichnung der Beweisenden hilft nicht dabei, Dritte zu überzeugen, denn die Sequenz hätte im Voraus gegenseitig abgesprochen werden können.

Vergleich mit klassischen Protokollen

Während die Behauptung, das Geheimnis zu kennen, allein nicht überzeugend ist, kompromittiert das Durchsickern von Informationen bei klassischen Protokollen, bei denen Beweisende (B) die Kenntnis eines Geheimnisses gegenüber Verifizierenden (V) beweisen, immer noch das Geheimnis. Wenn B ein Passwort an V übermitteln, dann erhalten V und alle, die diese Übertragung abgehört haben, alle Daten, um sich ab diesem Moment als B auszugeben. Bei einem Challenge-Response-Protokoll wird für jede Anwendung des Protokolls eine neue Aufgabe verwendet. Bei jeder Anwendung können V und Mithörende neue Informationen über das Geheimnis sammeln und es schließlich knacken. Wenn es sich bei der Aufgabe beispielsweise um die Verschlüsselung eines Klartextes handelt und die Angreifer diesen Klartext wählen können, dann handelt es sich um einen Angriff mit gewähltem Klartext.

Öffentlicher Schlüssel

Bei der Abwägung der Vorteile und Nachteile zwischen einem Zero-Knowledge-Beweis und einer digitalen Signatur mit einem privaten Schlüssel (verifiziert durch den entsprechenden öffentlichen Schlüssel) stellen wir fest, dass wie bei jedem Challenge-Response-Protokoll jede Signatur, d. h. die Verschlüsselung eines Dokuments mit dem privaten Schlüssel der Benutzer, Informationen preisgibt. Im Extremfall, wenn Angreifer diesen Klartext wählen können, handelt es sich um einen Angriff mit gewähltem Klartext. Allerdings erfordern Zero-Knowledge-Protokolle

weniger Rechenaufwand als Public-Key-Protokolle. Da digitale Signaturen praktisch sicher sind und viele Geräte (z. B. Chipkarten) nur über eine geringe Rechenleistung verfügen, werden digitale Signaturen häufiger verwendet.

Die Sicherheit der meisten Zero-Knowledge-Protokolle und Public-Key-Protokolle hängt von der unbewiesenen Annahme ab, dass die Kryptoanalyse rechnerisch so schwierig ist wie ein mathematisches Problem (wie das Quadratische-Reste-Problem, die Zerlegung einer ganzen Zahl in ihre Primfaktoren und der diskrete Logarithmus).

Ali Babas Höhle

Ali Babas Höhle veranschaulicht die Prinzipien hinter einem Zero-Knowledge-Beweis. In einer kreisförmigen Höhle gibt es eine vom Eingang aus unsichtbare Tür, die den Durchgang versperrt, wenn sie nicht mit einem Passwort (z. B. „Sesam“) geöffnet wird. Damit B gegenüber V beweisen können, dass sie das Passwort kennen, ohne es preiszugeben, betreten C die Höhle unbeobachtet von V, bis sie links oder rechts vor der Tür stehen. V verlangen nun von B, dass sie von links oder rechts kommend zum Eingang zurückkehren. Da die Wahrscheinlichkeit, dass B die Höhle auf der gleichen Seite betreten haben, von der sie – durch V aufgefordert – auch wieder zum Eingang zurückkehren sollen, 1/2 beträgt und alle Beweise unabhängig sind, beträgt die Wahrscheinlichkeit, dass C das Passwort nicht kennen, nach 10 Beweisen 2–10, also weniger als ein Tausendstel. Während V überzeugt sind, dass B das Passwort kennen, können sie niemanden sonst überzeugen. Selbst wenn V die Sequenz aufgezeichnet haben, könnte sie im Voraus gegenseitig abgesprochen worden sein.

Schnorr-Identifikation

Schnorr stellte ein Zero-Knowledge-Protokoll vor, das so einfach ist, dass es auf Chipkarten ausgeführt werden kann (Schnorr, 1991). Hierbei wird die Kenntnis von diskreten Logarithmen bewiesen, d. h. die Beweisenden B kennen eine ganze Zahl x, und die Verifizierenden V kennen gx mod p, wobei p eine Primzahl ist. Damit B beweisen können, dass sie x kennen, ohne es preiszugeben:

1. wählen B eine beliebige ganze Zahl a und sendet ga an V.
2. V werfen eine Münze und senden das Ergebnis c in {0, 1} an B.
3. B senden a an V, immer wenn c = 0 ist, oder a + x an V, immer wenn c = 1 ist.

Dies ist ein Zero-Knowledge-Protokoll, denn

* wenn c = 0 ist, dann wird nichts über x preisgegeben (sondern nur über a),
* wenn c = 1 ist, dann erfahren die Verifizierenden a + x mod p, aber wiederum, solange nichts über a offenbart wird (wobei wir mit der Schwierigkeit der Berechnung von log mod p rechnen), wird auch nichts über x offenbart.

Wenn c = 0, dann wird keine Kenntnis von x benötigt. Wenn jedoch c = 1 ist, können V durch ga+x = ga gx – wobei beide Werte auf der rechten Seite bekannt sind – verifizieren, ob B a + x kennen. Da die Wahrscheinlichkeit, dass c = 1 ist 1/2 beträgt und alle Beweise unabhängig sind, beträgt nach, sagen wir, 10 Beweisen die Wahrscheinlichkeit, dass V x nicht kennen, 2–10 < 1/1000.

### Biometrische Authentifizierung

Die biometrische Authentifizierung identifiziert Benutzer eines Computers entweder durch physische oder verhaltensbezogene menschliche Merkmale. Zu den physischen Merkmalen gehören Fingerabdrücke (die schon lange in diesem Kontext zum Einsatz kommen), Gesichtsmerkmale wie die relative Position der Augen, der Nase, der Lippen, des Kiefers und Augenmerkmale wie das Muster der Blutgefäße in der Iris. Zu den Verhaltensmerkmalen gehören Aspekte wie die Art und Weise des Tippens, z. B. die Geschwindigkeit der Tastenanschläge oder das Auftreten von Tippfehlern (besonders nützlich zur Ergänzung eines Login-Dialogs); die Art der Handschrift, entweder statisch, wenn ein Bild verwendet wird, oder dynamisch, wenn die Zeichen auf einem Tablet anhand von (zeitlichen) Funktionen wie x- und y-Koordinaten, Druck und Neigung ausgewertet werden; sowie Stimmeigenschaften (die Sprechererkennung ist besonders nützlich, um die Identität von Kunden am Telefon zu überprüfen). Jedes gesprochene Wort wird in seine Formanten, die dominanten Frequenzen, zerlegt. Danach werden physiologische und verhaltensbezogene Merkmale identifiziert, welche die physiologischen Merkmale (die die Form des Stimmtrakts eines Individuums beschreiben, d. h. der Nase, des Mundes, des Kiefers, der Zunge) oder die verhaltensbezogenen Merkmale (die die Bewegung der Nase, des Mundes, des Kiefers, der Zunge beschreiben, die den Akzent, den Ton, die Tonhöhe oder das Tempo verändern) beschreiben.

Die Authentifizierung durch das, was wir sind (z. B. ein Fingerabdruck) und was wir wissen (z. B. ein Passwort), hat den Vorteil, dass, selbst wenn diese bekannt ist, ein gewisser Aufwand betrieben werden muss, um sie nachzumachen, und dass sie nicht vergessen werden kann. Gegenüber der Authentifizierung mit dem, was wir haben (z. B. ein Token), hat dies den Vorteil, dass wir nichts mit uns herumtragen müssen und nichts verloren gehen oder gestohlen werden kann.

Diese Art der Authentifizierung erfordert ausgeklügelte Hardware, die offen liegt und somit imitierbar ist. Der Chaos Computer Club (CCC) und seine Mitstreiter haben zum Beispiel wiederholt gezeigt, wie einfach es ist, Fingerabdrücke zu fälschen, zum Beispiel mit Holzleim und sprühbarem Graphen. Die vielen erfolgreichen Fälschungsangriffe auf biometrische Identifizierungen, wie z. B. Fingerabdrücke und Foto-IDs, bringen uns zu dem Schluss, sie als Ergänzung für ein Passwort, eine Chipkarte, einen zweiten Faktor oder einfach als Ersatz für einen Benutzer zu behandeln, nicht als Ersatz.

### Authentifizierung in einem verteilten System

Wenn in einem verteilten System die Identität des Systems, gegenüber dem die Authentifizierung erfolgt, garantiert ist und niemand die Kommunikation abhören kann, dann könnte das Geheimnis selbst für die Authentifizierung sicher übermittelt werden. Um sicherzustellen, dass niemand außer beabsichtigten Empfängern das Geheimnis für die Authentifizierung sieht, darf das Geheimnis selbst niemals gesendet werden. Stattdessen überzeugen sich Benutzer und das System gegenseitig davon, dass sie das gemeinsame Geheimnis (normalerweise ein Passwort) kennen. Die Identifizierungsdaten selbst werden also nie gesendet, sondern nur der Nachweis, dass Benutzer Zugang zu diesen haben. In der Praxis ist bei passwortbasierten Systemen der beliebteste Ansatz das Challenge-Response-System.

Auch wenn das Geheimnis selbst nie gesendet wird, ist es dennoch empfehlenswert, die gesamte Kommunikation zu verschlüsseln, um die Authentifizierung zu gewährleisten, z. B. durch Verschlüsselung mit einem öffentlichen Schlüssel. Andere Ansätze umfassen die Preisgabe eines Teilgeheimnisses, z. B. werden in einem Einwegsystem die Identifikationsdaten nur einmal verwendet (TANs beim Online-Banking).

Wenn die Identifikationsdaten jedoch abgehört werden können und ihre Verwendung zur Authentifizierung und damit zur Ungültigmachung verhindert werden kann (z. B. durch Einloggen in eine gefälschte Kopie der Website der Bank), dann können sie später verwendet werden. Ein anderer Ansatz besteht darin, zusätzliche geheime Informationen über einen zweiten Kanal zu versenden, z. B. durch den Versand einer SMS im mobilen TAN-System (mTAN).

###### FIDO 2

Die FIDO (Fast IDentity Online) Alliance wurde im Februar 2013 gegründet, um in Zusammenarbeit mit vielen Unternehmen wie Google oder Microsoft offene und lizenzfreie Industriestandards für die Authentifizierung im Internet zu entwickeln. Am 9. Dezember 2014 wurde der erste FIDO-Standard veröffentlicht, der den Standard U2F (Universal Second Factor) für Hardware und Software zur Zwei-Faktor-Authentifizierung und den Standard UAF (Universal Authentication Framework) für das zugehörige Netzwerkprotokoll zur passwortlosen Authentifizierung spezifiziert. Diese Standards zielen darauf ab, die Authentifizierung im Internet zu erleichtern, indem sie das Eigentum von Benutzern (was sie haben) – wie z. B. Sicherheitstokens –, oder Eigenschaften (was sie sind) – wie z. B. einen Fingerabdruck –, anstelle von Wissen (was sie wissen) – wie z. B. Passwörter oder persönliche Identifizierungsnummern – akzeptieren. So müssen sich Benutzer nicht mehr zahlreiche sichere Passwörter merken, was allerdings auch einige Nachteile mit sich bringt. Im Vergleich zu früheren Methoden der Zwei-Faktor-Authentifizierung, wie z. B. SMS-Bestätigungscodes, setzt FIDO2 voraus, dass sich der Schlüssel, z. B. das Smartphone, physisch in der Nähe des Computers befindet.

FIDO2 (2020) setzt sich aus folgenden Teilen zusammen:

1. W3C Web Authentication Standard (WebAuthn), der den Zugriff auf das Internet mit biometrischen Informationen, mobilen Endgeräten oder FIDO-Sicherheitsschlüsseln ermöglicht. WebAuthn wird von verschiedenen Betriebssystemen, wie Windows 10 und Android, aber auch von Browsern, wie Google Chrome, Mozilla Firefox, Microsoft Edge oder Apple Safari unterstützt.
2. Client to Authenticator Protocol (CTAP) der FIDO Alliance, das auf U2F basiert. Mit der Veröffentlichung von FIDO2 wurde U2F in CTAP1 umbenannt. CTAP wird unter anderem für die Authentifizierung in Desktop-Anwendungen und Webdiensten verwendet.

Persönliche Daten und private Schlüssel befinden sich immer und ausschließlich in den Händen der Benutzer und werden nicht auf öffentlichen Servern gespeichert. Um ein Konto über FIDO2 zu registrieren, senden die Server eine Anfrage, und der FIDO2-Schlüssel generiert einen öffentlichen und einen geheimen Schlüssel aus einem geheimen Anfangsschlüssel und der Serveradresse. Dieser öffentliche Schlüssel wird an den Server übertragen, der ihn speichert und den FIDO2-Schlüssel in Zukunft eindeutig identifizieren kann.

Auf diese Weise kann sich der FIDO2-Schlüssel bei jedem Server mit einem individuellen Schlüssel identifizieren, ohne dass der Server Informationen über die Schlüsselpaare für denselben FIDO2-Schlüssel auf anderen Servern erhält. Der FIDO2-Schlüssel erzeugt für jeden Dienst ein eigenes Schlüsselpaar. Anhand der Domain der anderen Partei, z. B. Ebay oder Google, können sie nicht feststellen, welche ihrer Benutzer denselben FIDO2-Schlüssel haben. In der Praxis ist dies ein Vorteil (auf der Serverseite!) der Authentifizierung durch ein Passwort, wenn Benutzer häufig ähnliche Passwörter auf verschiedenen Servern verwenden.

Schlüssel

Ein FIDO2-Schlüssel (oder Authentifikator oder Token) ist ein Vehikel, mit dem wir uns bei einem Dienst authentifizieren können. Dabei kann es sich einerseits um ein externes Gerät handeln, das über USB, NFC oder Bluetooth mit dem PC oder Smartphone der Benutzers verbunden wird, wie z. B. ein Sicherheitstoken zum Einstecken in einen USB-Anschluss, eine Chipkarte zum Einstecken in einen Kartenleser, ein NFC-Token, Smartphones mit Bluetooth-Schnittstelle IEEE 802.15.1 und Bluetooth-Token (Bluetooth V4.0 Low Energy 2,45 GHz). Andererseits kann es aber auch ein interner Authentifikator sein, d. h. eine Software, die den Kryptochip des PCs, Smartphones oder Tablets für FIDO2 nutzt, unterstützt von Windows 10, Android 7 und höher. Zum Schutz vor Missbrauch des FIDO2-Schlüssels kann dieser zusätzlich biometrisch oder mit einem Passwort/einer PIN gesichert werden. Wenn der Stick verloren geht, steht entweder ein registrierter Backup-Schlüssel zur Verfügung oder wir müssen uns über die Handynummer in Kombination mit einer E-Mail-Adresse o. ä. erneut identifizieren.

Adaption

Ein FIDO2-Schlüssel kann entweder anstelle eines Passworts oder zusätzlich zu diesem als zweiter Faktor verwendet werden. Je nachdem, wie ein Dienst FIDO2 implementiert hat, reicht der Schlüssel für die Anmeldung (Ein-Faktor-Authentifizierung) oder die Eingabe eines zusätzlichen Passworts (Zwei-Faktor-Authentifizierung) aus. Die Ein-Faktor-Authentifizierung mit FIDO2 ist bereits für Microsoft.com, Outlook.com, Office 365 und OneDrive im Edge-Browser verfügbar. Die Zwei-Faktor-Authentifizierung mit FIDO2 funktioniert zum Beispiel mit Google, GitHub, Dropbox und Twitter.

###### Kerberos

Kerberos ist nach Cerberus benannt, der dreiköpfigen Bestie, die das diamantharte Eingangstor zum Hades bewacht. Cerberus überprüft die Identität von toten Seelen, und dementsprechend überprüft Kerberos die Identität von Benutzern über ein Netzwerk (Neuman & Ts'o, 1994). Für das Problem der Schlüsselverteilung, d. h. der geheimen Weitergabe desselben Schlüssels an alle Kommunikationspartner, und für die vielen Schlüssel, die eine Gruppe von Kommunikationspartnern benötigt, um sicher miteinander zu kommunizieren, gibt es eine Lösung. Alle Kommunikationspartner müssen einer zentralen Stelle bedingungslos vertrauen. Außerdem müssen alle Benutzer in der Lage sein, für jede Kommunikation einen sicheren Sitzungsschlüssel zu erhalten, so dass alle Kommunikationspartner nur einen Schlüssel schützen muss, während die Verantwortung für den Schutz aller Schlüssel unter den Kommunikationspartnern auf die zentrale Stelle verlagert wird.

Komponenten zur Authentifizierung und Ticketausgabe

Kerberos ist ein Netzwerkprotokoll (wie in RFC 1510-The Kerberos Network Authentication Service V5 spezifiziert), das es Benutzern ermöglicht, sich gegenseitig über ein unsicheres Netzwerk durch eine vertrauenswürdige dritte Partei, das Key Distribution Center (KDC), zu authentifizieren. Sobald ein Benutzer authentifiziert ist (Kerberos), wird er durch Zugriffsprotokolle wie LDAP (Lightweight Directory Access Protocol) autorisiert (Neuman & Ts'o, 1994).

Das KDC speichert die symmetrischen Schlüssel aller registrierten Benutzer (Client oder Server), um sie als externer Vermittler zu authentifizieren. Aufgrund seiner entscheidenden Funktion ist es eine bewährte Praxis, ein zweites KDC als Ausweichlösung zu haben.

Kerberos gruppiert Benutzer in Clients und (Dienst-)Server, welche Dienste für die Clients hosten. Das Protokoll authentifiziert einen Client nur einmal, sodass alle Dienstserver ihm für den Rest der Sitzung vertrauen. Dies wird durch zwei Faktoren erreicht:

1. ein Ticket, ein einmalig verwendbarer Berechtigungsnachweis, der vom KDC ausgegeben wird, um einen Client gegenüber einem Server zu authentifizieren, von dem er einen Dienst anfordert, und der mit dem Schlüssel des Servers verschlüsselt ist. Es enthält die ID des Servers und des Clients, die Netzwerkadresse des Clients, einen Zeitstempel, eine Lebensdauer und einen Sitzungsschlüssel, der mit dem Schlüssel des Clients verschlüsselt ist.
2. einen Authentifikator, einen Berechtigungsnachweis, der mit dem zwischen dem Client und dem Server geteilten Sitzungsschlüssel verschlüsselt ist und der das Ticket begleitet, um den Client zu authentifizieren. Sie enthält die ID des Clients, die Netzwerkadresse des Clients und einen Zeitstempel.

Das KDC ist in einen Authentifizierungsserver (AS) und einen Ticket-Granting-Server (TGS) aufgeteilt. Der AS authentifiziert alle Benutzer im Netzwerk. Er speichert einen symmetrischen Schlüssel für alle Benutzer, sei es ein Client oder ein Dienstserver, der nur ihm selbst, dem AS und dem Benutzer bekannt ist. Nachdem der AS einen Client mit dem Schlüssel des Client authentifiziert hat, sendet er ihm ein Ticket-Granting-Ticket (TGT) und einen mit dem Schlüssel des Clients verschlüsselten Sitzungsschlüssel. Der TGS erzeugt nach der Authentifizierung einen Sitzungsschlüssel als Teil eines Tickets zwischen zwei Benutzern des Netzwerks. Nachdem ein Client sein TGT gesendet hat, vom TGS authentifiziert wurde und einen Dienst eines Dienstservers anfordert, sendet der TGS ein Ticket und zwei Kopien eines Sitzungsschlüssels. Der eine wird mit dem TGT-Sitzungsschlüssel des Benutzers verschlüsselt, der andere mit dem Dienstserver-Schlüssel, um den Client gegenüber des Dienstservers zu authentifizieren und ihre Kommunikation zu sichern.

Authentifizierungsprotokoll

Damit Benutzer (oder Clients) sicher mit anderen Benutzern (Dienstserver oder Anwendungsserver) über das KDC kommunizieren können, definiert das Kerberos-Protokoll zehn Nachrichten, darunter (Neuman & Ts'o, 1994)

|  |  |
| --- | --- |
| Code | Bedeutung |
| KRB\_AS\_REQ | Kerberos Authentication Service Request |
| KRB\_AS\_REP | Kerberos Authentication Service Reply |
| KRB\_TGS\_REQ | Kerberos Ticket-Granting Service Request |
| KRB\_TGS\_REP | Kerberos Ticket-Granting Service Reply |
| KRB\_AP\_REQ | Kerberos Application Request |
| KRB\_AP\_REP | Kerberos Application Reply |

Die Authentifizierung und anschließende Ticketvergabe zwischen einem Benutzer und einem Server wird durch einen Kerberos-Server vermittelt (Pixis, 2019):

1. Zum Authentifizieren des Dienstservers führt der Client folgende Schritte durch:
   1. Er authentifiziert den Authentifizierungsserver (AS) (KRB\_AS\_REQ) unter Verwendung eines dauerhaften gemeinsamen Geheimnisses (Schlüssel des Clients), und
   2. er erhält vom Authentifizierungsserver ein kurzzeitiges gemeinsames Geheimnis (Sitzungsschlüssel) und ein (Ticket-Granting) Ticket (KRB\_AS\_REP).
2. Zum Authentifizieren des Dienstservers über den Authentifizierungsserver führt der Client folgende Schritte durch:
   1. Er authentifiziert den Dienstserver mithilfe des TGT, und
   2. er fordert (KRB\_TGS\_REQ) ein Ticket vom TGS (KRB\_TGS\_REP) an, das einen Sitzungsschlüssel zwischen dem Client und dem Dienstserver enthält.
3. Der TGS führt folgende Schritte zum Erstellen des Tickets durch:
   1. Er erzeugt einen Client-to-Server-Sitzungsschlüssel;
   2. er verschlüsselt den Sitzungsschlüssel mithilfe des Client-Schlüssels;
   3. er verschlüsselt mithilfe des Dienstserver-Schlüssels das Ticket zwischen Client und Server, das die ID des Clients, die Netzwerkadresse des Clients, einen Zeitstempel, eine Lebensdauer und den Sitzungsschlüssel enthält; und
   4. er sendet die Ergebnisse der beiden Verschlüsselungen an den Client.
4. Der Client führt folgende Schritte durch, um sich direkt beim Dienstserver zu authentifizieren:
   1. Er entschlüsselt den Client-to-Server-Sitzungsschlüssel mit seinem eigenen Schlüssel, und
   2. er sendet folgendes an den Dienstserver (KRB\_AP\_REQ und KRB\_AP\_REP)
      * das Client-to-Server-Ticket, verschlüsselt mit dem Schlüssel des Dienstservers, und
      * einen Authentifikator, der die ID des Clients und einen Zeitstempel N enthält, verschlüsselt mit dem Client-to-Server-Sitzungsschlüssel.
5. Der Dienstserver
   1. ruft den Client-Server-Sitzungsschlüssel ab, indem er das Client-Server-Ticket mit seinem eigenen Schlüssel entschlüsselt,
   2. entschlüsselt mithilfe des Sitzungsschlüssels den Authentifikator und überprüft ihn. Wenn die Überprüfung erfolgreich ist, kann der Server dem Client vertrauen.
   3. sendet den um 1 erhöhten Zeitstempel verschlüsselt über den Sitzungsschlüssel an den Client.

Mit dem Sitzungsschlüssel kann der Client den Zeitstempel entschlüsseln, um ihn zu überprüfen. Wenn dies gelingt, ist das Vertrauen hergestellt und die Dienstanfragen an den Server können beginnen. Der Server stellt dem Client natürlich die gewünschten Dienste zur Verfügung (Pixis, 2019).

### Chipkarten

Eine Chipkarte ist wie eine Kreditkarte geformt, enthält aber einen Mikroprozessor, der Informationen sicher speichert und verarbeitet. Im Gegensatz dazu speichert eine Magnetstreifenkarte nur wenige Informationen (etwa <100 Byte) und kann diese nicht verarbeiten. Die Verarbeitung von Informationen auf einer Chipkarte ist durch Sicherheitsalgorithmen gesichert; für den Zugriff auf die Daten ist eine erfolgreiche Autorisierung erforderlich (Borst et al., 2001).

###### Komponenten

Der Großteil der Verarbeitung in Chipkarten ist kryptografischen Verfahren gewidmet, insbesondere der Verschlüsselung zwischen On-Chip-Komponenten. Für die Verwendung benötigt eine Chipkarte eine externe Stromversorgung und ein Avvclock-Signal, das durch den Kontakt mit einem Chipkartenleser bereitgestellt wird (den die Authentifizierung normalerweise auch beschreiben kann). Das Betriebssystem der meisten Chipkarten implementiert einen Standardsatz von Steuerbefehlen, wie sie in ISO 7816 oder CEN 726 standardisiert sind. Die Komponenten einer Chipkarte sind:

1. Random-Access Memory (RAM), der Daten liest und schreibt, aber Informationen nur vorübergehend speichert, solange Strom vorhanden ist. Normalerweise hat eine Chipkarte 1 kByte RAM.
2. Read-Only Memory (ROM) speichert Informationen dauerhaft. Das Betriebssystem und die Verschlüsselungsalgorithmen werden gespeichert. Normalerweise hat eine Chipkarte 32 kByte ROM. Zur Verbesserung der Sicherheit ist der ROM in tieferen Schichten des Halbleiters verborgen.
3. EEPROM (Electrically Erasable Programmable Read Only Memory) speichert Informationen dauerhaft, ist aber langsam und kann nur für eine begrenzte Anzahl von Schreib-/Lesevorgängen verwendet werden (etwa 100 000 Mal). Normalerweise hat eine Chipkarte 8-128 kByte EEPROM. Zur Verbesserung der Sicherheit ist der EEPROM mit einer Metallschicht ummantelt.
4. Der Prozessor war früher ein 8-Bit-Mikrocontroller, aber es werden zunehmend leistungsfähigere 16- und 32-Bit-Chips verwendet. Um die Geschwindigkeit der Verschlüsselungsberechnungen zu erhöhen, wird häufig ein Ko-prozessor eingesetzt.

Es gibt einen einzigen Eingangs-/Ausgangsanschluss, der durch kleine Datenpakete namens APDUs (Application Protocol Data Units) gesteuert wird. Denn Daten fließen nur mit etwa 9600 Bits pro Sekunde und im Halbduplex-Verfahren, d. h. die Daten können entweder vom Leser zur Karte oder von der Karte zum Leser fließen, aber nicht bidirektional. Eine Chipkarte kann langsam ausgelesen werden, während sich sowohl das Lesegerät als auch die Chipkarte gegenseitig durch ein Challenge-Response-Protokoll mit einer symmetrischen Schlüsselverschlüsselung authentifizieren. Zur Verschlüsselung generiert die Karte eine Zufallszahl, gibt sie an das Lesegerät weiter, das die Zahl wiederum mit einem gemeinsamen Schlüssel verschlüsselt und an die Chipkarte zurückgibt. Schließlich vergleicht die Karte das Verschlüsselungsergebnis mit ihrer eigenen Verschlüsselung. Nach der gegenseitigen Authentifizierung wird jede ausgetauschte Nachricht durch einen Nachrichtenauthentifizierungscode (HMAC) verifiziert, der aus der Nachricht, dem Verschlüsselungscode und einer Zufallszahl berechnet wird (Borst et al., 2001).

###### SIM-Karten

Die Universal Integrated Circuit Card (UICC) oder das Universal Subscriber-Identity Module (USIM) ist für mobile Endgeräte in GSM- und UMTS-Netzen sehr beliebt. Es handelt sich um eine Chipkarte mit einer Größe von einigen hundert kB, die die Sicherheit und Integrität der persönlichen Daten gewährleistet. UICCs Subscriber Identification Module (SIM) ist eine Anwendung, welche die Authentifizierungsinformationen speichert. Die UICC speichert nicht nur den Identifizierungsschlüssel der Mobiltelefonteilnehmer und die Teilnahmeinformationen, sondern auch dessen Einstellungen, Textnachrichten, Kontakte und zuletzt gewählte Nummern. Im Vergleich zu Kreditkarten ist ein größerer EEPROM verfügbar.

###### Vorteile

Kryptografische Schlüssel müssen nicht gespeichert werden und können beliebig komplex sein, aber kryptografische Schlüssel können nicht aus der Distanz abgerufen werden. Das Verfahren ist weniger anspruchsvoll, verwendet weniger teure Hardware und die kryptografischen Schlüssel werden sicher auf einem Gerät gespeichert. So werden Fälschungen mit Fingerabdrücken oder Venenscannern unterbunden. Auf einer Chipkarte gespeicherte Schlüssel sind viel schwieriger zu lesen und hinterlassen weniger Spuren. Sie geben zum Beispiel nie den Schlüssel preis, sondern liefern nur die Informationen, nach denen gefragt wurde, und sind immun gegen Keylogger, die Tastenanschläge aufzeichnen.

### Identität und Anonymität

Anonymität stammt vom griechischen Wort für „namenlos“ (OPTED o. J.). Umgangssprachlich bedeutet es, dass die Identität einer Person unbekannt ist. Hier bedeutet es, dass ein Element (z. B. eine Person oder ein Computer) innerhalb einer Menge (z. B. einer Gruppe oder eines Netzwerks) nicht identifizierbar ist (innerhalb dieser Menge). Der Schutz der eigenen Identität vor der Preisgabe ist nicht nur für jemanden notwendig, der gegen das Gesetz verstößt, z. B. bei dem Versuch, einen Computer in einem Netzwerk zu missbrauchen, sondern auch als Vorsichtsmaßnahme gegen einen möglichen Missbrauch. Entsprechende Gegenmaßnahmen sind verfügbar:

* + ein Proxy-Server zwischen Benutzern und dem Internet, der neben anderen Aufgaben wie dem Zwischenspeichern häufig genutzter Daten und der Beschränkung des Zugriffs auf bestimmte Benutzer auch IP-Adressen (die einen Computer in einem Netzwerk identifizieren) verbergen kann,
  + das Tor-Projekt, das den gesamten Datenverkehr der Benutzer verschlüsselt und durch viele Zwischenstationen leitet, die sich gegenseitig nicht kennen, und
  + Virtuelle private Netzwerke, wie OpenVPN oder IPsec, die den gesamten Datenverkehr der Benutzer verschlüsseln und an einen zentralen Server weiterleiten.

Diese Gegenmaßnahmen sind jedoch mit Unannehmlichkeiten verbunden, wie z. B. einer aufwändigen Einrichtung und einer langsameren Datenübertragung. Eine weniger beeinträchtigende praktische Maßnahme ist die Anwendung von Best Practices zum Schutz der eigenen Privatsphäre im World Wide Web. So können wir beispielsweise Browser-Add-ons verwenden, die Tracker herausfiltern, z. B. aus Cookies, Referrern (die URL der vorherigen Webseite, von der aus ein Link verfolgt wurde) und Anfragen an zentrale Content Delivery Networks (CDNs), wie uBlock Origin, Privacy Badger, Don't track me Google und Decentraleyes.

###### Identitätsdiebstahl

Mit dem Begriff Identität, vom lateinischen *identitas*, (*idem*, gleich und *entitas*, Wesen), bezeichnen wir die Merkmale, durch die jemand oder etwas eindeutig erkennbar ist, d. h. die keine andere Person oder Sache besitzt (OPTED, o.J.). Identitätsdiebstahl ist die Aneignung der Identität einer anderen Person, in der Regel des Namens einer anderen Person oder anderer persönlicher Informationen wie PINs, Sozialversicherungsnummern, Führerscheine oder Bankdaten, um einen Betrug zu begehen, z. B. um ein Bank- oder Kreditkartenkonto für Betrugszwecke zu eröffnen. Mit dem Aufkommen digitalisierter Datensätze und der Anonymität im Internet ist Identitätsdiebstahl zunehmend zum Problem geworden.

Beim SIM-Karten-Tausch zum Beispiel verschaffen sich Angreifer die Handynummer eines Opfers, um vorübergehend dessen Online-Identität anzunehmen. Die Angreifer verschaffen sich zunächst persönliche Daten des Opfers, in der Regel den Namen, die Handynummer und die Postanschrift. Sie nutzen dann die Tatsache aus, dass Mobilfunkbetreiber ihren Kunden in der Regel eine neue SIM-Karte anbieten, z. B. nachdem das Telefon verloren gegangen ist, und übernehmen die bisherige Telefonnummer.

Die Angreifer geben sich nun als echte Kunden aus und telefonieren mit dem Kundendienstzentrum. Tatsächlich reicht es aus, die Handynummer zu kennen, um das Passwort eines Instagram-Kontos zurückzusetzen.

Zusammenfassung

Bei der Identifikation wird einem Computer mitgeteilt, wer die Benutzer sind, in der Regel durch den Namen individueller Benutzer (oder deren Konten). Darauf folgt die Authentifizierung, die Verifizierung der Identität der Benutzers, d. h. die Überzeugung eines Computers, dass eine Person diejenige ist, die sie vorgibt zu sein. Um sich zu authentifizieren, weisen Benutzer ihre Identität durch Informationen nach, die nur sie kennen (z. B. ein Passwort), die nur sie haben (Hardware wie eine Chipkarte) und die nur sie beschreiben (z. B. biometrische Identifikationsmerkmale).

Jede Methode hat ihre jeweiligen Vor- und Nachteile. Insbesondere Passwörter, die am häufigsten verwendete Methode, müssen leicht zu merken sein, was sie wiederum in der Praxis schwächt. Der FIDO2-Standard zielt darauf ab, sie durch Hardware und biometrische Authentifizierung zu ersetzen (oder zumindest zu ergänzen).

Anstatt das Geheimnis selbst bei der Authentifizierung preiszugeben, ist es sicherer, nur dessen Kenntnis zu beweisen. In einem Challenge-Response-Authentifizierungsprotokoll erfordert die erfolgreiche Beantwortung der Aufgabe die Kenntnis der Aufgabe, wie z. B. die Ver- und Entschlüsselung von Zufallsdaten mit dem geheimen Schlüssel. Bei einem Zero-Knowledge-Protokoll können im Gegensatz zu einem Challenge-Response-Protokoll keinerlei Informationen über den geheimen Schlüssel gewonnen werden, sofern die Berechnung einer mathematischen Funktion als nicht möglich angesehen wird.

Das Kerberos-Protokoll vermittelt zwischen Benutzern und Servern über einen zentralen Server, der die symmetrischen Schlüssel aller Parteien speichert. Anstatt dass die Parteien gegenseitig die Kenntnis ihrer symmetrischen Schlüssel beweisen, wird für alle Kommunikationspartner ein Sitzungsschlüssel von begrenzter Gültigkeit erstellt.



# Lektion 5

## Kryptoanalyse – Verschlüsselungen brechen

#### LERNZIELE

Nach der Bearbeitung dieser Lektion wissen Sie …

… gegen welche Szenarien wir uns hinsichtlich der Kryptoanalyse schützen müssen.

… wie Sie die Dauer eines Brute-Force-Angriffs abschätzen und ihn beschleunigen können.

… welche Rolle differentielle Kryptoanalyse und Frequenzanalyse beim Brechen von Chiffren spielen.

1. Kryptoanalyse – Verschlüsselungen brechen

### Einführung

Kryptoanalyse  
Das Wiederherstellen oder Erzeugen von verschlüsselten Informationen ohne Kenntnis des Schlüssels wird als Kryptoanalyse bezeichnet.

Kryptoanalyse ist die Kunst, verborgene Schriften zu entschlüsseln, d. h. Chiffren zu brechen. Das Präfix „Krypto“ kommt vom griechischen Wort für „verborgen“, „Analyse“ von dem Wort für „auflösen“ (OPTED, o. J.). Die Geschichte berichtet von zahlreichen Erfolgen der Kryptoanalyse, unter anderem der Entschlüsselung der deutschen Rotor-Chiffriermaschine Enigma durch polnische und britische Streitkräfte im Zweiten Weltkrieg. Wir werden einige bewährte Prinzipien der Kryptoanalyse vorstellen. In der Praxis waren die Intuition der Kryptoanalytiker und deren Fähigkeit, subtile Muster im Geheimtext zu erkennen, von größter Bedeutung, jedoch sind diese nur schwer zu vermitteln. Heute jedoch basiert die Kryptoanalyse auf der Mathematik und wird durch den effizienten Einsatz enormer Rechenleistung angewandt (Dooley, 2013, Kapitel 5).

### Frequenzanalyse

Die Kryptoanalyse von Ein-Schlüssel-Kryptosystemen beruht auf Mustern im Klartext, die sich auf den Geheimtext übertragen. Bei einer monoalphabetischen Substitutions-Chiffre zum Beispiel sind die Häufigkeiten des Auftretens der Buchstaben im Klartext-alphabet dieselben wie im Geheimtextalphabet. Dies lässt sich kryptoanalytisch nutzen, wenn wir erkennen, dass die Chiffre eine monoalphabetische Substitutions-Chiffre ist. So lassen sich die wahrscheinlichsten Kandidaten für die Klartextbuchstaben ermitteln.

###### Lateinisches Alphabet

Eine Substitution durch eine beliebige Permutation der Buchstaben des Alphabets, wie z. B.

A B … Y Z

…

E Z … G A

besitzt

26 · 25!1 = 26! > 1026

Schlüssel, weshalb ein Brute-Force-Angriff rechnerisch undurchführbar ist. Allerdings verstößt diese Methode gegen die Ziele der Diffusion und der Konfusion. Wenn der Schlüssel (also die gegebene Permutation des Alphabets) den Buchstaben α gegen den Buchstaben β austauscht, dann ergibt sich eine mangelhafte Konfusion. Das liegt daran, dass die Substitution von β im Schlüssel nur die Substitution jedes Buchstabens β im Geheimtext impliziert. Ebenso impliziert die Substitution eines Buchstabens α im Klartext nur die Substitution des entsprechenden Buchstabens β im Geheimtext. Der Algorithmus ermöglicht statistische Angriffe auf die Häufigkeit von Buchstaben, Bigrammen (Buchstabenpaaren) und Trigrammen (Dreiergruppen von Buchstaben). Im Englischen zum Beispiel ist der häufigste Buchstabe „e“, das häufigste Bigramm ist „th“ und das häufigste Trigramm ist „the“.

Somit bieten die folgenden Methoden einen guten Ansatz zur Entschlüsselung eines englischen Textes:

* Ersetzen des häufigsten Buchstabens des Geheimtextes mit dem häufigsten Buchstaben der englischen Sprache (e),
* Ersetzen des häufigsten Bigramms des Geheimtextes mit dem häufigsten Bigramm der englischen Sprache (th),
* Ersetzen des häufigsten Trigramms des Geheimtextes mit dem häufigsten Trigramm der englischen Sprache (the).

Je mehr Geheimtext uns vorliegt, desto wahrscheinlicher ist es, dass diese Substitution mit dem Geheimtext übereinstimmt.

Wenn wir diese Häufigkeiten zum Beispiel auf den Geheimtext ACB ACBGA ACSBDQT anwenden, erhalten wir THE THE\*\* TH\*E\*\*\* für die noch zu entschlüsselnden Buchstaben, die mit \* gekennzeichnet sind. Durch die Beschränkungen des englischen Vokabulars und der Satzstruktur ergibt sich THE THEFT THREAPS.

###### Homophone

Ein Ansatz zum Verbergen der Buchstabenhäufigkeiten ist die Verwendung mehrerer Homophone im Verhältnis zur Häufigkeit des Buchstabens im Klartext, z. B. doppelt so viele Symbole für E wie für S, so dass jedes verschlüsselte Zeichen im Durchschnitt gleich oft im Geheimtext vorkommt. Der deutsche Mathematiker und Astronom Carl Friedrich Gauß nahm an, dass seine Einführung von Homophonen eine solche Chiffre unknackbar machte. Andere Häufigkeiten im Klartext überstehen die Verschlüsselung jedoch (teilweise), wie z. B. Digraphen. „TH“ kommt am häufigsten vor – etwa 20 Mal so häufig wie „HT“ – und so weiter, was die Kryptoanalyse bei einem ausreichend langen Geheimtext immer noch erheblich erleichtert (Jagetiya & Krishna, 2020).

### Brute-Force-Angriffe

In der Praxis hängt die Sicherheit eines Kryptosystems in erster Linie von seiner Widerstandsfähigkeit gegenüber den effizientesten (bekannten) Methoden der Kryptoanalyse und dem Rechenaufwand ab, der zur Überprüfung aller Schlüssel erforderlich ist – einem Brute-Force-Angriff. Mit genügend Zeit und Rechenkapazität, wird der richtige Schlüssel irgendwann gefunden. Wenn nur der Geheimtext verfügbar ist, würde ein Brute-Force-Angriff den Geheimtext Block für Block und Schlüssel für Schlüssel entschlüsseln, um einen sinnvollen Text zu erhalten. Im Vergleich zur Ein-Schlüssel-Kryptografie, die sich auf rechnerisch schwierige mathematische Probleme verlässt (d. h. die Laufzeit wächst exponentiell mit der Bitlänge der Eingabe), ist die Kryptoanalyse der Zwei-Schlüssel-Kryptografie die der Computermathematik. Die Aufgabe besteht darin, einen Algorithmus zu finden, der die Berechnungszeit verkürzt, idealerweise einen mit polynomialer Laufzeit bezüglich der Anzahl der Eingabebits.

Zwar können wir in der Praxis für die bei RSA genutzte Primfaktorzerlegung oder den diskreten Logarithmus beim Diffie-Hellman-Schlüsselaustausch Algorithmen finden, deren Laufzeit langsamer als exponentiell (subexponentiell) mit der Anzahl der Eingabebits wächst, jedoch keinen mit polynomialer Laufzeit. Das Mooresche Gesetz besagte ursprünglich, dass sich die verfügbare Rechenleistung jedes Jahr ungefähr verdoppelt. Seit seiner Einführung beträgt der Geschwindigkeitszuwachs bei der Primfaktorzerlegung etwa sechs zusätzliche Binärziffern pro Jahr. Da sich also der Rechenaufwand für jeweils drei zusätzliche Ziffern verdoppelt, vervierfacht sich der Geschwindigkeitszuwachs pro Jahr. Um die Sicherheit eines Schlüssels über das Jahr 2020 hinaus zu gewährleisten, sind derzeit mindestens 2048 Binärziffern erforderlich.

Homophone

Unterschiedliche Chiffrierzeichen für denselben Klartextbuchstaben werden als Homophone bezeichnet.

###### Schlüsselgrößen im Vergleich

Diese Tabelle zeigt einen Vergleich der Schlüsselgrößen in Bits mit einem vergleichbaren Sicherheitsniveau zwischen einem symmetrischen Algorithmus wie dem AES, einem asymmetrischen Algorithmus mit elliptischen Kurven und einem asymmetrischen Algorithmus wie Diffie-Hellman oder RSA (Lenstra & Verheul, 2001).

|  |  |  |
| --- | --- | --- |
| Symmetrischer Schlüssel | Asymmetrisch-elliptischer Schlüssel | Klassischer asymmetrischer Schlüssel |
| 80 | 160 | 1024 |
| 112 | 224 | 2048 |
| 128 | 256 | 3072 |
| 192 | 384 | 7680 |
| 256 | 512 | 15360 |

Die Grundlage für die Werte der Tabelle bilden die schnellsten bekannten Algorithmen zur Lösung des kryptografischen Problems. Wenn ein Eingabeschlüssel mit n Bits gegeben ist, dann gilt:

* Bei einem symmetrischen Algorithmus wie AES besteht der schnellste Algorithmus darin, alle möglichen Schlüssel auszuprobieren, was einer Komplexität (= Anzahl der Operationen) von O(2n) entspricht.
* Für den Logarithmus über einer endlichen elliptischen Kurve ist der schnellste derzeitige Algorithmus der generische Babystep-Giantstep-Algorithmus (oder, etwas schneller, der ρ-Algorithmus von Pollard), dessen Komplexität ungefähr O(2√n) beträgt.
* Für klassische asymmetrische Algorithmen, entweder RSA oder Diffie-Hellman, auf einem endlichen Körper ist der schnellste Algorithmus das „allgemeine Zahlkörpersieb“, dessen Komplexität für ein großes n O(2√3n) ist.

In der Praxis beschleunigen die kleineren ECC-Schlüssel die kryptografischen Verfahren im Vergleich zu RSA und Diffie-Hellman um einen Faktor > 5 (zusätzlich zur Erleichterung des Austauschs zwischen Personen und zur Bandbreiteneinsparung). Allerdings hat ECC im Vergleich zu RSA auch Nachteile, so hängt sein Signaturalgorithmus beispielsweise von einem Pseudo-Zufallszahlengenerator ab, der bei schlechter Programmierung den privaten Schlüssel preisgibt.

### Rainbow-Tables

Rainbow-Tables (Regenbogentabellen) sind eine Methode zum Brechen von Passwörtern, bei der ein Passwort-Hash mit vorberechneten Hashes der wahrscheinlichsten Passwörter verglichen wird, d. h. ein Zeit-Speicher-Kompromiss – mehr Speicher für weniger Berechnungen.

Eine Rainbow-Table ist eine Tabelle mit kryptografischen Hashes der gebräuchlichsten Passwörter. Auf diese Weise werden häufigere Passwörter früher aufgedeckt. Die Generierung der Tabelle richtet sich nach der kryptografischen Hashfunktion und dem verwendeten Zeichensatz, der Länge des Passworts, der Anzahl der Tabelleneinträge und so weiter. Gängige kryptografische Hash-Algorithmen wie MD4/5 und SHA sind schnell. Sie sind daher für die Erstellung von Passwörtern ungeeignet, da sie anfällig für Brute-Force-Angriffe sind. So ist beispielsweise MD5 als kryptografische Hashfunktion auf Schnelligkeit ausgelegt und eignet sich daher gut für einen Rainbow-Table-Angriff, während Hashfunktionen wie PBKDF1, PBKDF2, bcrypt, scrypt und das neuere Argon2 so konzipiert wurden, dass sie diese Art von Angriff verhindern, indem sie absichtlich langsam (bcrypt) und/oder absichtlich speicherhungrig (scrypt) sind. Ein Rainbow-Table-Angriff kann jedoch am effektivsten verhindert werden, indem die verwendete Hashfunktion für jedes Passwort einzigartig gemacht wird. Dies wird durch die Verwendung eines Salt erreicht.

###### Lookup-Tabelle

Eine Lookup-Tabelle (LUT) ist eine Datenstruktur, die dazu dient, Laufzeitberechnungen durch einfache Array-Indizierungsoperationen zu ersetzen. In der LUT werden alle Einträge vorberechnet und anschließend lediglich nachgeschlagen, um Laufzeit zu sparen, da Array-Indizierungsoperationen im Allgemeinen schneller sind als Berechnungen (Kumar et al., 2013).

###### Meet-In-The-Middle-Angriff auf DES

Die Schlüsselgröße des symmetrischen Industriestandard-Kryptoalgorithmus DES betrug lediglich 56 Bit, so wenig, dass er schon kurz nach seiner Einführung Brute-Force-Angriffen zum Opfer fiel. Daher wurde angenommen, dass eine doppelte Verschlüsselung für zwei verschiedene Schlüssel die Schlüsselgröße effektiv auf 112 Bit verdoppeln würde. Der Meet-in-the-Middle-Angriff von Diffie und Hellman tauscht jedoch Speicher gegen Zeit, um den Schlüssel in nur 2n+1 Verschlüsselungen (unter Verwendung von etwa 2n gespeicherten Schlüsseln) statt der erwarteten 22n Verschlüsselungen zu finden (van Oorschot & Wiener, 1996).

Nehmen wir an, die Angreifer kennen einen Klartext P und seinen Geheimtext C, d. h.

C = EK: EK′ P ,

wobei E die Verschlüsselung mit K’ bzw. K’’ bezeichnet. Die Angreifer (1) berechnen EK(P) für alle möglichen Schlüssel K und speichern die Ergebnisse im Speicher, (2) entschlüsseln den Geheimtext, indem sie DK(C) für jeden K berechnen, und (3) suchen nach Übereinstimmungen zwischen diesen beiden Mengen, deren Schlüssel wahrscheinlich mit denen übereinstimmen, die zur Verschlüsselung von P zu C verwendet wurden.

Daher war eine dreifache Verschlüsselung, 3DES, notwendig, um die Schlüsselgröße effektiv zu verdoppeln und die Entschlüsselung für die zukünftige Computerleistung abzusichern.

### Bekannte und ausgewählte Klar- und Geheimtexte

Die asymmetrische Kryptografie nutzt mathematische Methoden – genauer gesagt modulare Arithmetik – zur Verschlüsselung. Die Sicherheit, also die Schwierigkeit der Entschlüsselung, der asymmetrischen Kryptografie basiert auf mathematischen Berechnungsproblemen, die seit Jahrhunderten als schwierig gelten. Bei der symmetrischen Kryptografie (z. B. Hashfunktionen) werden kunstvollere Verschlüsselungsmethoden genutzt, die darauf abzielen, die Diffusion und Konfusion zu maximieren – hauptsächlich durch iterative Substitution und Permutation. Die Sicherheit der symmetrischen Kryptografie beruht einfach darauf, dass sie über Jahre hinweg ständigen Angriffen standhält, d. h. sie ist aus praktischer Sicht zufriedenstellend, weniger jedoch im Hinblick auf das Ringen um ewige Wahrheiten.

###### Perfekte Sicherheit

Perfekte Sicherheit

Die Wahrscheinlichkeit, dass ein Klartext und ein Schlüssel, die einen bestimmten Geheimtext erzeugen, für alle Klartexte und alle Schlüssel gleich sind.

Klartexte kommen in der Regel nicht mit der gleichen Wahrscheinlichkeit vor. Dies hängt zum Beispiel von der Sprache, dem Jargon oder dem verwendeten Protokoll ab. Eine Chiffre ist vollkommen sicher, wenn keiner ihrer Geheimtexte etwas über den entsprechenden Klartext verrät. Außerdem ist die Wahrscheinlichkeit für jeden Klartext (stochastisch) unabhängig von jedem Geheimtext.

Bezeichnen wir mit p einen Klartext und mit P(p) seine Wahrscheinlichkeit. In Formeln ausgedrückt gilt für jeden Klartext p und jeden Geheimtext c, dass P(p|c) = P(p). In der Praxis bedeutet dies, dass Angreifer, die einen Geheimtext c abfangen, keinen Vorteil haben, d. h. die Wahrscheinlichkeit, dass sie den Klartext kennen, ist die gleiche als würden sie c nicht kennen.

Shannon (1949) bewies den folgenden Satz über die Bedingungen, unter denen eine Chiffre vollkommen sicher ist: Gegeben eine endliche Anzahl von Schlüsseln und Klartexten mit positiven Wahrscheinlichkeiten, d. h. P(p) > 0 für jeden Klartext p, ist die Chiffre perfekt sicher, wenn gilt:

* die Wahrscheinlichkeitsverteilung ist gleichmäßig, d. h. alle Wahrscheinlichkeiten sind gleich, und
* für jeden Klartext p und jeden Geheimtext c gibt es einen eindeutigen Schlüssel k, um c aus p zu erhalten.

Statistische Abweichungen neigen dazu, das Kryptosystem zu schwächen. Insbesondere ist es wichtig, einen zuverlässigen Zufallszahlengenerator für die Schlüssel zu verwenden.

###### One-Time-Pad

Das einzige vollkommen sichere Kryptosystem ist das One-Time-Pad. Ein solches vollkommen sicheres Kryptosystem ist jedoch unpraktisch. Für Echtzeitanwendungen, wie z. B. im Internet, wird es kaum verwendet. Das One-Time-Pad addiert (durch die XOR-Operation) jedes Bit des Klartextes t mit dem (positionsmäßig) entsprechenden Bit eines Schlüssels c , der die gleiche Länge hat, und wird nach Gebrauch verworfen, d. h. er wird nicht zur Verschlüsselung anderer Klartexte verwendet.

Der Geheimtext T = (T1, T2, …) ist somit

T = t ⊕ c = t1 ⊕ c1, t2 ⊕ c2, . . . .

Diese Chiffre ist so sicher wie theoretisch möglich!

Wenn der Klartext aus einem einzigen Block t besteht, dann ist diese einfache (XOR)-Addition eines Schlüssels, des One-Time-Pads, ein sicherer Algorithmus. Es ist jedoch oft unpraktisch oder sogar fast unmöglich, einen Schlüssel zu verwenden, der so lang ist wie der Klartext, z. B. um die Kommunikation über ein Netzwerk zu verschlüsseln. Wir müssen hierbei vorher wissen, wie viele Daten übertragen werden sollen.

Es ist eine schlechte Idee (wenngleich naheliegend), denselben Schlüssel für zwei verschiedene Blöcke zu verwenden. Wenn der Klartext zum Beispiel aus zwei Blöcken b’ und b’’ besteht, dann ergibt sich, dass bei diesem Algorithmus die Summe (XOR)

One-Time-Pad

Bei einem One-Time-Pad ist der Schlüssel genauso lang wie der Klartext. Die Schlüssel werden Buchstabe für Buchstabe (oder Bit für Bit) addiert, um den Geheimtext zu erhalten.

b′ ⊕ c ⊕ b: ⊕ c = b′ ⊕ b:

der beiden Chiffrierblöcke b’ ⊕ c und b’’ ⊕ c gleich der Summe b’ ⊕ b’’ der beiden klaren Blöcke ist. Das liegt daran, dass die Addition XOR per Definition selbstinvers ist, d. h. x ⊕ x = 0 unabhängig davon, ob die binäre Ziffer x = 0 oder x = 1 ist!

Es kann vorkommen, dass die Chiffrierung des ersten Blocks durch ein One-Time-Pad dem zweiten Block entspricht. Leider ist der zweite Block kein guter Schlüssel. Weil er nicht zufällig ist, ähnelt sein Inhalt in der Regel dem des ersten Blocks, was bedeutet, dass der Schlüssel vorhersehbar ist.

###### Erprobte Sicherheit

Da perfekte Sicherheit nicht realisierbar ist, wird Sicherheit nur durch die Widerstandsfähigkeit gegen bekannte Angriffe oder durch die Reduzierung der rechnerischen Schwierigkeit auf die eines als schwierig anerkannten (rechnerischen, mathematischen) Problems nachgewiesen: bewiesene Sicherheit. Es gibt zwar bewiesene sichere symmetrische Kryptosysteme, aber die effizientesten und am weitesten verbreiteten Algorithmen, wie z. B. AES, erweisen sich nur gegen bekannte Angriffe, wie z. B. die der differentiellen oder linearen Kryptoanalyse, als resistent.

Formal bewiesene Sicherheit

Die mathematischen Probleme, auf denen die Schwierigkeit der Entschlüsselung bei asymmetrischen kryptografischen Algorithmen beruht, sind alle NP-vollständig, d. h. ihre Lösungen sind in polynomialer Laufzeit (in der Bitlänge der Eingabe) verifizierbar, und alle anderen derartigen Probleme können darauf reduziert werden. Jeder kryptografische Algorithmus verschlüsselt oder entschlüsselt also eine Nachricht mit dem Schlüssel in polynomialer Laufzeit bezüglich der Bitlänge des Schlüssels.

Im Gegensatz dazu benötigen alle bekannten Algorithmen zur Dechiffrierung ohne den Schlüssel exponentielle Laufzeit in Abhängigkeit von der Bitlänge des Schlüssels. Gemäß der P-NP-Vermutung gibt es keinen Algorithmus, der eine polynomiale Laufzeit (in der Bitlänge des Schlüssels) benötigt. Da die Vermutung jedoch bisher weder bewiesen noch widerlegt wurde, könnte es theoretisch polynomiale Algorithmen geben, die ohne den Schlüssel in polynomialer Laufzeit entschlüsseln. Nach jahrzehntelangen ununterbrochenen Bemühungen der Kryptoanalytiker weltweit gilt dies jedoch als unwahrscheinlich.

Das erste Beispiel für ein solches nachweislich sicheres Kryptosystem war die semantische Sicherheit von Goldwasser et al. (1982), die den Schwierigkeitsgrad der Entschlüsselung auf die Berechnung des quadratischen Restes reduziert. Gegeben x und N ein Produkt aus zwei Primzahlen, ist es schwierig zu bestimmen, ob x quadratisch modulo N ist (d. h., ob es y so gibt, dass x = y2 mod N oder nicht), wenn, und nur wenn, das Jacobi-Symbol für x +1 ist und die Primfaktoren von N unbekannt sind.

Das Goldwasser-Micali-Kryptosystem besteht aus:

1. Einem Algorithmus zur Schlüsselerzeugung. Er erzeugt den privaten Schlüssel als zwei Primzahlen p und q sowie den öffentlichen Schlüssel N = p q und eine Zahl x , die weder modulo p noch modulo q quadratisch ist (so dass das Jacobi-Symbol von x für N +1 und für ihre beiden Faktoren p und q –1 ist). Wenn zum Beispiel p, q ≡ 3 mod 4, dann genügt x = N–1.
2. Einem probabilistischen Verschlüsselungsalgorithmus. Wenn m = (m1, m2, …) die Bits des Klartextes sind, dann werden die Zahlen y1, y2, … , die nicht durch p und q teilbar sind, erzeugt und die chiffrierte Nachricht ist M = (M1, M2, …) mit M1 = y 2 xm1, M1 = y 2 xm1, …
3. Einem deterministischen Entschlüsselungsalgorithmus. Wenn M = (M1, M2, …) die verschlüsselte Nachricht ist, dann ist m1 = 0 nur genau dann, wenn M1 quadratisch modulo N ist, ... was durch die Kenntnis der beiden Faktoren p und q von N schnell festgestellt werden kann.

Paradox

Theoretische Sicherheit bleibt eine unzulängliche Idealisierung der Realität. Ajtai und Dwork (1997) haben zum Beispiel ein theoretisch sicheres Kryptosystem vorgestellt und bewiesen, das ein Jahr später gebrochen wurde. „Bewiesen“ bedeutet nicht „wahr“. Ein bewiesen sicheres System ist nicht notwendigerweise wirklich sicher, weil der Nachweis in einem formalen Modell erbracht wird, das bestimmte Verfahrensprinzipien, Angreifer, ein bestimmtes Sicherheitsziel und die Schwierigkeit des Problems voraussetzt, auf das der Beweis zurückgeführt wird.

Zum Beispiel unterscheidet sich das implementierte Kryptosystem vom formalen Kryptosystem. Ein Teilziel ist für Angreifende bereits ausreichend. Wenn das Sicherheitsziel zum Beispiel darin besteht, dass die Angreifer nicht den gesamten Klartext aus dem Geheimtext ableiten, dann kann es bereits ausreichen, dass sie einen Teil des Klartextes ableiten. Außerdem könnte der Nachweis falsch sein! Trotz dieser Ungewissheit ist ein Nachweis der Sicherheit ein nützliches Kriterium für die Sicherheit eines Kryptosystems, das theoretisch zwar notwendig ist, aber praktisch unzureichend sein kann).

###### Angriffsszenarien

Was bedeutet Sicherheit? Das Kriterium, dass Angreifer den Klartext nicht aus dem Geheimtext ableiten können, ist unzureichend, weil sie andere nützliche (Teil-)Informationen über den Klartext gewinnen könnten. Aber selbst wenn es unmöglich ist, nützliche Informationen über den Klartext abzuleiten, ist dies unter bestimmten Umständen unzureichend. Wenn die Public-Key-Verschlüsselung deterministisch ist (d. h. wenn dieselbe Eingabe immer dieselbe Ausgabe liefert, wie z. B. bei der RSA-Verschlüsselung gemäß Lehrbuch) und Angreifende die Anzahl der möglichen Klartexte einschränken können (sie wissen z. B., dass der Geheimtext „ja“ oder „nein“ lautet), dann können sie alle diese möglichen Klartexte mit dem öffentlichen Schlüssel verschlüsseln und die Geheimtexte mit den verschlüsselten Texten vergleichen. Bei einem asymmetrischen Algorithmus gehen wir davon aus, dass Angreifende den öffentlichen Schlüssel kennen. Sie können also jeden beliebigen Klartext verschlüsseln und ihn mit dem Geheimtext vergleichen, d. h. sie können einen Chosen-Plaintext-Angriff (CPA) durchführen.

Im Allgemeinen werden die Angriffsszenarien danach eingeteilt, wie viel die Kryptoanalytiker über den Geheimtext wissen. Kennen sie nur den Geheimtext oder gibt es wahrscheinliche oder bekannte Paare von Geheimtexten und Klartexten? Gibt es auch einen ausgewählten Klartext oder Geheimtext? Um zum Beispiel eine monoalphabetische Chiffre zu brechen, genügt dank der Frequenzanalyse in der Regel der Geheimtext allein. Oft kennen oder erraten Kryptoanalytiker jedoch einen Teil des Klartextes, z. B. die Präambel eines Briefes (z. B. eine förmliche Begrüßung) oder ein Computerdateiformat (z. B. einen Identifikator). Und schließlich können sie entweder Absender auffordern, einen von ihnen gewählten Klartext zu verschlüsseln oder Empfänger, einen von ihnen gewählten Geheimtext zu entschlüsseln.

Angriff nur mit Geheimtext

Angreifende verfügen über die Geheimtexte mehrerer Nachrichten, die mit dem gleichen Algorithmus verschlüsselt wurden. Ihre Aufgabe ist es, so viel Klartext wie möglich wiederherzustellen oder, besser noch, die verwendeten Algorithmen und Schlüssel zu finden.

Angriff mit mutmaßlichem Klartext

Angreifende sind im Besitz des Geheimtextes und vermuten, dass der Klartext bestimmte Wörter oder sogar ganze Sätze enthält. Ihre Aufgabe ist es, so viele Klartexte wie möglich wiederherzustellen oder, besser noch, die Algorithmen und Schlüssel zu finden, die verwendet wurden. Beispielsweise die kryptografische elektromechanische Rotor-Chiffriermaschine Enigma, die von den Achsenmächten im Zweiten Weltkrieg eingesetzt wurde, wurde durch die Wiederholungen der Nachrichten, die sie verschlüsselte, gebrochen: Der Wetterbericht wurde täglich gesendet und als solcher am Anfang jeder Nachricht angekündigt.

Angriff mit bekanntem Klartext

Angreifende verfügen über einen Geheimtext und den entsprechenden Klartext. Ihre Aufgabe ist es, den verwendeten (Algorithmus und) Schlüssel wiederherzustellen; so fällt z. B. die lineare Kryptoanalyse in dieses Szenario. Ein Angriff aus dem Jahr 2006 auf das WEP-Protokoll (Wired Equivalent Privacy) zur WLAN-Verschlüsselung nutzte beispielsweise die Vorhersagbarkeit von Teilen der verschlüsselten Nachrichten aus, nämlich die Header des 802.11-Protokolls.

Angriff mit ausgewähltem (oder adaptivem) Klartext (CPA)

Angreifende besitzen die Geheimtexte der Klartexte, die sie frei wählen können. Angreifende können den Klartext in Abhängigkeit von dem nach jeder Dechiffrierung erhaltenen Text frei anpassen und die daraus resultierenden Änderungen im Geheimtext analysieren. Ihre Aufgabe ist es, den verwendeten Algorithmus und Schlüssel wiederherzustellen. Die differentielle Kryptoanalyse fällt in dieses Szenario.

Dies ist das minimale Angriffsszenario, auf das wir bei asymmetrischer Kryptografie vorbereitet sein müssen! Da der Verschlüsselungsschlüssel öffentlich ist, können Angreifer Nachrichten nach Belieben verschlüsseln. Wenn sie also die Anzahl der möglichen Klartexte reduzieren können, z. B. weil sie wissen, dass diese entweder „Ja“ oder „Nein“ lauten, dann können sie alle möglichen Klartexte mit dem öffentlichen Schlüssel verschlüsseln und mit dem abgefangenen Geheimtext vergleichen. Auch der klassische RSA-Algorithmus ist beispielsweise anfällig für diesen Angriff. Um sich gegen diesen CPA-Angriff zu schützen, muss daher jede Implementierung dieses Algorithmus‘ den Klartext vor der Verschlüsselung mit Zufallsdaten auffüllen.

Angriff mit ausgewähltem (oder adaptivem) Geheimtext

Angreifende haben einen Geheimtext c und die Klartexte der Geheimtexte (außer von c), die sie frei wählen können. Die Angreifer können den Klartext in Abhängigkeit von dem nach jeder Dechiffrierung erhaltenen Text frei anpassen und die daraus resultierenden Änderungen im Geheimtext analysieren. Ihre Aufgabe ist es, den verwendeten Algorithmus und Schlüssel wiederherzustellen. Angreifer müssen zum Beispiel die Blackbox einer Chiffriermaschine analysieren, deren innere Funktionsweise unbekannt ist.

Nur wenige praktische Angriffe fallen in dieses Szenario, aber es ist wichtig für Nachweise der Sicherheit. Wenn die Widerstandsfähigkeit gegen die Angriffe dieses Szenarios nachgewiesen werden kann, dann ist die Widerstandsfähigkeit gegen jeden realistischen Angriff auf den gewählten Geheimtext gegeben.

###### Semantische Sicherheit

Wenn ein asymmetrischer Algorithmus verwendet wird, dann sollte davon ausgegangen werden, dass die Angreifer den öffentlichen Schlüssel kennen. Sie können also jeden beliebigen Klartext verschlüsseln und ihn mit dem Geheimtext vergleichen, d. h. sie können einen Chosen-Plaintext-Angriff (CPA) durchführen.

IND-CPA-sicher „Nicht unterscheidbar bei einem Chosen-Plaintext-Angriff“ bedeutet, dass kein Angreifer eine Wahrscheinlichkeit von mehr als der Hälfte hat, zwei Geheimtexte zu unterscheiden.

Eine Chiffre ist sicher gegen IND-CPA, wenn kein Angreifer unterscheiden kann, welcher von zwei zuvor ausgewählten Klartexten dem danach erhaltenen Geheimtext entspricht. Konkret ist das Kryptosystem unter CPA nicht unterscheidbar, wenn alle Angreifer in probabilistischer Polynomialzeit nur einen unbedeutenden „Vorteil“ gegenüber dem zufälligen Erraten haben (Abdalla et al., 2016).

Die vier Schritte des IND-CPA-Spiels mit polynomialer Laufzeitbeschränkung (in der Bitlänge des Schlüssels k) auf die Berechnungen der Angreifer (durchgeführt bei der Erstellung der beiden Klartexte, Schritt zwei, und bei der Wahl des Klartextes, der dem Geheimtext entspricht, Schritt vier) sind:

1. Es wird ein Schlüsselpaar erstellt, ein geheimer und ein öffentlicher Schlüssel, beide mit k Bits. Die Angreifer erhalten den öffentlichen Schlüssel.
2. Die Angreifer berechnen zwei Klartexte M0 und M1 von gleicher Größe.
3. Die Chiffriermaschine wählt zufällig ein Bit b ∈ {0, 1}, verschlüsselt Mb und gibt den Geheimtext an die Angreifer weiter.
4. Die Angreifer wählen ein Bit b’ ∈ {0, 1}.

Die Angreifer, die das Bit b’ im vierten Schritt zufällig wählen, liegen mit einer Wahrscheinlichkeit von ½ richtig, wobei die Wahrscheinlichkeit P(b = b’) – 1/2 nicht signifikant ist, wenn die Angreifer einen nicht signifikanten „Vorteil“ haben. Dies ist der Fall, wenn sie mit einer Wahrscheinlichkeit ≥1 / 2 + =(k) gewinnen, wobei = eine nicht signifikante Funktion in k ist, d. h., für jede (von Null verschiedene) Polynomfunktion p gibt es k0, so dass =(k) < 1/p(k) für alle k > k0. Eine nicht signifikante Differenz sollte eingeräumt werden, da die Angreifer ihre Erfolgswahrscheinlichkeit leicht über 1/2 erhöhen, indem sie einen geheimen Schlüssel erraten und versuchen, den Geheimtext damit zu entschlüsseln. Obwohl dieses Spiel für ein asymmetrisches Kryptosystem formuliert ist, kann es an den symmetrischen Fall angepasst werden, indem die Public-Key-Chiffre durch ein kryptografisches Orakel ersetzt wird, eine Blackbox-Funktion, die den geheimen Schlüssel aufbewahrt und beliebige Klartexte auf Anfrage der Angreifer verschlüsselt.

### Seitenkanalangriffe

Physikalische Komponenten, die von Seitenkanalangriffen verwendet werden, sind z. B. Messungen der Rechenzeit, des Stromverbrauchs und der elektromagnetischen oder akustischen Emissionen. Wir werden uns auf Rechenzeitangriffe (Timing Attacks) konzentrieren. Diese können aus der Ferne durchgeführt werden, allerdings leiden die Messungen oft unter Rauschen, d. h. unter zufälligen Störungen durch Quellen wie Netzwerklatenz, Festplattenzugriffszeiten und Korrektur von Übertragungsfehlern. Für die meisten Rechenzeitangriffe müssen Angreifer die Implementierung kennen. Diese Angriffe können jedoch auch für Reverse Engineering verwendet werden.

Wir werden uns auf das Beispiel eines Rechenzeitangriffs konzentrieren, der die Zeit für die Berechnung der Potenzen ganzer Zahlen misst. Dazu müssen wir zunächst verstehen, wie ganzzahlige Potenzen berechnet werden.

###### Exponentiation durch Quadratur

Die Exponentiation durch Quadrieren (auch bekannt als Square-and-Multiply-Algorithmus oder binäre Exponentiation) ist ein Algorithmus, der zur schnellen Berechnung großer ganzzahliger Potenzen einer Zahl durch binäre Expansion des Exponenten verwendet wird. Dies ist besonders nützlich in der modularen Arithmetik. Um bn zu berechnen, sind statt n Mal b multipliziert mit b zu berechnen, nur 2 • log2 n Multiplikationen nötig.

Bei einer nicht-negativen ganzen Zahl als Basis b und einem Exponenten e zerlegen wir zum Berechnen von be mod M den Exponenten binär, d. h.

e = e0 + e12 + e222 + ! + es2s mit e0, e1, . . . , es ∈ 0,1 ,

und berechnen

Seitenkanalangriff Dieser Angriff nutzt Informationen über die physische Implementierung einer Chiffriermaschine.

Rechenzeitangriff

Ein Angriff, der die Laufzeit kryptografischer Operationen misst und sie mit Schätzungen vergleicht.

1 2 22 2s

b , b ,b , . . . , b

mod M .

Aufgrund von b^{2n+1} = b^{(2n)2} = (b^{2n})2 (jede Potenz ist das Quadrat der vorhergehenden, und höchstens M) ist jede Potenz, eine nach der anderen, leicht berechenbar. Dies ergibt:

e e + e 2 + e 22 + ! + e 2s

e 2 e1

22 e2

2s e

b = b 0 1 2

s = b 0 b b

! b s .

Im Nachhinein zählen nur Potenzen mit e0, e1, …, es gleich 1; die anderen können weggelassen werden.

Zum Berechnen von 35 mod 7 zerlegen wir z. B.

5 = 1 + 0 · 21 + 1 · 22

und berechnen so:

31 = 3, 32 = 9 ≡ 2, 322 = 32 2 ≡ 22 = 4 mod 7

Daraus ergibt sich:

35 = 31 + 22 = 31 · 322 = 3 · 4 ≡ 5 mod 7 .

###### Anfällige Algorithmen

Für die Ausführung der binären Exponentiation betrachten wir den Exponenten, insbesondere die Anzahl der Bits, die gleich 1 sind. Seine Laufzeit hängt linear von dieser Zahl ab. Diese Zahl allein reicht zwar nicht aus, um einen Schlüssel zu finden, aber eine statistische Korrelationsanalyse der Exponentiationen mit verschiedenen Basen hilft bei der Ableitung des Exponenten.

Kryptoalgorithmen, die mit Exponentiation modulo einer großen Primzahl verschlüsseln, sind für diesen Angriff anfällig. Dazu gehören RSA, Diffie-Hellman und Elgamal. Bei RSA zum Beispiel ist die Nachricht die Basis b und der Schlüssel der Exponent e. Brumley und Boneh (2005) demonstrierten einen netzwerkbasierten Rechenzeitangriff auf SSL-fähige Webserver unter Verwendung von RSA, der den privaten Schlüssel innerhalb eines Tages erfolgreich wiederherstellte. Dies führte zum weit verbreiteten Einsatz von Verschleierungstechniken, um Korrelationen zwischen Schlüssel und Verschlüsselungszeit zu verbergen.

###### Beispiel für die Exponentiation (wie in Diffie-Hellman verwendet)

Kochis (1996) hat im folgenden Algorithmus zur Berechnung der modularen Exponentiation, d. h. zur Berechnung von R(y) = yx mod n für n öffentlich und y bekannt, aber x geheim, einen Fehler aufgedeckt. Die Angreifer können durch die Berechnung von R(y) für mehrere Werte von y und bei Kenntnis von n, y sowie der Rechenzeit x wie folgt ableiten:

Sei w die Bitlänge von x und s\_0 = 1. Für k von 0 bis w – 1: wenn das k-te Bit von x den Wert 1 hat, dann sei R\_k = (s\_k \* y) mod n; sonst sei R\_k = s\_k. Sei s\_{k+1} = R\_k^k mod n

Ende (der Für-Schleife) Rückgabe von (R\_{w-1})

Je nach dem Wert des k-ten Bits von x wird entweder (sk × y) mod n oder nichts berechnet. Daher wird die Ausführungszeit des Algorithmus für verschiedene Werte von y schließlich den Wert des k-ten Bits ergeben.

### Moderne kryptoanalytische Algorithmen

Um die Gründe für die Auswahl der einzelnen Schritte eines Blockchiffrieralgorithmus wie AES zu verstehen, müssen wir zunächst wissen, welche Angriffe er abwehrt. Ein leistungsfähiger und moderner kryptoanalytischer Algorithmus ist die differenzielle Kryptoanalyse, die auf Blockchiffren anwendbar ist. Es setzt einen CPA voraus; Angreifende senden Paare von (leicht) unterschiedlichen Klartexten, deren Geheimtexte sie erhalten. Dann untersuchen sie, wie sich Unterschiede in der Eingabe (auf Differenzpfaden) durch das Netzwerk von Verschlüsselungsumwandlungen zu Unterschieden in der Ausgabe fortpflanzen. Daemen und Rijmen (1999) bewiesen die Widerstandsfähigkeit von AES gegen breite Pfade.

###### Prototypische Feistelchiffre

Wir wollen diese Technik anhand des einfachen Modells einer Feistelchiffre (Heys, 2002) demonstrieren, die den Klartext in Blöcke von 16 Bits aufteilt und jeden Block in vier Blöcke von vier Bits unterteilt.

Für jede Runde gibt es einen entsprechenden (unabhängigen) Schlüssel. In jeder der ersten drei Runden gilt:

1. Addiere den Rundenschlüssel C zum Block B / B ⊕ C.
2. Ersetze die Bits der 4 Teilblöcke entsprechend der Tabelle (in hexadezimaler Notation)

|  |  |
| --- | --- |
|  | |
| 2. | 0 |
| 3. | 1 |
| 4. | 2 |
| 5. | 3 |
| 6. | 4 |
| 7. | 5 |
| 8. | 6 |
| 9. | 7 |
| 10. | 8 |

|  |  |
| --- | --- |
| 11. | 9 |
| 12. | A |
| 13. | B |
| 14. | C |
| 15. | D |
| 16. | E |
| 17. | F |
| 18. | E |
| 19. | 4 |
| 20. | D |
| 21. | 1 |
| 22. | 2 |
| 23. | F |
| 24. | B |
| 25. | 8 |
| 26. | 3 |
| 27. | A |
| 28. | 6 |
| 29. | C |
| 30. | 5 |

|  |  |
| --- | --- |
| 31. | 9 |
| 32. | 0 |
| 33. | 7 |

1. Vertausche Bit i aus dem Teilblock j mit j aus dem Teilblock i.
2. Addiere in der vorletzten vierten Runde den Rundenschlüssel zu dem Block B /B ⊕ C und ersetze jeden der vier Teilblöcke mit vier Bits durch die Tabelle.
3. Addiere in der letzten fünften Runde den Rundenschlüssel zum Block, B /B ⊕ C.

In der vierten Runde wird der letzte Schritt, die Permutation, ausgelassen, da er nur den letzten Schlüssel der fünften Runde permutieren würde. Aus kryptografischer Sicht ist das überflüssig.

In der fünften (letzten) Runde werden die letzten beiden Schritte (Substitution und Permutation) ausgelassen, da der Algorithmus öffentlich ist (nach dem Kerckhoffs’schen Prinzip) und von allen Dechiffrierenden ohne Kenntnis des Schlüssels rückgängig gemacht werden kann. Aus kryptografischer Sicht sind diese überflüssig.

Die Substitutionstabelle stammt aus dem DES-Algorithmus und wird gemeinhin als Substitutionsbox (S-Box) bezeichnet.

###### Differenzielle Kryptoanalyse

Der Traum von Kryptoanalytikern ist es, zu erfahren, ob ein Teil des gewählten Schlüssels korrekt ist, d. h. ob er mit dem entsprechenden Teil des Schlüssels übereinstimmt, der zur Verschlüsselung des Textes verwendet wurde. Bei der Heys-Chiffre hat der Schlüssel zum Beispiel eine Länge von 16 Bit. Wenn Kryptoanalytiker herausfinden, dass die Hälfte (acht Bits) des versuchten Schlüssels mit der entsprechenden Hälfte des richtigen Schlüssels übereinstimmt, müssen sie nur alle möglichen Kombinationen dieser acht Bits und der restlichen acht Bits testen. Damit reduziert sich die Anzahl der zu testenden Kombinationen von 216 = 65536 auf 2 • 256 = 512.

Kriterium für die Entschlüsselung

Bei einem Brute-Force-Angriff entschlüsseln Kryptoanalytiker den verschlüsselten Text mit jedem nur möglichen Schlüssel. Um nun festzustellen, ob der versuchte Schlüssel korrekt ist, prüfen sie, ob der Inhalt verständlich ist. Dazu könnten beispielsweise die Häufigkeiten der Buchstaben, Paare und Dreiergruppen des entschlüsselten Textes gezählt und mit den Häufigkeiten der vermuteten Sprache des Klartextes verglichen werden. Wenn sie nahe beieinander liegen, ist der Klartext wahrscheinlich verständlich und der versuchte Schlüssel wurde von den jeweiligen Verschlüsselnden verwendet.

Wenn die Chiffre eine einzige Runde hat, dann ist dieses Kriterium anwendbar. Wenn die Chiffre jedoch zwei oder mehr Runden hat und Dechiffrierende die letzte Runde des Entschlüsselungsalgorithmus mit einem bestimmten Schlüssel ausführen, dann ist dieses Kriterium nicht mehr anwendbar, da der erhaltene Text die Ausgabe des Verschlüsselungsalgorithmus (mit demselben Schlüssel) aus der vorletzten Runde ist. Stattdessen ist das Kriterium der differenziellen Kryptoanalyse, den richtigen Schlüssel gefunden zu haben, probabilistisch. Der getestete Schlüssel ist wahrscheinlich korrekt, wenn für eine bestimmte „eingehende“ Differenz ∆X und eine bestimmte „ausgehende“ Differenz ∆Y, Klartextpaare mit Differenz ∆X mit einer bestimmten Wahrscheinlichkeit in Geheimtextpaaren mit Differenz ∆Y resultieren.

Damit die differenzielle Kryptoanalyse anwendbar ist, müssen Kryptoanalytiker in der Lage sein, eine beliebige Anzahl von frei gewählten Klartexten mit demselben Schlüssel zu verschlüsseln und die verschlüsselten Texte zu untersuchen.

Ein Differenzial ist ein Paar D = (∆X, ∆Y) aus Eingangs- bzw. Ausgangsdifferenzen ∆X

bzw. ∆Y. Die differenzielle Kryptoanalyse nutzt die hohe Wahrscheinlichkeit einer Differenz

∆X := X’ ⊕ X’’ zwischen zwei Klartexten X’ und X’’, die sich zu einer Differenz ∆Y = Y’ ⊕ Y’’ zwischen zwei Geheimtexten Y’ und Y’’ fortpflanzen (für X’ und X’’) (in der vorletzten Runde). Hier ist X’ ⊕ X’’ die bitweise XOR-Addition, d. h. ∆X zeigt alle Bits an, bei denen sich X’ und X’’ unterscheiden). Das Paar D = (∆X, ∆Y) ist das Differenzial. Die Differenz von X’ und X’’ ist ∆X := X’ ⊕ X’’*.*

Damit die differenzielle Kryptoanalyse effizient ist, muss es ein Differenzial D mit hoher Wahrscheinlichkeit pD geben. Daher ist unter allen eingehenden Paaren mit der Differenz ∆X die Wahrscheinlichkeit eines ausgehenden Paares mit der Differenz ∆Y (in der vorletzten Runde) pD. Genauer gesagt verschlüsseln Chiffrierende eine statistisch signifikante Anzahl von Paaren (> 1/pD) von Klartexten mit der Differenz ∆X , um die Anzahl der verschlüsselten Paare von Geheimtexten mit der Differenz ∆Y zu zählen.

Häufigkeit der Differenzen für eine Substitutionstabelle

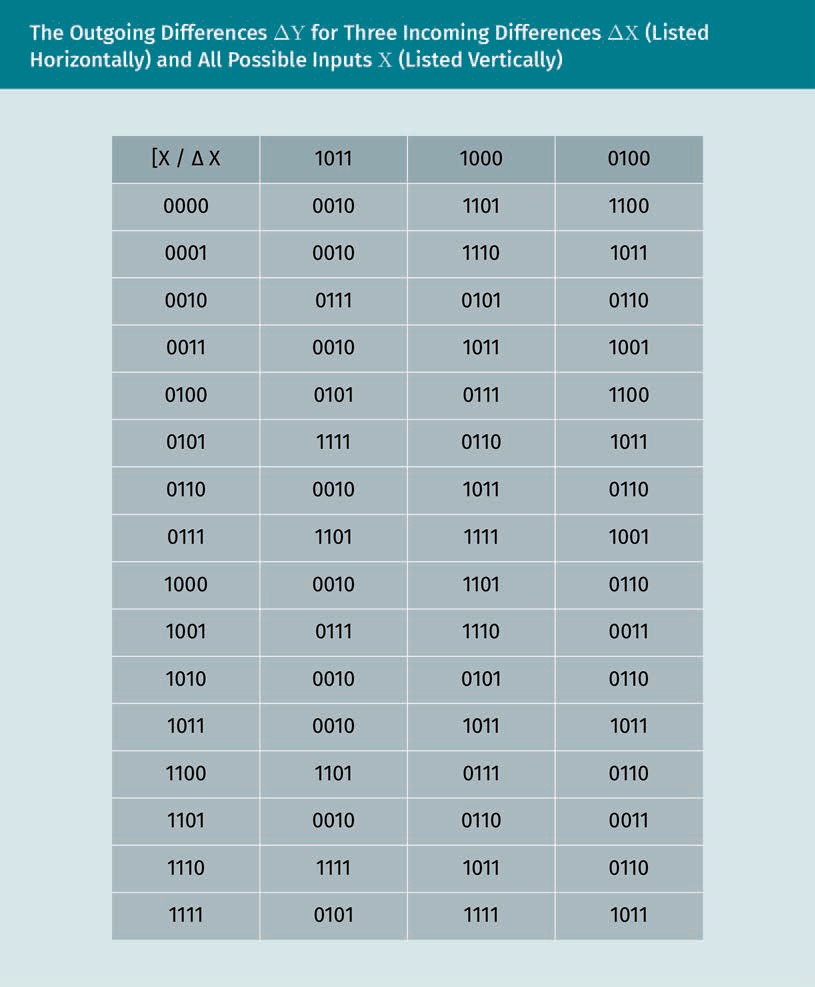
Eine affine Transformation A ist die Zusammensetzung aus einer linearen Anwendung, d. h. A(x ⊕ y) = A(x) ⊕ A(y) für alle x und y und einer Translation, d. h. A(x) = x ⊕ x0 für ein vorgegebenes x0.

Für eine affine Transformation A ist die ausgehende Differenz ∆Y unabhängig von dem eingehenden Paar X’ und X’ (sondern nur abhängig von ∆X). Für die Transformation A gilt:

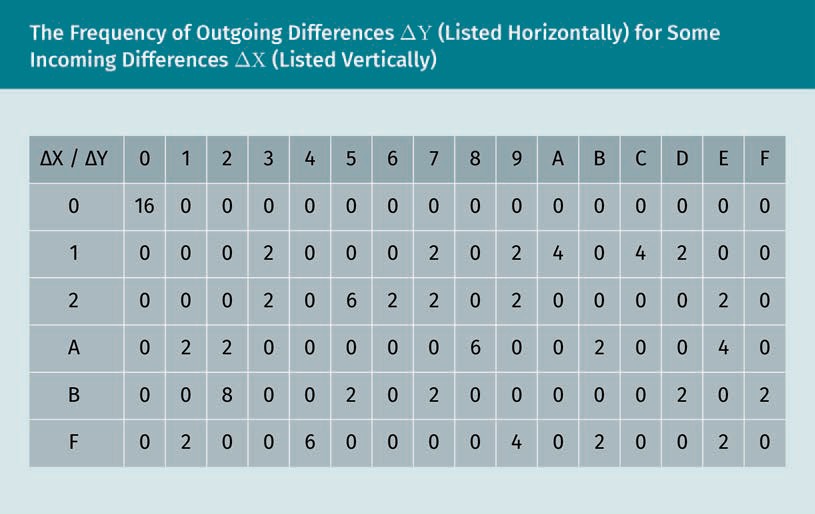
* Sofern sie linear ist, ist für jedes eingehende Paar mit der Differenz ∆X die ausgehende Differenz ∆Y = A(∆X), und
* sofern sie eine Translation ist, gilt immer ∆Y = ∆X.

Was die erste und zweite Funktion jeder Runde einer Feistelchiffre betrifft, so ist die Addition des Schlüssels eine Translation, wohingegen die Permutation linear ist.

Daher ist die ausgehende Differenz unabhängig von dem eingehenden Paar. Die ausgehende Differenz ∆Y der Substitution wird jedoch nicht allein durch die eingehende Differenz ∆X bestimmt, sondern hängt von X’ und X’’ ab! Wir prüfen die Substitutionstabelle, um ein Differenzial D mit hoher Wahrscheinlichkeit pD zu finden, also eine eingehende Differenz ∆X mit einer großen Anzahl von Paaren X’ und X’’, die eine ausgehende Differenz ∆Y ergeben. Bei ∆X gibt es 24 = 16 mögliche Eingaben X’ (die X’’ = X’ ⊕ ∆X bestimmen), und wir zählen die Häufigkeiten der 24 = 16 möglichen ausgehenden Differenzen ∆Y = 0, 1, …, F.



Zählen wir nun für jede eingehende Differenz ∆X wie oft jede ausgehende Differenz ∆Y unter allen eingehenden Paaren X’ und X’’ auftaucht, so dass X’ ⊕ X’’ = ∆X.



Die Einträge für jede Zeile summieren sich zu 16, der Anzahl aller möglichen Paare für eine bestimmte Differenz. Die erste Zeile bestätigt, dass zwei gleiche Eingaben zu zwei gleichen Ausgaben führen. Die höchste Zahl ist 8 und wird erreicht für ∆X = B und ∆Y = 2. Außerdem taucht die Zahl 6 fünfmal auf. Wir werden unsere Differenziale unter denjenigen mit diesen hohen Häufigkeiten auswählen.

Beispiel

In der Häufigkeitstabelle

* einer Translation, wie z. B. dem Hinzufügen des geheimen Schlüssels, sind alle Felder null, außer denen in der ersten Spalte, die den Wert 16 haben.
* sind für eine lineare Operation wie eine Permutation von Bits in jeder Zeile alle Einträge null, außer einem mit dem Wert 16.

Differenzpfade

Ein Differenzpfad ist ein Tupel von Differenzen, so dass jeder Eintrag ∆Ui die Eingabe der S-Box der i-ten Verschlüsselungsrunde ist. Gegeben sei eine Feistelchiffre und einen Differenzpfad (∆U1, ∆U2,…), sowie die ausgehende Differenz ∆Vi der S-Box der Runde i, so ist die eingehende Differenz der nächsten Runde ∆Ui+1 das Ergebnis der Anwendung der Permutation auf ∆Vi. (Die Addition des Schlüssels ändert als Translation nichts an der Differenz.)

Wir wollen den wahrscheinlichsten Differenzpfad D in der Heys-Chiffre finden:

D = ∆U1, ∆U2, ∆U3, ∆U4

Kryptoanalyse – Verschlüsselungen brechen

Oder zumindest wollen wir einen Pfad finden, bei dem jedes Differenzial (∆Ui, ∆Vi) zu den wahrscheinlichsten gehört. Jedes Differenzial besteht aus vier Teildifferenzialen, die jeweils den vier Teilblöcken von vier Bits entsprechen, die einen Block von 16 Bit bilden. Um einen solchen wahrscheinlichen Differenzpfad zu finden, maximieren Sie in jeder Runde die Häufigkeit jedes Teildifferenzials, d. h. die Anzahl, wie oft die S-Box die eingehende Differenz (des Teildifferenzials) in die ausgehende Differenz transformiert. Minimieren Sie insbesondere die Anzahl der (aktiven) Teildifferenzen, die ungleich Null sind.

Ein Beispiel für einen solchen Pfad D sehen wir hier: Die Differenz in der ersten Runde sei

∆U1 = 0000 1011 0000 0000 ,

die durch S-Box 2 ersetzt wird durch

∆V1 = 0000 0010 0000 .

Durch die anschließende Permutation erhalten wir als Differenz beim Eintritt in die zweite Runde

∆U2 = 0000 0000 0100 0000

die durch S-Box 3 ersetzt wird durch

∆V2 = 0000 0000 0110 0000 .

Da die Anzahl der Bits 2 und 3 von Null verschieden sind, erhalten wir durch die anschließende Permutation als eingehende Differenz der dritten Runde, dass bei zwei aktiven Teildifferenzialen

∆U3 = 0000 0010 0000

die durch S-Box 2 und 3 ersetzt wird durch

∆V3 = 0000 0101 0000 .

Durch die anschließende Permutation ist die Eingabe der vierten Runde schließlich

∆U4 = 0000 0110 0000 0110 .

Bezeichnen wir mit Si,j die Substitution des Teilblocks j durch die S-Box in der i-ten Runde. Auf unserem Differenzpfad tragen wir für jede Runde i = 1,2,3 und für jedes Teildifferenzial j = 1,2,3,4 abweichend von Null die Wahrscheinlichkeit der Substitution Si,j ein, die die eingehende Differenz X (in hexadezimaler Notation) in eine ausgehende Differenz Y umwandelt:



Wenn wir annehmen, dass die Differenziale einer Runde unabhängig von den Differenzialen der vorherigen Runde sind (was eine vernachlässigbar unzutreffende Vereinfachung ist), dann ist die Wahrscheinlichkeit pD der verketteten Substitutionen, die

∆U1 = 0000 1011 0000 0000

in

∆U4 = 0000 0110 0000 0110 .

transformiert, das Produkt aus den Wahrscheinlichkeiten der einzelnen Substitutionen,

pD = 8/16 · 6/16 · 6/16 · 6/16 = 27/1024.

Um den Schlüssel zu finden, müssen Kryptoanalytiker für jede mögliche Kombination vonK5,5, …, K5,8 und K5,13, …, K5,16 ein (ganzzahliges) Vielfaches m von 1/pD ≈ 38 Paaren von Klartexten U’1 und U’’1 mit Differenz ∆U1 Folgendes tun:

* 1. Das Paar U’1 und U’’1 verschlüsseln,
  2. die Chiffre bis zur S-Box-Eingabe in der vierten Runde mit dem Rundenschlüssel umkehren: K5 = 0000 K5,5, . . . , K5,8 0000 K5,13, . . . , K5,16,

um das Paar υ’4 und υ’’4 mit Differenz υ4 zu erhalten, und dann

* 1. die Differenz ∆υ4 mit ∆U4 vergleichen. Wenn sie übereinstimmen, dann erhöhen wir den Zähler n um 1.

Wenn für eine Kombination von Teilblöcken K5,5, …, K5,8 und K5,13, …, K5,16 die Zählung n/m ≈ pD ergibt, also das Verhältnis zwischen der Anzahl n der übereinstimmenden Paare und der Anzahl m der gesamten Paare nah bei der Wahrscheinlichkeit pD liegt, dann sind diese Teilblöcke wahrscheinlich die Teilblöcke 2 und 4 des von der Chiffre verwendeten Rundenschlüssels 5.

Die Schlussfolgerung, dass wir die richtigen Teilblöcke gefunden haben, basiert auf den folgenden Hypothesen: (a), dass die Differenziale einer Runde unabhängig von den Differenzialen der vorherigen Runde sind und (b), dass eine Wahrscheinlichkeit für übereinstimmende Paare, die nahe an der berechneten liegt, auf den richtigen Schlüssel hinweist.

Beide haben keine strenge mathematische Grundlage, sondern sind nur plausibel, weil jede Runde versucht, so viel wie möglich zu streuen, d. h. den Wert jedes Ausgangsbits praktisch unabhängig von allen Eingangsbits zu machen. Es ist unwahrscheinlich, dass es einen Schlüssel gibt, der anders ist, aber die gleiche Wahrscheinlichkeit reproduziert.

Zusammenfassung

Die Kryptoanalyse, also die Kunst, Verschlüsselungen zu brechen, basiert derzeit auf der Mathematik und wird durch den effizienten Einsatz enormer Rechenleistung angewandt.

Die Sicherheit eines Kryptosystems gegen Brute-Force-Angriffe und die empfohlenen Schlüsselgrößen hängen in erster Linie von dessen Widerstandsfähigkeit gegen die effizientesten (bekannten) Methoden der Kryptoanalyse (unter Verwendung einer Hintertür bzw. Falltür) und dem Rechenaufwand ab, der erforderlich ist, um alle Schlüssel zu überprüfen (unter Verwendung der Vordertür), indem die entschlüsselte Ausgabe auf wahrscheinliche Muster eines Klartextes geprüft wird.

Wenn die geheimen Informationen als kryptografische Hashes gespeichert sind, verwendet ein Brute-Force-Angriff eine Rainbow-Table. Dies ist vielversprechend gegen schnell berechnete Hashfunktionen wie MD4/5 und kann verhindert werden, indem die verwendete Hashfunktion für jedes Passwort einzigartig gemacht und ein Salt hinzugefügt wird.

Das einzige vollkommen sichere Kryptosystem ist das One-Time-Pad, bei dem ein Schlüssel in der Größe des Klartextes zum Klartext hinzugefügt wird. Die asymmetrische Kryptografie nutzt zum Verschlüsseln mathematische Methoden, insbesondere die modulare Arithmetik. Die symmetrische Kryptografie nutzt kunstvollere Methoden der Chiffrierung, welche die Diffusion und Konfusion durch wiederholte Substitution und Permutation maximieren. Die Sicherheit symmetrischer kryptografischer Algorithmen beruht schlicht und einfach auf ihrer Widerstandsfähigkeit gegen jahrelang andauernde Angriffe.

Bei einem Seitenkanalangriff werden Informationen aus der physischen Implementierung einer Chiffriermaschine verwendet, wie z. B. Messungen der Rechenzeit, des Stromverbrauchs, der elektromagnetischen Emissionen oder der Schallemissionen. Ein Rechenzeitangriff misst die Laufzeiten von kryptografischen Verfahren und vergleicht sie mit den geschätzten Laufzeiten, z. B. wird hier ausgenutzt, dass die Laufzeit der Berechnung einer Potenz von der Anzahl der von Null verschiedenen Bits ihres Exponenten abhängt (Bei RSA und Diffie-Hellman ist dies der Schlüssel.).

Die differenzielle Kryptoanalyse wird auf Blockchiffren unter der Annahme eines Angriffs mit ausgewählten Klartexten angewendet. Angreifende senden Paare von unterschiedlichen Klartexten, deren Geheimtexte sie erhalten. Dann untersuchen sie, wie sich Unterschiede in der Eingabe (auf Differenzpfaden) durch das Netzwerk von Verschlüsselungsumwandlungen zu Unterschieden in der Ausgabe fortpflanzen.

# Lektion 6

## Kryptologie und das Internet



#### LERNZIELE

Nach der Bearbeitung dieser Lektion wissen Sie …

… mehr über die Geschichte des Internets.

… über den Zweck von virtuellen privaten Netzwerken und Transport-Layer-Security-Protokollen Bescheid.

… wie E-Mails abgesichert werden.

… wie Sie Internetdomains abrufen, authentifizieren und verschlüsseln können.

1. Kryptologie und das Internet

### Einführung

Vor dem Internetzeitalter war es unvorstellbar, Kryptografie im Alltag einzusetzen. Heutzutage wäre das tägliche Leben im Internet ohne (Public-Key-)Kryptografie unvorstellbar, z. B. für sicheres Online-Shopping. Neben dem Verschlüsseln schafft Kryptografie dort Vertrauen, wo früher Papierdokumente zur Unterzeichnung, Identitätsauthentifizierung, Erteilung von Vollmachten, Lizenzen oder Eigentum verwendet wurden. Die Kryptografie erreicht dies auf sicherere Weise – während sich eine handschriftliche Unterschrift fälschen lässt, ist eine digitale Signatur eindeutig mit dem unterzeichneten Dokument verknüpft.

Um Transaktionen im Internet abzusichern, z. B. beim elektronischen Bankverkehr und Handel oder beim E-Mail-Verkehr, stellt ein kryptografisches Protokoll (wie Transport Layer Security (TLS), früher Secure Sockets Layer (SSL)) Vertrauen her und verschlüsselt den gesamten Datenverkehr. Vertrauen wird aufgebaut, wenn ein Client eine Verbindung zu einem Webserver herstellt. Die Identität des Servers muss dabei garantiert sein, um einen Man-in-the-Middle-Angriff zu verhindern. Bislang stellen die zentralen Stellen digitale X.509-Zertifikate aus. Der Datenverkehr wird verschlüsselt; in einem offenen WLAN, wie z. B. bei Anmeldeportalen an öffentlichen Orten, wird kein Datenverkehr zwischen dem Router (der die Verbindung zum Internet herstellt) und dem Client verschlüsselt, weder vom Router noch vom Client.

### Internetprotokolle

Die verschiedenen Protokolle, die die Verarbeitung der im Internet ausgetauschten Daten standardisieren, können in Schichten eingeteilt werden. Diese sind danach angeordnet, wie stark die verarbeiteten Daten strukturiert sind. Die unteren Schichten dienen der Formatierung der Rohdaten und als Schnittstellen für die oberen Schichten, während die oberen Schichten näher an den Anwendungen des Benutzers sind und abstraktere Daten handhaben. Unter diesen Protokollen sind die beiden wichtigsten (und die zuerst definierten) das Transmission Control Protocol (TCP) und das Internet Protocol (IP). Sie legen fest, wie die Daten formatiert, adressiert, übertragen, weitergeleitet und am Zielort empfangen werden sollen. Das TCP/IP-Protokoll stellt dabei zwar zuverlässig Datenpakete über das Internet zu, garantiert jedoch weder Sicherheit, Vertraulichkeit und Authentizität, noch verwaltet es Sitzungen zwischen einem Client und dem Server (z. B. Unterbrechung, Beendigung und Neustart einer Sitzung).

OSI-Modell  
Das Modell der International Organization for Standardization (ISO) ordnet die verschiedenen Protokolle des TCP/IP-Referenz­modells in sieben Abstraktionsschichten an.

Die meisten Internetanwendungen stützen sich auf eine höhere (Anwendungs-)Schicht, wie z. B. das HTTP-Protokoll für Webserver. Die bekanntesten Schichtenmodelle von Internetprotokollen sind das OSI-Referenzmodell (Open Systems Interconnection) als internationaler Standard für die Architektur von Computernetzwerken und das TCP/IP-Referenzmodell, das in Abschnitt 1.1.3 von Braden (1989) spezifiziert wird. Es wird manchmal als DoD(-Layer)-Modell bezeichnet, weil die Entwicklung in den späten 1960er Jahren mit einer Studie unter der Aufsicht des Verteidigungsministeriums der Vereinigten Staaten (Department of Defense, DoD) begann.

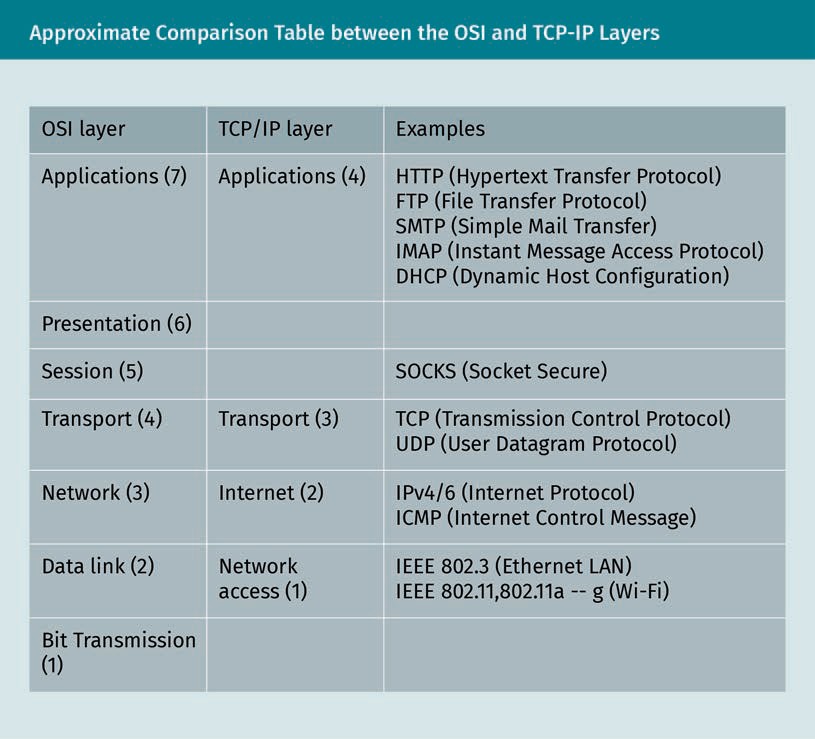
###### Vergleich zwischen dem OSI- und dem TCP/IP-Referenzmodell

Das Internet basiert auf dem TCP/IP-Referenzmodell, das aus nur vier Schichten besteht. Der ISO-Standard mit sieben Schichten ist daher eher eine theoretische Abstraktion als ein praktischer Standard. Im Gegensatz zum ISO-Standardmodell mit sieben Schichten hat sich das TCP/IP-Modell mit vier Schichten durch den praktischen Einsatz entwickelt und ist daher dafür bekannt, dass es, selbst plattformübergreifend, funktioniert. Während also das TCP/IP-Referenzmodell deskriptiv ist, sollte das OSI-Referenzmodell präskriptiv sein. Das OSI-Modell ist ein wunderbares abstraktes Konstrukt, auch wenn das tatsächlich existierende und funktionierende Netzwerk ihm nicht vollständig folgt. Das OSI-Modell ist jedoch von historischem und konzeptionellem Interesse. Da es dem älteren Modell vorausgeht, gelten die gleichen Prinzipien.

Wenn die Modelle (fälschlicherweise) austauschbar verwendet werden, werden beide als Internet-Referenzmodell bezeichnet.

TCP/IP-Referenzmodell

TCP/IP ordnet die verschiedenen Protokolle in vier Schichten an.



1. Die Bitübertragungsschicht (Physical Layer) bietet ausschließlich die Mittel zur Übertragung von rohen Bits. Es werden elektrische Details wie Netzwerkhardware oder Verkabelung spezifiziert.
2. Die Sicherungsschicht (Data Link Layer) enthält keine Protokolle der TCP/IP-Familie, sondern fasst die Protokolle für die Datenübertragung von Punkt zu Punkt zusammen, um verschiedene Teilnetze zu verbinden, wie Ethernet, Point-to-Point Protocol (PPP) oder 802.11 (WLAN).
3. Die Vermittlungsschicht (Network Layer) umfasst alle Protokolle für die Weiterleitung und das Routing von Paketen, d. h. die Bestimmung des nächsten Zwischenziels für ein empfangenes Paket und die Weiterleitung des Pakets dorthin. Der Kern dieser Schicht ist das Internetprotokoll in seiner Version IPv4 oder IPv6, das für Aufgaben wie Segmentierung, Fehlererkennung und -korrektur zuständig ist.
4. Die Transportschicht (Transport Layer) umfasst alle Protokolle zum Aufbau, zur Priorisierung, zur Aufrechterhaltung und zur Beendigung der Kommunikation zwischen zwei Computern in einem Netzwerk und zur Überprüfung, ob die von einem Computer zum anderen gesendeten Daten ihr Ziel korrekt erreicht haben. Am wichtigsten sind das Transmission Control Protocol (TCP) zum zuverlässigen Senden von Datenströmen, aber auch unzuverlässige Protokolle wie das User Datagram Protocol (UDP).
5. Die Sitzungsschicht (Session Layer) verfolgt den Fortschritt der Datenübertragungen für den Ruhezustand der Sitzung (Checkpointing), die Aussetzung, die Beendigung und den Neustart, z. B. nach einem Übertragungsfehler oder einer Unterbrechung.
6. Die Darstellungsschicht (Presentation Layer) ist für die Formatierung von Nachrichten zuständig, d. h. für die Umwandlung von Daten in ein für eine Anwendung wie einen Webserver verständliches Format, z. B. für die Ver- und Entschlüsselung (oder die Umwandlung von Codesätzen, z. B. von ISO Latin-1 in UTF-8).
7. Die Anwendungsschicht (Application Layer) umfasst alle Protokolle zum Austausch anwendungsspezifischer Daten über das Netzwerk, z. B. die Mozilla HTML-Engine, die von Firefox und Chrome verwendet, oder das SMTP-Protokoll, das von E-Mail-Programmen genutzt wird.

Verschiedene Anwendungsprotokolle (z. B. HTTP, FTP oder IMAP) implementieren die Funktionen der Schichten 5, 6 und 7 jedoch unterschiedlich und trennen diese Schichten nicht streng voneinander. Daher fassen Praktizierende, wie z. B. Netzwerktechnikfachkräfte, alle diese Schichten als Schicht 5+ bzw. als die Anwendungsschicht zusammen, wie es auch das TCP/IP-Referenzmodell macht.

### IPsec

Ein privates IP-Netzwerk ist ein Netzwerk, das in der Regel aus lokalen Netzwerken (LANs) in privaten und kommerziellen Umgebungen besteht, deren Computer IP-Adressen haben, die in die von IPv4 spezifizierten Bereiche fallen (in RFC 1918, analog in IPv6):

* Von 10.0.0.0 bis 10.255.255.255,
* von 172.16.0.0 bis 172.31.255.255 und
* von 192.168.0.0 bis 192.168.255.255.

Diese Adressen können ohne Genehmigung einer Internet-Registry verwendet werden. Wir verwenden den Begriff „Computer“ oft als Bezeichnung für einen Endpunkt des Netzwerks. Dieser Begriff schließt auch Tablets, Smartphones und andere Netzwerkgeräte ein.

Ein virtuelles privates Netzwerk (VPN) wird verwendet, um Netzwerke zwischen zwei Unternehmen (zum Aufbau eines Extranets), verschiedene Netzwerke innerhalb desselben Unternehmens (Intranets) und einen einzelnen Client über das Internet mit einem Intranet zu verbinden (Fernzugriff), was die häufigste Verwendung von VPNs für den Endbenutzer ist.

Um den Schutz des Datenverkehrs zu gewährleisten, kommen bei den Verbindungen zwischen zwei geschlossenen Netzwerken Authentifizierung und Verschlüsselung zum Einsatz. Die Parteien authentifizieren sich gegenseitig mit einem zuvor geteilten Geheimnis wie einem Passwort oder einem Zertifikat. Dann werden die ausgetauschten Daten an den Endpunkten verschlüsselt und entschlüsselt.

###### Gateway

Ein Gateway ist vereinfacht gesagt ein Gerät, das zwei Netzwerke miteinander verbindet, wie z. B. ein Router, der den Netzwerkverkehr weiterleitet. Die Internetverbindung zu Hause verwendet in der Regel einen Router, um Internet-Datenpakete an die Geräte im Haus zu liefern. Das erste Gerät, das eine Verbindung zum Internet herstellt, wird auch als Standard-Gateway bezeichnet. Der Konvention nach hat das Gateway die niedrigste IP-Adresse im Subnetz, eine ursprüngliche Adressgruppe.

Eine Firewall filtert Datenpakete, um ein inneres (privates) Netzwerk vor einem äußeren (öffentlichen) Netzwerk zu schützen. Sie befindet sich in der Regel auf einem Gateway oder möglicherweise als Software auf dem Computer der Benutzer, z. B. als Teil des Betriebssystems wie die Firewall bei Microsoft Windows.

Da viele Verbrauchergeräte Router und Firewall in einem sind, werden die drei Begriffe Gateway, Router und Firewall manchmal synonym verwendet.

###### NAT

Network Address Translation (NAT) ermöglicht es einem Internet-Protocol-(IP)-Netzwerk, öffentliche IP-Adressen in private zu übersetzen. NAT fängt sowohl eingehende als auch ausgehende IP-Datenpakete ab und ändert die Quell- oder Zieladresse im Header des Pakets (und passt die Prüfsummen an), um öffentliche IP-Adressen in private zu übersetzen. Normalerweise wird NAT auf Gateways an der Netzwerkgrenze implementiert. NAT sichert das private Netzwerk, indem es verhindert, dass externe Computer auf den IP-Bereich des internen Netzwerks zugreifen. In der Regel werden eingehende Verbindungen nur als Antwort auf ausgehende Verbindungen (von einem Computer innerhalb des privaten Netzwerks) zugelassen (Schell & Martin, 2006).

###### Endpunkte

Das VPN kann folgende Verbindungen nutzen:

VPN

Ein privates Netzwerk, das aus zwei oder mehr (räumlich getrennten) geschlossenen Netzwerken besteht, die über ein offenes Netzwerk (wie das Internet) verbunden sind, wird VPN genannt.

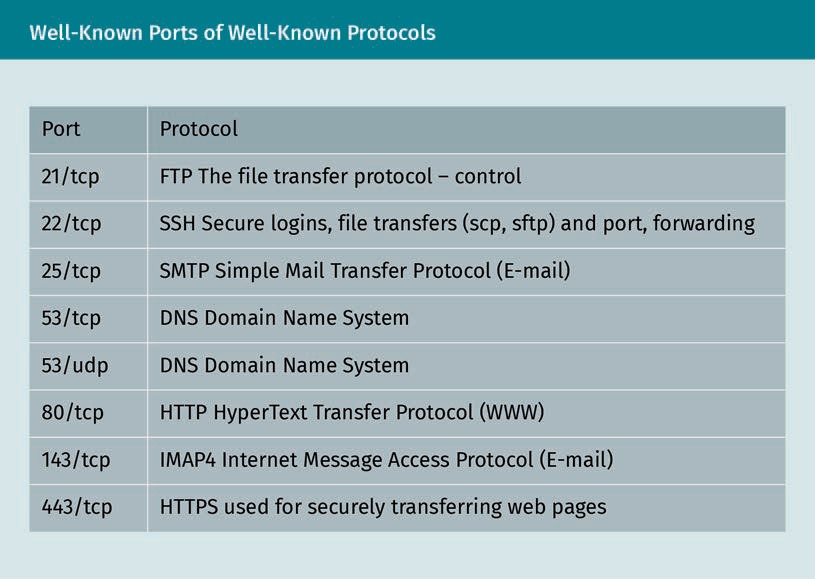
* Gateway-to-Gateway. Dies ist der einfachste Fall, bei dem beide Endpunkte mit fest programmierten Adressen und Ports direkt erreichbar, also z. B. im selben LAN oder beide öffentlich zugänglich sind.
* Host-to-Gateway (Fernzugriff). Dies ist der übliche Fall, wenn sich eine der beiden Parteien hinter einem Gateway befindet, z. B. das Notebook von Außendienstmitarbeitern oder der Computer von Heimanwendern. Sie befinden sich hinter einem (öffentlichen oder privaten) Router, der über das Internet mit einem öffentlichen Server, wie dem zentralen Server des Unternehmens, der unter einer festen IP-Adresse oder einer Domain im Internet erreichbar ist, verbunden ist.
* Host-to-Host (Peer-to-Peer). NAT ermöglicht es einem IP-Netzwerk, öffentliche IP-Adressen in private Adressen von Hosts hinter einem Gateway zu übersetzen. Dabei werden oft viele private IP-Adressen auf eine einzige öffentliche Adresse abgebildet, indem ihnen verschiedene Ports zugewiesen werden (Port Address Translation, PAT). Dieser Fall ist seltener.

###### Zugrunde liegende Transportprotokolle

Eine VPN-Verbindung verwendet in der Regel entweder das TCP- oder das UDP-Protokoll. Das TCP-Protokoll ist zuverlässiger, aber langsamer. Es ist nützlich, um den VPN-Verkehr so zu verschleiern, dass er wie normaler HTTPS-Datenverkehr aussieht, wodurch die Gefahr, blockiert zu werden, verringert wird. Bei UDP werden die Pakete ohne jegliche Bestätigung gesendet, d. h. ohne Garantie, dass die gesendeten Daten korrekt angekommen sind. Diese Aufgabe wird auf die Anwendungen verlagert, die das Protokoll verwenden, z. B. VOIP-Anwendungen. Das User Datagram Protocol (UDP) ist die schnellere und zu bevorzugende Option für die Verbindung eines VPNs, wenn die beiden oben genannten Einschränkungen, die TCP umgeht, nicht zutreffen.

Ports

Ein Port ist ein Softwarekonzept. Es ist eine Zahl (zwischen 0 und 65535), die für einen Datenkanal in und aus einem Computer in einem Netzwerk steht. Im TCP/IP-Referenzmodell verbindet er das TCP oder das UDP mit einem übergeordneten (Anwendungs-)Protokoll.



###### IPsec

Internet Protocol Security (IPsec) ist eine Sammlung von Protokollen, welche die IP-Kommunikation durch Authentifizierung und optionale Verschlüsselung jedes Pakets über öffentliche und unsichere Netzwerke absichert. Es wird hauptsächlich für VPNs verwendet und ist, zumindest auf dem Geschäftskundenmarkt, das etablierteste Protokoll (Friedl, 2005).

Zwischen Gateways wird in der Regel der Tunnelmodus über das Internet verwendet, der zwei Netzwerke zwischen zwei Gateways miteinander verbindet. Die Endgeräte selbst, die über die beiden Netzwerke verbunden sind, müssen IPsec nicht unterstützen. Die Verbindungssicherheit wird nur auf der Teilstrecke zwischen den beiden Gateways gewährleistet. Es wird ein neuer externer IP-Header verwendet. Die IP-Adressen der beiden Kommunikationsendpunkte befinden sich im inneren geschützten IP-Header.

Der Transportmodus wird normalerweise verwendet, wenn das endgültige Ziel kein Gateway ist. Er verwendet einen zusätzlichen IPsec-Header zwischen dem IP-Header und den transportierten Daten. Er ist weniger sicher als der Tunnelmodus.

Tunnelmodus

In diesem Modus wird das komplette IP-Paket verschlüsselt.

Transportmodus Dieser Modus verschlüsselt den Datenteil, aber belässt die ursprünglichen IP-Adressen im Klartext.

###### IPsec-Protokolle

IPsec besteht im Wesentlichen aus den Protokollen Internet Key Exchange (IKE) und Encapsulated Security Payload (ESP). IKE ist die technische Implementierung des Internet Security Association and Key Management Protocol (ISAKMP) Framework. IKE verwendet UDP auf Port 500 für den ersten Schlüsselaustausch und Port 50 für die IPSEC-verschlüsselten Daten. ESP (für NAT-Traversal) verwendet UDP-Port 4500 und TCP-Port 10.000.

IKE nutzt UDP auf Port 500 für den ersten Schlüsselaustausch und erstellt einen gemeinsamen geheimen Schlüssel entweder manuell durch den Austausch von öffentlichen Schlüsseln oder automatisch durch die Zertifikate eines vertrauenswürdigen Zertifikatservers. Zur Verwendung mit VPNs für maximale Sicherheit wird IKEv2 mit IPSec gekoppelt. Im Vergleich zu anderen VPN-Protokollen ist der wichtigste Vorteil von IKEv2 die Fähigkeit, die Verbindung nach einem VPN-Verbindungsabbruch rasch wiederherzustellen.

ESP (spezifiziert in RFC 3948) verschlüsselt kritische Informationen, indem es den TCP- oder UDP-Datenabschnitt mit einem ESP-Header kapselt. Im Tunnelmodus, in dem das gesamte ursprüngliche IP-Paket mit einem neu hinzugefügten Paket-Header eingekapselt wird, wird der ESP-Schutz für das gesamte innere IP-Paket (einschließlich des inneren Headers) gewährleistet, während der äußere Header (einschließlich aller äußeren IPv4-Optionen oder IPv6-Erweiterungs-Header) ungeschützt bleibt (Manualzz, 2020, S. 63).

###### Andere VPNs

OpenVPN ist ein beliebtes, nicht standardisiertes Open-Source-VPN-Protokoll über dem UDP- oder TCP-Protokoll, das TLS für den Schlüsselaustausch und OpenSSL für die Verschlüsselung verwendet. Es unterstützt dynamisch zugewiesene IP-Adressen hinter NAT-Gateways. Bei Verwendung von TLS ist es nicht kompatibel mit IPSec. Es ist als Software implementiert, die dank der Verwendung von OpenSSL stabil und sicher ist, auf allen gängigen Betriebssystemen wie Windows, Linux, macOS, Solaris, OpenBSD und Android läuft und auf Tausende von Clients skaliert werden kann.

WireGuard ist eine minimalistische und moderne Open-Source-VPN-Software, die in Linux-Kernel 5.5 (und höher) integriert ist. Sie ist einfach, benutzerfreundlich und leicht einzurichten. Dank neuester kryptografischer Algorithmen und bewährter Verfahren ist die Software sicher. Sie hat einen kurzen Quellcode (anfangs etwa 4000 Zeilen im Vergleich zu Hunderttausenden bei OpenVPN) und erlaubt nur UDP auf IPv4 oder IPv6. Es fehlt die TCP-Unterstützung. WireGuard überprüft weder die Identität des Servers anhand von Zertifikaten noch kann es IP-Adressen dynamisch verwalten. Aus diesem Grund verzichten viele Anbieter auf den Einsatz von WireGuard, weil sie befürchten, den Schutz der Daten ihrer Kunden zu gefährden (trotz der Zero-Log-Richtlinien).

###### VPN-Software

Um ein gemeinsam genutztes VPN sicher nutzen zu können, müssen wir seinen Betreibern und seinen Benutzern vertrauen. Auch wenn sich die meisten Benutzer gut benehmen, kann es sein, dass eine Person dies nicht tut, weswegen Strafverfolgungsbehörden schließlich den gesamten Netzwerkdatenverkehr ins Visier nehmen könnten. Um ein eigenes VPN einzurichten, gibt es verschiedene Software-Optionen wie SoftEther („Software Ethernet“) VPN. SoftEther ist kostenlos und quelloffen (Open Source) und kann viele gängige VPN-Protokolle wie SSL-VPN (HTTPS), OpenVPN und IPsec verwenden. Die Software unterstützt auch NAT-Traversal über SSL-VPN-Tunneling, um VPN-Server hinter Firewalls zu betreiben, indem sie HTTPS verwendet, so dass selbst Deep Packet Inspection (die sowohl die Metadaten als auch die Daten untersucht) die VPN-Transportpakete von SoftEther nicht erkennen kann.

### Transport Layer Security

TLS und sein Vorgänger SSL verschlüsseln Daten in beide Richtungen und garantieren die Identität von Server und Client. Ursprünglich von der Netscape Corporation entwickelt, wird es heute von allen großen Browsern unterstützt und ist das am häufigsten verwendete Sicherheitsprotokoll im World Wide Web (Lee et al., 2007)

Für Anwendungsprogrammierende bieten beide ein Protokoll, auf das fast wie auf einfaches TCP zugegriffen werden kann. Für die Nutzer stellen sie einen sicheren Kanal über das Internet her, über den deren private Daten wie Kreditkarten- oder Bankkontonummern mit Hilfe von Zertifikaten, d. h. öffentlichen, symmetrischen Schlüsseln (gekennzeichnet durch ein kleines Vorhängeschloss in der Adressleiste des Webbrowsers), sicher übertragen werden können. TLS befindet sich auf der Transportschicht (im OSI-Referenzmodell Schicht 4), da es eine zuverlässige Datenübertragung erfordert.

###### X.509-Zertifikate

Das X.509-Zertifikat dient der Authentifizierung und Verschlüsselung. Diese sind hierarchisch organisiert und vermitteln Vertrauen von der oberen zur unteren Ebene. Diejenigen, die an der Spitze stehen und denen bedingungslos vertraut wird, werden als Stammzertifizierungsstellen bezeichnet. In der Praxis wird dieses unbedingte Vertrauen durch den Einsatz ihrer selbstsignierten Zertifikate, z. B. als Teil der Installation eines Internetbrowsers erreicht.

Die Signatur eines X.509-Zertifikats ist die Verschlüsselung des Hashes der Verkettung (V,SN,AI,CA,TA,A,KA) mit dem privaten Schlüssel, wobei Folgendes gilt:

* V ist Version X.509,
* SN ist die Seriennummer des Zertifikats,
* AI ist die Identifikationsnummer des Algorithmus,
* CA ist der Name der Zertifizierungsstelle,
* TA ist die Gültigkeitsdauer des Zertifikats,
* A ist der Name des Subjekts, und
* KA ist der öffentliche Schlüssel des Subjekts.

Das System der hierarchischen Stellen wurde geschaffen, um Vertrauen durch Maschinen zu schaffen und den Schlüsselaustausch zu automatisieren.

TLS/SSL

Die kryptografischen Übertragungsprotokolle TLS/SSL bieten Authentifizierung, Vertraulichkeit und Authentizität für Daten, die über einen zuverlässigen Transportweg, in der Regel TCP, übertragen werden.

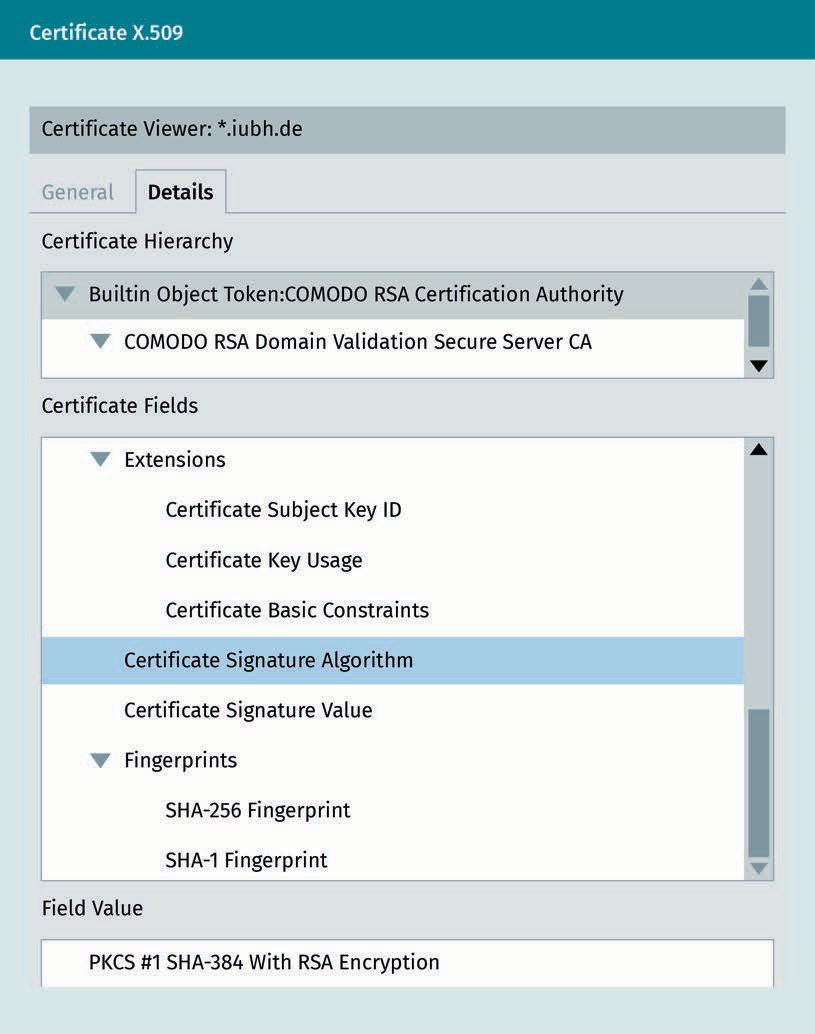
X.509-Zertifikat Diese Datei ist von einer Zertifizierungsstelle unterzeichnet und enthält den Namen, die Adresse und den öffentlichen Schlüssel der Website.

Das Vorhängeschloss in der Adressleiste des Browsers zeigt die Sicherheitsstufe an. Während „Let's Encrypt“-Zertifikate nur den Besitz der Domain per E-Mail verifizieren, bieten Unternehmen ein Extended-Validation-Zertifikat ((EV-Zertifikat) an, das die Identität der Besitzer verifiziert. Bei der Ausstellung eines herkömmlichen Zertifikats, z. B. eines kostenlosen Zertifikats von „Let's Encrypt“, erfolgt die Verifizierung völlig automatisch und ohne persönliche Überprüfung der Identität. Um das Zertifikat zu erhalten, genügt der Zugang zur Domain, z. B. durch Hochladen einer empfangenen Datei. Bei der Ausstellung eines EV-Zertifikats erfolgt die Überprüfung des Website-Inhabers persönlich.

###### Handshake

Das Herzstück des TSL/SSL-Protokolls ist der Handshake, mit dem die Verschlüsselung der Sitzung eingerichtet wird. Die maßgebliche Referenz ist RFC5426, eine Schritt-für-Schritt-Anleitung, die jedes einzelne ausgetauschte Byte verfolgt (Driscoll, 2019).

Die folgenden Schritte sind die ersten Schritte zwischen einem Client und dem Server (z. B. einer E-Commerce-Website) zum Aufbau einer verschlüsselten Verbindung (z. B. zum Empfang von Kreditkartendaten vom Client).



In der obigen Abbildung ist der Server www[.]iubh[.]de. Das Zertifikat ist (a) von der zwischengeschalteten Stelle Comodo RSA Domain Validation Secure Server CA signiert, (b) von der Stammzertifizierungsstelle Comodo RSA Certification Authority signiert und (c) selbstsigniert.

Der Client erstellt ein Vorgeheimnis (Pre-Secret, eine zufällige (Pseudo-)Zahl mit einer Größe von 48 Byte), verschlüsselt es mit dem öffentlichen Schlüssel (unter Verwendung des ursprünglich vereinbarten asymmetrischen Algorithmus) und sendet es an den Server. Der Server entschlüsselt dann das Vorgeheimnis mit seinem privaten Schlüssel.

Der Client und der Server berechnen das Geheimnis (Master Secret), eine Zahl mit einer Größe von 48 Bytes, durch eine Funktion PRF, master\_secret = PRF(pre\_master\_secret, ClientHello.random + ServerHello.random), die als Eingabe das Pre-Secret und die Nonces verwendet, die während des „Hello“ vom Client-Server übermittelt wurden.

Der Client und der Server leiten vier symmetrische Schlüssel für den ursprünglich vereinbarten Algorithmus ab, z. B. für AES, die jeweils 16 Byte lang sind und das Geheimnis bilden. Die ersten beiden dienen zur Überprüfung der Datenauthentizität, die letzten beiden zur Verschlüsselung der Daten.

### Sichere E-Mail

Die meisten Internetprotokolle, darunter auch die für den E-Mail-Versand, wie POP3, IMAP und SMTP, ignorierten zunächst Sicherheitsbedenken und tauschten alle Daten im Klartext aus. Seitdem sind verschiedene Verfahren entstanden, um die Daten entweder nur während der Übertragung (z. B. TLS) zu verschlüsseln, was bequemer ist, weil es einfacher einzurichten und zu verwenden ist, oder als Ende-zu-Ende-Verschlüsselung (z. B. S/MIME oder OpenPGP). Im Allgemeinen bietet die Ende-zu-Ende-Verschlüsselung ein höheres Sicherheitsniveau. Wie der Name schon sagt, verschlüsselt und entschlüsselt es die Daten auch an den Endpunkten: den Absendern und den Empfängern. Eine mit Ende-zu-Ende-Verschlüsselung versandte E-Mail ist für die Mailserver unlesbar, und kein Dritter kann die E-Mail auf das Vorhandensein von Malware scannen. Stattdessen muss dies direkt nach der Entschlüsselung auf dem Computer der Benutzer geschehen.

Ende-zu-Ende-Protokolle sind jedoch mit zusätzlichem Aufwand verbunden und bieten dennoch nur einen teilweisen Schutz. Sie erfordern, dass die Benutzer Paare von öffentlichen und privaten Schlüsseln einrichten und die öffentlichen Schlüssel veröffentlichen. Außerdem schützen sie nur den Inhalt der E-Mail, aber keine Metadaten, so dass Dritte weiterhin feststellen können, wer wem eine E-Mail geschickt hat.

###### TLS

STARTTLS

Hierbei handelt es sich um ein TLS-über-Klartext-Protokoll ähnlich wie IMAP4 und POP3, wie in RFC2595 beschrieben.   
Es ermöglicht E-Mail-Servern, alle zwischen den Servern sowie zwischen Servern und Clients ausgetauschten Daten zu verschlüsseln.

Das gebräuchlichste E-Mail-Protokoll für die Verschlüsselung beim Transport ist STARTTLS. Die Verifizierung von Zertifikaten ist jedoch optional, da eine fehlgeschlagene Verifizierung als weniger schädlich angesehen wird als eine fehlgeschlagene E-Mail-Zustellung, d. h. die meisten E-Mails, die über TLS zugestellt werden, sind nur opportunistisch verschlüsselt.

Die Verwendung von STARTTLS ist unabhängig davon, ob der Inhalt der E-Mail verschlüsselt ist. Angreifende können den verschlüsselten E-Mail-Inhalt nicht sehen, aber er wird entschlüsselt und ist somit bei jedem zwischengeschalteten E-Mail-Vermittler sichtbar. Mit anderen Worten: Die Verschlüsselung findet zwischen den Servern statt, aber nicht zwischen Absendern und Empfängern.

###### S/MIME

Die Secure Multipurpose Internet Mail Extension (S/MIME) ist ein Protokoll, das die Verschlüsselung mit öffentlichen Schlüsseln und das Signieren von E-Mails standardisiert. Die IETF-Spezifikation von S/MIME ist eine Weiterentwicklung der PEM-Spezifikation (Privacy Enhanced Mail) aus den 1990er Jahren. Während ursprünglich die RSA-Verschlüsselung mit öffentlichen Schlüsseln verwendet wurde, wird seit RFC5753 (Ramsdell, 2009) auch ECC eingesetzt.

Die meisten E-Mail-Programme, wie Microsoft Outlook oder Mozilla Thunderbird, unterstützen sichere S/MIME-E-Mails. Vor der Verwendung muss ein individuelles Schlüsselzertifikat installiert werden. Erstellen Sie vor der Installation ein (mehrere) Schlüsselpaar(e) und senden Sie es an die (interne oder öffentliche) Zertifizierungsstelle, damit diese es/sie signiert. Nach erfolgreicher Verifizierung erstellt die Zertifizierungsstelle ein Zertifikat des Schlüssels, indem sie ihn mit ihrem privaten Signaturschlüssel signiert.

###### Web of Trust

Das Web of Trust (Netz des Vertrauens) basiert auf der Ausbreitung des persönlichen Vertrauens. Sein Hauptvorteil besteht darin, dass es sich um ein Peer-to-Peer-System handelt, und es damit unabhängig von einer bestimmten dritten Stelle ist. Allerdings muss es persönlich gewartet werden, was seine größte Unannehmlichkeit darstellt. In der Praxis ist es realistischer, anstelle des Web of Trust das Vertrauen über einen anderen unpersönlichen Kanal herzustellen (durch den Austausch der Fingerabdrücke der öffentlichen Schlüssel), z. B. per Post, Telefon oder Messenger.

###### OpenPGP

Da Vertrauen eine persönliche Angelegenheit ist, ist das automatische, bedingungslose Vertrauen der Benutzer in die Zertifizierungsstellen (hauptsächlich Unternehmen) unbefriedigend. Dennoch verwenden nur wenige Menschen OpenPGP. Die meisten halten den konkreten persönlichen Aufwand für die Wartung der Schlüssel, der eine Einschätzung des Vertrauens der Nutzer in die Schlüssel voraussetzt, für unverhältnismäßig groß im Vergleich zu dem abstrakten Vorteil einer größeren Privatsphäre und Sicherheit.

###### Beispiele für OpenPGP-Programme

Wir stellen einige Programme vor, die das OpenPGP-Protokoll verwenden, wie z. B.:

* Das Kommandozeilenprogramm GPG zum Erstellen von Schlüsseln und zum Ent- bzw. Verschlüsseln und Signieren/Authentifizieren von Schlüsseln,
* die Erweiterung Enigmail für das E-Mail-Programm Thunderbird, und
* die Erweiterung Mailvelope für Firefox und Chrome zur Verschlüsselung von E-Mails in Web-Oberflächen.

S/MIME

Ein E-Mail-Verschlüsselungsprotokoll, das Zertifizierungsstellen nutzt, um Vertrauen für die Schlüsselverteilung zu schaffen.

OpenPGP

Dieses E-Mail-Verschlüsselungsprotokoll nutzt für die Schlüsselverteilung das Web of Trust.

GnuPG

Gnu Privacy Guard (GnuPG oder GPG) wurde geschrieben, um der Öffentlichkeit offene und freie kryptographische Methoden bereitzustellen. Es ist ein Befehlszeilenprogramm zum Ver- und Entschlüsseln von Daten (z. B. E-Mails) und zum Erstellen und Überprüfen digitaler Signaturen, zur Sicherstellung der Authentizität von Daten. GnuPG bildet die Grundlage für die kryptografische Funktionalität vieler kryptografischer Anwendungen mit einer grafischen Benutzeroberfläche.

Enigmail

Das Programm Enigmail ist eine Erweiterung des grafischen E-Mail-Programms Thunderbird um diesem Funktionen zum Ver- und Entschlüsseln sowie zum Signieren und Überprüfen von E-Mail-Signaturen hinzuzufügen. Benutzer können diese Funktionen über Schaltflächen in Thunderbird selbst aufrufen. Enigmail nutzt GnuPG als Fundament der Implementierung. Version 78 von Thunderbird, herausgekommen im Sommer 2020, verfügt über Funktionen für die E-Mail-Verschlüsselung und digitale Signaturen unter Verwendung des eingebauten OpenPGP-Standards und ersetzt das Enigmail-Add-on und die Abhängigkeit von GnuPG, dessen Installation für Anfänger zu mühsam war (Snipes, 2019).

Mailvelope

Mailvelope ist eine von der Mailvelope GmbH entwickelte Erweiterung für die Browser Firefox und Chrome, welche die Web-Oberflächen gängiger E-Mail-Anbieter um Ver- und Entschlüsselungsfunktionen erweitert. Mailvelope ist Open Source und basiert auf OpenPGP.js, einer OpenPGP-Bibliothek für JavaScript. Es ist sehr benutzerfreundlich, aber der Komfort geht auf Kosten der Sicherheit, weswegen es sicher ist, ein E-Mail-Programm wie Thunderbird zu verwenden. Mailvelope ist potenziell anfällig für Cross-Site-Scripting-Angriffe (XSS-Angriffe), bei denen eine Website auf lokale Daten zugreift, die für eine andere gespeichert sind.

###### Automatischer Schlüsselaustausch

Programme wie die Erweiterungen AutoCrypt (Thunderbird), prettyeasyprivacy (Outlook) oder der Messenger Delta-Chat (Android) bieten nur „opportunistische Sicherheit – die meiste Zeit etwas Schutz“, also Schutz vor passiven, aber nicht aktiven Mithörern, bei denen die Verschlüsselung die Nutzer nur so lange schützt, wie sich niemand für sie interessiert (Dukhovni, 2014). Ein solches Programm ist gerade wegen der fehlenden Verifizierung der Besitzer des privaten Schlüssels, der dem öffentlichen Schlüssel entspricht, anfällig für den MITM-Angriff, bei dem sich die Angreifer zwischen die beiden kommunizierenden Parteien schaltet. Es ist durchaus möglich, den Namen einer anderen Person in einer E-Mail oder einem WhatsApp-Konto zu verwenden. Die Verschlüsselung der Kommunikation verhindert nur, dass sie von Dritten gelesen werden kann, bietet aber keine Gewähr für die Identität der anderen Kommunikationspartner. Um diesen Angriff zu unterbinden, müssen wir persönlich oder über einen anderen Kanal, wie z. B. ein Telefon, den Fingerabdruck (eine kryptografische Prüfsumme) des öffentlichen Schlüssels des anderen Kommunikationspartners überprüfen. Das ist mühsam, aber unvermeidlich.

### Secure DNS

Das Domain Name System (DNS) ist eine verteilte Datenbank, die einem Telefonbuch des Internets entspricht. DNS übersetzt für Menschen lesbare alphabetische Internetdomainadressen, wie z. B. https://www [.]iubh[.]de, in computerfreundliche numerische IP-Adressen, wie z. B. 194.6.193.105 (diese können mit dem Unix-Befehlszeilenprogramm nslookup ermittelt werden). Diese Internetdomainadresse besteht aus dem Domainnamen eines Rechners, gefolgt von einer Top-Level-Domain (TLD), getrennt durch Punkte – iubh.de hat z. B. den Domainnamen iubh und die TLD de.

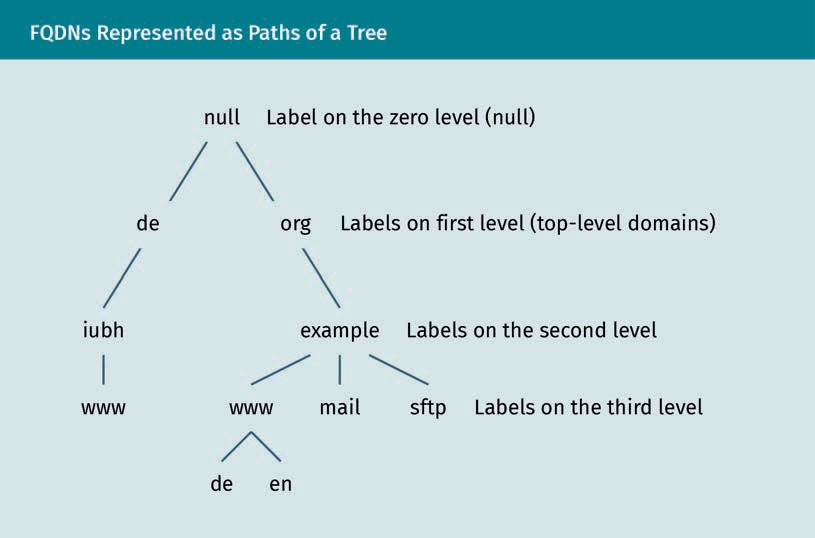
Ein vollständig qualifizierter Domainname (Fully Qualified Domain Name, FQDN) ist ein weltweit eindeutiger Name zur Adressierung einer IP-Adresse im Internet, der nach den von der Internet Corporation for Assigned Names and Numbers (ICANN) festgelegten Regeln frei gewählt werden kann.

Jeder FQDN endet auf eine Top-Level-Domain (z. B. „.de“). Eine Subdomain („www“) kann einem FQDN („iubh.de“) durch Voranstellen eines Punktes hinzugefügt werden. Übliche Subdomain-Bezeichnungen sind [„www](http://www/).“ für Webserver und „mail.“, „smtp.“, „pop3.“ und „imap.“ für (ausgehende und eingehende) Mailserver. Jede Domain (oder Bezeichnung wie „iubh“) darf höchstens 63 Zeichen enthalten. Der gesamte FQDN [(„www.iubh.](http://www.iubh.de/)de“) ist auf 255 Zeichen begrenzt.

Die FQDNs werden durch die Einträge von Nameservern mit der IP-Adresse in Verbindung gebracht. Die Bezeichnungen der FQDNs werden als Knoten eines Baumes dargestellt. Ein FQDN ist dann ein Pfad des Baums. Der oberste Knoten ist die Null- oder Wurzelbezeichnung, die einen leeren Namen darstellt. Unterhalb des höchsten Knotens befinden sich diejenigen, die eine Top-Level-Domain („de“) darstellen. Unterhalb der Knoten auf der ersten Ebene befinden sich die Knoten, die eine Domain darstellen („iubh“). Unterhalb der Knoten auf der zweiten Ebene befinden sich die Knoten, die eine Subdomain der Domain („www“) darstellen.

DNS

Das DNS ist eine Datenbank aller Internetdomainnamen, die auf hierarchisch organisierten Servern über das Internet verteilt ist.



DNS-Server verwenden eine Reihe von Datenbanken, die auf Servern im Internet verteilt und hierarchisch organisiert sind. DNS nutzt das UDP- oder TCP-Protokoll auf Port 54 (Mockapetris, 1987).

Eine DNS-Zone ist die Teilmenge, oft eine einzelne Domain (wie iubh.de), der DNS-Hierarchie, die durch eine Zonentextdatei beschrieben wird. Eine Liste von Einträgen, die Resource Records (RRs) genannt werden, ordnet einen FQDN seiner IP-Adresse zu. Das ursprünglich für die Software Berkeley Internet Name Domain (BIND) spezifizierte Zonenformat wird von den meisten DNS-Servern verwendet. Ein Root-Nameserver (Root-Server) ist der Nameserver, der alle FQDNs einer Top-Level-Domain (TLD) wie „.com“ auflöst.

###### DNSsec

DNSsec Dieses Protokoll authentifiziert die Auflösungen eines Domainnamens in eine IP-Adresse.

Die DNSsec-Auflösung wird bei der Registrierung von dem für die Zonendatei zuständigen DNS-Server signiert. Dies gewährleistet die Authentizität der Auflösung, aber weder die Vertraulichkeit noch die Authentifizierung des DNS-Servers. Zu diesem Zweck wurden weitere sichere DNS-Protokolle wie DNScrypt, DNS-over-TLS und DNS-over-HTTPS entwickelt.

Protokoll

DNSsec ist ein Protokoll, das DNS erweitert. Es umfasst alle Einträge (Resource Records oder RRs) eines DNSSEC-Pakets. RFC 4034 spezifiziert die Ergänzung der vier RRs DNSKEY, RRSIG, NSEC und DS.

Der DNSKEY-Eintrag übermittelt einen öffentlichen Schlüssel zwischen dem Resolver und dem Nameserver. Dieser öffentliche Schlüssel ist derjenige, der mit dem privaten Schlüssel verbunden ist, mit dem der Server der Zertifizierungsstelle Hashes von RRSET-Datensätzen signiert. Der Resolver verwendet den öffentlichen Schlüssel im DNSKEY-Eintrag, um die Nachricht des Servers der Zertifizierungsstelle zu authentifizieren, indem er seine Signatur überprüft. Der RRSIG-Datensatz enthält den Namen des Unterzeichners und die Signatur des vom Server der Zertifizierungsstelle gesendeten Datensatzes, also die Signatur, die der Resolver später verifizieren wird. Es gibt einen RRSIG-Datensatz für jeden Zonendatensatz in der signierten Zonendatei.

Signaturen

DNSsec authentifiziert jeden RR mit einer digitalen Signatur. Der RR-Eigentümer ist der primäre autorisierende Nameserver, wie er im Eintrag Start-of-Authority RR (SOA-RR) der Zonendatei angegeben ist.

* 1. Für jede Zone wird ein ZSK, ein Paar aus einem öffentlichen und einem privaten Schlüssel, erzeugt.
  2. Der öffentliche Schlüssel wird der Zonendatei im DNSKEY-RR hinzugefügt.
  3. Jeder RR der Zone wird dann gehasht und mit dem privaten Zonensignierschlüssel signiert.
  4. Der Resolver erhält den öffentlichen Schlüssel des autorisierenden Nameservers über den DNSKEY-Eintrag.
  5. Der Resolver kann dann die Authentizität des übergebenen RR überprüfen, indem er prüft, ob der entschlüsselte Hash des RR mit dem öffentlichen Schlüssel (der dem privaten Schlüssel des primären Nameservers entspricht) mit dem Hash des RR übereinstimmt.

Stimmt kein Server mit der Anfrage überein, beweist DNSsec, dass es keinen solchen RR gibt, und verwendet einen neuen Typ von Eintrag, der den Namen der ersten (in alphabetischer Reihenfolge) existierenden Domain übermittelt.

###### Virtuelles Hosting

Beim virtuellen Hosting können mehrere DNS-Namen auf einem einzigen Server (in der Regel ein Webserver) unter derselben IP-Adresse gehostet werden.

Die Unterscheidung zwischen der Domain auf dem Server, die angefordert wurde, erfolgt auf Anwendungsebene. Die angeforderte Domain wird immer unverschlüsselt in folgender Form gesendet:

* + - Im SMTP-Protokoll während des SMTP-Handshakes.
    - Im HTTP-Protokoll durch das HTTP-Header-Feld Host, das vom Client gesendet wird, wie es seit HTTP/1.1 obligatorisch ist (was heute üblich ist, aber dessen Einbeziehung vom Server nicht erzwungen werden kann).
    - Im HTTPS-Protokoll müssen für eine geeignete Zuordnung von Zertifikaten zu Domains sowohl der Client als auch der Server die Server Name Indication (SNI) unterstützen

###### Reverse DNS

Ein Reverse DNS (rDNS) Lookup bzw. eine Auflösung funktioniert in umgekehrter Reihenfolge. Es handelt sich um eine Abfrage an das DNS, um zu prüfen, ob eine bestimmte IP-Adresse in Gebrauch ist und für welchen Domainnamen sie verwendet wird. Dies kann zu Identifizierungszwecken im Internet verwendet werden, da empfohlen wird, dass jeder Host mit einem Namen übereinstimmen sollte (RFC 1912).

###### Secure DNS

DNSsec authentifiziert DNS-Antworten mit einer Signatur des autorisierenden DNS-Servers, auf dem der Domainname registriert wurde, nicht unbedingt des DNS-Servers, der die DNS-Anfrage beantwortet hat. DNSsec bietet also die Authentizität der DNS-Einträge, aber keine Vertraulichkeit, da alle ausgetauschten Daten unverschlüsselt sind, und auch keine Authentifizierung, da sich der korrekte DNS-Server nicht authentifizieren muss.

Sichere DNS-Protokolle verhindern Ersteres, indem sie alle ausgetauschten Daten verschlüsseln, und Letzteres, indem sie den DNS-Server gegenüber dem DNS-Client authentifizieren. Die vier Hauptanwärter für verschlüsselte und authentifizierte DNS-Anfragen und -Antworten sind das NSCrypt-Protokoll, DNS-over-TLS (oder DoT), die DNS-Auflösung über TLS, und DNS-over-HTTPS (oder DoH), die DNS-Auflösung über HTTPS (Hounsel et al., 2020).

Weder Windows, noch macOS oder Linux unterstützen standardmäßig verschlüsselte DNS-Anfragen beim Schreiben. Android 9 unterstützt jedoch DoT. Firefox unterstützt verschlüsselte DNS-Abfragen durch DoH, und Google testet DoH über Chrome. DNSCrypt hat weniger Rückhalt bei den großen Unternehmen.

###### Aspekte des Datenschutzes

Da alle anderen gesendeten Metadaten in der Regel den angeforderten DNS enthalten, werden DNS-Anfragen durch die Verschlüsselung hauptsächlich von einer Partei (dem DNS-Server ohne Verschlüsselung) zu einer anderen (dem DNS-Server mit Verschlüsselung) verlagert. Da bei den meisten Anfragen im Internet der Domainname als Klartext übertragen wird, z. B. durch Reverse-DNS-Lookup oder durch Protokoll-Header, liegt der Vorteil sicherer DNS-Anfragen weniger im Schutz gegen Abhören als vielmehr in der Authentifizierung des DNS-Servers. Die Authentifizierung der angeforderten Domain wird jedoch in der Regel bereits durch TLS gewährleistet. Der Nachteil ist die Zentralisierung aller DNS-Anfragen (weg von den Internetdiensteanbietern) bei DNS-Anbietern, was deren gebündelte Bearbeitung ermöglicht. Während die Daten zwischen dem Client und dem Server während des Transports verschlüsselt werden, werden sie an jedem Ende entschlüsselt.

DNSCrypt  
Nicht standardisiertes sicheres DNS-Protokoll ohne vertrauenswürdige Zertifizierungsstellen, das DNSCrypt genannt wird.

DNSCrypt vertraut einem öffentlichen Signierschlüssel anstelle von X.509-Zertifikaten einer Zertifizierungsstelle. Der von Anbietern erhaltene öffentliche Schlüssel verifiziert Schlüssel durch herkömmliche DNS-Abfragen, die als kurzzeitige öffentliche Schlüssel für den Schlüsselaustausch oder Chiffre-Identifikatoren verwendet werden. Während Server bereit sind, erzeugte, kurzfristige Schlüsselpaare wiederzuverwenden, suchen Clients bei jeder Anfrage nach neu erzeugten Schlüsselpaaren (DNSCrypt, o. J.). Das Protokoll basiert auf den Transportprotokollen TCP (obligatorisch) und UDP (optional). Obwohl es das Protokoll bereits seit 2013 gibt, ist es nicht standardisiert (DNSCrypt, o. J.).

DNS-over-HTTPS wurde in RFC8484 spezifiziert und verwendet HTTPS so, dass es von anderem HTTPS-Verkehr nicht zu unterscheiden ist und daher praktisch nie blockiert wird. Firefox hat Unterstützung für DoH durch die DNS-Server von Cloudﬂare hinzugefügt. Google plant, DoH mit Chrome zu testen.

DNS-over-TLS wird dank der IETF-Standardisierung in RFC7858 am häufigsten in Software unterstützt. DoT-Clients authentifizieren den Dienst, mit dem sie sich verbinden, mithilfe der Simple Public-Key Infrastructure (SPKI), einer gemeinsamen Maßnahme der IETF zur Vereinfachung der traditionellen X.509 PKI und ein unterstützter Standard für den Aufbau von Vertrauen.

DoT ist reiner DNS-Verkehr innerhalb einer TLS-Verbindung über einen speziellen Port 853 (und gelegentlich über Port 443). Bis zur Verschlüsselung durch TLS ist es dasselbe wie DNS über TCP/IP anstelle von UDP. Da TLS als Verschlüsselungsprotokoll zur Sicherung fast aller anderen Internetdienste verwendet wird, ist die Technologie gut bekannt und wird ständig verbessert.

Zusammenfassung

Die Protokolle, welche die Kommunikation im Internet standardisieren, können in Schichten eingeteilt werden: das OSI-Referenzmodell (Open Systems Interconnection) mit sieben und das TCP/IP-Referenzmodell mit vier Schichten. Die Schichten sind danach geordnet, wie stark die verarbeiteten Daten strukturiert sind: je höher die Schicht, desto näher an den Anwendungen der Nutzer befindet sie sich. Die beiden wichtigsten Internetprotokolle sind TCP und IP, die festlegen, wie Daten formatiert, adressiert, übertragen, weitergeleitet und am Zielort empfangen werden.

Ein VPN ist ein privates Netz, das aus zwei oder mehr geschlossenen Netzwerken besteht, die über ein offenes Netzwerk verbunden sind. Das IPsec-VPN verwendet den Transportmodus, um eine Punkt-zu-Punkt-Kommunikation zwischen zwei Endpunkten herzustellen, oder den Tunnelmodus, um zwei Netzwerke über zwei Gateways zu verbinden. Im IPSEC-Tunnelmodus werden IP-Pakete in anderen IP-Paketen eingekapselt (getunnelt).

Das kryptografische Transportprotokoll TLS und sein Vorgänger SSL bieten Authentifizierung, Vertraulichkeit und Authentizität für Daten, die über TCP gesendet werden. Die Authentifizierung und Verschlüsselung erfolgt durch ein X.509-Zertifikat einer Zertifizierungsstelle. Diese sind hierarchisch aufgebaut und geben das Vertrauen von der obersten zur untersten Ebene weiter, wobei die oberste Ebene als Stammstelle bezeichnet wird.

E-Mails werden entweder nur während des Transports oder durchgängig (Ende-zu-Ende) verschlüsselt, was sicherer ist. Bei letzterem werden die Daten an den Endpunkten, bei den Absendern und Empfängern, ver- und entschlüsselt, so dass die gesendeten E-Mails auch für die Mailserver nicht lesbar sind.

Das DNS ist eine verteilte Datenbank im Internet, die von Menschen lesbare Domainnamen in maschinenlesbare IP-Adressen auflöst. DNS bietet weder Datenschutz, da alle ausgetauschten Daten unverschlüsselt sind, noch Vertrauen, da sich der DNS-Server nicht authentifizieren muss. DNSsec bietet Authentizität durch die Signierung aller DNS-Einträge, während neuere sichere DNS-Protokolle wiederum alle ausgetauschten Daten verschlüsseln und den DNS-Server gegenüber einem Client authentifizieren.



# Lektion 7

## Praktische Aspekte der Kryptologie

#### LERNZIELE

Nach der Bearbeitung dieser Lektion wissen Sie …

… wo, warum und wie Zufallszahlen in der Kryptografie generiert werden, und kennen den Unterschied zwischen physischer Zufallszahlengenerierung und Pseudo-Zufallszahlengenerierung.

… wie langfristige Sicherheit in Bezug auf die zukünftige Stärke der Schlüssellängen und die eingesetzten Mittel gewährleistet werden kann.

… welche möglichen zukünftigen Bedrohungen durch die Nutzung von Quantencomputern entstehen und wie diese abgewehrt werden können.

… wie Sie Kryptografie bei der Anwendungsentwicklung am besten einsetzen.

… wie Sie gesetzliche Vorgaben für den Datenschutz durch die Nutzung von Verschlüsselung erfüllen.

… welcher gesetzlichen Fallstricke und Sicherheitslücken Sie sich bewusst sein müssen.

1. Praktische Aspekte der Kryptologie

### Einführung

Auch wenn die kryptografischen Algorithmen in der Theorie sicher sind, kann in der Praxis bei der Implementierung kryptografischer Verfahren viel schiefgehen. Als Softwareentwickler müssen wir daher Vorsicht walten lassen und bewährte Methoden (Best Practices) anwenden.

Insbesondere sollten wir geprüfte Softwarebibliotheken verwenden, die kryptografische Funktionen wie Verschlüsselung, Entschlüsselung, Signierung und Verifizierung implementieren. Neben den kryptographischen Algorithmen ist auch die Implementierung des Zufallszahlengenerators, der für Angriffe berüchtigt ist, von entscheidender Bedeutung. Wie wir bei der Kryptografie mit elliptischen Kurven (ECC) gesehen haben, gilt: Wird derselbe kurzlebige Schlüssel (in der Regel zufällig generiert) zweimal verwendet, um verschiedene Dokumente mit demselben privaten Signaturschlüssel zu signieren, dann wird der kurzlebige Schlüssel bekannt und verrät den geheimen Signaturschlüssel.

Selbst bekannte und bewährte kryptografische Open-Source-Bibliotheken, z. B. Open-SSL, weisen Sicherheitslücken auf. Der Heartbleed-Bug führte z. B. dazu, dass verbreitete Webserver auf Anfrage geheime Daten, wie Passwörter oder Serverschlüssel, preisgaben. Auch wenn diese Sicherheitslücke rasch beseitigt wurde, stellt sich die Frage, ob die Geheimdienste diese Lücken durch sogenannte Backdoors in Verschlüsselungssoftware ausnutzen.

Selbst ein optimales kryptografisches Verfahren kann das Mooresche Gesetz nicht unterbinden. Dieses Gesetz besagt, dass sich die Rechenleistung jedes Jahr verdoppelt, was auch bedeutet, dass die Schlüssel immer schwächer werden. Neben den kontinuierlichen Fortschritten kann es zudem technologische Sprünge geben, wie z. B. den Quantencomputer, der viele gängige asymmetrische Chiffren wie Diffie-Hellman, RSA, ECC und aufwändigere Alternativen brechen könnte.

### Generierung von Zufallszahlen

Die Kryptografie benötigt Zufallszahlen zur Schlüsselgenerierung (symmetrische und asymmetrische Algorithmen) sowie Nonces, Salts und IVs (Initialisierungsvektoren). IVs sind Zahlen, die in der Regel nach dem Zufallsprinzip generiert, offengelegt und einmalig in einem kryptografischen Verfahren verwendet werden, um dessen Sicherheit durch Einzigartigkeit zu erhöhen. Beispiele hierfür sind eine Sitzung zwischen einem Client und einem Server oder eine Hashfunktion, die zur Speicherung eines Passworts verwendet wird.

Echte Zufälligkeit ist für die Schlüsselgenerierung von entscheidender Bedeutung. Bei nur einmal verwendeten Nonces ist dies weniger wichtig. Hier wird manchmal ein Schlüssel als Nonce verwendet, für den Eindeutigkeit ausreichend sein kann. Der starke GnuPG-Zufallszahlengenerator für kryptografische Schlüssel bildet beispielsweise einen Pool von 600 physisch ungeordneten Bytes mit Entropie und hasht diese mit SHA-1. Der GnuPG-Nonce-Generator wiederum fügt 20 Byte mit der Prozess-ID-Nummer (PID) und der Zeit in Sekunden sowie acht Byte hinzu, die zufällig von einem starken Zufallszahlengenerator kommen, und hasht diese mit SHA-1.

###### Geheimnisse

Ein Geheimnis, d. h. eine geheime Bitfolge oder eine geheime Zahl, muss meist vom Computer erzeugt werden, wenn …

* … das Geheimnis automatisch generiert werden soll. Um beispielsweise den Diffie-Hellman-Schlüsselaustausch zu einem Public-Key-Algorithmus zu erweitern, erzeugt der Elgamal-Algorithmus für jeden zu verschlüsselnden oder zu signierenden Klartext einen kurzlebigen geheimen Schlüssel. Die populären Signaturalgorithmen DSA und sein Analogon über elliptische Kurven, ECDSA, das dem Elgamal-Signaturalgorithmus zugrunde liegt, erzeugen für jeden zu signierenden Klartext ein kurzlebiges Schlüsselpaar.
* … das Geheimnis einem bestimmten Format entsprechen muss. Zum Beispiel bei RSA mit Modul N = p q für die Primzahlen p und q gilt: Ein Schlüssel E ist eine Zahl, die keinen (Prim-)Faktor gemeinsam hat mit (N) = (p – 1)(q – 1).
* … das Geheimnis garantiert zufällig werden soll. Menschen wählen meist Passwörter, die bestimmten Mustern folgen, um sich diese leichter merken zu können.
* … die Bitlänge des Geheimnisses wesentlich größer sein muss als die eines Passworts, das wir uns merken können. Damit beispielsweise die Entschlüsselung eines mit RSA verschlüsselten Geheimtextes so lange dauert wie die Suche nach einem 112-Bit-Geheimschlüssel, muss der geheime RSA-Schlüssel 2048 Bit lang sein, was etwa 340 ASCII-Armor-Buchstaben entspricht.

Für die Geheimhaltung ist es erforderlich, dass das Ergebnis der Generierung unvorhersehbar ist. Da der Generierungsalgorithmus in der Regel bekannt ist, muss seine Eingabe somit unvorhersehbar sein. Dies schließt z. B. die Verwendung von Computer-Uhrzeiten oder einer Zahl einer bekannten Folge als Eingabe aus.

###### Einzigartigkeit

Wenn zum Beispiel der geheime kurzlebige Schlüssel, der von den Signaturalgorithmen DSA und seinem Analogon über elliptische Kurven, ECDSA, verwendet wird, bekannt ist, kann der permanente geheime Schlüssel der Signierenden abgeleitet werden. Angreifende können also die Signaturen der Signierenden fälschen.

Das gilt insbesondere wenn zwei kurzlebige öffentliche Schlüssel derselben Signierenden, die für verschiedene Dokumente verwendet werden, übereinstimmen. Dann kann der geheime kurzlebige Schlüssel, der von einem dieser Signaturalgorithmen verwendet wird, abgeleitet werden. Die Standard-Java-Bibliothek für die Generierung von Zufallszahlen unter Android generiert beispielsweise immer sich wiederholende Zufallszahlen und ermöglicht so die Fälschung von Signaturen der Nutzer einer Android-Bitcoin-App (Ducklin, 2013). In diesem Fall muss die erzeugte Zahl also einzigartig sein.

###### Zufallszahlengeneratoren (Random Number Generators, RNGs)

Während echte Zufallszahlen für sicherheitskritische Anwendungen entscheidend sind und mit Hardware-Zufallszahlengeneratoren erzeugt werden können, sind für weniger kritische Anwendungen Pseudozufallszahlen oft ausreichend, z. B. für probabilistische experimentelle Simulationen wie die Monte-Carlo-Methode.

###### Pseudo-Zufallszahlengenerator

Pseudo-Zufallszahlengenerator

Ein Pseudo-Zufallszahlengenerator erzeugt Zahlenfolgen, die unabhängig voneinander erscheinen, d. h. die statistische Tests auf Zufälligkeit erfüllen.

Ein Pseudo-Zufallszahlengenerator erfordert eine sorgfältige mathematische Analyse, wie z. B. die BigCrush Test Suite (L'Ecuyer & Simard, 2007), wird aber durch ein genau bestimmtes mathematisches Verfahren erzeugt. Für die meisten Anwendungszwecke ist diese scheinbare Zufälligkeit jedoch ausreichend. Die von den meisten Softwarebibliotheken bereitgestellten Zufallszahlengeneratoren sind pseudozufällig, wie einer Liste solcher Pseudo-Generatoren entnommen werden kann („List of random number generators“, 2020). Ein alter, einflussreicher und einfacher ist der lineare Kongruenzgenerator, der für einen Multiplikator a, Offset c und Modul m die Folge X0, X1, … erzeugt, induktiv gegeben durch

Xn + 1 ≡ a Xn + c mod m.

Sein Nachfolger, der linear rückgekoppelte Schieberegister verwendet, ersetzt die Arithmetik in /m durch die im binären Polynomring 2 [X]. Xorshift+ 128 ist eine effiziente Option, welche die BigCrush Test Suite erfolgreich besteht. Es handelt sich um eine Anpassung, um die Testsuite von Xorshift zu bestehen, die eine anfängliche n-Bit-Zeichenkette ungleich Null iterativ mit einer invertierbaren Matrix der Ordnung 2n – 1 multipliziert. Andere Generatoren, die die Testsuite bestehen, sind die Hashfunktion SHA-1 und der symmetrische Algorithmus AES mit Anfangswerten.

###### Hardware-Zufallszahlengenerator

Ein Hardware-Zufallszahlengenerator ist ein Computergerät, das Zufallszahlen aus einem physikalischen Prozess erzeugt, der theoretisch völlig unvorhersehbar ist, wie z. B. thermisches Rauschen, Spannungsschwankungen in einer Diodenschaltung oder Quantenoptik. In Unix-Betriebssystemen wird die Zufälligkeit aus den Geräten /dev/random und /dev/urandom gewonnen. GnuPG verwendet in Ermangelung dessen Prozessstatistiken, unterstützt aber auch die Hardware-Zufallszahlengeneratoren in der Padlock-Engine von VIA-(Centaur)-CPUs und x86-CPUs mit der RDRAND-Anweisung.

### Langfristige Sicherheit

Wie können wir sicherstellen, dass die heute verschlüsselten Daten auch in den kommenden Jahrzehnten noch sicher sind? Dies ist insbesondere eine für bestimmte langfristige Anwendungen wie Gesundheitsdaten relevante Frage. In der Praxis hängt die Sicherheit einer Chiffre und damit die empfohlene Schlüsselgröße in erster Linie von ihrer Widerstandsfähigkeit gegenüber den effizientesten (bekannten) Methoden der Kryptoanalyse (unter Verwendung einer Hintertür) und dem Rechenaufwand ab, der zur Überprüfung aller Schlüssel (unter Verwendung der Vordertür) erforderlich ist, indem die entschlüsselte Ausgabe auf wahrscheinliche Muster eines Klartextes überprüft wird.

Dabei müssen wir berücksichtigen, dass die Rechenleistung zunimmt, so dass die stärksten bekannten Kryptoanalyse-Algorithmen angewendet werden können, und dass neue Algorithmen (weniger vorhersehbar) und Geräte (besser vorhersehbar) am Horizont erscheinen, insbesondere was die Quantencomputer betrifft. Außerdem können wir vorbeugende Maßnahmen ergreifen, indem wir die für die Verschlüsselung verwendeten Schlüssel variieren, sodass ein einzelner, kompromittierter Schlüssel so wenig Geheimtext wie möglich kompromittiert. Perfect Forward Security erzeugt für jede neue Sitzung einen neuen Schlüssel (Paar), d. h. einen Austausch von Geheimtexten.

###### Schlüssellängen

Der schnellste Supercomputer der Welt ist der Summit von IBM, der sich im Oak Ridge National Laboratory in Tennessee, USA, auf eine Fläche von über 520 Quadratmetern erstreckt. Er hat eine Leistung von etwa 122 Peta-Flops, d. h. 1,22 · 1017 Gleitkommaoperationen pro Sekunde. Die Anzahl der zur Prüfung eines Schlüssels benötigten Flops hängt z. B. davon ab, ob der Klartext bekannt ist oder nicht, kann aber optimistisch mit 1000 angenommen werden. Summit kann also ca. 1. 22 · 1014 Schlüssel pro Sekunde überprüfen, und da ein Jahr rund 31536000 Sekunden hat, ca. 3,85 · 1023 Schlüssel pro Jahr.

Um der steigenden Rechenleistung entgegenzuwirken, wenden wir mit Bedacht das Mooresche Gesetz an. Alle zwanzig Jahre steigt die Rechenleistung um einen Faktor 220 = 1,05 · 106. Um sicherzustellen, dass ein Schlüssel bei einer einjährigen Suche durch den schnellsten Supercomputer der Welt in z. B. sechzig Jahren nicht gefunden wird, müssen mindestens 4,44 · 1041 Schlüsselkombinationen verwendet werden.

Für einen Schlüssel der Bitlänge n ist die Anzahl aller möglichen Schlüssel 2n. Wenn n = 80 , dann gibt es 280 mögliche Schlüsselkombinationen. Diese Zahl ist zwar für den Moment ausreichend, aber der voraussichtlich schnellste Supercomputer in zwanzig Jahren wird den Schlüssel wahrscheinlich finden.

AES

Für den symmetrischen Algorithmus AES ist der schnellste derzeit bekannte Algorithmus die erschöpfende Schlüsselsuche, d. h. alle möglichen Schlüssel auszuprobieren, deren Komplexität (= Anzahl der Operationen) 2nbeträgt. Die minimale AES-Schlüssellänge beträgt 128 Bit, d. h. es gibt 2128 mögliche Schlüsselkombinationen. Die Chance, dass der schnellste Supercomputer der Welt in sechzig Jahren den geheimen Schlüssel findet, liegt also bei 10–8, also einem Millionstel Prozent. Wir kommen somit zu dem Schluss, dass die minimale AES-Schlüssellänge in den nächsten Jahren gegen Brute-Force-Angriffe sicher ist.

Asymmetrische Algorithmen

In der Ein-Schlüssel-Kryptografie nutzt die Kryptoanalyse statistische Muster aus, indem sie auf rechnerisch schwierige mathematische Probleme zurückgreift, also solche, deren Laufzeit exponentiell mit der Bitlänge der Eingabe wächst. Die Kryptoanalyse der Zwei-Schlüssel-Kryptografie beruht auf der Computermathematik, also einen Algorithmus zu finden, der die Lösungen des schwierigen mathematischen Problems schnell berechnet. Für die klassischen asymmetrischen Algorithmen, die Primfaktorzerlegung bei RSA oder den diskreten Logarithmus beim Diffie-Hellman-Schlüsselaustausch, ist der schnellste Algorithmus das allgemeine Zahlkörpersieb (Lenstra, 2006), dessen Anzahl der Operationen für eine große Anzahl von Eingabebits M, wobei N = log (2) M ist, ungefähr L(N, 1 / 3 , (64 / 9)1 / 3^) beträgt. Hier gilt:

L N. r, c . = exp cNr logN 1 − r .

Um dies mit der Anzahl der Operationen 2m zu vergleichen, die für die erschöpfende Schlüsselsuche eines Schlüssels der Bitlänge m erforderlich sind, legen wir n = log (2) m fest, setzen die beiden Anzahlen von Operationen gleich und erhalten:

Nlog N 2 ≈ 9n/64 ≈ n3/7

Diese Gleichung muss numerisch für N gelöst werden. Für m = 80 , d. h. n , finden wir M = 1024 , d. h. N, um log (N)2= 43: N log (N)2 500 und 553 / 7 000 zu erfüllen. So ist der Rechenaufwand für das Auffinden eines privaten 1024 Bit langen Schlüssels für RSA und Diffie-Hellman durch das allgemeine Zahlkörpersieb mindestens so hoch wie für das Auffinden eines geheimen 80 Bit langen Schlüssels (z. B. für AES) durch erschöpfende Schlüsselsuche.

Für den Logarithmus über einer endlichen elliptischen Kurve ist der schnellste derzeitige Algorithmus der generische Babystep-Giantstep-Algorithmus (oder Pollards ρ-Algorithmus), dessen Komplexität ungefähr 2√n beträgt. Damit der Rechenaufwand für das Auffinden eines privaten Schlüssels der Bitlänge N für ECC mit dem für das Auffinden eines geheimen Schlüssels der Bitlänge n (z. B. für AES) durch erschöpfende Schlüsselsuche vergleichbar ist, muss die Schlüssellänge verdoppelt werden, d. h. N 2n.

Mit diesen Formeln lässt sich berechnen, dass die Schlüssellänge eines (privaten) RSA- oder Diffie-Hellman-Schlüssels mindestens 3072 Bit betragen muss, um genauso sicher zu sein wie ein AES-Schlüssel mit 128 Bit. Außerdem muss die Schlüssellänge eines ECC-Schlüssels mindestens 256 Bit betragen, damit er genauso sicher ist wie ein AES-Schlüssel mit 128 Bit.

In der Tat sind diese Schlüsselgrößen für die nächsten Jahrzehnte ausreichend, vorausgesetzt, es wird kein schnellerer Algorithmus als die bekannten entdeckt.

###### Quantencomputer

Quantenforscher hoffen, Computer bauen zu können, die das Phänomen der Superposition nutzen, bei dem ein Quantensystem mehrere Zustände annehmen kann, bis es durch eine Messung in einen einzigen Zustand zerfällt.

Quantencomputer Ein Quantencomputer nutzt Qubits, die „verschränkt“ werden können, so dass er mehrere Berechnungen gleichzeitig durchführen kann.

Ein klassischer Computer speichert Informationen in Bits. Jedes Bit entspricht einem der Zustände „Ein“ oder „Aus“. Ein Quantencomputer, der Qubits verwendet, hat ein Endergebnis, das von den Interferenzen abhängt, die von ihnen erzeugt werden (Steane, 1998). Wenn wir einen nützlichen Quantencomputer bauen wollen, stehen wir vor einigen Herausforderungen Ein Quantencomputer muss seine Qubits lange genug verschränkt halten, um eine Berechnung durchzuführen. Fehler, die durch unvermeidliche Wechselwirkungen mit der Umwelt entstehen, müssen herausgefiltert und korrigiert werden. Darüber hinaus wird der Zustand eines Quantensystems durch die Messung dessen, was behoben werden muss, beeinflusst.

Quantenzustände

Die an einem Teilchen gemessenen Eigenschaften hängen von den Operationen ab, die an allen anderen Teilchen durchgeführt werden. Wenn ein Teilchen zwei Zustände annehmen kann (bezeichnet als 0 und 1), kann ein System aus zwei Teilchen z. B. die Zustände 00 oder 11 oder sogar eine Überlagerung dieser Zustände annehmen. Beispielsweise wird jedes der beiden isoliert betrachteten Teilchen zufällig als 0 oder 1 gemessen, aber die Teilchen sind „Zwillinge“, d. h. die Messung eines Zustands eines der beiden Teilchen zwingt das andere Teilchen in denselben Zustand. Die Verletzung einer statistischen Ungleichung beweist, dass dies eine Eigenschaft der Teilchen ist und nicht auf eine (versteckte) Verbindung zurückzuführen ist (Bell, 1964; Aspect et al., 1982).

Eine physikalische Größe wird in der Quantenmechanik als beobachtbar bezeichnet und befindet sich normalerweise in einem überlagerten Zustand. Nur in speziellen (Eigen-)Zuständen hat sie einen eindeutig bestimmten (Eigen-)Wert. Im Allgemeinen ist der Eigenzustand das Ergebnis der Anwendung einer Beobachtungsgröße auf einen (überlagernden) Zustand und der Wahl eines Eigenwerts.

Formal ist ein überlagerter Zustand ein komplexer Einheitsvektor, d. h. ein Vektor der Länge eins mit komplexen Einträgen, und eine Beobachtungsgröße A ist ein (selbstverbundener) Operator auf dem Vektorraum aller Zustände. Da A selbstverbunden ist, gibt es eine Basis bi , so dass A bi = ai bi für reelle ai. Jedes bi ist ein Eigenzustand und ai dessen Eigenwert.

Ein Eigenzustand bi ist ein Eigenvektor von A und wird als mögliches Ergebnis einer Messung für eine bestimmte Beobachtungsgröße interpretiert. Ein allgemeiner Zustand |= ci bi ist eine Linearkombination von Eigenzuständen. Der komplexe Koeffizient ci von bi wird als Wahrscheinlichkeitsamplitude von bi bezeichnet; sein Betrag |ci | in [0, 1] ist die Wahrscheinlichkeit der Messung von bi.

Quantenbits

Während ein digitaler Computer Bits verarbeitet, die sich in einem von zwei Zuständen befinden können, nämlich 0 oder 1, verarbeitet ein Quantencomputer Quantenbits, deren Zustände sich überlagern. Der Zustandsraum, die Menge aller möglichen Zustände, eines Computers mit n Bits ist gegeben durch die Menge aller Zeichenketten

Q . = 0,1 n = 0, . . . , 0 , . . . , 1, . . . , 1

n

von n Bits, von denen es 2n gibt. Der Zustandsraum eines Quantencomputers mit n Qubits ist durch alle Wahrscheinlichkeitsamplituden (oder Schrödinger-Wellenfunktionen) auf Qn gegeben, d. h. durch alle Zeichenketten mit 2n Einträgen komplexer Zahlen von Einheitslänge. Eine Wahrscheinlichkeitsamplitude ist ein Vektor einheitlicher Länge mit komplexen Einträgen, die durch Qn indiziert sind. Sie ist also eine Kette Qn aus 2n komplexen Zahlen

λ = λ 0, . . . , 0 , . . . , λ 1, . . . , 1

so dass ~q Qn ~ |(q)|2 = 1 gilt.

Die Basiszustände sind diejenigen Wahrscheinlichkeitsamplituden, die einen einzigen Eintrag ungleich 1 haben, von denen es 2n gibt. Der Zustand, dessen Eintrag mit dem Wert 1 in Qn liegt, wird mit |bezeichnet. Ein einzelnes Qubit, das zwischen 0 und 1 überlagert ist, wird beispielsweise mit a | 0 + b | 1 bezeichnet, so dass a2 + b2 = 1 ist. Die Wahrscheinlichkeitsamplitude von zwei Qubits, die sich zwischen 00, 01, 10 und 11 überlagern, wird bezeichnet mit

α · 00 > + β · 01 > + γ · 10 > + δ · 11 >,

mit ||2 + ||2 + ||2 + ||2 = 1.

Jede elementare Operation im Zustandsraum wird dann durch eine (einheitliche) Matrix mit 2n Spalten (orthogonal zueinander) beschrieben. Jede Spalte ist die Wahrscheinlichkeitsamplitude, die aus der Anwendung der Operation auf den entsprechenden Basiszustand erhalten wird.

###### Quantenbeschleunigung und Post-Quanten-Kryptografie

Ein Quantencomputer bietet viele Möglichkeiten, klassische Probleme zu lösen, die ein klassischer Computer nicht bietet (Di et al., 2004). Um einen Eintrag in einer ungeordneten Liste von N Einträgen auf einem klassischen Computer zu finden, sind im Durchschnitt - , d. h. O(n) Operationen erforderlich.

2

Es gibt jedoch einen Quantenalgorithmus, der dies in nur etwa

π , d. h.

4

N

O Operationen erreicht. Es gibt auch Anwendungen, bei denen ein Quantencomputer überlegen ist, z. B. beim Lösen des Simon-Problems (Brassard & Hoyer, 1997) oder bei der Primfaktorzerlegung und diskreten Logarithmen mit dem Shor-Algorithmus (Shor, 1994).

N

Es überrascht nicht, dass die derzeitigen Verschlüsselungssysteme mit öffentlichen Schlüsseln mit einem ausreichend großen Quantencomputer leicht zu brechen sind. So kann beispielsweise der Shor-Faktorisierungsalgorithmus für die RSA-Verschlüsselung verwendet werden, während das Faktorisieren mit klassischen Algorithmen recht zeitaufwendig ist. Aber auch quantenmechanische Effekte sind Wegbereiter für die Quantenverschlüsselung. Obwohl mit Quantencomputern viele klassische Probleme viel schneller gelöst werden können als mit einem klassischen Ansatz, wird vermutet, dass nicht alle Probleme auch auf einem Quantencomputer schwierig bleiben, z. B. NP-schwere Probleme wie die Suche nach dem nächstgelegenen Vektor zu einem Gitter.

Gitter-Chiffren

Die Gitter-Chiffre oder Goldreich-Goldwasser-Halevi-Chiffre (GGH-Chiffre) verwendet eine einseitige Falltürfunktion, die auf dem Problem des nächstliegenden Vektors basiert, das NP-schwer ist (Goldreich et al., 1997). Bei jeder Basis eines Gitters ist es einfach, einen Vektor in der Nähe eines Gitterpunktes zu erzeugen, indem ein kleiner Fehlervektor zu diesem addiert wird. Um zum Gitterpunkt zurückzukehren, können wir eine besondere Art von Basis nehmen. Der private Schlüssel besteht aus einer Gitterbasis b, d. h. einer invertierbaren Matrix mit ganzzahligen Einträgen, die aus nahezu orthogonalen Vektoren besteht. Außerdem besteht er aus einer unimodularen Matrix u, d. h. einer invertierbaren Matrix mit ganzzahligen Einträgen, deren Invers ganzzahlige Einträge hat, die Vektoren mit ganzzahligen Einträgen transformieren.

Der öffentliche Schlüssel ist die Basis B = u b des Gitters L. Wenn eine Nachricht m (ein Vektor (m1 , …, mn ) aus ganzen Zahlen) und ein öffentlicher Schlüssel B gegeben sind, dann berechnen wir zur Verschlüsselung des Klartextes m den Gitterpunktvektor v = m B, wählen einen kleinen Fehlervektor e, z. B. einen mit Einträgen in 0, und berechnen den Geheimtext c = v + e (= m B + e). Wenn ein Geheimtext c und der private Schlüssel, bestehend aus einer Gitterbasis b und einer unimodularen Matrix u , gegeben sind, dann wird zur Entschlüsselung des Geheimtextes c folgendes berechnet:

c · b−1 = m · B + e b−1 = m · u · b · b−1 + e · B−1 = m · u + e · B−1 .

Während m u ein Vektor aus ganzen Zahlen ist, ist der Fehlerterm e B–1 gebrochen. So kann der ganzzahlige Summand m u vom gebrochenen Summanden e B–1 unterschieden werden, indem der gebrochene Fehlerterm e B–1 entfernt wird, um den ganzzahligen Vektor m u zu erhalten. Schließlich berechnen wir den Klartext m = (m u) u–1.

Codebasierte Kryptografie

Codebasierte Post-Quanten-Chiffren sind asymmetrische Chiffren, die auf fehlerkorrigierenden Codes basieren, um Bits über einen verrauschten Kanal zu übertragen. Um zu vermeiden, dass Andrea beispielsweise 01 sendet, Robert aber 11 empfängt, wäre eine einfache Lösung, dass Andrea jedes Bit dreimal wiederholt, 000111, und Robert für jede Dreiergruppe das Bit nimmt, das am häufigsten vorkommt, z. B. würde 100110 zu 10 dekodiert werden. Allerdings ist dieses Kodierungsschema auf ein fehlerhaftes Bit in jeder Gruppe von drei Bits beschränkt. Stattdessen multipliziert ein linearer Code den Bit-Vektor v mit einer Matrix M, d. h. er berechnet w = v M. Für eine beliebige Anzahl von fehlerhaften Bits n kann die Matrix M so gewählt werden, dass höchstens n fehlerhafte Bits in w korrigierbar sind (zu v). Ein bekanntes Beispiel ist die McEliece-Chiffre, die lineare Fehlerkorrekturcodes verwendet (McEliece, 1978).

Der private Schlüssel besteht aus drei Matrizen, einer Fehlerkorrekturmatrix c und den invertierbaren Matrizen S und P. Der öffentliche Schlüssel ist C = S c P und kann n Fehlerbits korrigieren. Um zu verschlüsseln, berechnen wir w = v C + e für einen Fehlerterm e, der n Fehler enthält. Um P–1 und *c* zu entschlüsseln, müssen wir S e entfernen und dann S–1anwenden.

Hash-basierte Kryptografie

Die Hash-basierte Kryptografie stützt sich auf die Sicherheit kryptografischer Hashfunktionen, nicht auf die Schwierigkeit mathematischer Probleme. Ein Schema für eine einmalige Signatur wird von einer Einwegfunktion, wie einer kryptografischen Hashfunktion, abgeleitet (Lamport, 1978).

Sei k der private Schlüssel und eine ganze Zahl L vorgegeben. Der öffentliche Schlüssel ist K = HashL (k), die L-fach verschachtelte Anwendung der Hashfunktion auf k.

HashL k = Hash . . . Hash k .

Die Signatur S einer Nachricht, die durch eine ganze Zahl M < L gegeben ist, mit dem privaten Schlüssel k ist die M-fach verschachtelte Anwendung der Hashfunktion auf k, d. h. S = HashM (k). Die Signatur S wird geprüft durch die Gleichheit

HashL − M S = HashL k = K .

Dieses einfache Verfahren ist unsicher, z. B. können Signaturen gefälscht werden. Aus der Signatur S von M leiten wir die Signatur von M + 1 ab durch:

Hash K M+ 1 = Hash S .

Wir müssen also nicht nur M durch k signieren, sondern auch L – M durch einen anderen Schlüssel. Dennoch ist dieses System rechnerisch zu aufwändig, weil Nachrichten in kürzere aufgeteilt werden müssen, damit die Verschlüsselung rechnerisch machbar ist. Außerdem kann das Verfahren nur für die sichere Signierung von einer Nachricht pro Schlüssel verwendet werden. Wenn ein privater Schlüssel mehr als eine Nachricht signiert, kann eine Signatur gefälscht werden. Wenn die Nachrichten m und M mit dem Schlüssel k’ signiert werden und L – m und L – M mit dem Schlüssel k’’, dann fälschen weitere iterierte Anwendungen von H auf die Signatur (s’ oder S’ entsprechend dem kleineren Wert zwischen m und M für k’ oder k’’) Signaturen für Nachrichten zwischen m und M.

###### Perfect Forward Secrecy

Perfect Forward Secrecy bedeutet, dass Kommunikationspartner, die ihre (permanenten) öffentlichen Schlüssel ausgetauscht und gegenseitiges Vertrauen aufgebaut haben, einen (kurzlebigen) Sitzungsschlüssel erstellen und diesen mit ihren (permanenten) privaten Schlüsseln signieren. Dies geschieht, um einen Man-in-the-Middle-Angriff zu verhindern. Nach der Korrespondenz löschen alle Kommunikationspartner den kurzlebigen privaten Schlüssel. Auf diese Weise kann die Korrespondenz, selbst wenn sie aufgezeichnet wurde, später nicht entschlüsselt werden, d. h. sie kann nicht entschlüsselt werden, indem der private Schlüssel einer Kommunikationspartei beschafft wird.

Ein Beispiel ist das TLS-Protokoll Version 1.2, mit dem ein Großteil der Kommunikation über das Internet verschlüsselt wird. Genauer gesagt tauschen der Client und der Server im Handshake zwischen Client und Server, nachdem der Client das Serverzertifikat erhalten (und ihm vertraut) hat, einen kurzlebigen öffentlichen Schlüssel aus, der zur Verschlüsselung der Kommunikation dieser Korrespondenz dient. Dieser kurzlebige Schlüssel ist mit dem permanenten öffentlichen Schlüssel des Servers signiert. Die Erstellung dieses asymmetrischen Schlüssels in Perfect Forward Secrecy macht die Erstellung eines symmetrischen vorübergehenden Schlüssels durch den Client im TLS-Handshake überflüssig.

Der gegenseitige geheime Schlüssel kann z. B. durch die Protokolle DHE (Diffie-Hellman Ephemeral) oder ECDHE (Elliptic Curve Diffie-Hellman Ephemeral) erstellt werden. Dieser Schlüssel wird dann als Eingabe verwendet, um einen geheimen symmetrischen Schlüssel zu erzeugen, z. B. für den AES-Verschlüsselungsalgorithmus.

### Kryptografie in der Anwendungsentwicklung

###### Risikoanalyse

Die Risikoanalyse misst die Wahrscheinlichkeit, dass die Sicherheit eines Unternehmens verletzt wird, z. B. durch Angreifer, die Schwachstellen im Netzwerk ausnutzen. Die Risikoanalyse berechnet den voraussichtlichen Schaden einer solchen Verletzung, sei er materiell (z. B. Verletzung des Datenschutzes) oder immateriell (z. B. Reputationsverlust). Das Risiko wird oft als finanzielles Risiko gemessen und ähnlich wie bei einer Versicherung gegen Bedrohungen wie Diebstahl bekämpft. Bei der Risikoanalyse werden die Bedrohungen (durch Angreifer oder Insider des Unternehmens) für das Computernetzwerk des Unternehmens bewertet. Laut einer CSI/FBI-Umfrage sind die bedeutendsten davon, in dieser Reihenfolge, (1) Viren, (2) Missbrauch von Netzwerken durch Mitarbeitende, (3) Denial of Service (DoS) und (4) gestohlenes geistiges Eigentum.

Die Risikoanalyse beurteilt die Werte des Vermögens des Unternehmens. Diese setzen sich aus materiellen Werten, wie dem Computernetzwerk des Unternehmens, und immateriellen Werten, wie dem Ruf des Unternehmens, zusammen. Im Rahmen einer Risikoanalyse werden die Schwachstellen des Unternehmens aufgedeckt, d. h. wo und wie ein Sicherheitsverstoß geschehen könnte. Vorbeugende Maßnahmen, wie z. B. zufällige Erstellung und regelmäßige Überprüfung von Passwörtern, Antivirensoftware und Systemen zur Erkennung von Eindringlingen, wie z. B. Firewalls, sind vorgeschrieben. Schließlich hilft die Risikoanalyse bei der Erstellung von Plänen zur schnellen Wiederherstellung nach einem Angriff.

Programmieren Sie keinen eigenen Kryptografie-Code!

Die erste Regel der Kryptografie lautet: Implementieren Sie Kryptografie nicht selbst im Produktionscode. Überlassen Sie das stattdessen den Experten und verwenden Sie eine bewährte Bibliothek, die sich in den Augen von Kryptoanalytikern bewährt hat, anstatt eine selbstentwickelte.

In der Praxis ist ein kryptografischer Algorithmus sicher, wenn er sich durch Abwehr aller Angriffe als widerstandsfähig erwiesen hat. Es sollte ein bewährter kryptografischer Algorithmus verwendet werden. Empfehlenswert sind bewährte Chiffren wie AES und RSA. Die Sicherheit kann nicht gewährleistet werden, wenn ein neuer Algorithmus verwendet wird. Insbesondere ist es eine schlechte Idee, selbst einen neuen kryptografischen Algorithmus zu entwickeln. Gleichermaßen ist eine Softwarebibliothek sicher, wenn sie sich als widerstandsfähig erwiesen hat, indem sie allen Angriffen getrotzt hat. Es sollte eine etablierte Softwarebibliothek verwendet werden. Empfohlen werden gängige Softwarebibliotheken wie Libsodium (oder das weit verbreitete OpenSSL).

###### Authentifizierte Verschlüsselung

Programmierende verwechseln oft Verschlüsselung und Authentifizierung. Während die Verschlüsselung die Vertraulichkeit gewährleistet, sorgt die Authentifizierung für die Integrität. Wenn beide Schritte getrennt durchgeführt werden müssen, verschlüsseln Sie zuerst und dann authentifizieren Sie. Berechnen Sie z. B. einen Message Authentication Code (MAC), und überprüfen Sie den MAC vor der Entschlüsselung. Grundsätzlich gibt es drei Optionen für die authentifizierte Verschlüsselung.

1. Verschlüsseln und authentifizieren. Der Klartext wird mit einem MAC des Klartextes verschlüsselt und an den Geheimtext angehängt (wie bei SSH).
2. Authentifizieren, dann verschlüsseln. Der berechnete MAC des Klartextes und der Klartext werden verschlüsselt (wie bei SSL).
3. Verschlüsseln, dann authentifizieren. Der Klartext wird verschlüsselt und ein MAC des Geheimtextes angehängt (wie bei IPsec).

Auf diese Weise können der MAC verifiziert und Texte ohne Entschlüsselung verworfen werden. So wird die Anzahl der DoS-Angriffe reduziert und gleichzeitig die gebotene „Angriffsfläche“ minimiert

Nur „Verschlüsseln, dann authentifizieren“ ist nachweislich sicher, d. h. sicher gegen IND-CCA (Nichtunterscheidbarkeit des Geheimtextes für ausgewählte Geheimtexte). Ausgehend davon können Angreifende jedes beliebige Paar von Geheimtexten zur Dechiffrierung anfordern (mit Ausnahme des fraglichen Geheimtextes) und dennoch nicht unterscheiden, welcher von zwei Klartexten dem Geheimtext entspricht.

Die Tatsache, dass weder „Verschlüsseln und authentifizieren“ noch „Authentifizieren, dann verschlüsseln“ theoretisch sicher sind, „Verschlüsseln, dann authentifizieren“ jedoch schon, bedeutet in der Praxis weder, dass die beiden ersteren unsicher sind, noch, dass die letzteren sicher sind. Es gab jedoch bereits eine Reihe von Schwachstellen für die erste Variante, während es für die zweite Variante bislang keine gab. Daher ist es am besten, eine Bibliothek zu verwenden, welche die authentifizierte Verschlüsselung als Ganzes übernimmt, anstatt diese Funktionen nur separat anzubieten, wobei diese manuell korrekt zusammengestellt werden müssten. Halten Sie sich an die goldene Regel, so viel wie möglich wiederzuverwenden, anstatt neu zu implementieren.

###### Speicherung von Passwörtern

Um ein Passwort zu speichern, verwenden wir eine Hashfunktion zur Schlüsselableitung und löschen das Klartext-Passwort aus dem Speicher, sobald wir es erhalten haben. Mögliche Hashfunktionen wären scrypt, das eine Nonce und eine Iterationszahl verwendet, um Rainbow-Angriffe abzuwehren, PBKDF2, zusätzlich zu einem Passwort. Wir sollten keine schnelle Hashfunktion wie MD5 einsetzen, die anfälliger für Rainbow-Table-Angriffe ist. Auch bei weniger sensiblen Anwendungen sollten wir sehr vorsichtig sein, denn manche Benutzende könnten diese Passwörter für sensiblere Anwendungen wiederverwenden.

Schließlich erinnern wir uns daran, dass Kodierung (z. B. base64-Kodierung) und Komprimierung (z. B. Zip-Komprimierung) keine Verschlüsselung sind, da sie kaum Informationen verschleiern. Kodierungs- und Komprimierungsalgorithmen sind beide reversible, schlüssellose Transformationen von Daten. Kodierungen transformieren Daten für eine bessere Verarbeitung, während Komprimierungen die Daten so weit wie möglich verkleinern.

### Rechtliche und regulatorische Aspekte

Datenschutz-Grundverordnung (EU) 2016/679

Die DSGVO regelt den Umgang mit personenbezogenen Daten, die von Hand oder per Computer verarbeitet und gespeichert werden. Seit dem 25. Mai 2018 gilt sie für alle Unternehmen innerhalb der Europäischen Union.

Neben der DSGVO können ggf. auch andere nationale Datenschutzgesetze gelten. In Deutschland sind dies das *Bundesdatenschutzgesetz* (BDSG) und die Datenschutzgesetze der deutschen Bundesländer (Datenschutz, 2020).

Die DSGVO trägt den Interessen von Unternehmen und Verbrauchern im digitalen Zeitalter Rechnung und schützt das Grundrecht eines jeden Bürgers auf informationelle Selbstbestimmung, indem sie dem betroffenen Bürger Transparenz und letzte Entscheidungsgewalt bei der Verarbeitung seiner personenbezogenen Daten einräumt. Mit anderen Worten: Die Verarbeitung personenbezogener Daten muss immer einen klaren Zweck haben, der von der betroffenen Person eindeutig bestätigt wurde. Außerdem müssen die Daten geschützt und nach Ablauf des angegebenen Zwecks gelöscht werden (z. B. bei Auskunftsbehörden wie der SCHUFA in Deutschland).

###### Schutz personenbezogener Daten

Alle Mitarbeitenden, die personenbezogene Daten verarbeiten, müssen in Bezug auf den Datenschutz geschult werden. Im Allgemeinen ist die Weitergabe von personenbezogenen Daten an Dritte ohne die Zustimmung der betroffenen Person unzulässig. Wenn eine Sondergenehmigung erteilt wird, müssen die Daten verschlüsselt und für jeden Zweck getrennt gesendet werden, damit Dritte die Daten weder abhören noch sammeln können.

Personenbezogene Daten im Sinne der DSGVO sind die folgenden:

* allgemeine persönliche Daten (z. B. Name, Geburtsdatum, Adresse)
* Identifikationsnummern (z. B. Steuer-ID, Krankenversicherungsnummer)
* körperliche Merkmale (z. B. Geschlecht, Haar- und Augenfarbe)
* Eigentumsmerkmale (z. B. Fahrzeug- und Immobilienbesitz)
* Werturteile (z. B. Schul- und Arbeitszeugnisse)
* Bankdaten (z. B. Kreditinformationen, Kontostände)
* Kundendaten (z. B. Bestellungen, Postadressen)
* Online-Daten (z. B. IP-Adresse, Standortdaten)

Besondere Daten, die eines speziellen Schutzes bedürfen, sind im Allgemeinen (aber ähnlich aufgelistet, z. B. in Paragraph 4, 9 der DSGVO): ethnische Herkunft, politische Meinungen, religiöse Überzeugungen, sexuelle Orientierung oder Mitgliedschaft in einer Vereinigung und biometrische Daten, wie z. B. Gendaten oder Gesundheitsdaten.

In der Gesundheitsfürsorge sind Patienteninformationen daher per Gesetz streng vertraulich zu behandeln. In Deutschland dürfen nur verschlüsselte Dokumente übermittelt und nur die für die Behandlung notwendigen Daten erhoben werden. Die Patientendaten müssen sicher aufbewahrt und von den Mitarbeitenden vertraulich behandelt werden. In Deutschland kann die unbefugte Weitergabe von Patientendaten, die dem Berufsgeheimnis unterliegen, nach § 203 des Strafgesetzbuches (StGB) mit einer Geldstrafe oder bis zu einem Jahr Gefängnis bestraft werden.

###### Behördenzugang

Durchgesickerte Dokumente zeigen, dass die amerikanische National Security Agency (NSA) über den Heartbleed-Bug nach Sicherheitslücken gesucht hat. Die NSA hat die internationalen Verschlüsselungsstandards sabotiert, indem sie die Algorithmen für eine spätere Entschlüsselung absichtlich geschwächt hat. Bei der Spezifizierung des ersten US Data Encryption Standard (DES) in den 1970er Jahren wurde die NSA beispielsweise verdächtigt, DES durch die Verkürzung der Schlüssellänge zu schwächen (de Leeuw & Bergstra, 2007).

Der Escrowed Encryption Standard (EES) ist ein chipbasiertes symmetrisches Verschlüsselungssystem, das von der NSA entwickelt wurde. Es wurde im Rahmen eines Projekts der US-Regierung entwickelt, um elektronische Geräte, die an die breite Öffentlichkeit verkauft werden, mit einem Sicherheitschip auszustatten. Der Verschlüsselungsschlüssel sollte den staatlichen Behörden zur Verfügung gestellt werden, die dann in der Lage wäre, die Kommunikation abzuhören, wenn es nötig wäre.

Der Hauptunterschied zu anderen Verschlüsselungsmethoden besteht darin, dass sich die US-Behörden bei Bedarf Zugang zu den Schlüsseln verschaffen können, die von zwei Nutzern zum Datenaustausch verwendet werden. Das Verfahren ist so spezifiziert, dass zum Abhören zwei Schlüssel benötigt werden, die bei verschiedenen Behörden hinterlegt sind und die nur per Gerichtsbeschluss gleichzeitig freigegeben werden sollten. Diese offizielle Zugangsmöglichkeit wird nicht durch eine eingebaute Hintertür im technischen Sinne erreicht, sondern durch die Hinterlegung von zwei Teilschlüsseln. Wenn die rechtlichen Voraussetzungen erfüllt sind, werden die beiden Teile des Schlüssels ausgegeben und miteinander verbunden.

Die Schweizer Crypto AG war ein international tätiges Unternehmen auf dem Gebiet der Informationssicherheit. Zwischen 1960 und 1990, auf dem Höhepunkt des Kalten Krieges, war die Crypto AG ein führendes Unternehmen für Verschlüsselungsgeräte in mehr als 130 Ländern. Die US Central Intelligence Agency (CIA) war besorgt darüber, dass sie nicht in der Lage war, ausländische Nachrichten zu entschlüsseln und wandte sich an europäische Hersteller, darunter auch die Crypto AG. Der (west-)deutsche Auslandsgeheimdienst BND und die CIA kauften das Unternehmen im Anschluss 1970 heimlich auf. Sie sorgten dafür, dass viele Staaten mit Maschinen mit schwächerer Verschlüsselung beliefert wurden, die von den Geheimdiensten entschlüsselt werden konnten (Operation Rubikon) (Leeuw & Bergstra, 2007; Theveßen et al., 2020).

Das Unternehmen ermöglichte es diesen beiden Diensten, zwischen den 1960er Jahren und dem Jahr 2010 verschlüsselte Nachrichten zu entschlüsseln. Obwohl die Sowjetunion und China nie zu den Kunden von Crypto gehörten, konnte die CIA dank Drittländern, die mit den manipulierten Geräten ausgestattet waren, von einigen ihrer Konversationen erfahren. Die CIA schätzt, dass sie auf diese Weise z. B. Folgendes erreichen konnte:

* + Lesen von etwa 85 % der verschlüsselten iranischen Nachrichten in den späten 1980er Jahren,
  + Ausspionieren der ägyptischen Kommunikation während der Camp-David-Verhandlungen 1978,
  + Ausspionieren argentinischer Nachrichten während des Falklandkriegs 1982 und
  + Sammeln entscheidender Informationen während der Invasion Panamas 1989.

###### Heartbleed

Heartbleed war eine Sicherheitslücke in der SSL-Implementierung OpenSSL, die in den beliebten Webservern Apache und nginx verwendet wurde und auf denen zum Zeitpunkt des Bugs zwei Drittel aller Webseiten liefen. Der sogenannte Heartbeat ermöglicht es dem Server und dem Client, eine TLS-Verbindung aufrechtzuerhalten, indem sie eine Nachricht beliebigen Inhalts (Payload, Nutzdaten) von einem Ende zum anderen senden, die dann auf die gleiche Weise zurückgeschickt wird, um zu zeigen, dass die Verbindung besteht.

Mit der RFC-6520-Erweiterung Heartbeat werden sichere TLS/DTLS-Kommunikations­verbindungen getestet. Sie ermöglicht einem Computer an einem Ende einer Verbindung, eine „Heartbeat-Anfrage“ zu senden, die aus Nutzdaten sowie der Länge dieser Nutzdaten als 16-Bit-Ganzzahl besteht. Empfangende müssen dann dieselben Nutzdaten an die Absendenden zurücksenden.

Versionen von OpenSSL, die von dem Heartbleed-Bug betroffen sind, weisen der zurückzugebenden Nachricht einen Speicherpuffer zu, der auf dem Feld „Länge“ in der Anforderungsnachricht basiert, unabhängig von der tatsächlichen Größe der Nutzdaten dieser Nachricht. Da die entsprechenden Grenzwerte nicht überprüft werden, besteht die zurückgegebene Nachricht aus den Nutzdaten, möglicherweise gefolgt von allem anderen, was im Speicherpuffer zugewiesen ist („Heartbleed“, Abschnitt „Behavior“, 2020).

Das Problem bei der Implementierung der TLS-Heartbeat-Funktion in OpenSSL war, dass das Programm nicht überprüfte, wie lang die empfangenen Nutzdaten tatsächlich waren. Die Angreifenden können beliebige Werte in das dafür vorgesehene Feld „payload\_length“ im Header des Nutzdatenpakets schreiben und so den Speicher der Gegenstelle lesen.

Auch wenn dieser Heartbleed-Angriff in beide Richtungen funktioniert, gehen wir davon aus, dass ein Client einen Server angreift.

1. Die Angreifenden senden dem Server Heartbeat-Nutzdaten, die 1 Byte groß sind, aber behaupten, dass sie z. B. 16 Kilobyte groß sind.
2. Der Server schreibt das Byte der Angreifenden in seinen Speicher in einen Puffer namens „pl“. Da die tatsächliche Größe der Nutzdaten nicht verglichen wird, geht der Server von der von den Angreifenden angegebenen Größe (Nutzdaten) aus, wenn die Nutzdaten zurückgegeben werden.
3. Der Server reserviert 16 Kilobyte Speicher (und ein wenig mehr für administrative Informationen): buffer = OPENSSL\_malloc(1 + 2 + Nutzdaten + Padding) (bp ist der Puffer.)
4. Der Server kopiert dann die von den Angreifenden angegebene Größe der Nutzdaten (etwa 16 Kilobyte) für die Antwort an diese Stelle: memcpy(bp, pl, Nutzdaten)k

Die zu kopierende Quelle „pl“ hat jedoch nur ein einziges Byte aus dem eingehenden Heartbeat. Die folgenden Bytes bestehen aus allen anderen Daten, die der Server gerade verarbeitet, z. B. Passwörter, Daten eines anderen Benutzers, die gerade entschlüsselt wurden, oder geheime Schlüssel für den Server. Der Server sendet dann das Datenpaket (bp) an den Client, der diesen Angriff nach Belieben wiederholen kann.

Der Abhörvorgang hinterlässt auf einem angegriffenen Computer kaum eine Spur. Wir können uns daher nicht sicher sein, inwieweit dieser Fehler ausgenutzt wurde. Es wurde zwar berichtet, dass die NSA den Heartbleed-Bug von Anfang an ausgenutzt hat, allerdings wurde dies vom NSA-Direktor umgehend dementiert: „Diese Regierung nimmt ihre Verantwortung ernst, zur Aufrechterhaltung eines offenen, interoperablen, sicheren und zuverlässigen Internets beizutragen [...] Es liegt im nationalen Interesse, die Schwachstelle verantwortungsvoll offenzulegen, anstatt sie für Ermittlungs- oder Geheimdienstzwecke zurückzuhalten“ (Hosenball & Dunham, 2014, Abs. 5 und 11).

Zusammenfassung

Bei der Implementierung von Kryptografie können viele Dinge schief gehen. Als Softwareentwickelnde empfiehlt es sich, Bibliotheken zu verwenden, die bereits verfügbar und geprüft sind. Open-Source-Softwarebibliotheken, die kryptografische Funktionen wie Verschlüsselung, Entschlüsselung, Signierung und Verifizierung implementieren, warten darauf, einen kryptografischen Originalalgorithmus zu entwickeln.

Auch wenn das meiste bereits von Kryptografie-Experten implementiert wurde, ist bei der Auswahl der richtigen Implementierung große Sorgfalt geboten. Zum Beispiel ist die Implementierung von Zufallszahlengeneratoren berüchtigt für Exploits und Audits eines Systems. Wenn bei ECC derselbe kurzlebige Schlüssel, der in der Regel zufällig generiert wird, zweimal zum Signieren verschiedener Dokumente verwendet wird, dann wird der kurzlebige Schlüssel bekannt und verrät den Schlüssel.

Selbst bei der Wiederverwendung bekannter und bewährter kryptografischer Open-Source-Bibliotheken wie OpenSSL können sich Bugs einschleichen, wie z. B. der Heartbleed-Bug, der dazu führte, dass gängige Webserver auf Anfrage aktuell verarbeitete geheime Daten preisgaben. Diese Fehler werden, sobald sie bekannt sind, schnell behoben. Es bleibt die Frage, ob die Geheimdienste diese Bugs ausnutzen.

Darüber hinaus erlassen Regierungen und Behörden Gesetze, welche die Verwendung von Verschlüsselung vorschreiben. Andererseits können Behörden dagegen arbeiten, indem sie z. B. verlangen, dass Hintertüren in kryptografische Software oder Geräte eingebaut werden.

Das Mooresche Gesetz besagt, dass sich selbst bei besten kryptografischen Verfahren durch den kontinuierlichen technologischen Fortschritt die Rechenleistung etwa jedes Jahr verdoppelt und damit die zukünftige Sicherheit der Schlüssel schwächt. Daher muss eine Sicherheitsspanne für die langfristige Sicherheit, etwa zwanzig Jahre, hinzugerechnet werden. Es könnte auch technologische Sprünge geben, wie den sich abzeichnenden Bau eines Quantencomputers, der viele gängige asymmetrische Chiffren wie Difﬁe-Hellman, RSA und ECC sowie aufwendigere Alternativen brechen würde.



# Lektion 8

## Anwendungen

#### LERNZIELE

Nach der Bearbeitung dieser Lektion wissen Sie …

... wie Kryptografie beim Online-Banking und bei Wahlen eingesetzt wird.

... wie die Authentifizierung und Zugangskontrolle über ein Einmal-Passwort abläuft.

... welche grundlegende Anatomie die Blockchain aufweist, mit der Transaktionen von Kryptowährungen gespeichert und abgesichert werden.

... wie Steganografie eine geheime Nachricht in eine öffentliche Nachricht einbettet und statistisch aufgedeckt werden kann.

... wie das Tor-Projekt für garantierte Anonymität im Internet sorgt.

DL-E-DLMCSEAITSC02

-U08

1. Anwendungen

### Einführung

Die Entwicklung der Kryptologie wird seit jeher durch ihre Anwendung vorangetrieben. Während es früher um das Abhören von Nachrichten in Kriegszeiten ging, betrifft dies heute nahezu alle geschäftlichen Transaktionen. Ihre Hauptanwendungen, die sichere Kommunikation und die Einrichtung sicherer Kommunikationskanäle, finden sich in aktuellen Anwendungen, z. B. sicheren E-Mails über S/MIME, in der Einrichtung sicherer Verbindungen über unsichere Kanäle, z. B. dem Internet über SSL/TLS und HTTPS, oder in der Einrichtung virtueller privater Netzwerke (VPNs) über IPSec. Jedoch sind noch viele weitere Anwendungen durchaus üblich, darunter die folgenden:

* Verstecken und geheimes Teilen von Daten,
* Passwortschutz,
* Identifikation,
* biometrische Sicherheit,
* digitale Signaturen,
* Erstellung und gemeinsame Nutzung öffentlicher Schlüssel,
* Zero-Knowledge-Verfahren.

Weitere prominente und anspruchsvollere Beispiele für die Kryptologie finden sich im elektronischen Handel (E-Commerce), im mobilen Handel (M-Commerce) und bei elektronischen Wahlen. Für Kauf und Verkauf, Online-Makeln (Brokerage), Geldtransaktionen und Wahlen muss die sichere Kommunikation über unsichere Kanäle gewährleistet sein. So sind Bankanwendungen, die alle Arbeitsabläufe in Online-Transaktionen abbilden, zu einer der wichtigsten Triebkräfte für die weitere Entwicklung der Kryptologie geworden.

Die Blockchain-Technologie, eine der innovativsten derzeit entwickelten Technologien, ist ebenfalls eine der treibenden Kräfte. Blockchain, ein Synonym für neue, alternative und disruptive Geschäftsmodelle, wird der Wegbereiter für Technologien vom Finanzwesen bis hin zum Industriesektor sein. Die Distributed-Ledger-Technologie, welche die Transparenz und Effizienz erhöht und gleichzeitig die Komplexität und Kosten reduziert, klingt durchaus vielversprechend. Blockchain nutzt PKIs, um Vertrauen in Form von Vertraulichkeit, Datenschutz und Sicherheit für die verwalteten Datenströme zu schaffen. Mit dem Vormarsch des Quantencomputers, der aktuelle Kryptografiesysteme entschlüsseln könnte, müssen Unternehmen aller Branchen ihre Systeme zum Schutz ihrer Daten und Prozesse auf quantenresistente Kryptografie umstellen.

### Onlinebanking

Der Begriff Onlinebanking umfasst die von Banken angebotenen Dienstleistungen, die darin bestehen, den Zugang zu einem Konto zu ermöglichen, um passive Vorgänge (z. B. die Überprüfung des Kontostands und der letzten Finanztransaktionen), aber auch aktive Vorgänge (z. B. die Durchführung von Transaktionen oder die Erteilung von Überweisungsaufträgen) durchzuführen. Onlinebanking wird über eine Benutzerschnittstelle angeboten, die über einen Webbrowser, eine Handy-App (Mobile Banking) oder per Telefon (Telefon-Banking) abgerufen wird. Der Zugang über die an Geldautomaten und Terminals angebotene Schnittstelle oder über einen Computer wird dem allgemeineren Begriff Electronic Banking zugeordnet.

###### Anforderungen

Wie bei jeder Webanwendung, die über einen Webbrowser oder über einen unsicheren Kanal wie das Internet läuft, können die Benutzenden von Online-Banking-Diensten einem Identitätsdiebstahl zum Opfer fallen, der z. B. durch eine Keylogger-Technik oder Phishing verursacht wird. Grundkenntnisse im sicheren Umgang mit Internet und E-Mail werden vorausgesetzt.

Aufgrund dieser Bedrohungen müssen die allgemeinen Anforderungen an die Informationssicherheit von Banken als Dienstleistungsanbieter erfüllt werden. Onlinebanking muss Folgendes garantieren:

* Vertraulichkeit, wobei die Daten so verschlüsselt werden, dass die Geheimtexte nicht ohne den Schlüssel und die Kenntnis der richtigen Chiffre gelesen werden können;
* Integrität, indem unbefugte Änderungen an den Inhalten durch angemessene Identifizierungs- und Authentifizierungsverfahren verhindert werden; und natürlich
* Verfügbarkeit durch eine redundante und sichere Serverinfrastruktur.

Verträge können mit einer elektronischen Signatur vollständig online unterzeichnet werden. Als vertraulich eingestufte Dokumente, z. B. Ausweisdokumente, Kontoinformationen, Rechnungen und Einzahlungen, können sicher im Onlinebanking statt auf dem Postweg ausgetauscht werden.

###### Sicherheit

Onlinebanking bietet Bankinformationen über das Internet als unsicheren Kanal an. Die Sicherheit und der Schutz der ausgetauschten Daten ist ein wichtiges Anliegen beim Onlinebanking. Dennoch ist das System ein offensichtliches Ziel für Angriffe im Zusammenhang mit der Kunden- oder Benutzerauthentifizierung. Onlinebanking senkt die Kosten für Banktransaktionen und erhöht den Nutzen für die Kunden, indem es integrierte Dienstleistungen anbietet und viele Informationen und Transaktionsdienste über das Internet verfügbar sind. Es wird ein Höchstmaß an Sicherheit angestrebt, um mögliche betrügerische Transaktionen jeglicher Art zu unterbinden. Die Verschlüsselung von Informationen und Transaktionen ist das ultimative Mittel, um die Sicherheit und den Datenschutz beim Onlinebanking durch Passwörter, biometrische Daten, PINs, digitale Signaturen, Steganografie usw. zu gewährleisten. Die Banken müssen mehr in den Schutz und die Sicherheit von Daten und Informationen investieren, da die Nutzung der digitalen Kanäle und die damit verbundenen Bedrohungen stetig zunehmen. Kryptografie hat sich zum wichtigsten Instrument für den Schutz aller Transaktionen entwickelt. Ihre Hauptaufgaben sind:

* Nachrichten für nicht legitimierte Benutzende unverständlich oder unlesbar zu machen,
* Authentifizierung der Quelle von Nachrichten,
* Sicherstellung, dass eine Nachricht während der Verarbeitung nicht verändert wurde,
* Gegenprüfung der Transaktionsdaten,
* Unterbindung von Datenmanipulationen und
* Absicherung von Transaktionen.

Beim Onlinebanking kommen verschiedene Techniken von Kryptosystemen zum Einsatz, die das Ver- und Entschlüsseln von Daten und das Verstecken von Informationen mithilfe von kryptografischen Algorithmen ermöglichen. Kryptosysteme sind das beste Instrument, um gewaltsame Sicherheitsangriffe auf Kommunikationskanäle zu verhindern. Die ständige Arbeit an der Verbesserung der Sicherheit dieser Systeme stärkt das Vertrauen der Kunden zu ihrer Bank. Neben einer einfachen Authentifizierung durch Passwort, ID und PIN werden starke Authentifizierungsmethoden wie Token, Benutzerzertifikate und private Schlüssel verwendet. Zusätzlich zur verschlüsselten Datenübertragung durch das SSL-Protokoll werden Methoden für eine digitale Signatur angeboten.

###### Authentifizierung

Neben der Verschlüsselung von Inhalten wird Kryptografie auch zur Überprüfung der Identität der Benutzenden und zur Gewährleistung der Datenintegrität eingesetzt, indem Daten vor unbefugten Änderungen geschützt werden. Für die Online-Kommunikation ist es wichtig, dass zwischen den beiden Kommunizierenden Vertrauen besteht. Zu diesem Zweck kann eine digitale Signatur verwendet werden. Diese Signaturdaten sind mit den Eigentümern verknüpft oder logisch verbunden und garantieren deren Authentizität.

OTP (One-Time-Passwort)

Dieses Passwort wird einmalig zum Nachweis der Identität verwendet und ist nur für eine Anmeldung oder eine Transaktion gültig.

Die Authentifizierung oder Zugangskontrolle beim Online- oder Mobile-Banking kann mit einem OTP erfolgen. Ein OTP wird in einer SMS an eine zuvor registrierte Handynummer gesendet. Dieses OTP kann für Bankgeschäfte oder die Anmeldevorgänge verwendet werden. Außerdem muss die PIN des mobilen Geräts bekannt sein, um das OTP zu lesen, und weitere Transaktionen sind erst nach der Authentifizierung zum Entsperren des mobilen Endgeräts möglich. Dies bietet eine Ende-zu-Ende-Verschlüsselung für die OTP-SMS. Dadurch erhalten wir zwei Authentifizierungsebenen, da sowohl die PIN als auch das OTP korrekt sein müssen, um fortfahren zu können. Das generierte OTP wird in der Regel mit dem symmetrischen AES mit Schlüssellängen von 128, 192 oder 256 Bit verschlüsselt. Dies hat zwar den Nachteil einer hohen Systemlast für die Ver- und Entschlüsselung, bietet aber wiederum ein höheres Sicherheitsniveau im Vergleich zu einem statischen, gespeicherten Passwort (Rao et al., 2011).

Über das Internet abgewickelte Transaktionen sind weit verbreitet, aber auch Bedrohungen ausgesetzt. Diese können durch ein mehrschichtiges Sicherheitskonzept bewältigt werden. Das OTP, das als Klartext-SMS versendet wird, ist zwar durch das symmetrische AES-Verfahren gesichert, aber auch anfällig für Angriffe über den mobilen Kommunikationskanal. Natürlich ist ein OTP, das in einem mehrschichtigen Sicherheitskonzept verwendet wird, sicherer als ein einfaches OTP. Kritische Transaktionen erfordern eine Zwei-Faktor-Authentifizierung, z. B. eine Kombination aus der Benutzer-ID und einem verschlüsselten OTP. Zu Sicherheitszwecken könnte eine digitale Signatur verwendet werden, die durch ein OTP über SMS oder ein Hardware-Token erzeugt wird. Über eine Zwei-Faktor-Authentifizierung hinaus gibt es noch weitere Maßnahmen für eine höhere Sicherheit durch die Verwendung eines dreistufigen Systems mit einer Kombination aus Authentifizierung durch Benutzername und PIN-Code, mobiler Sicherheit mit einem OTP und biometrischer Sicherheit mit Fingerabdruck oder Iriserkennung.

Ein alternatives Zwei-Faktor-Authentifizierungssystem ist die eingebettete krypto-biometrische Authentifizierung, eine Kombination aus Kryptografie und einer biometrischen Technik. Das biometrische Bild, z. B. ein Fingerabdruck oder ein Iris-Scan, wird sofort mit einem symmetrischen Schlüssel verschlüsselt und mithilfe einer chaotischen 3D-Karte an einen Server gesendet. Die Verschlüsselungsschlüssel werden aus der zufälligen Pixelverteilung im biometrischen Rohbild einiger stabiler biometrischer Merkmale und einem Pseudozahlengenerator extrahiert. Dies geschieht mit mehreren Iterationen und unterschiedlichen Schlüsseln pro Iteration.

###### FinTS und HBCI

Das Home Banking Computer Interface (HBCI) ist ein offener Standard für Bankgeschäfte über das Internet. Die Schnittstelle definiert Übertragungsprotokolle, Nachrichtenformate und Sicherheitsverfahren, die für Home- oder Mobile-Banking verwendet werden können. HBCI ist ein europäischer Standard, der mit den US-Standards IFX (Interactive Financial Exchange) und OFX (Open Financial Exchange) korreliert (Kubicek & Diederich, 2015). Die wichtigsten Eigenschaften von HBCI sind:

* Unabhängigkeit vom Betriebssystem,
* Unterstützung von Konten bei mehreren Banken,
* Nutzung von DES- und RSA-Verschlüsselung und Signaturen, und
* Speicherung des Schlüssels auf einer Chipkarte.

Die FinTS-Spezifikation (Financial Transaction Services) ist der Nachfolger der Spezifikation für den HBCI-3.0-Standard (Deutsche Kreditwirtschaft, 2020). FinTS wurde erstmals 2002 als Version 3.0 veröffentlicht und unterstützt Online-Banking mit SWIFT, Protokollen und Standards für internationale Zahlungssysteme zwischen Tausenden von Finanzinstituten in 194 Ländern, darunter die Zentralbanken der meisten Länder.

Die Version 4.0 aus dem Jahr 2004 enthielt einige Aktualisierungen. Erstens werden alle Daten in das universelle XML-Format (Extensible Markup Language) kodiert. Zweitens werden Daten über die Protokolle HTTP, HTTPS (synchrone, permanente Verbindung zwischen Kunde und Bank) und SMTP (asynchrone, instabile Verbindung zwischen Kunden und Bank) ausgetauscht, um die Integration mit anderen Zahlungssystemen zu erleichtern (und die Kommunikation mit Kunden hinter einer Firewall sicherzustellen). Die zugrunde liegenden Transportprotokolle werden entsprechend dem Anwendungsprotokoll ausgewählt. Während unverschlüsselte Protokolle (wie das PIN/TAN-Verfahren) ein verschlüsseltes HTTPS-Protokoll verwenden müssen, können verschlüsselte Protokolle (wie RAH-7 oder RAH-9) das unverschlüsselte HTTP-Protokoll nutzen.

Im Jahr 2014 wurde die Version 4.1 veröffentlicht. Sie enthält Verbesserungen, die sich aus jahrelanger praktischer Erfahrung ergeben haben. Diese übernahm SEPA (Single Euro Payments Area), eine Selbstregulierungsinitiative des europäischen Bankensektors, der im European Payments Council vertreten ist. Damit werden (technische) Standards festgelegt, die den Zahlungsverkehr in der Europäischen Union standardisieren (BIC, IBAN usw.). Im Jahr 2015 verabschiedete die Europäische Union die überarbeitete Zahlungsdienste-Richtlinie (PSD2), die alle Zahlungsdienstleister der Mitgliedsstaaten bis 2018 umsetzen mussten, um Online-Zahlungen sicherer zu machen und insbesondere die Rechte der Kunden zu schützen.

Die PSD2 wurde 2017 durch technische Regulierungsstandards (2018/389) für die (Zwei-Faktor-)Kundenauthentifizierung ergänzt.

###### Rollen

FinTS unterscheidet zwischen den folgenden Rollen (*Deutsche Kreditwirtschaft*, 2020).

* + Der Nachrichtendienst (Messenger) signiert und überträgt die Nachricht und muss nichts über ihren (möglicherweise verschlüsselten) Inhalt wissen.
  + Der Aussteller signiert den Auftragsteil einer Benutzernachricht und verschlüsselt ihn möglicherweise (und kann auch der Bote sein).

SWIFT

Die 1977 gegründete Society for Worldwide Interbank Financial Transactions, kurz SWIFT, befindet sich im Besitz der Mitgliedsbanken.

* + Der Zeuge signiert den Auftragsteil einer Benutzernachricht zusätzlich zum Aussteller, wenn die Signatur des Ausstellers allein nicht zuverlässig genug ist.
  + Der Intermediär vermittelt zwischen den Kunden und der Bank als technische Schnittstelle, die mit unterschiedlichen Berechtigungen gegenüber der Bank ausgestattet ist (z. B. die des Ausstellers oder des Boten). Die Kommunikation zwischen dem Intermediär und dem Kreditinstitut ist immer verschlüsselt.

###### Verschlüsselung und Signierung

Die Transaktionen werden entweder auf der FinTS-Protokollschicht mit einem Schlüssel verschlüsselt, der auf einer Chipkarte (Bankensignaturkarten, die das SECCOS-Betriebs­system der *Deutschen Kreditwirtschaft* verwenden) oder in einer durch ein Passwort gesicherten Datei gespeichert ist. Alternativ werden sie über das TSL-Protokoll (das dem HTTPS-Protokoll zugrunde liegt) unter Verwendung von PIN/TAN mit indizierten (iTAN) und mobilen (mTAN) Transaktionsnummern, d. h. einmaligen Passwörtern (OTP), verschlüsselt. Hinweis: Die iTAN wurde 2019 durch die EU-Zahlungsverkehrsrichtlinie PSD2 abgeschafft.

PIN (Persönliche Identifikationsnummer)

Die PIN ist ein persönlicher Code, der für die Autorisierung bei Mobiltelefonen, Online-Banking, Zugangspanels und Kartenzahlungen verwendet wird.

TAN (Transaktionsauthentifizierungsnummer)

Die TAN ist eine Art OTP, das beim Online-Banking zur Autorisierung von Finanztransaktionen verwendet wird.

Die Verwendung einer Chipkarte ist die sicherste Option, da alle kryptografischen Verfahren durchgeführt werden, ohne dass der geheime Schlüssel jemals die Chipkarte verlässt, auf der er sicher gespeichert ist. Die PIN wird über das Tastenfeld des Kartenlesers eingegeben. Wenn eine Datei verwendet wird, muss der Schlüssel mit einem von den Benutzenden gewählten Passwort verschlüsselt werden, das nur nach manueller Eingabe zugänglich ist. Das PIN/TAN-Verfahren ist bequemer, z. B. auf Reisen, da es kein Kartenlesegerät benötigt. Bei der Verwendung von PIN/TAN erfolgt die Zugangsautorisierung mit einer etwa vierstelligen PIN, während für die Transaktionsautorisierung eine zusätzliche TAN erforderlich ist. Die TAN bietet mehr Sicherheit, indem sie als eine Art Zwei-Faktor-Authentifizierung fungiert. Wenn Angreifende in den Besitz eines Dokuments oder Geräts gelangt, das eine TAN enthält, ist diese ohne Kenntnis der Anmeldeinformationen, d. h. der PIN, unbrauchbar. Wenn sie wiederum die Anmeldedaten mit der PIN erhalten, können sie ohne eine gültige TAN keine Transaktion durchführen (Kubicek & Diederich, 2015).

Der HBCI-Standard deckt das gesamte Verfahren auf Kunden- und Bankseite ab. Die Verfahren werden mit einer speziellen Chipkarte mit einem 128-Bit-Schlüssel durchgeführt. Hierfür benötigen wir eine spezielle Software und einen Chipkartenleser. Für das PIN/TAN-Verfahren brauchen wie wiederum eine von der Bank ausgegebene Identifikationsnummer und eine Transaktionsnummer. Jede TAN ist nur für eine Transaktion gültig, was diese Methode sicher macht. Zunächst erstellt die Bank eine Liste mit einmaligen TANs für einzelne Benutzende. Diese benötigen den Benutzernamen und das Passwort, um sich anzumelden. Damit können sie zwar auf die Kontoinformationen zugreifen, aber es reicht nicht aus, um Transaktionen durchzuführen. Die Benutzenden können Transaktionen nur tätigen, indem sie eine nicht verwendete TAN aus ihrer individuellen Liste eingeben. Die Bank prüft, ob die TAN zu der Liste gehört, die für die jeweiligen Benutzenden erstellt wurde, und ob sie nicht bereits verwendet wurde. Wenn beides der Fall ist, lässt die Bank die Transaktion zu, andernfalls wird sie abgelehnt. Wenn die TAN-Liste offengelegt oder gestohlen wird, können Benutzer sie ungültig machen, indem sie die Bank benachrichtigt. Dieses System ist anfällig für Phishing-Angriffe, wenn Angreifende das Passwort und einige TANs aus der Liste erlangen. Es bietet außerdem auch keinen Schutz gegen Man-In-The-Middle-Angriffe. Angreifende fangen beispielsweise die TAN während der Übertragung ab und können sie später für eine betrügerische Transaktion verwenden.

Eine iTAN (indizierte TAN) verringert das Risiko von Phishing. Zum Autorisieren einer Transaktion müssen Benutzer eine vordefinierte TAN aus der Liste eingeben. Die Position der TANs in der Liste wird von der Bank nach dem Zufallsprinzip ausgewählt, so dass eine zufällig gewonnene TAN für Angreifende praktisch wertlos ist. Während iTANs immer noch anfällig für MITM-Angriffe sind, könnte mTAN (Mobile TAN) hier Abhilfe schaffen. Bei der Verwendung von mTAN generiert die Bank eine TAN und sendet sie als SMS an die angemeldeten Benutzenden. Diese SMS enthält in der Regel auch Transaktionsdaten, anhand derer die Kunden überprüfen können, dass die Transaktionsdaten beim Versand an die Bank nicht verändert wurden. Die Sicherheit dieser Methode hängt von der Sicherheit des für den Versand der SMS verwendeten Mobilfunknetzes ab.

### Blockchain

Die Blockchain ist wie der Name schon sagt eine Kette von Blöcken. Es gibt einen Anfangsblock, den Genesis-Block, und Blöcke, die auf ihre Vorgänger verweisen. Der „Pfeil“, der auf den Vorgänger zeigt, ist ein Hash, ein Nachweis des gesamten Inhalts des vorherigen Blocks. Es ist eine Adresse, die im Header des Blocks erscheint. Der Rumpf des Blocks besteht aus durchschnittlich 2000 Transaktionen zwischen Bitcoin-Nutzern (Wang et al., 2018).

###### Hashes

Da der Hash des Blocks vom gesamten Inhalt des Blocks abhängt, ändert die einfache Änderung eines Bits den Hash, d. h. der Pfeil des Nachfolgers wird ungültig. Damit die Blöcke weiterhin eine Kette bilden, müssen wir also diesen Pfeil ändern. Dadurch ändert sich der Nachfolger-Hash. Damit die Blöcke weiterhin eine Kette bilden, muss der Pfeil des Nachfolgers geändert werden, und so weiter, wodurch eine Kettenreaktion entsteht. Wenn wir eine Kleinigkeit ändern, z. B. eine Transaktion im ersten Block, müssen alle folgenden Pfeile (Hashes) neu berechnet werden.

###### Mining

Mining macht diese Änderung sehr schwierig, denn nur Blöcke, deren Hashes klein sind, d. h. mit vielen Nullen beginnen, werden vom Netzwerk (d. h. von den prüfenden Knoten) akzeptiert. Es reicht nicht aus, eine Transaktion zu ändern und die neuen Hashes aller Nachfolger zu berechnen. Die Änderung ist so vorzunehmen, dass alle Hashes klein sind! Es ist sehr schwierig, eine solche Änderung zu finden. Derzeit dauert die Suche nach einem solchen Block auf einem normalen Computer eine Milliarde Jahre, im Mining-Netzwerk jedoch nur zehn Minuten. Während Angreifende mit der Suche nach akzeptablen Blöcken gerade erst begonnen haben, hat das Netzwerk bereits mehrere andere Blöcke erstellt, die den potenziellen Angriff unterbinden.

###### Transaktionen

Alle existierenden Bitcoins wurden durch Mining (Schürfen) erzeugt, d. h. sie wurden (durch die Coinbase-Transaktion) als Belohnung an denjenigen vergeben, der einen Block (mit einem kleinen Hash) erzeugt hat. Alle anderen Transaktionen haben einen Input und einen Output. Dem Eintrag wird ein ausreichender Betrag hinzugefügt, um das zu bezahlen, was ausgegeben wird. Sie müssen alles ausgeben! Normalerweise beinhaltet eine Transaktion den Absender als Empfänger, also die Änderung. Das Unerwartete gewinnt am Ende. Der oder die Schürfende nimmt die Transaktion in das Paket auf und verlangt einen Preis dafür (derzeit etwas 40 USD). Die Empfänger werden durch ihre öffentlichen Schlüssel identifiziert und müssen erst beim Ausgeben der Coins nachweisen, dass sie im Besitz des entsprechenden privaten Schlüssels sind, mit dem sie die Transaktion unterzeichnen.

###### Elliptische Kurven

Bitcoin verwendet die Verschlüsselung durch elliptische Kurven, um Transaktionen zu signieren. Diffie-Hellman verwendet endliche Zahlenringe wie {0, 1, ..., m} (z. B. m = 12 für die Uhr und eine Primzahl wie p = 2255 – 19 mit 100 Ziffern in Bitcoin). Das von Bitcoin verwendete Konzept ähnelt dem von Diffie-Hellman, nur dass statt der Zahlen 0, 1, 2 … Zahlenpaare (x, y) verwendet werden, d. h. Punkte einer elliptischen Kurve. Das Schöne an dieser Kurve ist, dass Punkte auf ihr addiert werden können. p + q + r = 0 , wenn diese drei Punkte p, r und q auf der gleichen Linie liegen. Anstatt dieselbe Zahl mehrmals zu multiplizieren, addieren wir denselben Punkt. Es ist einfach, mit dieser Formel Punkte auf diesen endlichen Ringen zu addieren, aber es ist sehr schwierig zu wissen, wie oft ein Punkt auf sich selbst addiert wurde, um den resultierenden Punkt zu erhalten. Die Chiffrierung entspricht der Addition, die Entschlüsselung dem Wissen um diese Anzahl der Additionen.

Schließlich ist das Signaturschema eine Variation der Elgamal-Signatur. Die Signatur zeigt, dass Besitzer des privaten Schlüssels in der Lage waren, eine schwierige Gleichung zu lösen. Es ist praktisch unmöglich, sie ohne diesen privaten Schlüssel zu lösen, der eine Abkürzung (Trapdoor) bietet (Puthal et al., 2018).

### Wahlen

Bei einer Wahl müssen Wählende wie bei jeder sensiblen Transaktion authentifiziert werden, und ihre Stimme muss nicht nur integer sein, sondern auch vertraulich behandelt werden. Folglich muss auch der Stimmzettel anonym sein, d. h. Empfangende dürfen Absendende nicht erkennen, und es gibt keine Verbindung zwischen den Wählenden und deren Stimmzettel. Das Wahlgeheimnis garantiert, dass nur die Wählenden selbst von ihrer Wahl wissen, und niemand sonst.

###### Anonymität versus Integrität

Integrität bezieht sich darauf,

* ob eine Stimme wie beabsichtigt abgegeben wird (was bei einer Wahl auf Papier offensichtlich ist) und
* ob eine Stimme als abgegeben registriert wird und alle Stimmen als registriert gezählt werden (bei Wahlen auf Papier müssen wir uns hierbei auf die zuständigen Wahlhelfer und Prüfer verlassen).

Integrität kann nur auf Kosten der Anonymität erreicht werden, und es müssen Vorkehrungen getroffen werden, um diese so weit wie möglich zu wahren. In einem durchgängig überprüfbaren (oder von Wählenden verifizierbaren) Wahlsystem erhalten alle Wählenden eine verschlüsselte ID, mit der sie auf einer öffentlichen Liste überprüfen können, ob ihre Wahl wahrscheinlich wie vorgesehen abgegeben, registriert und ausgezählt wurde. Je mehr Wählerkontrollen stattfinden und Fehler begangen werden, desto wahrscheinlicher ist es, dass einer von ihnen entdeckt wird (Smart, 2012).

Die Integritätsprüfung erfordert jedoch eine Datenbank, die jede Stimme eines jeden Stimmzettels einzelnen Kandidierenden zuordnet und die sich irgendwann im Besitz der Wahlbehörden befindet. Auch wenn die Integrität sehr wahrscheinlich erreicht werden kann, so kann mit den folgenden Aktionen das Wahlgeheimnis gebrochen werden:

* Identifikation des Stimmzettels anhand seiner ID und der ausgefüllten Zettel,
* Durch Zugriff auf die Zuordnungsdatenbank Kandidaten identifizieren, die einer Stimme entsprechen, und
* Identifikation von Wählenden über ihren Stimmzettel, z. B. anhand ihres Fingerabdrucks auf dem Stimmzettel und durch Vergleich mit einer Datenbank mit Fingerabdrücken.

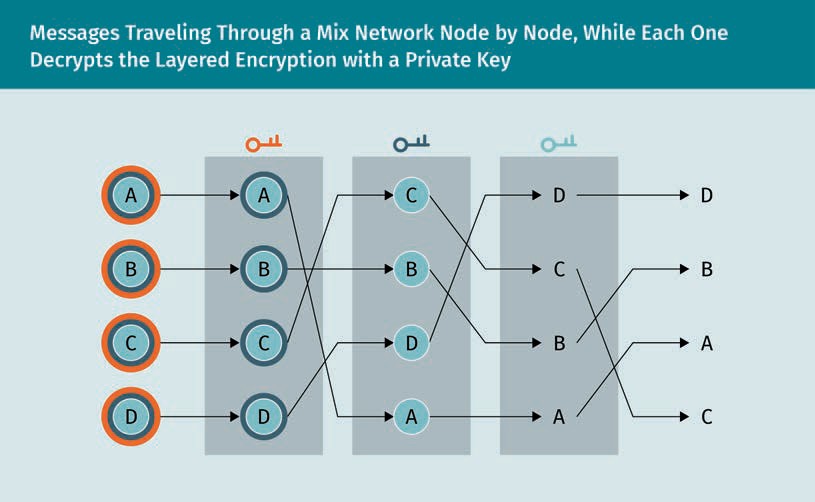
Um sicherzustellen, dass diese Zuordnung zwischen den Stimmen der Stimmzettel und den Kandidierenden während der Stimmabgabe unverändert bleibt, legen sich die Wahlbehörden vor der Wahl darauf fest, indem sie die Informationen jedes Stimmzettels (wie seine ID und die Stimmennummern) und die Zuordnungsdatenbank kryptografisch verschlüsseln. Danach werden diese Hashes veröffentlicht.

###### Integrität und Rückverfolgbarkeit

In vielen Ländern sollte der einer Demokratie zugrunde liegende Wahlvorgang für jeden verständlich sein (anstatt dem Computer vertrauen zu müssen), was den Einsatz von Kryptografie ausschließt. So hat das Bundesverfassungsgericht in seiner Entscheidung vom 3. März 2009 den Einsatz der Wahlcomputer bei der Wahl zum 16. Deutschen Bundestag (und zum Europäischen Parlament) wegen unzureichender Nachvollziehbarkeit für verfassungswidrig erklärt. Das dafür verwendete System war nicht öffentlich, und eine Prüfung der Integrität des Quellcodes war der interessierten Öffentlichkeit nicht gestattet. Die wesentlichen Schritte des Wahlvorgangs und die Ermittlung des Wahlergebnisses müssen durch die Bürger jedoch zuverlässig und ohne besondere Sachkenntnis verifizierbar sein. Dies ergibt sich aus Artikel 38 in Verbindung mit Artikel 20 Absatz 1 und 2 des Grundgesetzes, die vorgeben, dass alle wesentlichen Schritte der Wahl der öffentlichen Kontrolle unterliegen (Bundesverfassungsgericht, 2020). Ein weiteres Beispiel: Namibia war das erste Land in Afrika, das bei den nationalen Wahlen 2014 Wahlautomaten einsetzte. Die Verwendung der Automaten ohne einen Verifizierungsausdruck wurde im Februar 2020 vom Obersten Gerichtshof Namibias für verfassungswidrig erklärt (Ndeunyema, 2020). Stattdessen entscheiden sich die meisten Länder nach wie vor für Wahlen mit Papierstimmzetteln, die von Hand ausgezählt werden. Daher werden anstelle von Wahlautomaten für die Stimmabgabe, ‑registrierung und ‑auszählung Wahlverfahren wie Scantegrity erforscht. Dabei werden zwar Stift und Papier benutzt, aber die Stimmzettel werden kryptografisch mit den Wählenden identifiziert. So wird die Authentizität gewährleistet, aber gleichzeitig die Anonymität gewahrt.

###### Mix-Netzwerke

Mix-Netzwerke sollen die Anonymität einer Wahl wahren. Das 1981 von David Chaum erfundene Verfahren basiert auf der Änderung der Reihenfolge der gesendeten Nachrichten in Kombination mit Public-Key-Kryptografie. Das Netzwerk besteht aus Knoten, deren Aufgabe es ist, eingehende Pakete zwischenzuspeichern, zu permutieren, zu entschlüsseln und zu versenden. Alle an der Kommunikation beteiligten Parteien (Mixer, Sender und Empfänger) haben ein öffentliches und ein privates Schlüsselpaar. Es wird davon ausgegangen, dass jeder Knoten den öffentlichen Schlüssel des anderen Knotens kennt. Eine in einem Knoten empfangene Nachricht wird neu geordnet und in zufälliger Reihenfolge an den nächsten Knoten zurückgeschickt. Das Abhören der Ende-zu-Ende-Kommunikation ist aufgrund der fehlenden Verbindung zwischen Quelle und Ziel der Anfragen somit recht schwierig. Ein Knoten kennt nur seinen Vorgängerknoten, von dem er Nachrichten erhält, und seinen Nachfolgerknoten, an den er Nachrichten sendet.



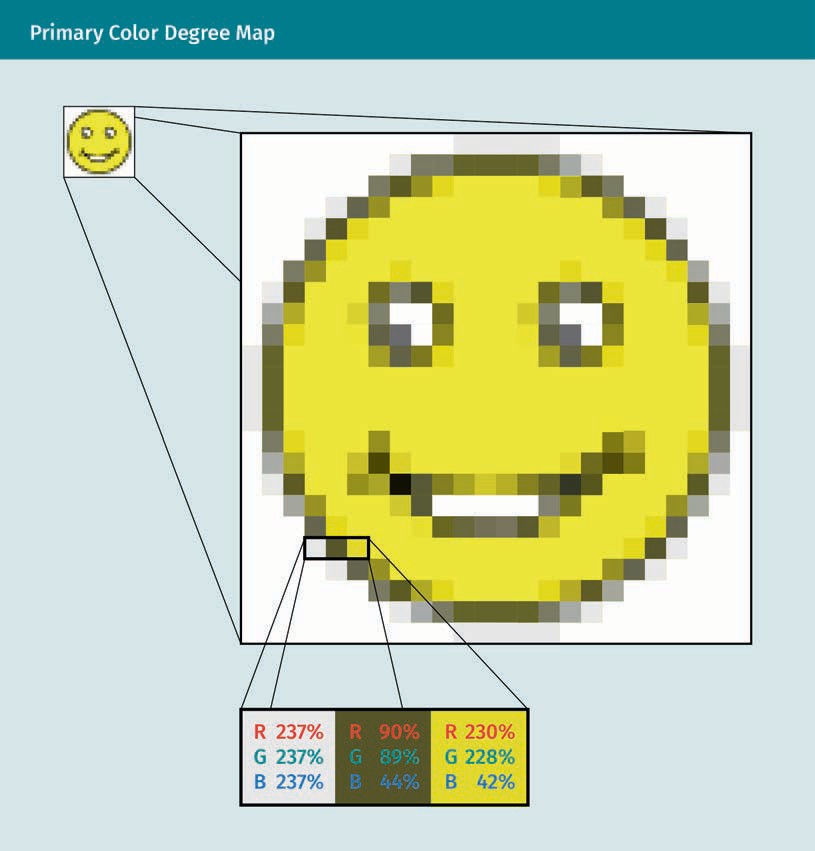
Jede Nachricht wird mit asymmetrischer Kryptografie verschlüsselt und von Knoten zu Knoten übertragen. Die resultierende Chiffre ist geschichtet und die Nachricht befindet sich in der innersten Schicht. Jeder Knoten entschlüsselt die verschlüsselte Nachricht aus seiner eigenen Schicht, um herauszufinden, wohin die Nachricht als nächstes gesendet werden soll.

### Steganografie

Durch Steganografie wird öffentlich gemacht, dass mutmaßlich wichtige Informationen absichtlich versteckt werden. Dies erregt Aufmerksamkeit und bringt interessierte Kryptoanalytiker möglicherweise dagegen auf. Ein Beispiel für Steganografie findet sich in der Geschichte von Herodot, wo Histäus den Kopf seines vertrauenswürdigsten Sklaven rasierte und eine Nachricht auf seinen Kopf Tätowierte, die versteckt war, sobald sein Haar nachgewachsen war (Watkins, 2001).

###### Implementierung

Bei der Datenverarbeitung wird die Nachricht häufig in einem Bild versteckt, aber auch eine Audiodatei ist möglich. Dies kann z. B. verwendet werden, um im Bild selbst einen Urheberrechtshinweis der Eigentümerschaft eines Bildes zu verstecken (Coded Anti-Piracy). Da eine typische Bilddatei Hunderte von Kilobytes umfasst, können einige wenige Bytes verändert werden, um eine geheime Nachricht zu übermitteln, ohne dass dies auffällt. Die geheimen Informationen werden normalerweise in den Randbereichen des Bildes gespeichert, z. B. in einer Bitmap.



Die versteckte Nachricht kann dann als Bitstrom gespeichert werden, z. B. in dem niedrigsten Bit der acht Bits für jede Primärfarbe jedes Pixels. Für jedes Byte als Zahl b zwischen 0 und 255 ist das verborgene Bit 1 genau dann, wenn b ungerade ist.

Steganografie

Aus dem Griechischen für „verborgen“, ist die Kunst, eine Nachricht zu verbergen, indem sie in eine andere Nachricht eingebettet wird, so dass niemand außer dem beabsichtigten Empfänger von der Existenz der Nachricht weiß.

###### Steganalyse

Um den verborgenen Inhalt eines Mediums zu lesen, muss er aufgespürt und extrahiert werden. Steganalyse ist die Technik, den versteckten Inhalt in einem Medium aufzuspüren und zu extrahieren, ohne zu wissen, wo der versteckte Inhalt ist oder welcher Algorithmus verwendet wurde.

Versteckte Inhalte können statistisch aufgespürt werden, indem wir nach signifikanten Abweichungen zu ähnlichen Referenzdaten ohne versteckte Inhalte suchen. Damit diese Abweichungen statistisch weniger signifikant sind, sollte der versteckte Inhalt selbst statistisch zufällig und musterfrei sein. Dies wird durch Verschlüsselung erreicht.

Die Substitution mit dem kleinsten signifikanten Bit kann beispielsweise mit einem Histogramm erkannt werden, einem Diagramm mit Balken für jede Farbe im Bild, deren Höhe proportional zur Anzahl der Pixel dieser Farbe ist. Da ursprünglich ungerade Bytes nur um eins verringert werden und gerade Bytes nur um eins erhöht werden können, werden die Höhen jedes Paares benachbarter Balken, d. h. die Häufigkeit der Pixel benachbarter Farben, deutlich näher an ihrem Mittelwert liegen als im Originalbild. Die Substitution mit dem kleinsten signifikanten Bit kann also noch zuverlässig erkannt werden, wenn die versteckte Information weniger als ein Prozent des gesamten Bildes ausmacht.

Um dieser Erkennungsmethode entgegenzuwirken, addiert oder subtrahiert das Least Significant Bit Matching zufällig eins zum Byte. Wenn das Bit der versteckten Nachricht „1“ ist, dann entscheidet ein Münzwurf, ob das Byte inkrementiert oder dekrementiert wird (anstatt das Byte zu dekrementieren, wenn es ungerade ist, und es zu dekrementieren, wenn es gerade ist). Wie bei der Substitution mit dem kleinsten signifikanten Bit ist das versteckte Bit für jedes Byte mit einer Zahl *b* zwischen 0 und 255 genau dann 1, wenn b ungerade ist. Allerdings liegen die Höhen jedes Paares benachbarter Balken im Histogramm nicht mehr signifikant näher an ihrem Mittelwert als statistisch erwartet. Das Problem der Aufdeckung des Kerckhoffs‘schen Prinzips, dass Angreifende, die den stenografischen Algorithmus zur Speicherung der versteckten Informationen kennen, diese nicht entdecken können, ist in der Steganografie schwieriger zu lösen als in der Kryptografie (Watkins, 2001).

###### Anwendung

Das Kommandozeilen-Tool Tomb von Roio (2020) (alias Jaromil) ist ein Linux-Shell-Skript. Es verschlüsselt Ordner mit dm-crypt, das Teil des Kernels ist. Die verschlüsselten Ordner werden Tombs genannt, während dm-crypt sie Container nennt. Diese können mit ein paar Befehlen auf der Kommandozeile erstellt und in das laufende System integriert werden, wofür allerdings teilweise administrative Rechte erforderlich sind. Um die höchste Sicherheit zu gewährleisten, sollten Tomb und Schlüssel nicht auf demselben Gerät gespeichert werden. Wenn sich das Tomb auf einem PC oder Notebook befindet, kann das Tomb auf einem USB-Stick gespeichert werden. Wenn der Schlüssel jedoch auf demselben Gerät wie das Tomb gespeichert werden muss, kann Tomb den Schlüssel mithilfe von Steganografie in einem JPEG-Bild verstecken. Dies verbirgt den Schlüssel vor unbefugten Blicken und hilft, sich an den Speicherort des Schlüssels zu erinnern (Roio, 2020).

Tomb kann Ordner anzeigen (mounten), die andere Anwendungen zur Laufzeit benötigen, z. B. das Postfach eines E-Mail-Programms. Zu diesem Zweck benötigt es das Paket steghide. Der Schlüssel kann in einem kleinen JPEG-Bild von tomb bury versteckt und von tomb exhume extrahiert werden.

|  |  |
| --- | --- |
| Verbergen eines Schlüssels in einem Bild | tomb bury secret.tomb -k picture.jpg |
| Abrufen eines Schlüssels aus einem Bild | tomb exhume picture.jpg -k secret.key |
| Öffnen eines Tombs mit einem Steganografie-Schlüssel | tomb open secret.tomb -k picture.jpg |

### Das Tor-Projekt

Tor ist eine Implementierung des Onion-Routings der zweiten Generation zur Gewährleistung der Anonymität im Internet. Es wurde ursprünglich vom US Naval Research Laboratory gefördert, dann aber zwischen Ende 2004 und 2005 von der Electronic Frontier Foundation (EFF) unterstützt (Tor Project, o. J.).

###### Verbindung zum Netzwerk

Um sich mit dem Netzwerk zu verbinden, ruft jeder Client eine Liste von Tor-Knoten von einem Server ab, wählt automatisch einen zufälligen Pfad (der sich nach einer Weile ändern kann) und baut eine Verbindung auf, in dem jeder Knoten nur seinen Vorgänger und seinen Nachfolger kennt.

Der erste Knoten der Verbindung kennt die angeforderte IP-Adresse. Aber ab dem zweiten Knoten erfolgt die Aushandlung über die bereits aufgebaute Teilverbindung, so dass der zweite Knoten z. B. nur die IP-Adresse des ersten Knotens (und eventuell des dritten Knotens) kennt. Die weiterzuleitenden Pakete werden durch einen (zum Zeitpunkt der Erstellung der Verbindung gewählten) Code identifiziert, der dem Eigentümer der Verbindung zugeordnet ist. Jeder Knoten der Verbindung erhält seinen eigenen privaten (asymmetrischen) Schlüssel, der mit dem öffentlichen (asymmetrischen) Schlüssel des jeweiligen Knotens verschlüsselt ist.

Paketaustausch zwischen Client und Server

Bevor der Client ein TCP-Paket an den Server sendet, verschlüsselt er es so oft, wie es Knoten gibt (a), mit dem öffentlichen Schlüssel des letzten Knotens mit der Nummer n; (b) mit dem öffentlichen Schlüssel des vorletzten Knotens mit der Nummer n - 1; und (c) mit dem Schlüssel von n - 2. Beim letzten Mal verschlüsselt er mit dem des ersten Knotens.

An diesem Punkt umschließen alle Schichten der Zwiebel das TCP-Paket. Die Zwiebel wird geschält, wenn der Client sie an die von ihm erstellte Verbindung sendet. Zunächst entschlüsselt der erste Server in der Verbindung das Paket mit Schlüssel Nummer 1 und sendet es an den zweiten Server. Anschließend entschlüsselt der zweite Server dieses Paket mit Schlüssel Nummer 2 und sendet es an den dritten Server. Am Ende entschlüsselt der letzte Server dieses Paket mit seinem eigenen privaten Schlüssel Nummer n und erhält das ursprüngliche Klartextpaket.

Onion-Routing

In dieser Kette kennt jeder Knoten nur seinen unmittelbaren Vorgänger und seinen Nachfolger, und der gesamte Datenverkehr zwischen den beiden Endpunkten ist für jeden Knoten außer den Endpunkten nicht zu entziffern.

Proxy

Benutzende im Tor-Netzwerk können ihren Webbrowser so einrichten, dass er einen persönlichen Proxy-Server für den Zugang zu Tor verwendet (z. B. Privoxy), um sich z. B. mit IUBH.de zu verbinden. Folgende Schritte werden ausgeführt:

* + 1. Ihr Webbrowser sendet die HTTP-Anfrage an Privoxy.
    2. Privoxy entfernt die nicht-anonymen Informationen und leitet die Informationen über SOCKS an den Tor-Client weiter.
    3. Der Tor-Client baut eine Verbindung auf (sofern er dies nicht bereits getan hat), verschlüsselt die zu sendenden Daten und leitet sie an den ersten Knoten weiter.
    4. Der erste Knoten entschlüsselt einen Teil des Umschlags (Envelope) und leitet die Daten an den Exit-Knoten weiter.
    5. Dieser Exit-Knoten sendet die Anfrage an IUBH.de.

Damit die IUBH-Website eine Verbindung mit den Benutzenden herstellen kann, werden die Schritte in umgekehrter Reihenfolge ausgeführt.

Implementierungen

Die benutzerfreundlichste Möglichkeit zur Nutzung des Tor-Netzwerks ohne fortgeschrittene Computerkenntnisse ist der Tor-Browser. Dieser Webbrowser, der für Linux, Microsoft Windows und Mac verfügbar ist, passt Mozilla Firefox so an, dass er im Netzwerk und auf dem Computer so wenige Spuren wie möglich hinterlässt. Daraus folgt unter anderem:

* + Der Browserverkehr wird standardmäßig über diejenige Tor-Instanz umgeleitet, die bei der Initialisierung gestartet wurde,
  + Es gibt keinen Browserverlauf,
  + DuckDuckGo ist als Standardsuchmaschine konfiguriert,
  + Die Erweiterungen NoScript und HTTPS-Everywhere sind standardmäßig aktiviert.

Tails (The Amnesic Incognito Live System) ist ein Betriebssystem, das standardmäßig das Tor-Netzwerk verwendet. Es ist so konzipiert, dass es keine Spuren auf dem verwendeten Computer hinterlässt. Es ist für den Betrieb auf Wechseldatenträgern (wie USB-Laufwerken) ausgelegt und basiert auf der Linux-Distribution Debian.

Zusammenfassung

Das Online-Banking-Protokoll FinTS standardisiert den Datenaustausch zwischen Kunden und ihrer Bank mit dem XML-Format und gewährleistet die Umsetzung der bewährten Methoden (Best Practices) der Kryptografie wie zum Beispiel der Speicherung der geheimen Schlüssel auf einer Chipkarte. Diese verlassen die Karte nie, sondern werden nur benutzt zur Beantwortung von Abfragen, welche die Kenntnis des Schlüssels voraussetzen, zur permanenten Verschlüsselung und Authentifizierung aller sensiblen Daten und zur Verwendung bewährter Verschlüsselungsalgorithmen.

Blockchain ersetzt einen vertrauenswürdigen Dritten durch eine Datenbank mit Einträgen, die sukzessive aufeinander verweisen. Jeder Verweis (Zeiger) ist ein Hash des gesamten vorherigen Eintrags und gewährleistet die Integrität der Datenbank. Eine Änderung des Eintrags hat also eine Änderung des Hashs zur Folge, wodurch die Zeiger und die gesamte Kette ungültig werden. Da die Hashes viele führende Nullen haben müssen, erfordert das Auffinden gültiger Einträge oder das Mining (Schürfen) eine beträchtliche Rechenleistung und wird praktisch unmöglich.

Die Verifizierbarkeit der Stimmauszählung bei einer Wahl mit Kryptografie stellt zusätzliche Anforderungen an ihre Durchführung, denn die Stimmabgabe muss anonym sein und die Auszählung muss für die durchschnittlichen Wählenden nachvollziehbar sein. Während die Kryptografie eine Nachricht durch reversible Verschlüsselung verbirgt, versteckt die Steganografie sie durch Einbettung in eine Klartext­nachricht. Statistische Analysen können das Vorhandensein verborgener Informationen verraten. Wenn sie jedoch sorgfältig versteckt werden, werden sie selten entdeckt. Schließlich erreicht das Tor-Netzwerk eine anonyme Datenübertragung im Internet durch Onion-Routing, bei dem der Datenverkehr eine Kette von Knoten durchläuft, bei der jeder Knoten nur seinen unmittelbaren Vorgänger und Nachfolger kennt und der gesamte Datenverkehr zwischen den beiden Endpunkten für jeden Knoten außer den Endpunkten nicht zu entziffern ist.



# Anhang 1

## Literaturverzeichnis

Literaturverzeichnis

Abdalla, M., Benhamouda, F., & Pointcheval, D. (2016). Public-key encryption indistinguishable under plaintext-checkable attacks [Public-Key-Verschlüsselung nicht unterscheidbar unter Klartext-überprüfbaren Angriffen]. *IET Information Security, 10*(6), 288–303.

Acdx. (2008). Digital signature diagram [Diagramm der digitalen Signatur]. https://commons.wikimedia.org/w/index.php? title=File:Digital\_Signature\_diagram.svg&oldid=362770490

Ajtai, M., & Dwork, C. (1997). A public-key cryptosystem with worst-case/average-case equivalence [Ein Public-Key-Kryptosystem mit Worst-Case/Average-Case-Äquivalenz]. In *STOC ’97 (El Paso, TX)* (S. 284–293). ACM.

Aspect, A., Grangier, P., & Roger, G. (1982). Experimental realization of Einstein-Podolsky-Rosen-Bohm Gedankenexperiment: A new violation of Bell’s inequalities [Experimentelle Umsetzung des Einstein-Podolsky-Rosen-Bohm Gedankenexperiments: Eine neue Verletzung der Bellschen Ungleichungen]. *Physical Review Letters*, *49*(2), 91–94. https://doi.org/10.1103/PhysRevLett.49.91

Bauer, C. P. (2016). *Secret history: The story of cryptology* [Geheime Historie: Die Geschichte der Kryptologie]. CRC Press.

Bauer, F. L. (2000). *Entzifferte Geheimnisse: Methoden und Maximen der Kryptologie.*

Springer.

Bell, J. S. (1964). On the Einstein Podolsky Rosen paradox [Über das Einstein-Podolsky-Rosen-Paradoxon]. *Physics Physique Fizika*, *1*(3), 195–200. https://doi.org/10.1103/PhysicsPhysiqueFizika.1.195

Bellare, M., & Rogaway, P. (2000). Encode-Then-Encipher encryption: How to exploit nonces or redundancy in plaintexts for efﬁcient cryptography [Kodieren-dann-Chiffrieren-Verschlüsselung: Wie man Nonces oder Redundanz in Klartexten für efﬁziente Kryptografie ausnutzt]. In T. Okamoto (Hrsg.), *Advances in Cryptology–ASIACRYPT 2000 (*S. 317–330). Springer. https://doi.org/ 10.1007/3-540-44448-3\_24

Borst, J., Preneel, B., & Rijmen, V. (2001). Cryptography on smart cards [Kryptografie auf Chipkarten]. *Computer Networks*, *36*(4), 423–435.

Braden, R. (1989). *Requirements for internet hosts–communication layers* [Anforderungen für Internet-Hosts-Kommunikationsschichten]. Internet Engineering Task Force. http//tools.ietf.org/html/rfc1122

Brassard, G., & Hoyer, P. (1997). An exact quantum polynomial-time algorithm for Simon’s problem [Ein exakter Quanten-Polynomialzeit-Algorithmus für das Simon-Problem]. *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, 12–23.

Brumley, D., & Boneh, D. (2005). Remote timing attacks are practical [Rechenzeitangriffe aus der Ferne sind praktisch durchführbar]. *Computer Networks*, *48*(5), 701–716.

Bundesverfassungsgericht. (2020). BVerfG, Urteil des Zweiten Senats vom 03. März 2009, 2 BvC 3/07. Abgerufen am 7. Juli 2020 von https://[www.bundesverfassungsgericht.de/](http://www.bundesverfassungsgericht.de/) SharedDocs/Entscheidungen/EN/2009/03/cs20090303\_2bvc000307en.html

Buonafalce, A. (2014). Alberti cipher disk [Alberti-Chiffrierscheibe]. https://commons.wikimedia.org/w/index.php? title=File:Alberti\_cipher\_disk.JPG&oldid=121453667

Literaturverzeichnis

Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudonyms [Nicht zurückverfolgbare elektronische Post, Absenderadressen und digitale Pseudonyme]. *Communications of the ACM*, *24*(2), 84–90. https://doi.org/10.1145/358549.358563

Chen, L., Takabi, H., & Le-Khac, N.-A. (2019). *Security, privacy, and digital forensics in the cloud* [Sicherheit, Datenschutz und digitale Forensik in der Cloud]. John Wiley & Sons.

Corbellini, A. (2015a). Elliptic curve cryptography: A gentle introduction [Elliptische-Kurven-Kryptografie: Ein leichter Einstieg]. https:// andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

Corbellini, A. (2015b). Implementation of ECDHE in ‘Python‘ [Implementierung von ECDHE in ‚Python‘]. Abgerufen am 7. Juli 2020 von https://github.com/andreacorbellini/ecc/blob/master/scripts/ecdhe.py

Crypto, M. (2005). DES-f-function [DES-f-Funktion]. https://commons.wikimedia.org/w/index.php? title=File:DES-f-function.png&oldid=141155336

CrypTool-Projekt. (2008). Skytale3d de. https://commons.wikimedia.org/w/index.php? title=File:Skytale3d\_de.png&oldid=154927861

Daemen, J., & Rijmen, V. (1999). AES proposal: Rijndael [AES-Vorschlag: Rijndael]. <http://www.cs.miami.edu/home/> burt/learning/Csc688.012/rijndael/rijndael\_doc\_V2.pdf

Daemen, J., & Rijmen, V. (2002). *The design of Rijndael* [Der Aufbau von Rijndael]. Springer. https://doi.org/ 10.1007/978-3-662-04722-4

Datenschutz. (2020). https://datenschutz.bund.de/

Davies, D. (1997). A brief history of cryptography [Eine kurze Geschichte der Kryptografie]. *Information Security Technical Report*, *2*(2), 14–17. https://doi.org/10.1016/S1363-4127(97)81323-4

de Leeuw, K. M. M., & Bergstra, J. (2007). *The History of information security: A comprehensive handbook* [Die Geschichte der Informationssicherheit: Ein umfassendes Handbuch]. Elsevier.

Deutsche Kreditwirtschaft. (2020). FinTS. Abgerufen am 7. Juli 2020 von https://www.hbcizka.de/

Di, T., Hillery, M., & Zubairy, M. S. (2004). Cavity QED-based quantum walk [Cavity-QED-basierte Quantenwanderung]. *Physical Review A*, *70*(3), 032304. https://doi.org/10.1103/PhysRevA.70.032304

IBM. (n. d.). Authentication [Authentisierung]. In *Dictionary of IBM & Computing Terminology*. Abgerufen am 7. Juli 2020 von https://[www.ibm.com/ibm/history/documents/pdf/glossary.pdf?cv=1](http://www.ibm.com/ibm/history/documents/pdf/glossary.pdf?cv=1)

Difﬁe, W., & Hellman, M. (1976). New directions in cryptography [Neue Wege in der Kryptografie]. *IEEE Transactions on Information Theory*, *22*(6), 644–654.

DNSCrypt. (o. D.). Offizielle Website. https://dnscrypt.info/

Dooley, J. F. (2013). *A brief history of cryptology and cryptographic algorithms* [Eine kurze Geschichte der Kryptologie und kryptografischer Algorithmen].Springer International.

Driscoll, M. (2019). *Illustrated TLS connection 1.3 with each download explained* [Illustrierte TLS-Verbindung 1.3 mit Erklärung jedes Downloads]. https// tls13.ulfheim.net/

Ducklin, P. (2013). *Android random number ﬂaw implicated in bitcoin thefts* [Zufallszahlenfehler in Android bei Bitcoin-Diebstählen involviert]. Sophos Ltd. https.//nakedsecurity.sophos.com/2013/08/12/android-random-number-ﬂaw-implicated-in-bitcoin-thefts/

Dukhovni, V. (2014). *Opportunistic security: Some protection most of the time* [Opportunistische Sicherheit: die meiste Zeit etwas Schutz]. Internet Engineering Task Force. http//tools.ietf.org/html/rfc7435

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms [Ein Kryptosystem mit öffentlichem Schlüssel und ein auf diskreten Logarithmen basierendes Signatursystem]. In *Advances in Cryptology (Santa Barbara, Calif., 1984)* (Bd. 196, S. 10–18). Springer. https://doi.org/10.1007/3-540-39568-7\_2

FIDO2. (2020). https://ﬁdoalliance.org/ﬁdo2/ﬁdo2-web-authentication-webauthn/

Forma Estudio. (2007). Rijndael Animation v.4. Abgerufen am 7. Juli 2020 von https:// formaestudio.com/portfolio/aes-animation/

Friedl, S. J. (2005). An illustrated guide to IPsec [Ein illustrierter Leitfaden zu IPsec]. Abgerufen am 7. Juli 2020 von http// [www.unixwiz.net/techtips/iguide-ipsec.html](http://www.unixwiz.net/techtips/iguide-ipsec.html)

Goldreich, O., Goldwasser, S., & Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems [Public-Key-Kryptosysteme aus Gitterreduktionsproblemen]. In B. S. Kaliski (Hrsg.), *Advances in Cryptology–CRYPTO ’97* (S. 112– 131). Springer.

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof systems [Die Wissenskomplexität von interaktiven Nachweissystemen]. *STOC85: Annual ACM Conference on Theory of Computing*, 291–304.

Goldwasser, S., Micali, S., & Tong, P. (1982). Why and how to establish a private code on a public network [Warum und wie man einen privaten Code in einem öffentlichen Netzwerk einrichtet]. *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, 134–44.

Gordon, D. M. (1993). Discrete logarithms in gf(p) using the number ﬁeld sieve [Diskrete Logarithmen in gf(p) unter Verwendung des Zahlenfeldsiebes]. *SIAM Journal on Discrete Mathematics*, *6*(1), 124–138. https://doi.org/10.1137/0406010

Grau, S. (2018a). Elliptic curve plotter [Elliptischer Kurvenplotter]. <http://www.graui.de/code/elliptic2/> Grau, S. (2018b). Modular function plotter [Modularer Funktionsplotter]. <http://graui.de/code/ffplot/>

Gringer. (2011, October 26). Rgb-raster-image [RGB-Rasterbild]. https://commons.wikimedia.org/wiki/ File:Rgb-raster-image.svg

Heartbleed. (2020). In *Wikipedia*. https://en.wikipedia.org/wiki/Heartbleed

Heer, T. (2008). Introduction to network security [Einführung in die Netzwerksicherheit]. In A. Gurtov (Hrsg.), *Host Identity Protocol (HIP)* (S. 11–42). Wiley & Sons. https://doi.org/10.1002/9780470772898.ch2

Literaturverzeichnis

Heys, H. M. (2002). A tutorial on linear and differential cryptanalysis [Ein Tutorium zur linearen und differentiellen Kryptoanalyse]. *Cryptologia*, *26*(3), 189–221.

Hosenball, M., & Dunham, W. (2014, April 12). *White House, spy agencies deny NSA exploited ‘Heartbleed’ bug* [Weißes Haus und Geheimdienste bestreiten, dass die NSA den Heartbleed-Bug ausgenutzt hat].Reuters. https://www.reuters.com/article/us-cybersecurity-internet-bug-nsa/white-house-spy-agencies-deny-nsa-exploited-heartbleed-bug-idUSBREA3A1XD20140411

Hounsel, A., Borgolte, K., Schmitt, P., Holland, J., & Feamster, N. (2020). Comparing the effects of DNS, DoT, and DoH on web performance [Vergleich der Auswirkungen von DNS, DoT und DoH auf die Web-Performance]. *Proceedings of The Web Conference 2020 (WWW ’20)*. Association for Computing Machinery, 562–572. https://doi.org/ 10.1145/3366423.3380139

IANA. (2020). Service Names Port Numbers [Dienstnamen Portnummern].https://www.iana.org/assignments/servicenames-port-numbers/service-names-port-numbers.xhtml

ISO. (Juni 2005). ISO/IEC 17799:2005: Information technology–Security techniques– Code of practice for information security management [Informationstechnologie - Sicherheitstechniken - Verhaltenskodex für das Management der Informationssicherheit]. Abgerufen am 6. Juni 2020 von https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/96/39612.html

Jagetiya, A., & Krishna, C. R. (2020). Evolution of information security algorithms [Entwicklung von Algorithmen der Informationssicherheit]. In D. Goyal., S. Balamurugan, S.-L. Peng, & O. P. Verma (Hrsg.), *Design and Analysis of Security Protocol for Communication (*S. 29–77). Wiley Online Library.

Kku. (2019). Web of trust-en. https://commons.wikimedia.org/w/index.php? title=File:Web\_of\_Trust-en.svg&oldid=375289053

Kochis, P. C. (1996). Timing attacks on implementations of Difﬁe-Hellman, RSA, DSS, and other systems [Rechenzeitangriffe auf Implementierungen von Difﬁe-Hellman, RSA, DSS und anderen Systemen]. *Annual International Cryptology Conference*, 104–113.

Krawczyk, H., Bellare, M., & Canetti, R. (1997). *RFC2104: HMAC: Keyed-hashing for message authentication* [Keyed-Hashing für die Nachrichtenauthentifizierung]. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/ rfc2104

Kubicek, H., & Diederich, G. (2015). Sicherungsverfahren im Online-Banking im Vergleich In H. Kubicek & G. Diederich (Hrsg.), *Sicherheit im Online-Banking: PIN/TAN und HBCI im magischen Dreieck aus Sicherheit, Kosten und einfacher Bedienbarkeit* (S. 7–19). Springer Fachmedien. https://doi.org/10.1007/978-3-658-09960-2\_2

Kumar, H., Kumar, S., Joseph, R., Kumar, D., Singh, S. K. S., Kumar, P., & Kumar, H. (2013). Rainbow table to crack password using MD5 hashing algorithm [Rainbow-Table zum Knacken von Passwörtern mit dem MD5-Hash-Algorithmus]. *2013 IEEE Conference on Information & Communication Technologies*, 433–439.

Lamport, L. (1978). The implementation of reliable distributed multiprocess systems [Die Implementierung von zuverlässigen verteilten Multiprozesssystemen].

*Computer Networks*, *2*(2), 95–114. https://doi.org/10.1016/0376-5075(78)90045-4

L’Ecuyer, P., & Simard, R. (2007). TestU01: A C library for empirical testing of random number generators [Eine C-Bibliothek für empirische Tests von Zufallszahlengeneratoren]. *ACM Transactions on Mathematical Software*, *33*(4), 1–40.

Lee, H. K., Malkin, T., & Nahum, E. (2007). Cryptographic strength of ssl/tls servers: Current and recent practices [Kryptografische Stärke von SSL/TLS-Servern: Aktuelle und neue Verfahren]. *Proceedings of the 7th ACM SIGCOMM Conference on internet Measurement*, 83–92. https://doi.org/10.1145/1298306.1298318

Lenstra, A. K. (2006). Key lengths. In H. Bidgoli (Hrsg.), *Handbook Of Information Security, Volume 1: Key Concepts, Infrastructure, Standards And Protocols* [Handbuch der Informationssicherheit, Band 1: Schlüsselkonzepte, Infrastruktur, Standards und Protokolle]. Wiley.

Lenstra, A., & Verheul, E. (2001). Selecting cryptographic key sizes [Kryptografische Schlüsselgrößen auswählen]. *Journal of Cryptology*, *14*, 255–293

List of random number generators [Liste von Zufallszahlengeneratoren]. (2. Juli 2020). In *Wikipedia*. https://en.wikipedia.org/ wiki/List\_of\_random\_number\_generators

Manualzz. (2020). Security articles from Wikipedia [Wikipedia-Artikel zum Thema Sicherheit]. https://manualzz.com/doc/ 43781390/security-articles-from-wikipedia

McEliece, R. J. (1978). A public key cryptosystem based on algebraic coding theory [Ein Public-Key-Kryptosystem auf der Grundlage der algebraischen Kodierungstheorie]. *DSN Progress Report*, 42–44.

Melnikov, A. (2011). *RFC6331: Moving digest-md5 to historic* [RFC6331: Digest-md5 wird historisch]. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc6331

Menon-Sen, A., Williams, N., Melnikov, A., & Newman, C. (2010). *Salted challenge response authentication mechanism (SCRAM) SASL and GSS-API mechanisms* [Challenge-Response-Authentifizierungsmechanismus mit Salt (SCRAM) SASL- und GSS-API-Mechanismen]. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc5802

Mockapetris, P. (1987). *Domain names–implementation and speciﬁcation* [Domainnamen – Implementierung und Spezifizierung]. Internet Engineering Task Force. http//tools.ietf.org/html/rfc1035

Neuman, B. C., & Ts’o, T. (1994). Kerberos: An authentication service for computer networks [Kerberos: Ein Authentifizierungsdienst für Computernetzwerke]. *IEEE Communications Magazine*, *32*(9), 33–38.

Ndeunyema, N. (2020, February 17). *Vote, but you cannot verify: The Namibian supreme court’s presidential election decision* [Wählen, aber nicht nachprüfen: Die Entscheidung des Obersten Gerichtshofs von Namibia zur Präsidentschaftswahl]. Oxford Human Rights Hub. https:// ohrh.law.ox.ac.uk/vote-but-you-cannot-verify-the-namibian-supreme-courts-presidential-election-decision/

National Institute for Standards and Technology [NIST]. (2000). *AES competition* [AES-Wettbewerb]. https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-

crypto-projects/aes-development

Pixis. (2019). Kerberos en active directory [Kerberos in Active Directory]. https:// beta.hackndo.com/kerberos/

Literaturverzeichnis

Primepq. (13. Dezember 2008). Red de mezcla. https://commons.wikimedia.org/wiki/ File:Red\_de\_mezcla.png

Project Gutenburg & Webster's Unabridged Dictionary (1913) [OPTED]. (o. D.). *The online plaintext English dictionary* [Das Online-Klartext-Wörterbuch Englisch].

Puthal, D., Malik, N., Mohanty, S. P., Kougianos, E., & Yang, C. (2018). The blockchain as a decentralized security framework [future directions] [Die Blockchain als dezentrales Sicherheits-Framework [zukünftige Ausrichtungen]]. *IEEE Consumer Electronics Magazine*, *7*(2), 18–21. https://doi.org/10.1109/MCE.2017.2776459

RadioFan. (2010). Enigma rotor wiring. https://commons.wikimedia.org/w/index.php? title=File:Enigma\_rotor\_wiring.png&oldid=260095687

Ramsdell, B. (2009). *Secure/multipurpose internet mail extensions (s/mime) version 3.1*. Internet Engineering Task Force. http//tools.ietf.org/html/rfc3851

Rao, D. S., Kour, G., & Jyoti, D. (2011). One time password security through cryptography for mobile banking [Sicherheit von Einmal-Passwörtern durch Kryptografie für mobiles Banking]. *International Journal of Computer Technology and Applications*, *2*(5), 1563–1567.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems [Ein Verfahren zur Erlangung digitaler Signaturen und Public-Key-Kryptosysteme]. *Communications of the ACM*, *21*(2), 120–126.

Roio. (2020). A minimalistic commandline tool to manage encrypted volumes aka The Crypto Undertaker [Ein minimalistisches Kommandozeilen-Tool zur Verwaltung verschlüsselter Volumes, auch bekannt als The Crypto Undertaker]. Abgerufen am 1. Juli 2020 von https://github.com/dyne/Tomb

Schell, B., & Martin, C. (2006). *Webster’s new world hacker dictionary*. Wiley.

Schnorr, C.-P. (1991). Factoring integers and computing discrete logarithms via Diophantine approximation [Faktorisierung ganzer Zahlen und Berechnung diskreter Logarithmen durch diophantische Approximation]. In *Advances in cryptology–EUROCRYPT ’91* (Bd. 547, S. 281–293). Springer. https://doi.org/10.1007/3-540-46416-6\_24

Schoof, R. (1995). Counting points on elliptic curves over ﬁnite ﬁelds [Zählen von Punkten auf elliptischen Kurven über endlichen Feldern]. *Journal de Théorie Nombres de Bordeaux, 7*(1), 219–254. [http://jtnb.cedram.org/item?](http://jtnb.cedram.org/item) id=JTNB\_1995 7\_1\_219\_0

Shannon, C. E. (1949). Communication theory of secrecy systems [Kommunikationstheorie von Geheimhaltungssystemen]. *Bell System Technical Journal*, *28*(4), 656–715. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring [Algorithmen für Quantenberechnungen: Diskrete Logarithmen und Faktorisierung]. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134.

Smart, M. J. (2012). *Anonymity vs. traceability: Revocable anonymity in remote electronic voting protocols* [Anonymität vs. Rückverfolgbarkeit: Widerrufbare Anonymität in elektronischen Fernwahlprotokollen]. [Doktorarbeit, University of Birmingham]. https://etheses.bham.ac.uk/id/eprint/3386/

Snipes, R. (8. Oktober 2019). Thunderbird, Enigmail and OpenPGP [Thunderbird, Enigmail und OpenPGP]. *The Thunderbird Blog*. https//blog.thunderbird.net/2019/10/thunderbird-enigmail-and-openpgp/

Steane, A. (1998). Quantum computing [Quantencomputing]. *Reports on Progress in Physics*, *61*(2), 117–173. https://doi.org/10.1088/0034-4885/61/2/002

Theveßen, E., Müller, P. F., & Stoll, U. (2. Februar 2020). *Operation Rubikon* [Video]. ZDF. https://[www.zdf.de/politik/frontal-21/operation-rubikon-100.html](http://www.zdf.de/politik/frontal-21/operation-rubikon-100.html)

Tor Project. (n. d.). https://[www.torproject.org/](http://www.torproject.org/)

Tudhope, G. V. (1993). *The discovery of Francis Bacon’s cipher signatures in James Anderson’s constitutions of the Free Masons* [Die Entdeckung von Francis Bacons verschlüsselten Signaturen in den Satzungen der Freimaurer von James Anderson]. Health Research Books.

van Oorschot, P. C., & Wiener, M. J. (1996). Improving implementable meet-in-the-middle attacks by orders of magnitude [Verbesserung der implementierbaren Meet-in-the-Middle-Angriffe um Größenordnungen]. In N. Koblitz (Hrsg.), *Advances in Cryptology–CRYPTO ’96* (S. 229–236). Springer.

Wang, H., Zheng, Z., Xie, S., Dai, H. N., & Chen, X. (2018). Blockchain challenges and opportunities: A survey [Herausforderungen und Chancen der Blockchain: Eine Übersicht]. *International Journal of Web and Grid Services*, *14*(4), 352. https://doi.org/10.1504/IJWGS.2018.10016848

Watkins, J. (2001). Steganography-messages hidden in bits [Steganografie - in Bits versteckte Botschaften]. *Lehrmaterial zu Multimediasystemen, Fakultät für Elektronik und Informatik.* University of Southampton.

Zeilenga, K. D. (2008). *CRAM-MD5 to historic* [CRAM-MD5 zu historisch]. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/draft-ietf-sasl-crammd5-to-historic-00