5 Authentifizierung und Autorisierung

Durch den Prozess der *Authentifizierung* wird die Identität innerhalb eines IT-Systems überprüft (siehe Abbildung 1):

- Aus Sicht des **Benutzers** ist die *Authentisierung* die Prüfung des Identitätsnachweises auf Authentizität.
- Aus Sicht des **IT-Systems** findet zum einem die *Authentifizierung* und zum anderen die *Autorisierung* statt.
    a. Bei der *Authentifizierung* handelt es sich um die Überprüfung der tatsächlichen Identität des Benutzers. Sowohl Sender als auch Empfänger sollten in der Lage sein, die *Identität* des jeweils anderen Kommunikationspartners zu bestätigen.
    b. Die *Autorisierung* wird als eine Sicherheitsfunktion betrachtet. Es geht darum, welcher Benutzer welchen Zugriff auf welche Funktionen hat. Dies erfolgt durch Zuweisen und wiederholtes Überprüfen von Zugriffsrechten.

Abbildung 1 stellt schematisch den Prozessablauf der Authentisierung dar.

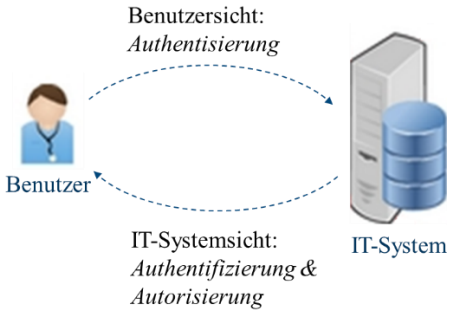

Abbildung 1: Prozessablauf der Authentifizierung

Im Authentifizierungsverfahren wird zwischen *Wissen*, *Besitz* und *Sein* unterschieden, wie der Nachweis der Echtheit der digitalen Identität zu überprüfen ist.

- *Wissen (what you know):* Hier wird der Nachweis durch die Kenntnis einer Information auf die Echtheit eines Nutzers geprüft. Beispiele sind Passwort, PIN, Antwort auf eine bestimmte Frage (Sicherheitsfrage) usw.
- *Besitz (what you have):* Die Verwendung eines Besitztums gilt als Nachweis für das Authentifizierungsverfahren. Beispiele sind Personalausweis, SIM-Karte im Smartphone, Hardware-Sicherheitsmodule (Smartcard, USB-Stick) usw.

- *Sein (what you are):* Bei diesem Authentifizierungsverfahren muss der Nutzer als Nachweis gegenwärtig sein. Beispiele sind Fingerabdruck, Retina Scan usw.
- *Weitere Verfahren:* Es können noch weitere unterstützende Faktoren für die Beurteilung der Echtheit des Nutzers herangezogen werden: Beispiele sind vergangene Transaktionen des Nutzers (Reputation), verwendete Endgeräte und Software des Nutzers (Technologie), Standort und Zeit des Authentisierungsprozesses.

Im Autorisierungsverfahren ist die *rollenbasierte Zugriffskontrolle* (engl. ***R**ole **B**ased **A**ccess **C**ontrol*, kurz *RBAC*)[1] eine Möglichkeit, die Zugriffsberechtigung der Nutzer innerhalb eines IT-Systems zu erteilen. Dieses *Sicherheits- und Berechtigungskonzept* ermöglicht die Vergabe von Rollen und Berechtigungen. Dabei werden die Zugriffsrechte anhand eines definierten Rollenmodells vergeben.

Zweifaktor Authentifizierung (2FA)

Für die *2FA* werden zwei unterschiedliche und unabhängige Faktoren für den Nachweis der Echtheit der digitalen Identität verwendet. Eine häufige Variante die mit Besitz und Wissen: Beispielsweise kann eine Smartcard (Hardware-Sicherheitsmodul) durch einen PIN aktiviert werden. Dabei muss der Nutzer die PIN kennen (Wissen) andererseits das Hardware-Sicherheitsmodul haben (Besitz).

Ein *Schlüsselverwaltungssystem* (engl. ***K**ey **M**anagement **S**ystem*, kurz *KMS*) verwaltet die für kryptographische Verfahren benötigten Schlüssel und Zertifikate in einem Unternehmen an zentraler Stelle, erinnert rechtzeitig vor deren Ablauf und unterstützt den kompletten *Key Lifecycle*. Einerseits soll der Überblick über die im Unternehmen im Einsatz befindlichen Schlüssel und Zertifikate gewährleistet, anderseits deren Gültigkeit überwacht und kontrolliert werden. Das *KMS* stellt sicher, dass Schlüssel echt sind und geheim gehalten werden. Es kann eine große Anzahl von Schlüsseln generieren, aufbewahren, bereitstellen, austauschen und schützen.

1.1 Schlüsselmanagement-Protokoll KMIP

Anwendungen und Systeme müssen Informationen mit dem *KMS* auf sicherem Weg austauschen können. Dafür wurde das ***K**ey **M**anagement **I**nteroperability **P**rotocol* (kurz *KMIP*)[2] entwickelt. Über dieses standardisierte Kommunikationsprotokoll können

[1] Siehe: Role Based Access Control – Wikipedia

[2] Siehe: Key Management Interoperability Protocol - Wikipedia

verschiedene Systeme für kryptografische Operationen derart verknüpft werden, damit eine zentrale Schlüsselverwaltung ermöglicht wird.

1.2 Public-Key-Infrastruktur (PKI)

Eine *PKI* dient zum Erstellen und Verwalten von digitalen Zertifikaten über deren gesamten Lebenszyklus, von der Erstellung über die Aufbewahrung und Verwendung bis hin zur Löschung. Dabei kommt es, außer auf die sichere Erstellung und Speicherung gültiger Schlüssel, auch auf die Verifizierung der ursprünglichen Identität der *PKI-Nutzer* an.

Eine *PKI* besteht aus Hardware, Software und einem abgestimmten Regelwerk, der Leitlinie. Diese definiert, nach welchen Sicherheitsregeln die Services für digitale Zertifikate erbracht werden. Dazu zählen das Betriebskonzept der PKI, die Nutzerrichtlinien sowie Organisations- und Arbeitsanweisungen.

2 Operative Sicherheit

Wird in einem Computernetzwerk der Datenverkehr, der in das Netz eintritt oder es verlässt, auf seine Sicherheit überprüft, protokolliert, verworfen oder weitergeleitet, dann sind *Firewalls*, *Intrusion-Detection-Systeme (IDS)* und *Intrusion-Prevention-Systeme (IPS)* daran beteiligt.

2.1 Firewall

Eine *Firewall*[3] ist eine Kombination aus Hardware und Software, die das *Intranet* einer Organisation vom *Internet* abschottet, wobei sie einige Datenpakete passieren lässt und andere blockiert. Eine *Firewall* ermöglicht einem Netzwerkadministrator die Kontrolle des Zugriffs der Außenwelt auf die Ressourcen innerhalb des von ihm verwalteten Netzes, indem das Sicherungssystem die Datenverkehrsströme zu und von diesen Ressourcen regelt (siehe Abbildung 2)

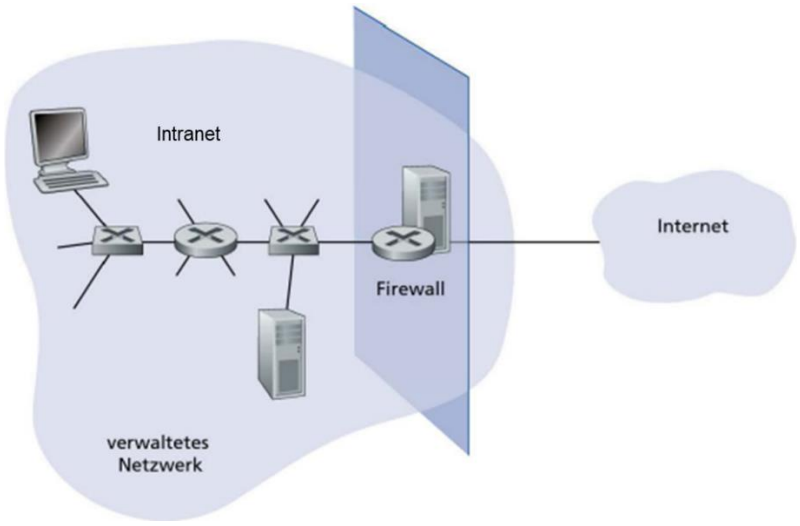

Abbildung 2: *Firewall*, platziert zwischen dem durch einen Administrator verwalteten Netzwerk und der Außenwelt

Eine *Firewall* hat drei zentrale Ziele:

1. Der komplette Datenverkehr von außen nach innen und umgekehrt passiert die Firewall.
2. Nur berechtigter Datenverkehr, definiert durch die lokale Sicherheitsrichtlinie, darf passieren.
3. Die Firewall selbst kann nicht kompromittiert werden, d.h. sie muss installiert und richtig aufgebaut sein (sonst entsteht ein falsches Gefühl der Sicherheit).

[3] Siehe: Firewall – Wikipedia

Die Aufgaben von Firewall-Systemen sind die *Zugangskontrolle* auf der *Netzwerk-, Nutzer- und Datenebene*.

Auf *Netzwerkebene* wird überprüft, welche IT-Systeme über das Firewall-System miteinander kommunizieren dürfen.

Auf *Nutzerebene* überprüft das Firewall-System, welche Nutzer und IT-System über das Firewall-System eine Kommunikation aufbauen dürfen. Dabei werden die Authentizität des Nutzers sowie der IT-Systeme überprüft.

Auf *Datenebene* überprüft das Firewall-System, welche Daten eines definierten Nutzers und IT-Systems über das Firewall-System übertragen werden dürfen.

Weitere Aufgaben von Firewall-Systemen sind

- die Alarmierung von sicherheitsrelevanten Ereignissen,
- die Beweissicherung und Protokollauswertung von Verbindungsdaten sowie sicherheitsrelevanten Ereignissen,
- die Kontrolle auf der Anwendungsebene (z.B. Spam, Malware),
- die Rechteverwaltung, die festlegt, mit welchen Protokollen, Diensten und zu welchen Zeiten eine Kommunikation erlaubt ist,
- die Entkopplung von Diensten, damit Schwachstellen und Konzeptionsfehler keine Angriffe von aussen ermöglichen.

2.2 Intrusion-Detection-Systeme

Das Firewall-System selbst bietet nur einen sehr geringen Schutz vor internen Angriffen. Um internen Angriffen entgegenzuwirken, müssen weitere, ergänzende Sicherheitsmechanismen eingeführt werden.

Hierfür gibt es ein Gerät, das Warnsignale auslöst, wenn es potenziell gefährlichen Datenverkehr beobachtet. Dieses Gerät wird als *Intrusion-Detection-System* (kurz *IDS*, System zur Erkennung von Eindringlingen) bezeichnet. Ein Gerät, das dagegen verdächtigen Verkehr herausfiltert, wird als *Intrusion-Prevention-System* (kurz *IPS*, System zur Vorbeugung gegen Eindringlinge) bezeichnet.

Ein *IDS* lässt sich einsetzen, um eine große Zahl von Angriffen zu erkennen, beispielsweise das Ausspähen von Netzwerken (Mapping, etwa mithilfe von nmap[4]), Port Scans,

[4] Siehe: Nmap: the Network Mapper - Free Security Scanner

Scans von TCP-Stacks, DoS-Angriffe[5], Würmer und Viren, Angriffe auf Schwachstellen von Betriebssystemen und Anwendungen.

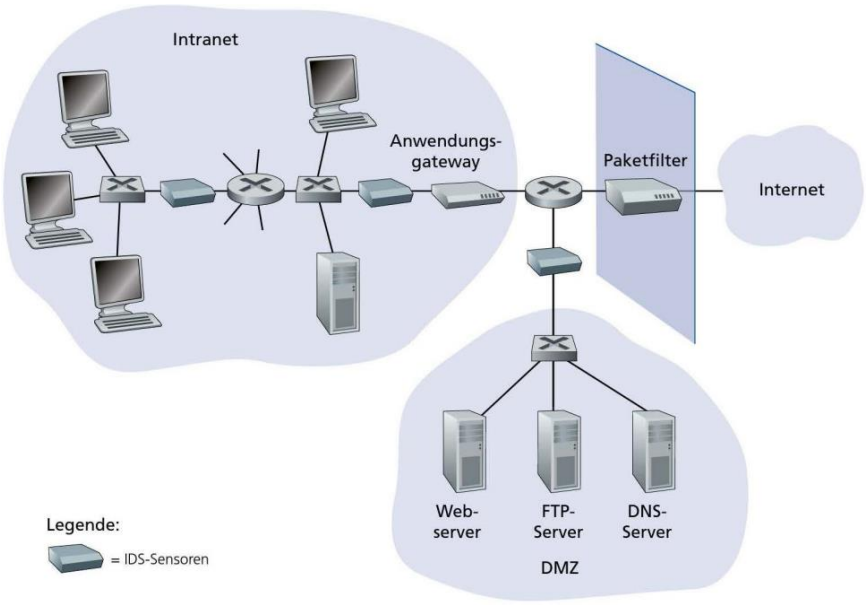

Abbildung 3: Eine Organisation, die einen Filter, ein Anwendungsgateway und IDS-Sensoren einsetzt

[5] Siehe: Denial of Service – Wikipedia

3 Best Practices

Der Begriff *Best Practice*[6] (dt. beste Methode, beste Praxis, beste Vorgehensweise) stammt aus der angloamerikanischen Betriebswirtschaftslehre und bezeichnet bewährte, optimale bzw. vorbildliche Methoden, Praktiken oder Vorgehensweisen im Unternehmen.

Die folgende Auflistung der *Best Practices für Informationssicherheit* sind Empfehlungen von IEEE[7], die von Fachleuten als gut oder notwendig akzeptiert werden:

I. Die Hardware/Software manipulationssicher machen (engl. tamper resistant)

- Viele IoT[8] Geräte sind dauernd (7x24 h) und unbeaufsichtigt im Betrieb
- Es gilt potenzielle Eindringlinge von den Daten fernzuhalten
- **Physische Sicherheit** durch spezielle Gehäuse, USB- und Ethernet-Port Schlösser, abgeschlossene Räume, etc.
- Gerätedeaktivierung, wenn Manipulation detektiert wird
- Starke *boot-level* Passwörter
- Bekannte Schwachstellen absichern (**engl. Hardening**)
  - ✓ Offene TCP/UDP-Ports
  - ✓ Offene serielle Schnittstellen
  - ✓ Password Prompts
  - ✓ Code Injektionen bei Web Servern
  - ✓ unverschlüsselte Kommunikation
  - ✓ etc.
- **Generell:** geschichteter Ansatz anwenden, d.h. eine Vielzahl von Hindernissen dem Angreifer in den Weg stellen, anstelle eines einzelnen Hindernisses

II. Firmware-Updates bereitstellen (conditio-sine-qua-non)

[6] Siehe: Best Practice – Wikipedia

[7] Institute of Electrical and Electronics Engineers: IEEE - The world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.

[8] Siehe: IoT-Plattform – Wikipedia

- Schachstellen werden unweigerlich zu Tage treten, wenn das Gerät in Betrieb ist
- Hersteller verdienen kein Geld mit der Bereitstellung von Firmware-Updates, nur mit dem Verkauf von Geräten
- **Geräte brauchen zwingend eine Update-Funktionalität**
- Firmware-Updates nur mit digitaler Unterschrift (man muss die Quelle des Updates überprüfen können)
- Für viele Geräte ist ein ***O**ver-**t**he-**A**ir-Update* (kurz *OTA*)[9] gar nicht vorgesehen (z.B.: LoRa[10])
- **Geräte ohne Update-Mechanismus können gar nicht sicher sein!**
- **Beispiel:** erstmals DDoS[11] Attacke mit IoT Geräten am 21.10.2016 (Mirai botnet)
- Hersteller müssen verpflichtet werden ihre Geräte über einen vereinbarten Lebenszyklus zu überwachen und zu warten
- Der Lebenszyklus eines Gerätes beginnt bereits bei der Herstellung und endet bei der Entsorgung

III. Durchführung von dynamischen Tests sind notwendig

- **Statisches Testen:**
  - ✓ Software-Techniken, bei denen das Testen durchgeführt wird, ohne den Code auszuführen
  - ✓ **Beispiel**: Statische Code Analysis Tools (z.B.: FindBugs), Code Reviews
- Mit statischen Tests können keine Schwachstellen im Gesamtsystem (HW + SW) gefunden werden
- Dynamisches Testen, das den Code auch ausführt, ist notwendig, um Schwachstellen im Code und in der zugrunde liegenden Hardware zu entdecken
- IoT Geräte müssen gründlich getestet werden, um eine Mindestanforderung an Sicherheit zu erfüllen

IV. Datenschutz bei der Entsorgung von Geräten

- IoT Geräte dürfen bei der Entsorgung keinerlei private Daten preisgeben

[9] Siehe: Over-the-Air-Update – Wikipedia
[10] Siehe: Long Range Wide Area Network – Wikipedia
[11] Siehe: Denial of Service – Wikipedia

- Aus nicht fachgerecht entsorgten IoT Geräte können Informationen über den ehemaligen Besitzer und dessen IoT Umgebung extrahiert werden (gilt auch für Geräte, die weitergegeben werden)
- Benutzer verfügen oft über zu wenig Wissen, um zu verstehen welche persönlichen Informationen in den IoT Geräten gespeichert sind
  - ✓ **Beispiel**: Kopierer mit eingebauter Festplatte
- Hersteller sollten Benutzern eine konkrete Beschreibung für die fachgerechte Entsorgung zur Verfügung stellen
- Hersteller sollten die Entsorgung der Geräte anbieten

V. Starke Authentifizierung verwenden

- Authentifizierung: what you know, what you have, what you are, combinations
- IoT Geräte sollten keine einfach zu erratenden Anmeldeinformationen (engl. *Credentials*) verwenden (z.B.: admin/admin)
- IoT Geräte sollten keine Hintertüren (*back doors*) oder versteckte Entwickler Credentials enthalten
- Jedes IoT Geräte sollte eine eindeutige Standard-Benutzername/Passwort Kombination anbieten
- Wenn immer möglich sollte Zweifaktor Authenfizierung (2FA) zum Einsatz kommen (**Beispiel**: Campus Buchs Gebäude)

VI. Starke Verschlüsselung und sichere Protokolle

- Daten sollten bei der Übertragung (*data in motion*) und bei der Speicherung (*data at rest*) verschlüsselt werden
- Verwendung der stärksten Verschlüsselung, vorzugsweise IPsec und/oder TLS/SSL
  - ✓ http → https, mqtt → mqtts, ws → wss, etc.
- Bei der Übertragung von unverschlüsselten und unsignierten Daten sollte sichergestellt sein, dass falsche Daten keinen oder nur geringen Schaden anrichten
- Key Management Systeme (KMS) und Key Management Interoperability Protocols (KMIP) verwenden

VII. Minimierung der Gerätebandbreite

- DDoS Attacken können auch mit schlecht gesicherten IoT Geräten durchgeführt werden (**Beispiel**: Mirai botnet)
- Vielen IoT Geräten steht viel zu viel Übertragungskapazität zur Verfügung
- Hersteller sollten den Netzverkehr ihrer Geräte auf ein vernünftiges Niveau limitieren (*kernel-level controls*), um die Geräte für DDoS Attacken unattraktiv zu machen
- Geräte sollten so programmiert sein, dass sie ungewöhnliches Verhalten detektieren und die Werkseinstellungen wiederherstellen
- Leistungsanforderungen jedes Gerätes berücksichtigen und angemessene Einschränkungen vornehmen, welche schwer umgangen werden können

VIII. Sensible Informationen schützen (need to know principle)

- **IoT Grundgedanke:**
  - ✓ Vernetzung von Gegenständen mit dem Internet oder anderen Netzwerken
  - ✓ IoT Geräte bieten Dienstleistungen an (sog. *Services*) und können von anderen Geräten gefunden und verwendet werden
- Oftmals wird zu viel sensible Information (unter anderem *PII*[12] (***P**ersonally **I**dentifiable **I**nformation*) z.B.: Eigentümer, Hostname, etc.) offengelegt, welche mit anderen Informationsquellen verknüpft werden kann, um Attacken durchzuführen
- Nur autorisierte Clients sollten in der Lage sein, Geräte im Netzwerk zu finden

IX. Ermutigung für Ethisches Hacken

- Forscher, die Schwachstellen entdecken und berichten, sind der Industrie dienlich

[12] Siehe: Personal data - Wikipedia

- Hacking für Forschungszwecke sollte der Gesetzgeber daher nicht unterbinden
- Hersteller können mehrfach von einem Bug-Bounty-System[13] profitieren
  - ✓ Milderung schlechter Pressenachrichten
  - ✓ Verbesserung der Produktqualität
  - ✓ Geringere Kosten im Vergleich zu Penetrationstests[14]

X. Einrichtung eines IoT Security und Privacy Certification Boards

- Hersteller müssen die Verantwortung für ihre Kreationen übernehmen bzw. für ihre Kreationen geradestehen
- IEEE und andere Organisationen (OST?) sollten Zertifizierungsprogramme anbieten
- Zertifizierungsstellen sollten überprüfen, dass
  - ✓ Daten verantwortungsbewusst gehandhabt, geschützt und geteilt werden
  - ✓ Protokolle keine sensiblen Daten offenbaren
  - ✓ Zertifizierte Hersteller auf Datenschutzprobleme umgehend reagieren
  - ✓ Authentifizierung angemessen stark ist
  - ✓ IoT Geräte nicht ungeschützt sind
  - ✓ IoT Geräte ein fälschungssicheres, identifizierendes Etikett haben mit einem Weblink, mit dem Kunden den Zertifizierungsstatus inkl. Gerätebeschreibung finden können

[13] Siehe: Bug-Bounty-Programm – Wikipedia

[14] Siehe: Penetrationstest (Informatik) – Wikipedia

4 Aufgaben

4.1 Aufgabe 1a: Prüfsummenberechnung von Nachrichten

In dieser Aufgabe berechnen Sie zur Überprüfung der Integrität die Prüfsumme (engl. Checksum) einer zusammengesetzten Nachricht (engl. Message). Abbildung 4 stellt schematisch den Ablauf der Prüfsummenbildung dar:

Abbildung 4: Prüfsummenbildung

Message beschreibt die Nachricht, für die eine Prüfsumme berechnet werden soll. *MessageDigest* stellt die kryptographische Hashfunktion SHA-256 bereit. *Digest/Hash* wird als Prüfsumme in einem *Byte-Array* gespeichert.

1. Öffnen Sie mit *BlueJ* aus dem Projektordner …*Security* die Projektvorlage *CheckSum*.
2. Kompilieren Sie die Klasse `CheckSum`.
3. Starten Sie in *BlueJ* `CheckSum` mit der Methode `checkSumOfMessage`. Öffnen Sie anschliessend das *Terminalfenster* und überprüfen Sie das Ergebnis: Welche Nachricht wurde von der Methode gesendet? Aus wieviel Bytes besteht die Prüfsumme?
4. Studieren Sie nun den Code in der Methode `checkSumOfMessage`.
5. Anschliessend implementieren Sie die Methode `checkSumOfCompositeMessage`: Teilen Sie den Inhalt der Nachricht `message` in drei gleiche Teile `m1`, `m2` und `m3` auf und fügen Sie die Teilnachrichten der Instanz von `MessageDigest` mit der Methode `update(Byte[])` hinzu. Bestimmen Sie die Prüfsumme und geben Sie diese sowie die Länge der Prüfsumme in Bytes und Bits im Terminalfenster aus.
6. Vergleichen Sie das Ergebnis der Prüfsummen von `checkSumOfMessage` und `checkSumOfCompositeMessage`. Was fällt Ihnen auf?

4.2 Aufgabe 1b: Prüfsummenberechnung von Dateien

Dateien können während eines Downloads abgefangen, manipuliert der zufällig verändert werden. Damit nachprüfbar ist, ob eine Datei unverändert heruntergeladen wurde und die Integrität gewährleistet bleibt, erzeugt der Ersteller der Downloaddatei vor der

Übertragung ins Internet mittels eines bestimmten Programmes eine Prüfsumme. Die Prüfsumme, für die zum Download angebotene Datei, stellt er dann zusätzlich zur Downloaddatei im Internet zur Verfügung.

1. Studieren Sie nun in der Projektvorlage *CheckSum* den Code in der Methode `checkSumOfFile`. Welche kryptographische Hashfunktion wird für die Bestimmung der Prüfsumme von der Datei verwendet?
2. Starten Sie in *BlueJ* `checkSumOfFile` und öffnen Sie anschliessend das *Terminalfenster* mit der erstellten Prüfsumme der Datei.
3. Öffnen Sie die Online-Berechnung und bilden Sie Prüfsumme von der zu überprüfenden Datei *...\Security\Checksum\20230223_171546.jpg*. Vergleichen Sie die Werte und überprüfen Sie, ob diese identisch sind.
4. Ersetzen Sie den Prüfsummen-Algorithmus SHA-256 durch SHA-512 und vergleichen Sie die Längen der Prüfsummen. Was fällt hier auf?

4.3 Aufgabe 2: Symmetrische Verschlüsselung

Führen Sie eine Verschlüsselung eines Klartextes (engl. Plaintext) mit anschliessender Entschlüsselung des Chiffriertextes (engl. Chiphertext) mit einem AES-Schlüssel durch.

Abbildung 5 stellt schematisch den Ablauf der Symmetrischen Verschlüsselung dar:

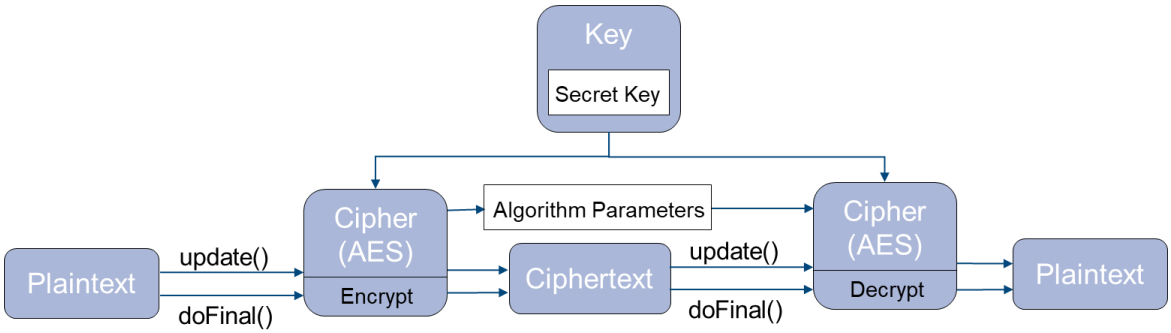

Abbildung 5: Symmetrische Verschlüsselung

Die Klasse *Cipher* stellt den Ver- und Entschlüsselungs-Algorithmus (engl. *Encrypt* and *Decrypt*) zur Verfügung. Dabei ist zu beachten, dass der **gleiche Schlüssel** sowohl für *Encrypt* als auch für *Decrypt* zu verwenden ist:

1. Öffnen Sie dazu mit *BlueJ* aus dem Projektordner …*Security* die Projektvorlage *AES*.
2. Kompilieren Sie die Klasse `AES`.
3. Studieren Sie nun den Code der statischen Methoden `encrypt` und `decrypt`. Aus wieviel Bits besteht der AES-Schlüssel? Was bewirkt `init` von der Instanz-Variable `cipher` in den beiden Methoden `encrypt` und `decrypt`?
4. Definieren Sie einen eigenen Klartext (`plainText`) und einen Schlüssel mit genau 16 Character und weisen Sie den Wert der Variable `myKey` zu. Starten Sie in *BlueJ* `AES` mit der Methode `main`. Öffnen Sie anschliessend das *Terminalfenster* und überprüfen Sie das Verschlüsselungsergebnis.

4.4 Aufgabe 3: Asymmetrische Verschlüsselung

Führen Sie eine Verschlüsselung eines Klartextes mit anschliessender Entschlüsselung des Chiffriertextes mit einem RSA-Schlüssel durch. Dabei ist zu beachten, dass für die Verschlüsselung einen öffentlichen (engl. *Private Key*) und für die Entschlüsselung einen privaten (engl. *Private Key*) Schüssel in Form eines **Schlüsselpaares** zu generieren ist:

1. Öffnen Sie dazu mit *BlueJ* aus dem Projektordner …*Security* die Projektvorlage *RSA*.

2. Kompilieren Sie die Klasse `RSA`.
3. Studieren Sie nun den Code der statischen Methoden `generateKey`, `encrypt` und `decrypt`. Aus wieviel Bits besteht der RSA-Schlüssel? Was bewirkt `init` von der Instanz-Variable `cipher` in den beiden Methoden `encrypt` und `decrypt`?
4. Definieren Sie einen eigenen Klartext. Starten Sie in *BlueJ* `RSA` mit der Methode `main`. Öffnen Sie anschliessend das *Terminalfenster* und überprüfen Sie das Verschlüsselungsergebnis.

4.5 Aufgabe 4: Verschlüsselte Internet-Kommunikation

Um die Vertraulichkeit und Integrität in der Kommunikation zwischen Webserver und Webbrowser zu ermöglichen, werden die Daten verschlüsselt im HTTPS-Protokoll übertragen. Dabei geschieht die Verschlüsselung mittels SSL/TLS. Eine HTTPS-Verbindung wird durch eine https-URL angewählt und durch das SSL-Logo angezeigt: Dies wird als kleines Schloss-Symbol in der Adresszeile dargestellt.

Selbstsigniertes SSL/TLS-Zertifikat erstellen

Bevor Sie einen HTTPS-Server implementieren und prüfen, ist ein selbstsigniertes SSL/TLS-Zertifikat zu erstellen. Dies bedeutet, dass das Zertifikat seine eigene CA ist und nicht von einer Zertifizierungsstelle ausgestellt wird

1. Öffnen Sie die Eingabeaufforderung als Administrator. Anschliessend wechseln Sie in der Eingabeaufforderung in den Java-Bin-Ordner (z.B. `C:\Program Files (x86)\Java\openjdk\bin>`)

2. Für die Zertifikatserstellung geben Sie folgenden Befehl ein:

   ```
   keytool -genkeypair -keyalg RSA -alias selfsigned -keystore testkey.jks -storepass password -validity 360 -keysize 2048
   ```

   Der Befehl fragt Sie nach Ihrem *Vor*- und *Nachnamen*, der *Organisationseinheit*, der *Organisation*, der *Stadt*, dem *Kanton* und dem zweistelligen *Ländercode*. Geben Sie alle Informationen ein und bestätigen Sie *Ja*.

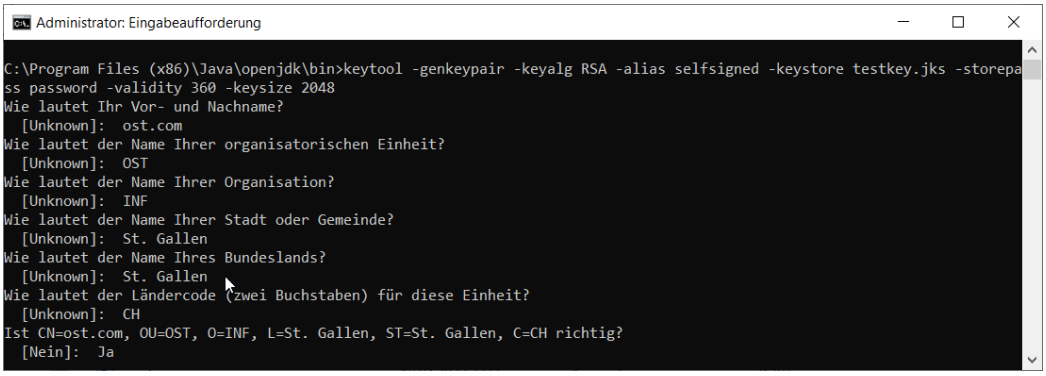

Abbildung 6: Zertifikatserstellung

3. Die Datei *testkey.jks* wurde im Java-Bin-Ordner neu erstellt. Kopieren Sie diese in Ihren Java-Projektordner *…\Security\HTTPSServer*, um das Zertifikat für die HTTPS-Verbindung zu verwenden.

HTTPS-Verbindung implementieren prüfen

1. Öffnen Sie dazu mit *BlueJ* aus dem Projektordner *…\Security* die Projektvorlage *HTTPSServer*.

2. Kompilieren Sie die Klasse `HTTPSServer`.

3. Studieren Sie nun den Code von der Methode `main`: Für den gesicherten Datenaustausch wird ein Schlüsselspeicher (Klasse `KeyStore`) verwendet. Die Instanz `keyStore` benötigt für das Laden der Zertifikatsdatei *testkey.jks* das Passwort. Weisen Sie das Passwort der Variable `password` zu (Hinweis: Studieren Sie dazu den Parameter `-storepass`, welcher Teil der Befehlseingabe für die Zertifikatserstellung ist).

4. Starten Sie in *BlueJ* `HTTPSServer` mit der Methode `main`.

5. Öffnen Sie den Webbrowser und testen Sie die Verbindung mit der Adresse `https://localhost:9000/page`.
   Hinweis: Da ein selbstsigniertes Zertifikat eingesetzt wird, erscheint nach der Adresseingabe im Webbrowser die Nachricht «Nicht sicher» – bei einem öffentlichen Zertifikat wird Schloss-Symbol angezeigt.

6. Klicken Sie auf das Symbol und suchen Sie in der Anzeige das Zertifikatssymbol. Öffnen den Zertifikats-Viewer mit Klick auf das Symbol und wählen Sie in der Rubrik *Details* das Zertifikatsfeld *Zertifikatsignaturalgorithmus* aus. Welcher Algorithmus wird hier verwendet?

7. Vervollständigen Sie nun die Klasse `MySecondTestHandler`: Fügen Sie der Instanz `buffer` vom Typ `StringBuffer` folgende HTML-Zeilen für die Servernachricht hinzu:

```
buffer.append("<HTML>\n");
buffer.append("<HEAD><TITLE>HTTPS
   Server</TITLE></HEAD>\n");
buffer.append("<BODY>\n");
buffer.append("<H1>Success!</H1>\n");
```

8. Starten Sie in *BlueJ* `HTTPSServer` mit der Methode `main` und öffnen Sie den Webbrowser. Testen Sie die Verbindung mit der Adresse `https://localhost:9000/secondpage` und überprüfen Sie die Nachricht.

5 Anhang

5.1 SSL/TLS

Die Abkürzung *SSL* bedeutet *Secure Sockets Layer*. Dabei handelt es sich um ein Protokoll, das sicherstellt, dass Sie mit einer verschlüsselten Verbindung im Internet Daten übertragen können. In Version 3 wurde das Protokoll in *TLS* (*Transport Layer Security*) umbenannt. Zunächst findet eine geschützte Authentifizierung und Autorisierung der der Kommunikationspartner statt. Mit Hilfe der asymmetrischer Verschlüsselung erfolgt der Schlüsseltausch für die jeweilige Session. Dabei dient der Schlüssel dient zur Verschlüsselung der Daten.

