Fachhochschule
Dortmund
University of Applied Sciences and Arts

Informationssicherheit – SoSe 2023
Asymmetrische kryptographische Verfahren: Grundlagen, DH, RSA, ECC, Digitale Signaturen

Prof. Dr. Holger Schmidt
holger.schmidt004[at]fh-dortmund.de

Fachhochschule Dortmund
Fachbereich Informatik
Professur für IT-Sicherheit, Informatik

Themen & Lernziele

Themen

- Schlüsselaustauschproblem
- Grundlegende Angriffsarten und Kryptoanalyse
- DH
- RSA
- Digitale Signaturen
- ECC

Lernziele

Die Studierenden sind in der Lage,

- Grundlagen der asymmetrischen Kryptographie zu differenzieren und zu erklären.
- DH, RSA, digitale Signaturen, ECC zu erklären und anzuwenden.

Kryptographie
Asymmetrische Verschlüsselung

Schlüsselaustauschproblem

- **Schlüsselaustauschproblem**
  - Manueller Schlüsselaustausch
  - Schlüsselaustausch mit Authentifizierung
  - **Asymmetrische Kryptographie**

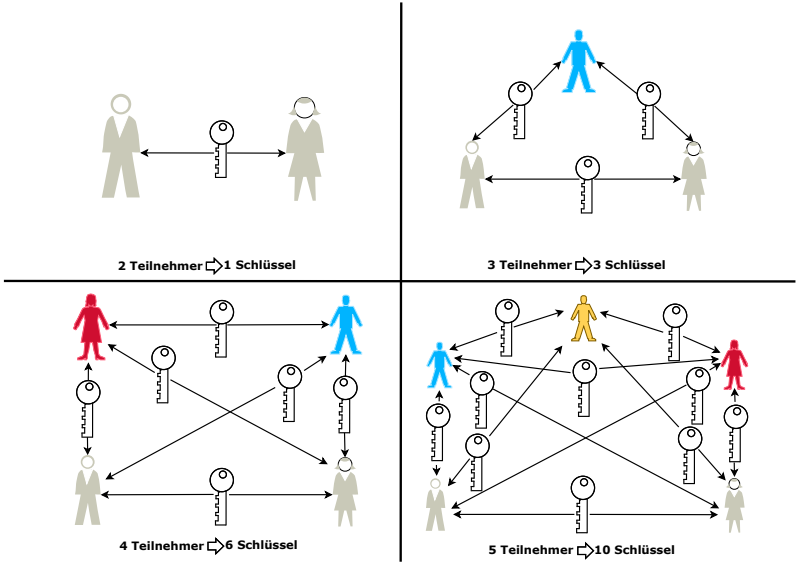

Abbildung 11-1 aus Schmeh, 2016

Asymmetrische Verschlüsselung I

- Problem: Alice möchte Bob eine geheime Nachricht senden.
- Bob übergibt an Alice ein **geöffnetes Vorhängeschloss**.
- Bob behält den **Schlüssel** für sich allein.

Foto selbst erstellt

Asymmetrische Verschlüsselung II

- Alice legt die geheime Nachricht in eine Schatulle.
- Alice verschließt die Schatulle mit dem Vorhängeschloss.
- Alice sendet die **verschlossene Schatulle** an Bob
- Bob besitzt als einziger den passenden Schlüssel und kann die Schatulle aufschließen, um dann die geheime Nachricht zu lesen.

Fotos selbst erstellt

Asymmetrische Verschlüsselung III

- Schutzziel: u. a. **Vertraulichkeit**
- Algorithmen in die ein öffentlicher Schlüssel („geöffnetes Vorhängeschloss“) und ein privater Schlüssel („Schlüssel“) mit einfließt
- m ist der Klartext, e der Verschlüsselungsalgorithmus, d der Entschlüsselungsalgorithmus, c der Geheimtext, a der öffentliche Schlüssel, x der private Schlüssel
- Verschlüsselungsalgorithmus: $e(m, a) = c$
- Entschlüsselungsalgorithmus: $d(c, x) = m$

Asymmetrische Verschlüsselung IV

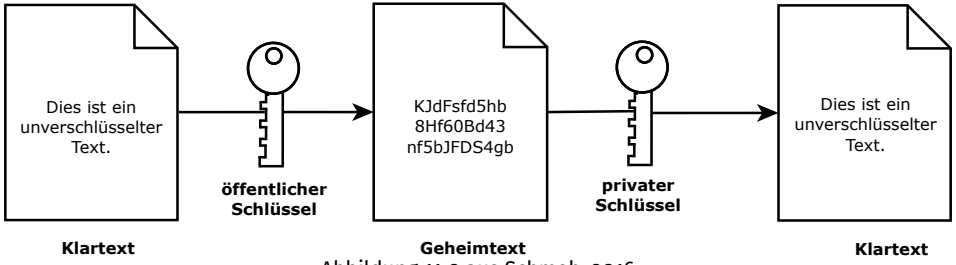

Abbildung 11-2 aus Schmeh, 2016

Exkurs: Mathematische Grundlagen – Existenz inverses Element

Wir beschäftigen uns mit **Modulo-Rechnung**, da diese typischerweise sehr einfach durchführbar ist, ihre Umkehrung ist jedoch sehr aufwändig.

- Wir betrachten also Modulo-Division bzw. Modulo-Multiplikation: $\frac{b}{a} \equiv b \cdot a^{-1} \ (mod\ n)$.
- Wir betrachten $n \in \mathbb{N}$ für das alle ganzen Zahlen zwischen 1 und $n - 1$ ein inverses Element (mod n) haben.
- Das ist der Fall, wenn alle Zahlen zwischen 1 und $n - 1$ **teilerfremd** zu n sind.
- Diese Anforderung ist genau dann erfüllt, wenn n eine **Primzahl** ist.
- Die ganzen Zahlen zwischen 1 und $p - 1$ bilden mit der Modulo-Multiplikation eine **Gruppe** mit 1 als neutralem Element, die wir $\mathbb{Z}_p^*(\cdot, 1)$ nennen[1].

[1] $\mathbb{Z}^* = \mathbb{Z} \setminus \{0\}$

Exkurs: Mathematische Grundlagen – Untergruppe und Generator

- Eine **Untergruppe** ist eine Teilmenge einer Gruppe, die selbst wieder eine Gruppe ist (mit derselben Verknüpfung und demselben neutralen Element).
- Die Anzahl der Elemente einer Untergruppe von $\mathbb{Z}_p^*(\cdot, 1)$ teilt immer $p - 1$.
- Nimmt man ein beliebiges $a \in \mathbb{Z}_p^*(\cdot, 1)$ und betrachtet $\{a, a^2, a^3, \ldots, a^{p-1} \ (mod\ p)\}$, dann erhält man eine Untergruppe. a ist **Generator** der Untergruppe.
- Jede Untergruppe von $\mathbb{Z}_p^*(\cdot, 1)$ hat einen Generator (und damit auch $\mathbb{Z}_p^*(\cdot, 1)$ selbst).

Exkurs: Mathematische Grundlagen – Diskreter Logarithmus

- Die Gleichung $a^x \equiv b \ (mod\ p)$ ist immer lösbar, wenn a ein Generator von $\mathbb{Z}_p^*(\cdot, 1)$ und $b \in \mathbb{Z}_p^*(\cdot, 1)$ ist.
- Anders ausgedrückt: Der **diskrete Logarithmus** in $\mathbb{Z}_p^*(\cdot, 1)$ existiert immer, wenn die Basis ein Generator von $\mathbb{Z}_p^*(\cdot, 1)$ ist.

Einwegfunktion und Falltürfunktion

- Eine Funktion, deren Umkehrung nur sehr aufwändig zu berechnen ist, nennt man **Einwegfunktion**.
- Eine Einwegfunktion, die eine „versteckte Abkürzung“ (z. B. mithilfe eines Schlüssels) enthält, wird **Falltürfunktion** genannt.

Diskreter Logarithmus und Faktorisierungsproblem

Diskreter Logarithmus:

- Die **diskrete Exponentialfunktion** $a^b \ (mod\ n)$ ist als Einwegfunktion selbst für sehr große Werte von a, b und n leicht berechenbar.
- Die Umkehrung, d. h. der diskrete Logarithmus und damit die Berechnung von b bei gegebener Basis a, Modul n und gewünschtem Ergebnis, verhält sich entgegengesetzt: die Berechnungszeit liegt für große Werte über der Lebensdauer unseres Universums.

Faktorisierungsproblem:

- Die Multiplikation zweier Primzahlen $n = p \cdot q$ ist für sehr große Werte leicht berechenbar.
- Die Umkehrung, d. h. die Bestimmung der Faktoren p und q mithilfe von n, verhält sich entgegengesetzt: die Berechnungszeit liegt für große Werte über der Lebensdauer unseres Universums.

Kryptographie
Diffie-Hellman Schlüsselaustausch

Dieser Abschnitt basiert auf Schmeh, 2016, Teil 2 – Kapitel 13. Aufgrund der Präsentation als Folien und Notizen sind die Texte der Quelle typischerweise paraphrasiert.

Diffie-Hellman Schlüsselaustausch I

- **Public-Key-Verfahren**
- Verwendung des diskreten Logarithmus zur Lösung des Schlüsselaustauschproblems
- Von Whittfield Diffie und Martin Hellman Mitte der 1970er entwickelt (Diffie & Hellman, 1976)
- Spezifiziert in Public Key Cryptography Standards (PKCS)#3 v1.4 (https://www.teletrust.de/fileadmin/files/oid/oid_pkcs-3v1-4.pdf, aufgerufen am 28. Juni 2023)

Diffie-Hellman Schlüsselaustausch II

- Alice und Bob einigen sich auf eine Primzahl p und $g \in \mathbb{N}$ mit g Generator von $\mathbb{Z}_p^*(\cdot, 1)$.
- p und g sind öffentlich (können also unverschlüsselt kommuniziert werden).
- Alice wählt außerdem $x \in \mathbb{N}$ mit $x < p$ und Bob wählt $y \in \mathbb{N}$ mit $y < p$.

1. Alice berechnet $a := g^x \ mod \ p$ und sendet a an Bob.
2. Bob berechnet $b := g^y \ mod \ p$ und sendet b an Alice.
3. Alice berechnet $k_1 := b^x \ mod \ p$.
4. Bob berechnet $k_2 := a^y \ mod \ p$.

Es gilt $k_1 = k_2$.

Diffie-Hellman Schlüsselaustausch III

- Mallory kann die Kommunikation abhören, jedoch k_1 bzw. k_2 nicht bestimmen, denn dazu müsste Mallory x oder y kennen, also den diskreten Logarithmus $a \equiv g^x \ mod \ p$ bzw. $b \equiv g^y \ mod \ p$ lösen.
- Der diskrete Logarithmus ist eine Einwegfunktion (jedoch keine Falltürfunktion).
- a ist der öffentliche und x der private (geheime) Schlüssel von Alice.
- b ist der öffentliche und y der private (geheime) Schlüssel von Bob.

Diffie-Hellman Schlüsselaustausch – Kryptoanalyse

- Brute Force ist nicht der beste Angriff auf das Diffie-Hellman-Schlüsselaustauschverfahren, denn der diskrete Logarithmus lässt sich schneller berechnen.
- Durch Werte ab 3000 Bit für p ist kein praktikabler Angriff auf das Diffie-Hellman-Schlüsselaustauschverfahren bekannt[2].
- Die Wahl von g beeinflußt die Sicherheit des Verfahrens: Die Untergruppe sollte möglichst groß sein; das wird durch die Wahl eines Generators der Untergruppe gewährleistet.
- **Man-in-the-Middle** Angriff ist möglich, wird in der Praxis jedoch durch **Public-Key-Infrastrukturen** bzw. **Authentifikation** verhindert.

[2] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf, 28. Juni 2023

Kryptographie
RSA-Algorithmus

Dieser Abschnitt basiert auf Schmeh, 2016, Teil 2 – Kapitel 13. Aufgrund der Präsentation als Folien und Notizen sind die Texte der Quelle typischerweise paraphrasiert.

RSA-Algorithmus

- **Asymmetrischer Verschlüsselungsalgorithmus**
- Von Ron Rivest, Adi Shamir und Leonard Adleman Ende der 1970er entwickelt (Rivest, Shamir & Adleman, 1978)
- Spezifiziert in RFC 8017 (https://www.rfc-editor.org/rfc/rfc8017, aufgerufen am 28. Juni 2023)
- Verschlüsselung entspricht einer **Modulo-Exponentation** und Entschlüsselung einem **Modulo-Wurzelziehen**.

Erzeugung RSA-Schlüsselpaar – Öffentlicher Schlüssel I

Szenario: Bob möchte eine Nachricht m verschlüsseln, sodass nur Alice diese entschlüsseln kann. Dazu muss Alice zunächst ein RSA Schlüsselpaar erzeugen. Für die Verschlüsselung ist dann Alice öffentlicher Schlüssel notwendig, für die Entschlüsselung Alice privater Schlüssel.

1. Alice wählt zufällig zwei große **Primzahlen** p und q mit $p \neq q$ aus.
   - Wichtig: *zufällige Auswahl* von p und q, beide *gleiche Größenordnung* - jedoch *nicht dicht beieinander liegend*
2. Alice berechnet den **RSA-Modulus** $n = p \cdot q$.
   - Die Sicherheit von RSA ist insbesondere dadurch gegeben, dass die Wiederherstellung von p und q (d. h. *Primfaktorzerlegung* von n) nur aus dem RSA-Modulus n nicht praktikabel ist.
   - n beziffert die RSA-Schlüssellänge.

Erzeugung RSA-Schlüsselpaar – Öffentlicher Schlüssel II

3. Alice berechnet die **Eulersche φ-Funktion** $\varphi(n) = (p-1)(q-1)$.
   - φ-Funktion gibt an, wie viele Zahlen $x \in \mathbb{N}$ mit $0 < x < n$ zu n teilerfremd sind, z. B. $\varphi(3) = 2$, denn 1 und 2 sind zu 3 teilerfremd (1 ist als trivialer Teiler teilerfremd zu allen $x \in \mathbb{N}$).
   - Die Gleichung $\varphi(n) = (p-1)(q-1)$ gilt nur für *Primzahlen* p und q, d. h. nur dann lässt sich $\varphi(n)$ einfach berechnen.
   - Die Primzahlen p und q werden nach der Berechnung von $\varphi(n)$ sicher gelöscht.
4. Alice wählt zufällig eine Zahl e mit $2 < e < \varphi(n)$, die teilerfremd zu $\varphi(n)$ ist.
   - n und e bilden zusammen Alice öffentlichen Schlüssel.
   - In der Praxis ist e eine kleine und konstante Zahl, z. B. $2^{16} + 1 = 65537$, damit die Verschlüsselung performant durchgeführt werden kann.

RSA-Verschlüsselung mit öffentlichem Schlüssel

Die Verschlüsselung der Nachricht m erfolgt durch Modulo-Potenzieren der Nachricht m mit dem Exponenten e und basierend auf dem RSA-Modulus n.

1. Alice gibt Ihren öffentlichen Schlüssel (n, e) bekannt.
2. Damit m überhaupt verschlüsselt werden kann, muss m in eine natürliche Zahl y umgewandelt werden, mit
   - $y < n$, da $y \in \mathbb{Z}_n$ sein muss. Tritt der Fall $y \geq n$ ein, dann muss blockweise verschlüsselt werden.
   - $y^e > n$, da sonst y durch Wurzelziehen (mit polynomialen Aufwand) aus y^e ermittelt werden kann.
3. Bob verschlüsselt damit die umgewandelte Nachricht y wie folgt: $c \equiv y^e \ (mod\ n)$. c ist der Geheimtext, den er an Alice übergibt. Es gilt natürlich $c \in \mathbb{Z}_n$.
   - Damit ist das Modulo-Potenzieren eine Falltürfunktion, wobei die Falltürinformation die **Faktorisierung des Modulus** ist.

Erzeugung RSA-Schlüsselpaar – Privater Schlüssel

Um das Modulo-Potenzieren rückgängig zu machen und aus dem Geheimtext c wieder die umgewandelte Nachricht y zu erhalten, wird die Modulo-Wurzel gezogen. Dazu wird das **modulare Inverse** d von e bzgl. des RSA-Modulus n genutzt.

1. Ausgehend von $c \equiv y^e \ (mod\ n)$ heißt y die e-te Wurzel von c bzgl. dem RSA-Modulus n.
2. Alice berechnet (mit dem erweiterten euklidischen Algorithmus) $d \equiv e^{-1} \ (mod\ \varphi(n))$. (n, d) bildet Alice privaten Schlüssel.
   - Das modulare Inverse d existiert, da e und $\varphi(n)$ teilerfremd sind.
   - Das modulare Inverse d lässt sich nur einfach berechnen, wenn $\varphi(n)$ bekannt ist.
3. $\varphi(n)$ wird nach Erzeugung des privaten Schlüssels sicher gelöscht.

RSA-Entschlüsselung mit privatem Schlüssel

1. Alice kann c entschlüsseln, indem sie $y \equiv c^d \ (mod\ n)$ berechnet. Alice berechnet also die e-te Wurzel von c bzgl. dem RSA-Modulus n, indem sie die d-te Potenz berechnet.
   - Es gilt für alle $y < n$ und e, d wie oben definiert: $(y^e)^d \equiv y \ (mod\ n)$. Daher lässt sich der Klartext mit dem modularen Inversen d berechnen.
2. Mallory kann das nicht, denn er kann d nicht ermitteln ohne $\varphi(n)$ zu kennen, wofür die die Faktorisierung von n notwendig ist. Die Primfaktorzerlegung ist jedoch unter den zuvor genannten Bedingungen praktisch unmöglich durchzuführen.

RSA-Algorithmus – Beispiel

1. Alice wählt $p = 5$ und $q = 17$ und berechnet $n = 5 \cdot 17 = 85$.

2. Alice wählt $3 \in \mathbb{N}$, die teilerfremd zu $\varphi(85) = 64$ ist. 85 und 3 bilden zusammen Alice öffentlichen Schlüssel.

3. Alice berechnet $d = 3^{-1} \equiv 43 \ (mod\ 64)$ (denn $3 \cdot 43 \equiv 1 \ (mod\ 64)$). 85 und 43 bilden zusammen Alice privaten Schlüssel.

4. Bob kennt Alice öffentlichen Schlüssel und verschlüsselt damit einen Klartext $m = 2$ wie folgt: $c = 2^3 \equiv 8 \ (mod\ 85)$ ist der Geheimtext, den er an Alice sendet. Bei der Berechnung hat Bob den privaten Schlüssel von Alice nicht benutzt.

5. Alice kann 8 mit ihrem privaten Schlüssel entschlüsseln, indem sie $m = 8^{43} \equiv 2 \ (mod\ 85)$ berechnet.

RSA-Algorithmus – Kryptoanalyse I

- RSA ist ein Verfahren mit **variabler Schlüssellänge** (Alice kann n wählen).
- Wesentliche Angriffstypen:
  - **Public-Key-Only Angriff**: Mallory kennt nur den öffenlichen Schlüssel von Alice.
  - **Chosen-Ciphertext Angriff**: Mallory kann einen Geheimtext wählen und diesen von Alice entschlüsseln lassen.
- **Faktorisierungsangriff**, d. h. Berechnung der Primzahlen p und q aus n, ist für große Werte praktisch aussichtslos.

RSA-Algorithmus – Kryptoanalyse II

- Aufgrund des Mooreschen Gesetzes sind RSA-Moduli mit 100 Dezimalstellen (ursprünglich von Rivest et al., 1978 empfohlen) längst nicht mehr sicher.
- **RSA-Factoring-Challenge** zahlte bis 2007 Preisgelder für Faktorisierungen. Heutige Community erreichte zuletzt z. B. Faktorisierung von RSA-250 (829 Bit) in 2700 Prozessorjahren auf einem Intel Xeon Gold 6130 (16 Kerne, 2,1 GHz) (https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2020-February/001166.html, aufgerufen am 28. Juni 2023).
- Schlüssellängen ab **3000 Bit gelten aktuell praktisch als sicher**[3].

[3] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf, 28. Juni 2023

RSA-Algorithmus – Kryptoanalyse III

- Wahl von p, q, e und n sowie komplizierte mathematische Berechnung bieten größere Angriffsfläche (im Vergleich zu z. B. DES).
- **Low-Exponent-Angriff** beruht auf der Verwendung kleiner Werte für e (typisch sind 3 und 17) und die Verschlüsselung des gleichen Klartexts für e verschiedene Empfänger. Gegenmaßnahmen: größere Werte für e (z. B. 65537), in RFC 8017 durch spezielle Aufbereitung von Klartexten.
- Zu kleine Klartexte erlauben ebenfalls Angriffe auf RSA, daher ist **immer ein Padding zu benutzen**, z. B. Optimal Asymmetric Encryption Padding (OAEP) (siehe RFC 8017).
- Verschiedene Angriffe die sich **Implementierungsfehler** zu nutze machen

Kryptographie
Elliptische Kurven

Elliptische Kurven / Elliptic Curve Cryptography (ECC)

Fachhochschule Dortmund
University of Applied Sciences and Arts

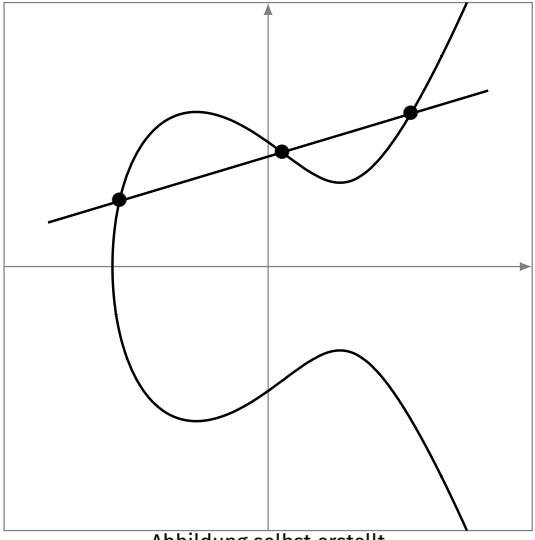

Abbildung selbst erstellt

- Problem: klassische asymmetrische Verfahren (z. B. RSA, DSA) erfordern große Schlüssellängen und Berechnungen sind daher aufwändig
- **Elliptische Kurve** erfüllt folgende (kurze) Weierstraß-Gleichung: $y^2 = x^3 + ax + b$
- Elliptische Kurven bilden abelsche Gruppe bzgl. Punktaddition.

Inverser Punkt auf elliptischer Kurve

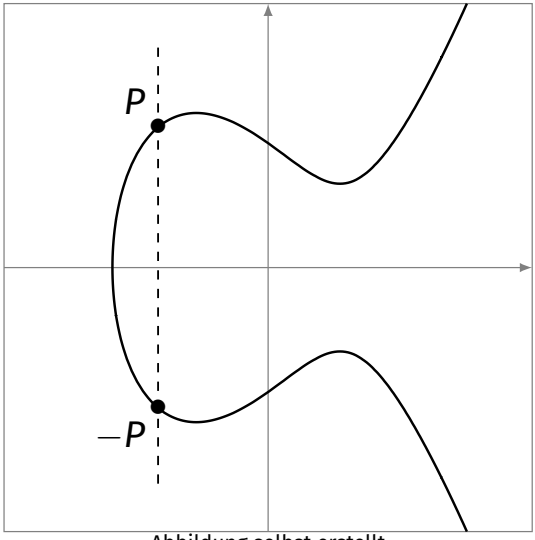

Abbildung selbst erstellt

- $-P$ ist inverser Punkt zu P
- $\forall P | P + (-P) = \infty$ ist neutrales Element

Addition zweier verschiedener Punkte auf elliptischer Kurve

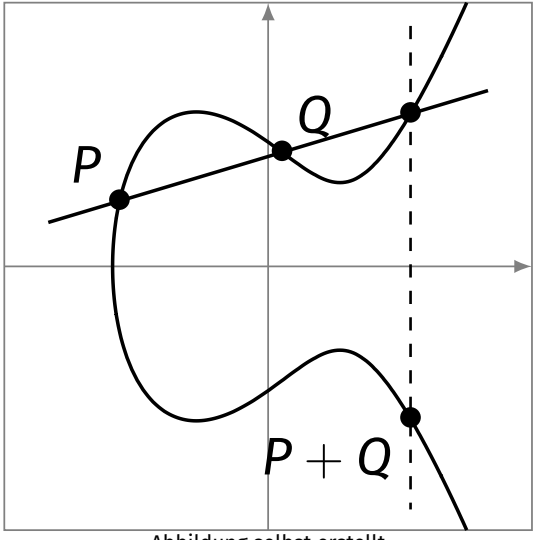

Abbildung selbst erstellt

- Eine Gerade mit zwei Schnittpunkten $P, Q, P \neq Q$ mit elliptischer Kurve hat einen dritten, welcher gespiegelt an x-Achse die Summe $P + Q$ ergibt (**Sekantenregel**).

Selbst-Addition eines Punktes auf elliptischer Kurve

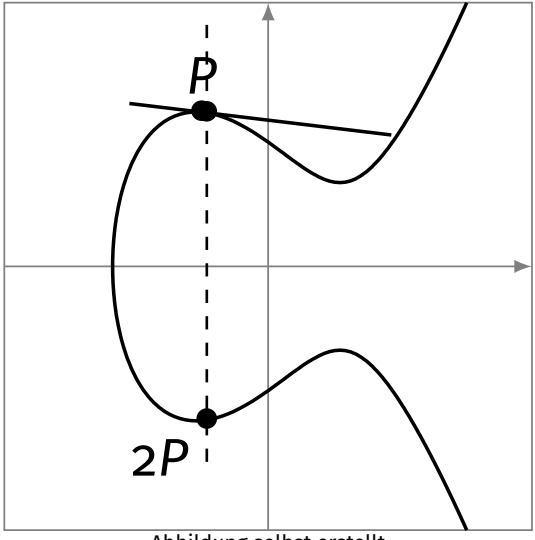

Abbildung selbst erstellt

- Gilt $P = Q$, so sind zwei Fälle möglich:
  1. $P + P = 2P$ falls Tangente durch P elliptische Kurve erneut schneidet
  2. $P + P = \infty$ anderenfalls
- Schnittpunkt Tangente an Punkt P, dann Spiegelung an x-Achse (**Tangentenregel**)

Analogie zum diskreten Logarithmus

- k-fache Addition $kP = P + P + \cdots + P = Q$
- Für ausgewählte elliptische Kurven gilt: p von $\mathbb{Z}_p$ sowie k ausreichend groß gewählt, so ist es schwierig (praktisch unmöglich) ausgehend von Q auf k zu schließen.
- Da das Problem deutlich schwieriger als diskreter Logarithmus und Primzahlfaktorisierung in Galois-Körpern ist, kommt ECC mit **deutlich kleineren Schlüsseln** aus, z. B. bietet ein 256 Bit ECC-Schlüssel ein ähnliches Sicherheitsniveau wie ein 3072 Bit RSA-Schlüssel.
- Eine populäre elliptische Kurve ist **Curve25519** (spezifiziert in RFC 7748[4]).

[4] https://tools.ietf.org/html/rfc7748, aufgerufen am 28. Juni 2023

Kryptographie
Hybridverfahren

Hybridverfahren I

Symmetrische Verschlüsselung ist wesentlich schneller als asymmetrische Verschlüsselung.

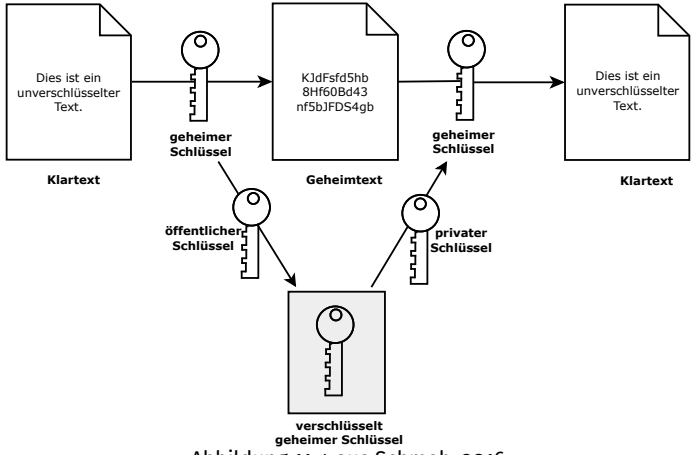

Abbildung 11-4 aus Schmeh, 2016

Hybridverfahren II

TLS arbeitet mit **cipher suites**, z. B.
`TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`

ECDHE steht für Diffie-Hellman Schlüsselaustausch mit ECC

RSA wird genutzt für ein Authentifikationsverfahren, damit man-in-the-middle Angriffe auf das Diffie-Hellman Verfahren praktisch ausgeschlossen sind

AES für Nutzdatenverschlüsselung, hier mit 256 Bit Schlüsseln

GCM steht für Galois/Counter mode (NIST 800-38D, 2007), ein Block Modus der bei der Nutzdatenverschlüsselung zusätzlich Authentizität gewährleistet

SHA384 wird für ein HMAC Verfahren und dieses als Basis für eine PRF (Pseudo-Random Function) genutzt, mit der z. B. aus dem ECDHE-Schlüssel ein *master secret* abgeleitet wird.

Kryptographie
Digitale Signaturen

Digitale Signaturen

- **Nachweis der Echtheit** eines digitalen Dokuments
- Analog zur eigenhändigen „Unterschrift“:
  - nicht zu fälschen
  - verifizierbar
  - nicht übertragbar auf andere Dokumente
  - dazugehöriges Dokument nicht unbemerkt veränderbar

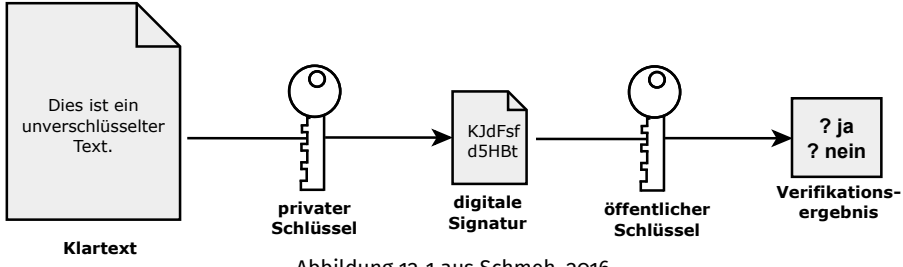

Abbildung 12-1 aus Schmeh, 2016

Digitale Signaturen – Rechtliche Rahmenbedingungen

- Europäische **electronic IDentification, Authentication and trust Services (eIDAS)**[5] seit dem 01. Juli 2016 europaweit geltendes Recht
- Harmonisierung des Binnenmarkts für Signatur, Siegel und Zeitstempel in EU und europäischer Freihandelsassoziation (EFTA)
- Signaturerstellungseinheiten müssen nach Common Criteria zertifiziert sein

[5] https://eur-lex.europa.eu/eli/reg/2014/910/oj, aufgerufen am 28. Juni 2023

Digitale Signaturen – PGP und S/MIME

- **Pretty Good Privacy (PGP)** spezifiziert in RFC 2440 (https://www.rfc-editor.org/rfc/rfc2440, aufgerufen am 28. Juni 2023)
  - **Web of Trust** durch transitive Vertrauensbeziehungen (Signieren von öffentlichen Schlüsseln)
- **Secure / Multipurpose Internet Mail Extensions (S/MIME)** spezifiziert in RFC 2633 (https://www.rfc-editor.org/rfc/rfc2633, aufgerufen am 28. Juni 2023)
  - S/MIME basiert auf **X.509 Zertifikaten** spezifiziert in RFC 5280 (https://www.rfc-editor.org/rfc/rfc5280, aufgerufen am 28. Juni 2023)
  - Zertifikat beinhaltet öffentlichen Schlüssel und wird von ausgebener Stelle **beglaubigt**
  - Streng hierarchisch strukturierte **Public Key Infrastrukturen (PKI)** mit **Wurzelzertifikat**

Digitale Signaturen – Basierend auf RSA

- Schutzziele: **Authentizität, Integrität, Nicht-Abstreitbarkeit**
- Algorithmen in die ein öffentlicher und ein privater Schlüssel einfließen (z. B. RSA)
- m ist das zu schützende Dokument, a der öffentliche Schlüssel, x der private Schlüssel, u der Signaturalgorithmus, s die Signatur, v der Verifikationsalgorithmus
- Signaturalgorithmus: $u(m, x) = s$
- Verifikationsalgorithmus: $v(a, s) = m'$ (Signatur s ist echt, falls $m = m'$)
- Statt direkt m zu signieren, wird typischerweise ein **Hashwert** von m erzeugt und dieser signiert.

Digitale Signaturen – RSA vs. ECDSA

- Bei Verwendung von RSA für digitale Signaturen ähnliche Angriffe wie bei RSA-Verschlüsselung möglich
- Digitale Signaturen **basierend auf ECC**:
  - **Digital Signature Standard (DSS)** FIPS-186-5, 2023 basiert auf **Elliptic Curve Digital Signature Algorithm (ECDSA)**
  - Vorteile gegenüber RSA:
    - kürzere Schlüssel bei vergleichbarer Sicherheit
    - performanter aufgrund kürzerer Schlüssel
  - Nachteile gegenüber RSA:
    - kein Message Recovery
    - kein Verschlüsselungsalgorithmus
    - komplexere Implementierung
    - zusätzlicher Zufall notwendig

Post-Quanten-Kryptographie I

- **Quantencomputer** basiert auf Quantenmechanik
- Bisher existiert keine praktikable Implementierung eines Quantencomputers
- Unentscheidbarkeit unbeeinflusst
- Vornehmlich asymmetrische kryptographische Verfahren bedroht, z. B. **Shor-Algorithmus** (Shor, 1997) zur Primzahlfaktorisierung in Polynomialzeit
- Bisher existiert jedoch kein Beweis, dass Quantencomputer prinzipiell schneller als gewöhnliche Computer sind.

Post-Quanten-Kryptographie II

- Symmetrische Verfahren eher weniger bedroht, z. B. **Grover-Algorithmus** (Grover, 1996) erlaubt Brute Force auf 2^n Schlüssel in $2^{\frac{n}{2}}$ Schritten (d. h. AES 128/192 Bit nicht sicher)
- **Post-Quanten-Kryptographie** beschäftigt sich folglich mit asymmetrischen kryptographischen Verfahren für die selbst Quantencomputer keine Bedrohung darstellen.

Fallstricke und Anwendungshinweise

- Schlüssellängen ab 3000 Bit bzw. bei ECC 250 Bit verwenden, siehe BSI TR-02102-1[6]
- Nicht mehr RSA und DSA, sondern stattdessen ECDSA oder Ed25519[7] für digitale Signaturen verwenden
- Nicht mehr DH, sondern stattdessen ECDH[8] als Schlüsselaustauschprotokoll verwenden

[6] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf, 28. Juni 2023

[7] https://ed25519.cr.yp.to/, aufgerufen am 28. Juni 2023

[8] https://www.rfc-editor.org/rfc/rfc6090, aufgerufen am 28. Juni 2023

Zusammenfassung

Zusammenfassung

- Grundlagen asymmetrische Kryptographie definiert
- DH, RSA, digitale Signaturen, ECC vorgestellt

Weiterführende Literatur

Weiterführende Literatur

- *Kryptografie – Verfahren - Protokolle - Infrastrukturen*, Kapitel 13, 14, 15.1 und 20 von Schmeh (2016)
- *IT-Sicherheit – Konzepte - Verfahren - Protokolle*, Kapitel 7.6, 7.7, 8.2 von Eckert (2023)
- Choosing safe curves for elliptic-curve cryptography[9]

[9] https://safecurves.cr.yp.to/, aufgerufen am 28. Juni 2023

Literatur I

- Diffie, W., & Hellman, M. E. (1976).New Directions in Cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654 (siehe S. 17).
- Eckert, C. (2023). *IT-Sicherheit: Konzepte - Verfahren - Protokolle* (11. Aufl.). De Gruyter Oldenbourg. (Siehe S. 53).
- FIPS-186-5. (2023). Federal Information Processing Standards Publication (FIPS 186-5). Digital Signature Standard (DSS). https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf (siehe S. 46).
- Grover, L. K. (1996).A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, 212–219 (siehe S. 48).
- NIST 800-38D. (2007). NIST Special Publication 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf (siehe S. 40).

Literatur II

- Rivest, R. L., Shamir, A., & Adleman, L. (1978).A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, *21*(2), 120–126 (siehe S. 22, 30).
- Schmeh, K. (2016). *Kryptografie – Verfahren - Protokolle - Infrastrukturen* (6. Aufl.). dpunkt.verlag. (Siehe S. 6, 10, 16, 21, 39, 42, 53).
- Shor, P. W. (1997).Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, *26*(5), 1484–1509 (siehe S. 47).

Anhang
RSA-Algorithmus FAQ

RSA-Algorithmus – FAQ I

- Falls p **und** q **keine Primzahlen** sind, dann ergeben sich Probleme:
  - Die Bedingung, dass p und q Primzahlen sind, ist nicht erfüllt, sodass die Anwendung der Formel $\varphi(n) = (p-1)(q-1)$ falsche Ergebnisse liefert. In der Folge arbeiten Ver- und Entschlüsselung nicht mehr so zusammen, dass sich z. B. ein Geheimtext korrekt entschlüsseln lässt. Selbst ausprobieren: Wähle $p = 24$, $q = 19$, $e = 41$, $m = 6$.
  - Die allgemeine Berechnungsformel von $\varphi(x)$ basiert auf der Primfaktorzerlegung von x. Wenn also p und q keine Primzahlen sind, dann ergibt die Primfaktorzerlegung entsprechend mehr als zwei Primfaktoren. Folglich kann die Primfaktorzerlegung wesentlich einfacher sein, z. B. bei geraden Zahlen, Zahlen die mit 5 enden, etc.
  - Trotzdem kann RSA auch mit einem RSA-Modulus funktionieren, der sich in mehr als zwei Primfaktoren zerlegen lässt (**Multi-Prime RSA** in RFC 8017). Dabei sind dann weitere Bedingungen zu erfüllen, z. B. das kein Faktor zu kurz ist, kein Mehrfachvorkommen eines Faktors, etc.

RSA-Algorithmus – FAQ II

- Die **RSA-Schlüssellänge** ist nicht direkt vergleichbar mit Schlüssellängen bei der symmetrischen Verschlüsselung, da nicht jede Zahl mit einer RSA-Schlüssellänge von k Bit ein RSA-Modulus ist, z. B. eine RSA-Schlüssellänge von 1024 Bit führt nicht zu 2^{1024} Schlüsseln, da nicht jede 1024 Bit Zahl ein RSA-Modulus ist. Um einen Vergleich herstellen zu können, betrachtet man die Anzahl notwendiger Operationen für die entsprechende Primfaktorzerlegung des RSA-Modulus, z. B. für 1024 Bit RSA-Schlüssellänge sind das ca. 2^{73} Operationen.
- Die Berechnung der Primzahlen p und q erfolgt mit **probabilistischen Primzahltests** wie z. B. dem **Fermat-Test**. Die Archillesferse dieses Tests ist, dass *Charmicael-Zahlen* den Test bestehen, welche keine Primzahlen sind. Eine Verbesserung in dieser Hinsicht ist z. B. der **Miller-Rabin-Test**.