Modul C1
Sicherheit in Web-Applikationen

Sicherheit

Philipp Gressly © 15. Juni 2006
Santis Training AG (Zürich)

Inhaltsverzeichnis

Präambel

Im folgenden Artikel wird für Personen in der Regel die kürzere Schreibweise verwendet. Selbstverständlich sind darunter Personen des jeweils anderen Geschlechts mitgemeint.

1 Sicherheit

1.1 Angriffe

Neben den normalen Problemen des Datenverlusts und der Fehlmanipulation gibt es eine Fülle von *bösartigen* Angriffen auf eine Web-Applikation. Überlegen Sie zu den folgenden Angriffen je eine Gegenmaßnahme und tragen Sie Ihre Überlegungen in die Tabelle (S. Kap. A.1 auf Seite 32) ein.

1.1.1 Spionage

Um an vertrauliche oder wertvolle Firmendaten zu gelangen gibt es eine Menge Tricks. Eine Variante ist das (illegale) herunterladen ganzer Datenbestände von Web-Applikationen. Wie können wir uns dagegen wehren?

Beispiel 1 (GET-IDs) *Eine einfache Webapplikation findet die Artikel oder Produkte mit einem GET-Parameter:* `http://www.xyz.com?art_id=35`*. Somit kann ein Angreifer alle Artikelnummern*[1] *ausprobieren und die ganze Datenbank in kurzer Zeit ausspionieren.*

1.1.2 Passwort Cracker / Passwort Guesser

Wer an einem System genügend *Logins* durchführen darf, kann mit einem einfachen, jedoch zeitaufwändigen Verfahren Passwörter herausfinden. Sogenannte "Cracker"-Angriffe probieren *brute-force*[2] alle Möglichkeiten durch. Password-Guesser hingegen versuchen aufgrund von Daten des Benutzers (Geburtsdatum, Namen von Verwandten, Beruf, ...) an mögliche und sinnvolle Passwörter heranzukommen.

Meistens geschehen Passwort-Angriffe aber nicht von extern (also von außerhalb der Firma). Wer im Besitz einer Passwortliste ist, kann autorisierten Zugriff auf verschiedene Konten erhalten.[3]

[1] `art_id=1, 2, ...`

[2] "Mit aller Kraft", stur, durch simples Probieren aller Varianten

[3] Das funktioniert auch für verschlüsselte Passwortlisten, falls diese mit öffentlich bekannten *Hash*-Funktionen arbeiten.

Unsichere SQL-Anfragen Eine spannende Art, sich Zugang zu einem Webserver ohne Rechte zu verschaffen, funktioniert mittels *unsicheren* SQL-Abfragen. Häufig werden Anfragen an die Datenbank wie folgt gestellt (hier ein PHP Beispiel):

```
<?php
  $sql="SELECT * FROM user WHERE name='$uname'";
  $result=mysql_query($sql);
?>
```

Ein schlauer Benutzer könnte nun bei der Anfrage nach seinem Benutzernamen ins Feld einfach folgenden Eintrag tätigen.

Benutzer: `blah'; UPDATE user SET password='simple`

Falls dieser Benutzername eins-zu-eins in die Variable `$uname` eingesetzt wird, so lautet die SQL-Anfrage nun wie folgt:

```
  $sql="SELECT * FROM user WHERE name='blah';
        UPDATE user SET password='simple'";
```

Der Benutzer ist zwar damit noch nicht *eingeloggt*, aber das Passwort wurde bei allen Benutzern nach `'simple'` abgeändert. Somit kann ein späteres *Einloggen* nicht wirklich schwierig sein.

Abhilfe Abhilfe verschafft man sich, indem vor alle Apostrophe, die vom Benutzer eingegeben wurden, ein `Back-Slash "\"` vorangestellt wird[4]. Somit kann ein SQL-Statement nicht mehr mutwillig beendet werden.

1.1.3 Horcher und "The-Man-in-the-Middle"

Ein Abhorcher (Horcher) schaut sich den ganzen Datentransfer zwischen zwei Sockets an und versucht so, Informationen über die Schwachstellen des Systems zu erhalten. Später kann er mit diesem Wissen das System direkt angreifen.

Ein *Man-in-the-Middle* dagegen fängt den gesamten Verkehr zwischen zwei Systemen ab, modifiziert den Inhalt und sendet die Änderungen ans Gegenüber. Somit ist es einem *Man-in-the-Middle* z. B. möglich, einem System vorzugaukeln, er sei ein legaler Kunde. Ein solches Einschleusen funktioniert nur bei "langsamen" Transaktionen (z. B. E-Mail), wo die Endpunkte nichts von der Verzögerung (die durch die Veränderung am Inhalt entsteht) mitbekommen.

1.1.4 E-Shop Lifting

Falls es einem Angreifer möglich ist, die Preise auf einer Webseite zu verändern und so Angebote billiger zu erschleichen, so sprechen wir von **E-Shop Lifting** oder auch von "virtuellem Ladendiebstahl" [CT Nr. 26 2002]. Das funktioniert z. B. dann, wenn der "Shop" die Preise in *Hidden-Fields* auf dem Client ablegt.

[4] PHP kennt hier die Methode `addslashes()`

1.1.5 Session Hijacking

Ein Angreifer *übernimmt* eine bestehende Sitzung. Dieses Vorgehen wurde bei TCP[5] eingehend untersucht. Natürlich ist dies auch bei simpel gewählten SessionIDs in HTML-Anwendungen (z. B. 1, 2, 3, ...) keine Hexerei.

1.1.6 Man-in-the-Middle Attacken

Bei Man-in-the-Middle Attacken wird die komplette Verbindung abgefangen, betrachtet, evtl. abgeändert und weitergeleitet. Das funktioniert einfach, bei langsamen Verbindungen (z. B. E-Mail). Falls die Nachrichten 1:1 weitergeleitet werden, haben die beiden Kommunikationspartner keine Möglichkeit den Man-in-the-middle zu erkennen. Aber auch wenn sich dieser geschickt anstellt, können die beiden nie herausfinden, dass sie in Tat und Wahrheit nicht mit ihrem vermeindlichen Partner kommunizieren, sondern eben lediglich mit “The Man in the Middle”.

1.1.7 Viren, Würmer und anderes Getier

Viren, Würmer und Enten (Hoax) gefährden in erster Linie die Endanwender und nicht die Web-Applikation. Es gibt jedoch immer wieder Fälle, wo auch die Webserver mehr oder weniger gezielt attackiert werden.

1.1.8 DoS Attacken, Trojaner und Hintertüren

Webserver werden eher Ziel einer **Denial of Service**-Attacke (DoS) als der gemeine Heimanwender. Dabei machen mehrere PCs gleichzeitig simple Anfragen an einen Webserver. Dieser Server wird dann durch die Fülle von Anfragen lahmgelegt.

Um DoS-Attacken vorzubereiten, werden häufig sogenannte **trojanische Pferde**[6] eingesetzt. Diese setzen, ohne das Wissen des PC-Besitzers, zu einem bestimmten Zeitpunkt Anfragen auf das Opfer der DoS-Attacke ab.

Trojanische Pferde können aber auch eingesetzt werden, um **Hintertüren** (sog. *Backdoors*) zu öffnen. Mit offenen Hintertüren ist es einem entfernten Angreifer möglich, alle Information über das System zu erhalten und dieses auch nach seinen Wünschen zu modifizieren.

1.2 Schutz

1.2.1 Autorisierung/Authentifizierung

Um auf einem entfernten System arbeiten zu können, braucht es Zugriff (Authentifizierung) und Berechtigungen (Autorisierung).

[5] Transfer Control Protocol

[6] Griechische Mythologie (Ilias): Die Griechen eroberten die Stadt Troja mit Hilfe des hölzernen Trojanischen Pferdes, in dessen hohlem Bauch sich die tapfersten Helden verbargen und so von den ahnungslosen Trojanern in die Stadt geführt wurden.

Die Authentifizierung geschieht im Normalfall mit Passwörtern. Es wird unterschieden zwischen schwacher (allein mittels Passwörtern) und starker Authentifizierung. Letztere benötigt *something to know* **und** *something to have* (Passwort **und** Streichliste). Die Autorisierung (Bevollmächtigung mit Privilegien) geschieht nach der Authentifizierung.

Definition 1 (Authentifizierung) ***Authentifizieren*** *heißt: "Die Echtheit von etwas bezeigen, beglaubigen." [Wahrig] In der Informatik wird ein Benutzer oder ein System (Software, Client, ...) authentifiziert. Der Server will wissen,* ***wer*** *den Dienst in Anspruch nimmt. Dieses Wissen über das Gegenüber erlaubt z. B. eine Autorisierung oder eine finanzielle Abrechnung. Zur starken Authentifizierung kann, neben Passwörtern mittels Streichlisten oder Secure-IDs vorgegangen werden.*

Definition 2 (Autorisierung) *Das WAHRIG Fremdwörterlexikon umschreibt* ***Autorisierung*** *mit "Bevollmächtigung". In verteilten Systemen ist es wichtig, dass nur ermächtigte Personen Privilegien auf bestimmten Daten erhalten: hinzufügen, suchen, ansehen, löschen, verändern, vergeben weiterer Rechte, ... Hier geht es darum,* ***was*** *eine Person oder ein System tun darf.*

1.2.2 Verfahren und Werkzeuge

Für die Autorisierung existieren grundsätzlich 3 Verfahren

- Etwas, das man **weiß** (knowledge) [PIN, Passwort, ...],
- etwas, das man **hat** (possesion) [Chipkarte (SmartCard) [7], Ausweis, Schlüssel, Token, SecurID[8], Streichlisten, ...] und
- etwas, das man **ist** (biology , Biometrie) [Fingerabdruck, Spracherkennung, Iriserkennung, DNA(?), ...].

Um die Sicherheit weiter zu erhöhen gibt das daneben noch weitere Verfahren:

- Backup / Restore,
- Zugangsschutz zu Räumen und Daten (Eingangskontrollen, Tresortüren, Tresore, Schlüssel),
- Firewalls
- Schutz vor Trojanern (Virenschutz, ...)
- ...

[7] Smartcards werden auch Chipkarten genannt. Darauf sind Zertifikate gespeichert und unter Umständen auch kleine Programme, die bei der Autorisierung gestartet werden können.

[8] SecurID: Nachteil ist, dass die RSA-Security das Verfahren nicht offenlegt, was die Sicherheit unter Umständen einschränkt.

1.2.3 Kryptographie

Kryptographie bezeichnet die Verschlüsselung von Daten, um diese vor fremder Einsicht zu schützen. Ebenso soll auch die Echtheit des Absenders überprüft werden können, und seine Rechte müssen definiert werden. Verfahren zur Kryptographie werden im Kapitel Kryptographie (S. Kap. 2 auf Seite 9) genauer behandelt.

1.2.4 E-Banking

Auch bei E-Banking ist Vorsicht geboten. Nicht immer sind die Systeme so sicher, wie man glaubt. Die größten Lücken bestehen aber typischerweise immer noch beim Anwender. Folgende Punkte sollten (neben der Wahl von *sicheren* Passwörtern und der Tatsache, dass Streichlisten und dazugehörige Passwörter nicht am selben Ort abgelegt werden sollten) beachtet werden, wenn "sichere" Transaktionen mit der Bank vorgenommen werden (Quelle [Tages Anzeiger 2003 11. 17.] [*luc*]):

1. Vor dem Einloggen alle Browserfenster schließen. Während des Bankgeschäfts keine anderen Internetseiten *ansurfen*. Die Anwendung immer über *Logout* verlassen.
2. Sicherheitsnummern und Passwörter nicht auf dem Computer speichern.
3. Das von der Bank zugeteilte Passwort sofort ändern. Alle paar Monate auswechseln.
4. Immer aktuelle Sicherheits*updates* für Browser und eventuell verwendete Software installieren.
5. Nach der Session den *Browser-Cache* (Zwischenspeicher) leeren.
6. Computer gegen Viren schützen.
7. Wenn Sie vom Finanzinstitut ein E-Mail erhalten mit einem Link und der Aufforderung zum *Log-in* oder mit einem *Attachment*, löschen Sie es. Im Zweifelsfall rufen Sie dort an und erkundigen sich, ob das E-Mail tatsächlich verschickt wurde.
8. Geben Sie nie telefonisch oder per E-Mail Passwort und Sicherheitsnummern durch. Finanzinstitute verlangen nie danach, vertrauen Sie deshalb niemandem, der sie erfragt (Phishing).
9. Darauf achten, dass in der Statuszeile des *Browsers* ein Schloss angezeigt wird (das heißt: verschlüsselte Verbindung). Andernfalls Internetverbindung kappen und nochmals einwählen.
10. Kontoauszüge immer genau kontrollieren.

2 Kryptographie

2.1 Überblick

- In diesem Dokument wird zunächst mit historischen Verfahren und auch mit dem XOR-Verfahren (S. Kap. 2.2.4 auf Seite 11) begonnen. Danach werden einige mathematische Grundlagen gegeben. Diese Grundlagen werden dann im Verfahren von Diffie/Hellman und RSA (S. Kap. 2.5 auf Seite 20) angewendet. Im letzten Abschnitt (S. Kap. 2.7 auf Seite 23) wird noch erklärt, wie die mathematischen Hilfsprogramme (JAVA) zu bedienen sind. Denn: wer rechnet schon gerne `;-)`

- Wer sich weder für die Geschichte der Kryptographie noch für die mathematischen Hintergründe interessiert, kann gleich zur Anwendung der Verfahren blättern: Diffie/Hellman (S. Kap. 2.4 auf Seite 18) / RSA (S. Kap. 2.5 auf Seite 20).

2.2 Einführung

- Überlegen Sie sich, wozu Verschlüsselungstechniken eingesetzt werden und was die Aufgabe eines Krypto-Analytikers ist.

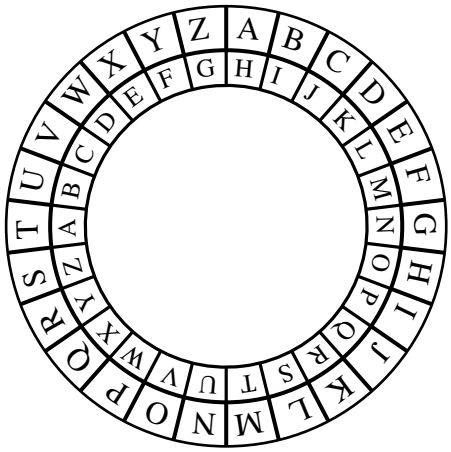
- In der vernetzten Welt gewinnen Verschlüsselungsverfahren wie z. B. Diffie/Hellman ssl[9] (`https://`), SSH (secure shell) , PGP[10], GnuPG[11], ... zunehmend an Bedeutung. Doch wie funktionieren sie? Sind solche Verfahren sicher?

- Mathematische Verschlüsselungsverfahren, die auf großen Primzahlen basieren, haben die Stärke, dass genau berechnet werden kann, wie groß der durchschnittliche (zeitliche, rechnerische) Aufwand sein wird, um eine Botschaft unrechtmäßig zu "entschlüsseln".

[9] secure socket layer
[10] Pretty good privacy
[11] Gnu Privacy Guard

2.2.1 Geschichte

Beginnen wir mit dem Verfahren der Spartaner (oder waren es die Babylonier?). Sie verwendeten bereits im 5. Jh. v. Chr. zur Verschlüsselung einen runden Stab und einen Papierstreifen (Linke Abbildung).



Nur wer einen Stab der selben Dicke besaß, war in der Lage, derart verschlüsselte Botschaften zu entziffern.

Überlegen Sie sich, mit welchem Aufwand dieses Verfahren "geknackt" werden kann.

2.2.2 Cäsar

Der Algorithmus von Cäsar ist nicht viel sicherer (Rechte Abbildung). Das Verfahren verschiebt die Buchstaben im Alphabet um eine vorgegebene Anzahl. Es kommen 26 Buchstaben im Alphabet vor, somit gibt es lediglich 26 mögliche Schlüssel. Bedenken Sie aber, dass um 50 v. Chr. noch fast niemand lesen oder schreiben konnte. Somit war das Verfahren für die damalige Zeit sicher genug.

Beispiel 2 (Cäsar) *Der Schlüssel sei 7. Das heißt, jeder Buchstabe wird im Alphabet um 7 Zeichen nach hinten verschoben.*

"Hallo" $\Rightarrow$ *"Ohssv".*

2.2.3 Monoalphabetische Substitutionen

A	B	C	D	E	F	G	...
x	π	σ	r	ϱ	!	ϑ	...

Häufig wird anstelle einer Translation des Alphabetes eine beliebige Permutation (Vertauschung) verwendet. Ob man wieder Buchstaben oder irgendwelche kryptisch anmutenden Zeichen verwendet, spielt hier keine Rolle.

Das Verfahren kann jedoch auch sehr rasch *geknackt* werden. Wie wohl?

Bemerkung: Das Verfahren von Cäsar ist ein Spezialfall einer monoalphabetischen Substitution.

2.2.4 Das XOR-Verfahren

Wie funktioniert das XOR-Verfahren? Wir gehen davon aus, dass eine Datei verschlüsselt werden sollte. Jede Datei können wir einfach als eine Byte- bzw. eine Bitfolge auffassen (wir können z. B. Unicode verwenden). Wenn nun der Sender wie auch der Empfänger vorab eine zufällige Bitfolge (Schlüssel) ausgetauscht hatten, so können Sie die Botschaft bitweise mit der exklusiven Oder-Operation (XOR) verknüpfen.

Das XOR-Verfahren ist absolut sicher, wenn wir davon ausgehen, dass der Schlüssel genug streut (die Null- und Einsbits sind rein zufällig gewählt). Jetzt kann das Verfahren nicht mehr geknackt werden. Um an die Information zu kommen, müsste schon der Schlüssel "geraubt" werden.[12]

Wichtig ist auch zu wissen, dass ein XOR-Schlüssel nur einmal eingesetzt werden sollte.

Das XOR-Verfahren kann z. B. auch mit einem Pseudozufallszahlen-Algorithmus gestartet werden. Hierbei ist der Schlüssel eine sogenannte "Random-Seed"-Zahl. Wenn zwei gleich gebaute Zufallszahlengeneratoren mit demselben Startwert beginnen, so liefern sie auch dieselbe Zahlenfolge. Das hat den Vorteil, dass nur eine kleine Information ausgetauscht werden muss. Das Verfahren verliert dabei aber an Sicherheit!

2.2.5 Public Key

James H. Ellis hat 1970 ein Verfahren entwickelt, bei dem Schlüssel, oder zumindest Teile davon, öffentlich übermittelt werden können. Auch wer diese Schlüsselteile kennt, kann Botschaften noch nicht genügend rasch knacken. Die Idee ist brillant, doch – werden Sie sich fragen – geht das überhaupt?

Die bisher kennengelernten Verfahren waren sogenannte symmetrische Verfahren . Sie haben die Eigenschaft, dass der Algorithmus zum Verschlüsseln identisch (bzw. leicht umkehrbar) ist mit demjenigen zum Entschlüsseln. Die im folgenden beschriebenen (asymmetrischen) Verfahren benötigen zum Entschlüsseln jedoch einen anderen Schlüssel. Manchmal werden auch Verfahren asymmetrisch bezeichnet, die eine mathematische Funktion benötigen, deren Umkehrfunktion ungleich mehr Rechenaufwand benötigt (Multiplizieren / Faktorisieren).

In der Regel wird bei einem solchen Public-Key[13]-Verfahren eine Rechenoperation eingesetzt, die nicht einfach umzukehren ist. Denken Sie z. B. an das Rechnen aus der Grundschule. Das Multiplizieren zweier Zahlen geht rasch und einfach. Jedoch die beiden Zahlen zu finden, die zu dem Produkt geführt haben, braucht schon wesentlich größeren Aufwand. Auf einer ähnlichen Grundidee basieren moderne Krypto-Verfahren.

Erst mittels solcher "Einwegfunktionen"[14] wird es möglich, dass zwei Parteien, die vorher noch nie miteinander in Kontakt getreten sind, geheime Botschaften austauschen! Bisher mussten (wie beim XOR-Verfahren) die Parteien vorher einen Schlüssel über einen geheimen Kanal verschicken!

[12] Ein analoges Verfahren zum XOR-Verfahren ist der *One Time Pad* von AT&T (1917).

[13] Public-Key = öffentlicher Schlüssel

[14] Im Gegensatz zu Hash-Funktionen (S. Kap. 2.3.6 auf Seite 17) sind diese Funktionen umkehrbar. Jedoch ist der Aufwand, die Funktion umzukehren enorm viel höher, als die Funktion zu berechnen (S. Kap. 2.3.5 auf Seite 16).

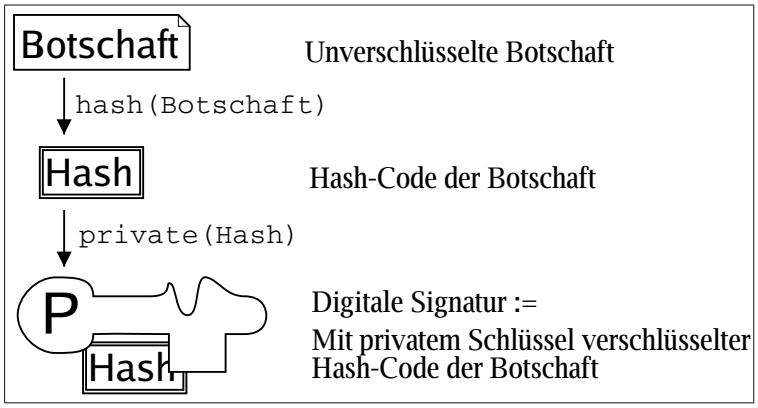
2.2.6 Digitale Signatur

Die digitale Signatur entspricht einer Unterschrift oder einem Siegel. Nur wer den Siegelring (hier den Private-Key) besitzt, kann die Signatur anfertigen.

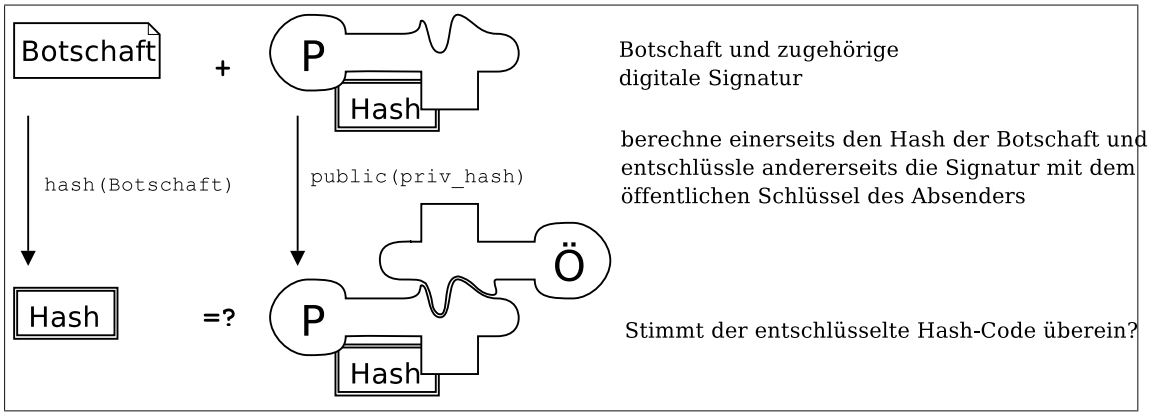
Im Gegensatz zur Geheimhaltung bleibt beim digitalen signieren die Botschaft unverschlüsselt. Es wird lediglich ein Hash-Code (S. Kap. 2.3.6 auf Seite 17) der Botschaft **verschlüsselt**; und zwar diesmal mit dem **Private-Key**.

Alle sollen die Herkunft der Botschaft überprüfen können. Hierzu wird mit dem Public-Key der verschlüsselte Hash-Code dechiffriert und mit dem Hash-Code der unverschlüsselten Botschaft verglichen. Da der Public-Key öffentlich zugänglich sein soll, ist es für jede Person möglich, die Unterschrift auf Echtheit zu prüfen; vorausgesetzt natürlich, dass bereits dem Public-Key vertraut werden kann ;-)

Signieren:



Signatur prüfen:



2.2.7 Digitale Zertifikate

Die Frage aus dem vorangehenden Kapitel soll hier nochmals aufgenommen werden: wie kann ich einem Public-Key trauen? Natürlich ist das einfach, wenn ich die zugehörige Person od. Institution kenne. Im Internet ist das jedoch selten der Fall. Zur Lösung dieses Problems bieten sich z. B. Zertifikate an:

Eine öffentlich bekannte Zertifizierungsstelle (ZS) unterschreibt sogenannte digitale Zertifikate.

Ein digitales Zertifikat ist die Unterschrift (digitale Signatur) der Zertifizierunsstelle auf

dem öffentlichen Schlüssel (public key). Etwas präziser: Ein Public Key wird zusammen mit dem Namen der zugehörigen Person (bzw. Institution) von der ZS signiert.

Ein digitales Zertifikat besteht aus den folgenden Komponenten:

- (A) Public Key des Inhabers
- (B) Name des Inhabers
- (C) Name der Zertifizierungsstelle
- Public Key der Zertifizierungsstelle
- Digitale Signatur von A,B und C

2.3 Mathematische Grundlagen zu Krypto-Verfahren

- Dieses Kapitel beleuchtet die mathematischen Hintergründe, die für die Anwendung der Verfahren RSA, ElGamal und Diffie/Hellman notwendig sind.
- Diese Einführung erhebt keinen Anspruch auf Vollständigkeit. Insbesondere werden wichtige Beweise weggelassen.
- Es geht in den nachfolgenden Kapiteln lediglich darum, dass die beiden Verfahren RSA und Diffie/Hellman in groben Zügen verstanden und angewendet werden können.
- Trotz meinem pragmatischen Ansatz werden einige Grundlagen der Zahlentheorie eingeführt:

2.3.1 ggT

Der "ggT" von natürlichen Zahlen, ist der **größte gemeinsame Teiler**. (engl. GCD = greatest common divisor). Def.: Der größte gemeinsame Teiler von zwei Zahlen ist die größte ganze Zahl, die beide Zahlen ohne Rest teilt.

Beispiel 3 (ggT) *größter gemeinsamer Teiler:*

```
ggT (48, 32) = 16.
ggT (10, 11) = 1
ggT (50, 70) = 10
ggT (35, 63) = 7
```

Bemerkung 1 *Zwei Zahlen a und b sind genau dann teilerfremd, wenn* $ggT(a, b) = 1$ *ist.*

Um den ggT von zwei großen Zahlen zu berechnen, verwenden Sie das Programm GCD:
`>java GCD i j`.

2.3.2 Primzahlen

Eine **Primzahl** ist eine Zahl, die **neben** sich selbst nur die 1 als Teiler hat. Mit anderen Worten: Eine Primzahl hat genau zwei Teiler. Die kleinste Primzahl ist 2.

Beispiel 4 (Primzahlen) *Hier die ersten zehn Primzahlen: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...*

Bemerkung 2 *Ist n eine beliebige (positive, ganze) Zahl, und ist p eine Primzahl, so gilt:*

ggT(n, p) = *p, wenn p Teiler von n ist.*
1, wenn p kein Teiler von n ist.

Für unsere Aufgaben müssen wir Primzahlen finden - je größer umso besser. Beispiele von Primzahlen finden wir z. B. auf dem Internet unter:

`http://www.geocities.com/primes_r_us/small/index.html`

Natürlich werden für sehr sichere Verschlüsselungen weitaus größere Primzahlen (ab 300 Stellen) verwendet. Abgesehen davon, dass die Primzahlverfahren sehr sicher sind, weisen sie noch eine zusätzliche Stärke auf: Es ist berechenbar, mit welchem durchschnittlichen Zeit- bzw. Rechenaufwand die Verfahren geknackt werden können.

2.3.3 Modulo p

Mit *Modulo* (mod) bezeichnen wir das Berechnen von Divisionsresten.

$m \bmod p :=$ Rest, der entsteht, wenn wir m durch p teilen.

Beispiele:

1. $17 \bmod 7 = 3$ (denn $17 = 2 * 7 + 3$)
2. $37 \bmod 5 = 2$ (denn $37 = 6 * 5 + 2$)

Bemerkung 3 (Distributivität von Modulo-Berechnungen) *Ist p eine Primzahl, so gelten folgende Regeln:*

$$(a+b) \bmod p = [(a \bmod p) + (b \bmod p)] \bmod p \quad (1)$$
$$(a*b) \bmod p = [(a \bmod p) * (b \bmod p)] \bmod p \quad (2)$$
$$(a)^n \bmod p = [(a \bmod p)^n] \bmod p \quad (3)$$

Beispiel 5 (Modulo Beispiele)

$$\begin{aligned}
(5+2) \bmod 3 &= ((5 \bmod 3) + (2 \bmod 3)) \bmod 3 = (2+2) \bmod 3 = 4 \bmod 3 = 1 \\
(8*15) \bmod 7 &= (1*1) \bmod 7 = 1 \bmod 7 = 1 \\
(14*35) \bmod 3 &= (2*2) \bmod 3 = 4 \bmod 3 = 1 \\
46^{10} \bmod 11 &= 2^{10} \bmod 11 = 4^5 \bmod 11 = (4*4^4) \bmod 11 = (4*16^2) \bmod 11 \\
&= (4*5^2) \bmod 11 = (4*25) \bmod 11 = (4*3) \bmod 11 = 1
\end{aligned}$$

Bemerkung 4 (RSA) *Für das RSA - Verfahren (S. Kap. 2.5 auf Seite 20) und das Verfahren von Diffie/Hellman (S. Kap. 2.4 auf Seite 18) brauchen wir $a^b \bmod c$ zu berechnen. Berechnen Sie zum Beispiel $23^{17} \bmod 7$ mit dem* JAVA *Hilfsprogramm* `AhBmC:`

```
>java AhBmC 23 17 7
```

2.3.4 Inverses Modulo p

Für das RSA-Verfahren benötigen wir noch die folgende Rechenoperation. Wenn Sie sich nur für Diffe-Hellman oder das Verfahren von ElGamal interessieren, so können Sie dieses Kapitel überblättern.

Wenn wir Modulo p rechnen und p eine Primzahl ist, so gibt es für jede Zahl m ein sogenanntes multiplikatives inverses Modulo p. Das heißt: für jedes m gibt es ein n, sodass $m * n \pmod p = 1$ ist.

$p = 7 \quad m = 3 \rightarrow n = 5$ (denn $3 * 5 \bmod 7 = 1$)
$p = 7 \quad m = 6 \rightarrow n = 6$ (denn $6 * 6 \bmod 7 = 1$)
$p = 11 \quad m = 5 \rightarrow n = x$ (berechnen Sie x selbständig)

Hier ein simples Vorgehen, um das Inverse (mod p) zu finden:

Nehmen wir z. B. $p = 13$ und $m = 5$

$5 * 2 \bmod 13 = 10 \bmod 13 = 10$
$5 * 3 \bmod 13 = 15 \bmod 13 = 2$
$5 * 4 \bmod 13 = 20 \bmod 13 = 7$
$5 * 5 \bmod 13 = 25 \bmod 13 = 12$
$5 * 6 \bmod 13 = 30 \bmod 13 = 4$
$5 * 7 \bmod 13 = 35 \bmod 13 = 9$
$5 * \mathbf{8} \bmod 13 = 40 \bmod 13 = \mathbf{1}$

Daraus ergibt sich 8 als das Inverse (mod 13) zu 5, denn $5 * 8 \bmod 13 = 1$.

Das JAVA Programm um das Inverse zu finden heißt `MInv`:

```
>java MInv 5 13
```

2.3.5 Einfach & Schwierig

Es gibt nun zwei Eigenschaften, die die Primzahlverfahren sehr sicher machen:

a) Die Zerlegung von großen Zahlen in ihre Primfaktoren ist schwierig, die Multiplikation dagegen ist einfach:

Sind p und q zwei 300-stellige Primzahlen, so ist $p*q$ einfach zu berechnen; die Primfaktorzerlegung einer 600-stelligen Zahl hingegen ist sehr zeitaufwändig ("pröbeln"). Genau diese Schwierigkeit nutzt das RSA Verfahren.

b) Exponenten Modulo einer Primzahl zu rechnen ist einfach. Die Umkehrung (den sog. diskreten Logarithmus) zu finden ist schwierig. Diese Schwierigkeit wird vom Diffie/Hellman-Verfahren wie auch vom ElGamal-Algorithmus ausgenutzt:

$a^b \bmod p$ zu berechnen ist einfach (siehe Beispiel 5).

Aus der Gleichung $a^n \bmod p = s$ das n zu berechnen, ist hingegen schwierig.

2.3.6 Hash-Funktionen

Hash-Funktionen sind schnell zu berechnende Schlüsseltransformationen. Sie bilden Daten (Suchschlüssel, Sortierschlüssel, beliebige Objekte, E-Mails, ...) auf einen vergleichsweise kleinen Wertebereich ab. So wird aus einem großen Objekt eine kleine Datenstruktur aus wenigen Bytes.

Hash-Funktionen haben die folgenden Eigenschaften:

funktionales Verhalten	Jede Hash-Funktion liefert bei mehrfachem Anwenden auf dieselben Daten auch wieder dasselbe Resultat.
schnell berechenbar	Eine Hash-Funktion soll sehr schnell berechnet werden können.
kleiner Wertebereich	Objekte beliebiger Größe werden auf wenige Bytes abgebildet. JAVA verwendet eine Hash-Funktion für Strings, die jede Zeichenkette auf lediglich 4 Bytes abbildet.
nicht umkehrbar	Hash Funktionen können nicht rückgängig gemacht werden. Es handelt sich hier um eine Art Einwegfunktion. Jedoch nicht so, dass die Funktion sehr schwierig umzukehren ist, wie dies bei den Verschlüsselungsverfahren der Fall ist, sondern, dass die Funktion überhaupt nicht umzukehren ist. Aus dem Hash-Wert können die ursprünglichen Daten nicht wieder rekonstruiert werden. Das ist übrigens eine einfache Konsequenz aus obiger Tatsache des *kleinen Wertebereiches*. Mehrere Objekte können denselben Hash-Wert erhalten.
gute Streuung	Hash-Funktionen sollen im Wertebereich stark streuen. Das heißt: Zwei unterschiedliche Objekte sollten mit großer Wahrscheinlichkeit zwei verschiedene Hash-Werte liefern.

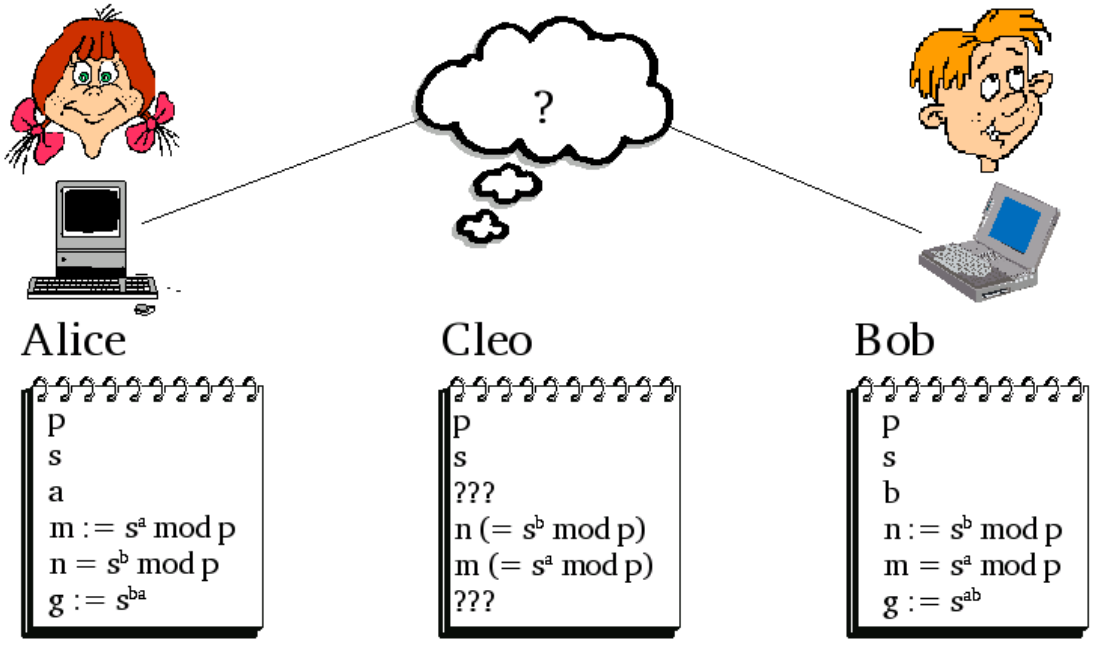
Einsatzgebiete:

- Digitale Signatur (S. Kap. 2.2.6 auf Seite 12)
- Digitaler Fingerabdruck
- Passwortlisten (S. Kap. 1.1.2 auf Seite 4)
- Hash-Tabellen (nicht Inhalt dieses Kurses.)

Beispiele:

- Der JAVA Hash-Code von “Geheimnachricht” ist 0x4A70B2C1.
- Der JAVA Hash-Code von “Geheim Nachricht” ist 0xAB60729F.

2.4 Das Verfahren von Diffie/Hellman



Das Verfahren von Diffie/Hellman (DH-Verfahren[15] 1976) erlaubt es, Daten verschlüsselt zu übermitteln, ohne vorab einen geheimen Schlüssel auf einem separaten Kanal zu transportieren[16].

2.4.1 Ausgangslage

- Ausgangssituation: Alice (A) und Bob (B) wollen Nachrichten über eine öffentlich zugängliche Leitung austauschen. Diese Leitung wird möglicherweise abgehorcht.
- Alice und Bob hatten vorher noch keine Schlüssel miteinander ausgetauscht.
- Das Diffie/Hellman-Verfahren erlaubt es, eine Botschaft über eine Leitung zu übermitteln, ohne dass ein "Horcher" die Nachricht verstehen kann.
  Der Horcher sei im Folgenden Cleo (C).
- Es wird davon ausgegangen, dass der "Horcher" sich nicht als Sender oder Empfänger ausgeben kann. (Die Transaktion sei zu schnell, als dass die *Man-in-the-Middle*-Attacke funktionieren könnte.)

[15] Whitfield Diffie, Martin Hellman und Ralph Merkle
[16] Das Verfahren wurde schon vor Diffie und Hellman von Malcolm Williamson vorgeschlagen.

2.4.2 Vorgehen im Diffie/Hellman-Verfahren

Das Diffie/Hellman-Verfahren besteht aus folgenden Schritten:

1. Die Kommunikationspartner A und B (Alice und Bob) entscheiden sich gemeinsam für eine große Primzahl p. (Je größer die Primzahl, umso sicherer das Verfahren.)

2. A und B suchen eine Zahl s, die kleiner als p ist (Bedingung: $1 < s < p$).[17] Wichtig: Die beiden Zahlen p und s können unverschlüsselt über die Telefonleitung übermittelt werden; sie sind also öffentlich zugänglich. Cleo (C) hört p und s, kann aber mit diesen Zahlen noch nichts anfangen.

3. Alice sucht sich im Geheimen eine Zahl a.
   Bob sucht sich im Geheimen eine Zahl b.
   Die beiden Zahlen sollten kleiner als $p - 1$ sein.

4. A berechnet $m := s^a \bmod p$ und schickt das Resultat an B.
   B berechnet $n := s^b \bmod p$ und schickt das Resultat an A.
   Cleo hört zwar m und n mit und kann damit nun theoretisch a und b berechnen. Er braucht dazu aber viel zu lange.

5. A berechnet die Geheimzahl $g = n^a \bmod p$
   B berechnet die Geheimzahl $g = m^b \bmod p$
   Bem. $g = n^a = m^b$, denn $n^a = (s^b)^a = s^{ba} = s^{ab} = (s^a)^b = m^b$
   C kann g nicht in nützlicher Frist berechnen.
   Das einzige mathematische Verfahren für C, um g zu "berechnen", heißt *ausprobieren*[18].

6. A und B können nun ihre Nachrichten mit g verschlüsseln und entschlüsseln (z. B. mit dem XOR-Verfahren (S. Kap. 2.2.4 auf Seite 11)).

2.4.3 Ein Zahlenbeispiel

1. Alice und Bob bestimmen gemeinsam die folgenden Zahlen und tauschen diese übers Internet aus:

$$\begin{aligned} p &= 467 \\ s &= 44 \end{aligned}$$

2. Alice wählt sich eine beliebige Geheimzahl a, Bob wählt sich die Geheimzahl b. Diese Zahl darf nicht übers Netz verraten werden! Auch nicht gegenseitig. Alice wird b nicht erfahren; ebenso wird Bob die Zahl a nie erfahren:

[17] Genaugenommen sollte s eine sogenannte Primitivwurzel modulo p sein (Siehe [Buchmann 2001]). Das Verfahren funktioniert auch für andere Zahlen, kann aber unter Umständen einfach *geknackt* werden.

[18] Es gibt einige mathematische Tricks (das Babystep-Giantstep-Verfahren von Shanks, der Pollard-ρ-Algorithmus oder das Verfahren von Pohlig-Hellman), um schneller zum Ziel zu kommen, und es gibt Fälle von *einfachen* Primzahlen, bei denen das Verfahren schneller *geknackt* werden kann als stures ausprobieren; vgl. [Buchmann 2001].

$$\begin{aligned} a &= 101 \\ b &= 133 \end{aligned}$$

3. Alice berechnet $m = s^a \bmod p = 44^{101} \bmod 467 = 292$. Alice schickt $m = 292$ an Bob.

   Bob berechnet $n = s^b \bmod p = 44^{133} \bmod 467 = 115$. Bob schickt $n = 115$ an Alice.

4. Alice berechnet $g = n^a \bmod p = 115^{101} \bmod 467 = 150$.

   Bob berechnet $g = m^b \bmod p = 292^{133} \bmod 467 = 150$.

   Oh Wunder, es gibt dieselbe Geheimzahl. Nur Alice und Bob kennen sie!

5. Jetzt können Alice und Bob die Zahl 150 als Basis für ein einfaches Verschlüsselungsverfahren benutzen (z. B. das XOR-Verfahren).

2.5 Das Verfahren von Rivest, Shamir und Adleman (RSA)

- Das RSA-Verfahren wurde von Ronald R. Rivest, Adi Shamir und Leonard M. Adleman entwickelt.
- Das RSA-Verfahren ist ein "Public Key Verfahren". Das heißt, der Algorithmus und der Schlüssel zum Verschlüsseln (codieren, chiffrieren) von Botschaften wird öffentlich bekannt gegeben. Nur der Schlüssel zum Entschlüsseln (decodieren, in Klartext zurückverwandeln) der Botschaften wird geheim gehalten.
- Das RSA-Verfahren basiert auf der Tatsache, dass es sehr einfach ist, zwei Primzahlen miteinander zu multiplizieren, dass es aber äußerst aufwändig ist, die beiden Primzahlen wieder zu finden, falls nur noch das Produkt bekannt ist.

2.5.1 Vorgehen im RSA-Verfahren

Das RSA-Verfahren besteht aus vier Schritten.

1. Der Empfänger generiert ein Schlüsselpaar: einen privaten (geheimen) und einen dazu passenden öffentlichen Schlüssel,
2. der Empfänger publiziert den öffentlichen Teil,
3. der Sender verschlüsselt mit dem öffentlichen Schlüssel des Empfängers seine Botschaft und
4. der Empfänger entschlüsselt die Botschaft mit seinem geheimen Schlüssel.

Der Trick dabei ist, dass nur der Empfänger die Botschaft in sinnvoller Zeit entschlüsseln kann, denn nur er kennt die Primfaktorzerlegung des Schlüssels, wie wir gleich sehen werden. Nun aber die 4 Schritte im Detail:

Schritt 1: Schlüssel generieren In diesem Schritt generiert der Empfänger einen öffentlichen Schlüssel. Alle können danach mit diesem Schlüssel Botschaften chiffrieren (verschlüsseln), aber nur der Empfänger kann sie wieder dechiffrieren (entschlüsseln).

a) Der Empfänger sucht zwei (möglichst große) Primzahlen p und q.

b) Der Empfänger berechnet $r = p \cdot q$.

c) Der Empfänger berechnet zudem $s = (p-1) \cdot (q-1)$.

d) Der Empfänger bestimmt ein beliebiges c mit den beiden folgenden Eigenschaften: $c < s$ und $\mathrm{ggT}(c, s) = 1$. Das erreicht der Empfänger zum Beispiel einfach, indem er eine Primzahl sucht, die kleiner als s ist. Hier kann das JAVA Programm `GCD` eingesetzt werden:

```
>java GCD c s
```

muss 1 ergeben!

Schritt 2: Veröffentlichen des Schlüssels Der Empfänger gibt r und c als öffentlichen Schlüssel bekannt. Zum Beispiel steht auf seiner Homepage vereinfacht: *Der öffentliche Schlüssel von phi@gressly.ch ist (r = 289073 und c = 353).*

Schritt 3: Verschlüsseln einer Botschaft

a) Der Sender zerhackt seine Botschaft in kleinere Stücke, die danach einzeln verschlüsselt werden. Sind die Stücke sehr klein, z. B. 8 Bit, so kommt das Verfahren fast einer monoalphabetischen Substitution gleich. Die Stücke dürfen jedoch nicht mehr Bits in Anspruch nehmen als die Zahl r. Ist z. B. $r = 134565$, so darf ein Stück nicht mehr als 17 Bit in Anspruch nehmen ($2^{17} = 131072$). Der Einfachheit halber würde man in einem solchen Fall wohl 2 Byte belegen (2 Byte = 16 Bit < 17 Bit).

In mathematischen Worten würde man wohl sagen: Die Originalbotschaft wird in Stücke B_i der Bitlänge des (ganzzahligen) Zweierlogarithmus von r unterteilt.

b) Diese "Kurzinformationen" B_i muss der Sender nun mit r und c verschlüsseln:

$$\begin{aligned} G_1 &= B_1^c \bmod r \\ G_2 &= B_2^c \bmod r \\ &\ldots \end{aligned}$$

Verwenden Sie das JAVA Hilfsprogramm

`>java AhBmC Bi c r`.

c) Diese "Chiffrate" (G_1, G_2, ...) sendet er/sie an den Empfänger. Das kann problemlos über die öffentliche Telefonleitung geschehen, denn nur der Empfänger kann diese entschlüsseln.

Schritt 4: Entschlüsseln der Botschaften

a) Der Empfänger berechnet $d := c^{-1} \bmod s$.

Diesen Dechiffrierexponenten d kann nur er berechnen, denn nur er kennt s!

Bem.: Alle können zwar rein theoretisch r in die Primfaktoren p und q zerlegen und so auch s berechnen; aber bei großen Primzahlen p und q ist dieser Aufwand immens.

Verwenden Sie, um d zu berechnen, das JAVA Programm

`>java Minv c s`.

b) Alle Chiffrate (G_1, G_2, ...) kann der Empfänger mit

$$\begin{aligned} B_1 &= G_1^d \bmod r \\ B_2 &= G_2^d \bmod r \\ &\ldots \end{aligned}$$

nun entschlüsseln. Verwenden Sie wiederum das JAVA Programm

`>java AhBmC Gi d r`.

2.5.2 Ein Zahlenbeispiel

Schritt 1: Empfänger

$$\begin{aligned} p &= 467 \\ q &= 619 \\ r &= 467 \cdot 619 = 289073 \\ s &= 466 \cdot 618 = 287988 \\ c &= 353 \end{aligned}$$

Schritt 2: Veröffentlichen Öffentlich bekannt geben: $r = 289073$, $c = 353$

Schritt 3: Sender Die Botschaft sei "Hallo". Die Botschaft wird in Stücke der Länge 2 Byte (= 16 Bit) unterteilt. (Die "x" steht für *hexadezimale Codierung.*)

$$\begin{array}{lrlrlr} B_1 & = & \text{“Ha”} & = & x4861 & = & 18592 \\ B_2 & = & \text{“ll”} & = & x6C6C & = & 27756 \\ B_3 & = & \text{“o”} & = & x6F & = & 111 \end{array}$$

$$\begin{array}{lrlrlr} G_1 & = & B_1^c \bmod r & = & 18529^{353} \bmod 289073 & = & 200883 \\ G_2 & = & B_2^c \bmod r & = & 27756^{353} \bmod 289073 & = & 212601 \\ G_3 & = & B_3^c \bmod r & = & 111^{353} \bmod 289073 & = & 12789 \end{array}$$

Übermittelt wird nun die chiffrierte Botschaft :

$(G_1, G_2, G_3) = (200883, 212601, 12789)$

Schritt 4: Empfänger Der Empfänger berechnet zunächst d, den sog. Dechiffrierexponenten.

$$d = c^{-1} \bmod s = 206405$$

$$\begin{array}{rcrcrcrcr} B_1 & = & 200883^d \bmod r & = & 18529 & = & x4861 & = & \text{“Ha”} \\ B_2 & = & 212601^d \bmod r & = & 27756 & = & x6C6C & = & \text{“ll”} \\ B_3 & = & 12789^d \bmod r & = & 111 & = & x6F & = & \text{“o”} \end{array}$$

2.6 Der eigene öffentliche Schlüssel

Wichtig bei öffentlich benutzten Verfahren ist es natürlich, dass die Zahlen, Schlüssel, Hash-Codes etc. in einem standardisierten Protokoll übermittelt werden.

Mit PGP (pretty good privacy: `www.pgpi.org`) von Phil Zimmermann existiert eine standardisierte Implementierung des Public Key Verfahrens. Unter Linux wird auch GnuPG (`www.gnupg.org`) als Open Source Version angeboten.

PGP arbeitet mit einem Vertrauens-Netzwerk. Jedem Schlüssel in meinem Schlüsselbund kann eine Vertrauensstärke angegeben werden. So gibt es Schlüssel, deren Herkunft ich eher traue als anderen. Wenn ich nun einen neuen Schlüssel in meinen Schlüsselbund aufnehme, so schaue ich nach, ob jemand, dem ich traue, diesem Schlüssel bereits vertraut: In solchen Fällen kann ich dem neuen Schlüssel auch eher trauen.

Vgl. Übung zu PGP (S. Kap. A.2 auf Seite 34).

2.7 Java Hilfsprogramme

Holen Sie sich die mathematischen Hilfsprogramme vom Internet und *entpacken* Sie diese auf Ihren lokalen Rechner:

`http://www.santis.ch/training/java/sicherheit/ (math.zip und XorKryptRandom.zip)`

2.7.1 `XorKryptRandom`

Das Programm `XorKryptRandom` hat den Zweck, eine mit dem XOR-Verfahren geheim gehaltene Botschaft zu entschlüsseln oder auch zu verschlüsseln. Da es sich um ein symmetrisches Verfahren handelt, kann für beide Schritte (verschlüsseln, entschlüsseln) dasselbe Programm eingesetzt werden.

Dem Programm wird beim Starten ein Startwert (*Seed*) für den Zufallszahlengenerator mitgegeben. Mit diesem *Seed* wird eine zufällige Bytefolge generiert. Diese Folge wird *bitweise* mit XOR mit dem Originaltext verknüpft. Das Resultat ist die verschlüsselte Botschaft.

Beispiel 6 (XorKryptRandom) *Sei* ***`original.txt`*** *die zu verschlüsselnde Botschaft. Der Startwert für den Zufallszahlenalgorithmus wird zufällig gewählt, muss aber beiden Parteien (Sender, Empfänger) bekannt sein; z. B. 56. Gehen Sie wie folgt vor, um die Botschaft* ***`original.txt`*** *in eine Datei* ***`krypt.cpt`*** *zu verschlüsseln:*

```
>java XorKryptRandom 56 original.txt  >krypt.cpt
```

entschlüsseln:

```
>java XorKryptRandom 56 krypt.cpt  >original.txt
```

2.7.2 Größter gemeinsamer Teiler: `GCD`

Mit dem Programm `GCD` (greatest common divisor = größter gemeinsamer Teiler) wird der *ggt()* von zwei Zahlen berechnet.

Beispiel 7 (ggt() von 38 und 57) `>java GCD 38 57`

2.7.3 Das Inverse modulo einer Primzahl: `MInv`

Mit dem Programm `MInv` (Modulo-Inverses) wird das Inverse modulo einer Primzahl p berechnet (S. Kap. 2.3.4 auf Seite 16).

Beispiel 8 (Inverses von $5(\text{mod}13)$**)** `>java MInv 5 13`
Ergebnis: ***8****.*

2.7.4 Potenzieren modulo einer Primzahl: `AhBmC`

Mit dem Programm `AhBmC` *"a hoch b* mod *c"* wird eine Zahl a b Mal mit sich selbst multipliziert. Danach wird der Divisionsrest mod c berechnet.

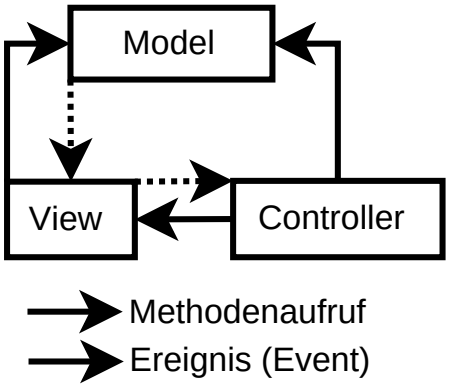
Beispiel 9 ($a^b \bmod c : 43^{11} \bmod 13$**)** `>java AhBmC 43 11 13`
Ergebnis: ***10****.*

3 Konzepte von Web-Applikationen

3.1 Techniken

3.1.1 Mehrschicht-Architektur

Eine Mehrschicht-Architektur (Multi-Tier) bezeichnet die Trennung der Applikation mehrere Schichten, wie z. B. Präsentationsschicht, Applikationsebene (Logik) und Datenbankebene. In *Stand-Alone*-Applikationen wird das Konzept häufig unter MVC (Model-View/Controller) angetroffen. In Web-Applikationen wird das Modell (die Daten) oft in einer Datenbank gespeichert. Die Präsentation wird dann oft mit der Logik in einer Web-Skripting-Sprache (ASP, PHP, JSP) verschmolzen. Perfekt wäre auch hier eine Trennung von Anzeige (View) und Applikationslogik (Controller). Meist ist das Trennen dieser beiden Schichten mit PHP oder ASP aber aufwändig. Der große Nutzen dieser Trennung zeigt sich dann mit dem **Anpassen** solcher Applikationen.



Model

- Kapselung des Zustandes der Applikation
- Reagiert auf Anfragen zur Zustandsänderung
- Stellt die Applikationsfunktionalität heraus
- Meldet dem “View” die Änderungen

View

- Zeichnet das Modell
- Behandelt Aktualisierungen vom Modell
- Sendet Benutzeranfragen an den Controller
- Erlaubt dem Controller die Darstellungsart zu wählen

Controller

- Definiert das Applikationsverhalten
- Bildet die Benutzeranfragen auf das Modell ab
- Wählt die Darstellung (bzw. View) aus

Vorteile der Trennung

- Schnelleres Finden von Fehlern
- Kürzere Einarbeitungszeit für neue Mitarbeiter
- Bessere Erweiterbarkeit / Anpassbarkeit

Nachteile

- Schlechtere Performance
- Höherer Initialaufwand

Templates In der Praxis werden zur Trennung von View und Logik – um das Rad nicht wieder neu zu erfinden – häufig sogenannte Template-Libraries (Bibliotheken von Vorlagen) eingesetzt. Das Modell wird meist mittels einem (relationalen) Datenbanksystem separiert. Hier einige Beispiele von Templates oder Frameworks für z. B. PHP/JSP:

- Mini Templator, Typo3 (PHP)
- ITX
- Smarty
- Struts, Java Server Faces (JSP/J2EE)
- Soap
- Cocoon

3.1.2 Anpassungen

Wichtig ist im Titel des Moduls "E-Business-Applikation anpassen" aber vor allem das Wort **anpassen**. Es ist nötig zu wissen, wie eine bestehende Software verändert wird. Welche Schritte sind nötig, um den Vorgehensprozess einer Änderung optimal zu verwalten (zu "managen"). Wichtig ist also auch der Begriff des "Change-Managements":

Definition 3 (Change Management) *Unter **Change Management** verstehen wir die Schritte, die notwendig sind, die **Anpassungen** in einem Programm zu definieren, die Adaption (Anpassungen und Erweiterungen) vorzunehmen und diese Änderungen letztlich zu prüfen.*

Zwei Arten von Anpassungen sind dabei zu berücksichtigen. Erstens das **Konfigurieren**[19] oder Verändern mittels vorgegebener Parameter und zweitens die **Codeänderungen** beim Anfügen, Entfernen oder Modifizieren bestehender Funktionen.

Ressourcen:

- http://home.att.net/~nickols/change.htm
- http://www.change-management.com/
- http://www.change-management-directory.com/

Bemerkung 5 (Vorgehen) *In der Praxis werden häufig Versionenverwaltungen (wie CVS[20]) eingesetzt. Da sich dieser Kurs nicht mit dem Installieren einer Versionenverwaltung aufhalten will, wählen wir das sogenannte "Poor-mans-CVS": Wir speichern mit jeder Version alle Dateien in einem neuen Verzeichnis ab.*

3.1.3 Usability

Eine gute Web-Applikation sollte ergonomisch gestaltet sein. Die folgenden Punkte geben einen Ansatz, worauf bei der Entwicklung besonders geachtet werden sollte.

Performance Die Antwortzeiten sollten – auch bei langsameren Internetverbindungen – angemessen sein. Vermeiden Sie "schwere" Grafiken.

Verfügbarkeit Wie für *Stand-Alone*-Applikationen gilt auch auf dem Web: Das System darf nicht abstürzen und sollte sich korrekt verhalten, auch wenn viele Benutzer gleichzeitig zugreifen. Wenn eine Stand-Alone Applikation abstürzt, so ist in der Regel nur ein Anwender betroffen. Wenn ein User hingegen via Browser den Server zum Absturz bringt, so sind alle betroffen. Beispiel (dümmster anzunehmender Programmierfehler): "logout"-¿ System.exit(0) zum Ausloggen in einer Web-Applikation verwenden :-(

Plattformunabhängigkeit Die Applikation sollte auf allen (oder zumindest den meisten) gängigen *Webbrowsern* gut lesbar sein.[21] Auch ist von Javascript abzuraten. Javascript ist a) nicht auf allen Browsern vorhanden bzw. eingeschaltet und b) verhält es sich nicht auf allen Systemen gleich. Beachten Sie zudem, dass es mehr und mehr vollblinde Internetbenutzer gibt, die schlecht durch Grafiken navigieren können. Verwenden Sie bei Bildern immer die `alt-` und `title-`*Tags*. Um die Plattformunabhängigkeit zu verifizieren reicht zwar ein automatischer Tester wie unter
`http://www.w3.org`
nicht; es ist aber sicher eine gute Hilfe, diesen HTML-Validator anzuwenden.

[19] Customizing
[20] Concurrent Versions System
[21] Firefox, lynx, Internet Explorer, Netscape, Mozilla, Opera, Konqueror, ...

Lesbarkeit Die Schrift sollte gut lesbar sein: Schriftgröße und Kontrast auf diversen Systemen prüfen. Hintergrundgrafiken können die Lesbarkeit extrem vermindern.

Seitenlänge Auf Internetseiten sollte möglichst wenig *gescrollt*[22] werden. Schreiben Sie nicht: `Siehe weiter unten im Text...` sondern verwenden Sie lieber einen sogenannten *Hyperlink* auf eine neue Seite. Auf der Hauptseite (*home*) sollte **nie** gescrollt werden müssen. Vorteile von kurzen Seiten: Bessere Navigierbarkeit, bessere Strukturierung möglich, schnellere *download*-Zeiten, bessere Wartbarkeit.

Orientierung Der Benutzer sollte jederzeit sehen, wo er sich gerade in der Informationshierarchie befindet.

1. Verwenden Sie immer das `<title>`-*Tag* im Seitenkopf, damit die User die Seite in den Favoriten (bzw. Bookmarks) einfach wieder finden können.
2. Auf jeder Seite eine Navigationshierarchie[23] anzugeben ist auch sinnvoll:

   `MyShop > Zahlungsmodalitäten > Rechnung > Howto`
3. Eine Hierarchie sollte maximal 9 Kapitel zur Auswahl haben, um die Benutzer nicht zu überfordern. Um nicht das Gefühl zu geben, man habe gar keine Wahl, sollten auch nicht weniger als 3-5 Unterkapitel zur Auswahl gegeben werden.
4. Auf jeder Seite sollte es die Möglichkeit geben, wieder auf die Hauptseite zu gelangen (Link auf HOME).

Konsistenz Alle Seiten einer Web-Applikation sollten sich immer gleichartig (eben konsistent) verhalten. Die Grafiken und Anordnungen der Steuerelemente sollten einheitlich sein. Verwenden Sie Fluchtlinien, um das Auge zu beruhigen. Beachten Sie, dass auch allfällige Bannerwerbung zum Layout passt.

Verwenden Sie für Abstände, Rahmen, Farben, Schrift und so weiter wenn immer möglich CSS[24]-Vorlagen. Die Applikation wirkt einheitlicher, ruhiger und professioneller. Zudem können Sie diese viel einfacher **anpassen**.

Didaktik Das System sollte leicht erlernbar sein – wenn möglich ohne Einführung oder Hilfe-Seiten. Alle Informationen und *Links*, die benötigt werden, um den nächsten Schritt (Warenkorb ansehen, Artikel zukaufen, Zahlungsbedingung aushandeln, ...) auszuwählen, sollten permanent ersichtlich sein.

?? Gibt es noch weitere Punkte zu erwähnen?

Natürlich ist für *B2B* (bzw. *B2E*) Applikationen die Usability nicht im Vordergrund. Hier sollte schnell gearbeitet werden können. Oft wird eine gewisse Einarbeitung in Kauf genommen, um danach eine noch höhere Performance zu erreichen.

[22] rollen mittels Rollbalken

[23] z. B. mittels Breadcrumb-trails:
(`http://psychology.wichita.edu/surl/usabilitynews/52/breadcrumb.htm`)

[24] Cascading Style Sheets

3.1.4 Session

Ein generelles Phänomen von Multi-User Applikationen (und somit im speziellen auch von Web-Applikationen) ist die Tatsache, dass jeder Benutzer seine eigenen Variablen besitzen muss.

Sessions (Sitzungen) implementieren auf verbindungslosen Protokollen (hier `http`) eine Beziehung vom Client zu einem zugehörigen Prozess auf dem Server. Es wird dem Client eine ID[25] zugewiesen. Jeder nun folgende Aufruf des Clients übergibt dem Server diese ID. Somit kann der Server seine Prozesse und Daten eindeutig den Clients zuweisen. Eine andere Möglichkeit wäre es, alle Nutzdaten auf dem Client abzulegen.

Mögliche Implementationen über `http`:

Cookies Speichern von Variablenwerten durch den Browser auf der Client-Maschine. Ein "Cookie" enthält den Namen, den Inhalt (Content), die *Domain*, den Pfad, ein Verfallsdatum, eine *Policy* und den Sicherheitszustand (*secure*).

URL-Rewriting Beim URL[26]-Rewriting wird jedem Hyperlink eine Identifikationsnummer mitgegeben
(z. B. `<a href="http://www.xyz.com?SessionID=Sz45Km232Kxw22">`).

Hidden-Fields Variablen können auch in versteckten Feldern der Formulare (`<form>`) mitgegeben werden.

SSL-Sitzung Bei Verbindungen über dem Secure Socket-Layer besteht per se eine Sitzung.

Überlegen Sie sich die Sicherheitsrisiken oder allfällige Datenschutzproblematiken der drei genannten Verfahren.

In `PHP` wird eine Session einfach mit `session_start();` generiert. Die Applikationsentwickler müssen sich nicht mehr um das darunterliegende Verfahren kümmern.

3.2 Architektur

3.2.1 Personalisierung

Individuelle Ansprache von bekannten Kunden: *Guten Tag Herr Gressly, gerade eben frisch eingetroffen das neueste PHP-MySQL Buch!* Auch könnte es möglich sein, dass ein Benutzer sein *Layout* (Aussehen) und das Verhalten der Applikation bis zu einem gewissen Grad selbst bestimmen kann.

3.2.2 Bannerwerbung

Bannerwerbungen sind für die Kunden lästig. Es ist jedoch eine einfache Möglichkeit, Werbefläche für bares Geld zu verkaufen. Die Werbefläche kann auch *per Click* oder *per Show* verkauft werden.

[25] Identifikationsnummer oder -string
[26] Uniform Resource Locator

3.2.3 Web-Shop Konzepte

Im Folgenden soll speziell auf Konzepte des Web-Shops eingegangen werden. Selbstverständlich finden wir einige dieser Konzepte in ähnlicher Form in anderen Applikationen wieder.

Preisfindung Preise werden im Warenkorb anhand der Artikel zusammengestellt. Jedoch können hier je nach Kunde und aktuellen Rabatten andere Resultate entstehen.

Beispiel 10 *Kunden, die im letzten Monat für mehr als CHF 200.- eingekauft haben, erhalten einen Sonderrabatt.*

Kunden, die mehr als 5 Artikel im Warenkorb haben, erhalten 10% Rabatt.

...

Auftragsbestätigung Bevor eine Bestellung wirklich ausgeführt wird, hat der Kunde die Möglichkeit, alle Artikel, Preise, Rechnungs- und Lieferanschriften, Rabatte und so weiter anzuschauen. Der Kunde sieht alle Produkte, die Lieferadresse und die Zahlungsart noch einmal, bevor er zuallerletzt auf den Schalter `Auftrag versenden` klickt.

Cross- und Up-Selling Mit diesen beiden Begriffen (Cross-Selling und Up-Selling) sind Angaben gemeint, die den Kunden darauf hinweisen, dass es noch verwandte Artikel (höherwertige, ergänzende) zum aktuellen Produkt gibt. Es können auch gleich Rabattpakete mit mehreren Artikeln angeboten werden.

Data-Mining In großen Warenhäusern werden Statistiken erzeugt, die Zusammenhänge im Kaufverhalten aufdecken, um besseres Cross- bzw. Upselling zu betreiben. Ebenso kann damit personalisiert und Aktionen können sinnvoll geplant werden.

Warenkorb Der Warenkorb ist eine zentrale Funktion des Webshops. Wichtig ist zu wissen, wie die Inhalte im Warenkorb gespeichert sind. Die Variante, die Inhalte *clientseitig* zu speichern, birgt Risiken. Auf der Serverseite gibt es zwei Varianten: a) persistent: Der Inhalt des Warenkorbes wird in einer Datenbank gespeichert und b) transient: Der Warenkorb lebt in der "Session" als temporäre Variable. Hier muss man sich überlegen, ob die Waren auch noch nach längerer Zeit im Korb liegen sollten. Oder macht es Sinn, die Session-Variablen nach einer halben Stunde – mit allen Waren im Korb – zu löschen. Die Anwenderin muss sich dann wieder neu anmelden.

Passantenfunktion Die **Passantenfunktionen**, evtl. auch als "uncommited Shopping" oder "Window-Shopping" bekannt, bezeichnet eine spezielle Einkaufsvariante, bei welcher der Käufer sich erst umsehen kann. Die Daten über die Zahlungsart, Lieferadresse und *Login* braucht der Gast erst anzugeben, wenn er am Schluss wirklich etwas kaufen will.

Produkte Auswahl Um Produkte eines Web-Shops zu finden, gibt es diverse Strategien. Wichtig ist, dass der Kunde rasch auf das gesuchte Produkt stößt. Das kann bei kleinen Anbietern eine einfache Tabelle mit allen im Lager befindlichen Artikeln sein. Eine **Suche** nach Artikeln kann diverse Stichworte berücksichtigen oder aber nach allen im Text vorkommenden Wörtern suchen (Index). Häufig werden auch Produktehierarchien angeboten. So kann sich ein Käufer wie im Supermarkt von Stockwerk zu Stockwerk und anschließend von Regal zu Regal bewegen, bis er beim gewünschten Produkt ankommt.

Beispiel 11 *Ein Anwender stöbert nach neuen Interpretationen alter Werke im CD-Audio Bereich. Die Hierarchie könnte wie folgt aussehen:*

```
Audio -> CD -> Klassik -> Neuheiten
```

Hier noch eine zugehörige mögliche Tabelle (S. 31), Hierarchien in relationalen Datenbanken zu speichern[27]*:*

Kat. ID	*Parent*	*Name*	*Beschreibung*
1	*null*	*Food*	*Esswaren*
2	*1*	*Frischwaren*	*Gemüse, Obst, Salate*
3	*1*	*Dosen*	*Eingemachtes*
4	*null*	*non-Food*	*Haushalt, Elektronik, Audio-Video*
5	*4*	*Haushalt*	*Haushaltsgeräte und Einrichtungsgegenstände*
6	*4*	*Elektronik*	*Fernseher, Audio-Abspielgeräte*
7	*4*	*Audio/Video*	*Musik und Filme*
8	*7*	*Audio*	*Schallplatten, Kassetten, CD*
9	*8*	*Schallplatten*	*Schallplatten (nur noch kleines Lager!)*
10	*8*	*Kassetten*	*bespielte Audio Kassetten (Musik, Hörspiel)*
11	*8*	*CD*	*Compact Disk Audio*
12	*11*	*Pop*	*Pop CDs (nur Neuheiten: 20. & 21. Jh)*
13	*11*	*Klassik*	*Klassik (Alte Einspielungen und Neuheiten)*
14	*13*	*Alte Klassik*	*Alte Klassik*
15	*13*	*Klassik Neuheiten*	*Neue Interpretationen alter Großmeister*
16	*6*	*Fernseher*	*Fernsehgeräte und Beamer*
17	*16*	*Beamer*	*Home Cinema und Beamer*

[27] In hierarchischen Datenbanken erhält man Hierarchien geschenkt!

A Aufgaben

A.1 Sicherheit

A.1.1 Schutz gegen Angriffe

Geben Sie zu vier der folgenden möglichen Angriffe aus dem Kapitel Sicherheit (S. Kap. 1.1 auf Seite 4) eine mögliche Abhilfe und notieren Sie sich diese in Ihr Arbeitsheft.

Spionage mittels GET-Parameter	.....
Passwort Cracker und Guesser	.....
Horcher, Man-in-the-Middle	.....
E-Shop Lifting	.....
Viren, Würmer, Enten	.....
Trojanische Pferde	.....
DoS Attacken	.....

A.1.2 Angriff

Beschreiben Sie, wie Sie als *Hacker* vorgehen würden, um die E-Banking-Sicherheitslöcher (S. Kap. 1.2.4 auf Seite 8) auszunutzen. Was ist zu tun, um in das System einzudringen, dieses auszuhorchen oder zu manipulieren, wenn der Benutzer den unter E-Banking verzeichneten Punkt nicht beachtet.

a) Passwort auf PC gespeichert	....
b) Passwort nicht geändert	....
c) Alte (unsichere) Browserversion	....
d) Cache nicht geleert	....
e) Kein Virenschutz	....
f) “Fake” E-Mails beantwortet	....
g) User gibt telefonisch Passwörter durch	....
h) User achtet nicht auf sichere Verbindung	....

A.2 PGP (als E-Learning Sequenz möglich)

Voraussetzungen

- Sie haben einen Internetzugang und einen gängigen Web-Browser.
- Sie haben die Möglichkeit, auf Ihrem Computer Software (JSDK, PGP) zu installieren (falls die beiden Pakete nicht bereits vorinstalliert wurden).
- Sie sind in der Lage, Text zu editieren.
- Sie sind in der Lage, E-Mails zu versenden.

Vorgehen

1. Entpacken Sie die folgenden Dateien auf Ihrem PC:

   `http://www.santis.ch/training/java/sicherheit/ (math.zip und XorKryptRandom.zip)`

2. Schreiben Sie die Lösung zu den drei folgenden Aufgaben in ein elektronisches Dokument:

   - Denken Sie sich in das XOR-Verfahren mit einem Random-Seed ein und versuchen Sie, den bereitgestellten Text aus **XorKryptRandom (WelcheLebensform.cpt)** zu "knacken". Schauen Sie dazu im Kapitel XOR (S. Kap. 2.2.4 auf Seite 11), im Kapitel `XorKryptRandom` (S. Kap. 2.7.1 auf Seite 23) und in der heruntergeladenen `README.TXT`-Datei nach. Der Zufallszahlengenerator von Java wurde verwendet. (PS: Der gewählte Seed liegt zwischen 0 und 100)
     - Wie lautet die gesuchte Lebensform?
     - Beschreiben Sie kurz Ihr Vorgehen.
   - Beschreiben Sie in wenigen Sätzen (100-200 Wörter), wie ein längerer Text, der mittels *monoalphabetischer Substitution* (S. Kap. 2.2.3 auf Seite 10) verschlüsselt ist, geknackt werden kann. Geben Sie alle allfälligen Quellen (auch Internet) an.
   - Beschreiben Sie (100-200 Wörter), wie mit dem Verfahren öffentlicher Schlüssel (Public-Key) ein Text verschlüsselt wird und wieder gelesen werden kann. Gehen Sie nicht auf Primzahlverfahren oder spezielle Schlüssel (ElGamal, RSA, Diffie/Hellman , DES, ...) ein, sondern erklären Sie, wer zu welchem Zeitpunkt mit welchem Schlüssel (oder Teil davon) was tun muss. Beschreiben Sie auch, wie eine digitale Signatur (Unterschrift) mit Public-Key-Verfahren funktioniert: Wessen Schlüssel wird wann eingesetzt?

3. Generieren Sie sich mittels PGP oder GnuPG ein *Public/Private-Key-Paar* zu **Ihrer** E-Mailadresse.

   Siehe dazu `http://biwidus.ch/pgp` oder im Anhang (GnuPG S. 41).

   Schicken Sie Ihren Public-Key an `phi@gressly.ch`.[28]

[28] In der Praxis können Sie Ihren Schlüssel auch auf einem sogenannten Key-Server publizieren.

4. Holen Sie den Public-Key von `phi@gressly.ch` vom Internet:

   `http://www.santis.ch/training/java/sicherheit/`

   `(Public Key (Philipp Gressly [phi (at) gressly.ch]) )`

   und legen diesen in Ihrem *Key-Ring* ab. Je nach System ist dem Schlüssel noch ein "trust-level" anzugeben.

5. Prüfen Sie den Schlüssel auf Echtheit mit folgendem digitalen Fingerabdruck (*Fingerprint*):

   `3B21 5FC7 E4F9 2C95 7EDE D9D5 7DCA 0BA1 2E01 5C63`

6. Verschlüsseln Sie Ihre Texte und senden Sie diese in einem verschlüsselten und digital signierten E-Mail an `phi@gressly.ch`.

A.3 *Feedback*

Hiermit bitte ich alle Teilnehmenden, sich in einigen kurzen Sätzen zu diesem Modul zu äußern. Dies, damit der Kurs in Zukunft besser gestaltet werden kann.

Schreiben Sie bitte zu den folgenden Fragestellungen je zwei bis drei Stichworte auf:

- Was war gut? Was hat Ihnen gefallen? Was sollte beibehalten werden?
- Was war schlecht? Was kann verbessert werden?
- Was haben Sie in diesem Modul gelernt?
- Welchen Themen wurde zu viel bzw. zu wenig Zeit gegeben? Welche Themen kann man weglassen? Welche Themen sollten auch noch behandelt werden?
- Was fehlt in diesem Büchlein? Wo habe ich zu viel geschrieben?

B PHP und MySQL

B.1 PHP Hilfeleistung im Internet

`http://tut.php-q.net` Tutorial

`http://www.php.net` Umfassende Referenz / Nachschlagewerk

`http://www.phpmyadmin.net` Administrations-Werkzeug für MySQL auf der Basis von PHP.

B.2 Wichtige MySQL-Befehle

MySQL ist eine freie Implementation der SQL[29].

B.2.1 Auf Datenbank verbinden: `mysql`

Mit dem Kommando `mysql` kann eine interaktive Verbindung zu einer laufenden Datenbank vollzogen werden:

```
#  Verbindung auf 172.23.0.45 auf die Datenbank 'shop_db' öffnen:
>  mysql --host=172.23.0.45 --user=phi --password=p3sq4 shop_db

#  Anfragen:
>> SELECT * FROM artikel;
#  Verbindung trennen:
>> quit
```

B.2.2 Backups

Die MySQL Datenbanken können sehr einfach *gebackupt* werden. Der Befehl `mysqldump` schreibt den Inhalt einer Datenbank als SQL (Create- und Insert-Befehle) in eine Textdatei. Das interaktive Programm `mysql` hat dann die Möglichkeit, die Daten wieder einzulesen.

```
#  Daten aus 'shop_db' in eine Datei 'backup.sql' schreiben:
>  mysqldump --host=172.23.0.45 --user=phi --password=p3sq4 shop_db > backup.sql

#  Datenbank 'shop_db' löschen:
>  mysql --host=172.23.0.45 --user=phi --password=p3sq4 shop_db
>> DROP DATABASE shop_db;
>> quit

#  Datenbank aus Backup 'backup.sql' wieder in die Datenbank einlesen.
#  Eventuell ist dabei die Datenbank zuerst zu erstellen (CREATE DATABASE):
>  mysql --host=172.23.0.45 --user=phi --password=p3sq4
>> CREATE DATABASE shop_db;
>> quit
#  Einlesen aus flat-File:
>  mysql --host=172.23.0.45 --user=phi --password=p3sq4 shop_db < backup.sql
```

B.2.3 Rechte vergeben / User definieren: `GRANT`

Das Vergeben von Rechten auf Datenbanken, Tabellen und Spalten geschieht mit dem Kommando `GRANT`. Damit kann auch gleich ein bisher unbekannter Benutzer erzeugt werden.

Beispiele:

```
#  Gib dem Benutzer "shopadmin" die Rechte, auf der Datenbank 'shop'
#    alles (ALL) zu tun:

>  GRANT ALL ON shop.* TO shopadmin@localhost IDENTIFIED BY 'krpt4';

#  Gib dem Benutzer "phi" die Rechte, Zeilen in die Tabelle 'wako' einzufügen:

>  GRANT INSERT ON shop.wako TO phi IDENTIFIED BY 'p3sq4';
```

[29] Standard Query Language

B.2.4 Einfache Anfragen

SQL-Anfragen können nun direkt eingegeben werden (S. Kap. B.3 auf Seite 40):

```
SELECT * FROM Artikel WHERE art_id = 20;
```

B.2.5 `quit`

Verlassen Sie den MySQL-Interpreten mit dem Kommando `quit`.

Notizen:

B.3 SQL-Befehle

B.3.1 SELECT

Einer der wichtigsten Befehle in SQL ist das SELECT-Statement. Mit SELECT können Anfragen "beliebiger" Komplexität an eine Datenbank gestellt werden:

```
SELECT * FROM Artikel;
SELECT name FROM Kunde WHERE kunden_id=23;
```

B.3.2 INSERT

Mit INSERT können Zeilen in eine Tabelle eingefügt werden:

Beispiel:

```
INSERT INTO kunden (name, vorname)
       VALUES ("Meier", "Vroni"), ("Keller", "Johann");
```

B.3.3 DELETE

Mit DELETE können Zeilen aus einer Tabelle wieder gelöscht werden:

Beispiel:

```
DELETE FROM kunden WHERE kunden_id = 4;
```

B.3.4 UPDATE

Mit UPDATE werden Zeilen in einer Tabelle verändert.

Beispiel:

```
UPDATE kunden SET vorname="Kim" WHERE kunden_id=17;
```

Achtung: Wird keine WHERE-Klausel angegeben, so werden alle Datensätze verändert!

B.3.5 CREATE

Mit CREATE werden neue Tabellen angelegt

Beispiel:

```
CREATE TABLE person (id              INT NOT NULL AUTO_INCREMENT,
                     name            VARCHAR(60),
                     vorname         VARCHAR(30),
                     PRIMARY KEY (id))
```

Notizen:

.....

.....

.....

.....

C GnuPG

Wie wird `GnuPG` von Hand eingesetzt? Hier einige wichtige Befehle und Optionen, falls Sie sich entscheiden, nicht die graphische Benutzeroberfläche von PGP zu verwenden.

C.1 Installation

Zuallererst (Voraussetzung) müssen Sie das gpg-Tool (z. B. `gpg.exe`) vom Internet (`www.gnupg.org` [30]) herunterladen, falls dies (wie auf einigen Santis-Rechnern) nicht schon geschehen ist.

1. Erstellen Sie ein Verzeichnis `c:\temp\gpg`[31] und kopieren Sie alles vom Netz dahinein.
2. Wechseln Sie entweder ins oben erstellte Verzeichnis (`cd c:\temp\gpg`) oder geben Sie dieses Ihrem System-Pfad bekannt.

C.2 GPG-Home Verzeichnis

GnuPG benötigt ein Heim-Verzeichnis, um die Schlüssel zu speichern. Wenn nichts angegeben wird, sucht sich GnuPG selbständig ein solches Verzeichnis aus. Wenn Sie auf den Maschinen nicht überall Schreibrechte haben oder wenn Sie das Verzeichnis explizit angeben wollen, verwenden Sie bei jedem Aufruf von `gpg` die Option `--homedir <Verzeichnis>`. Dabei steht `<Verzeichnis>` für das von Ihnen gewählte Verzeichnis.

C.3 Schlüssel generieren

Mit der Option `--gen-key` wird ein Schlüssel generiert. ElGamal-Schlüssel sind in der Regel sicher. Wegen einem Implementationsfehler sollten Sie aber nicht zum Unterschreiben eingesetzt werden. Für unsere kleine Übung ist das jedoch kein Problem, denn der Aufwand, den privaten Schlüssel dennoch zu "knacken", ist immer noch hoch genug.

```
gpg --gen-key
```

bzw.

```
gpg --homedir  c:\temp\gpg --gen-key
```

Achtung: Die gefragte "Passphrase" ist nichts anderes als ein Passwort, wie wir es gelernt hatten.

C.4 Importieren von Schlüsseln

Damit Sie später für jemanden etwas verschlüsseln können, müssen Sie seinen Public-Key importieren; im Volksmund gesagt: Hängen Sie den öffentlichen Schlüssel des Empfängers an Ihren Schlüsselbund (`Key-Ring`). Das geschieht mit der Option `import`.

```
gpg --import phi.key
```

bzw.

[30] Alternativ kann `www.pgpi.org` mit grafischer Benutzerschnittstelle verwendet werden. Siehe auch `www.biwidus.ch/pgp`.

[31] Falls Sie in beliebigen Verzeichnissen Schreibrechte haben, können Sie selbstverständlich das Programm auch in ein anderes Verzeichnis kopieren. Sie können dann auch auf den (im Folgenden immer erwähnten Zusatz "`homedir`") verzichten.

```
gpg --homedir c:\temp\gpg --import phi.key
```

Den erfolgreichen Import testen Sie am einfachsten mit dem Auflisten aller Schlüssel an Ihrem "Schlüsselbund".

```
gpg --list-keys
```

C.5 Schlüssel unterschreiben und beglaubigen

Es gibt noch zwei weitere Befehle, die je nach Anwendung der Schlüssel einzusetzen sind: Schlüssel unterschreiben (`--sign-key`) und Schlüssel beglaubigen (`--edit-key` und `trust`).

C.6 Verschlüsseln / Entschlüsseln einer Botschaft

Das Verschlüsseln für einen Empfänger (z. B. `phi@gressly.ch`) ist nun, nachdem die obigen Schritte durchgeführt sind, keine Hexerei mehr. Der Befehl `--encrypt` verschlüsselt den Text. Hier können mehrere Empfänger angegeben werden. Sobald eine Leerzeile eingegeben wird, wird der Text für alle angegebenen Empfänger verschlüsselt.

```
gpg  --encrypt geheim.txt
```

Zum Entschlüsseln verwenden Sie den Befehl `--decrypt`.

Notizen:

D Bibliographie

Literatur

[Amor 2001] D. Amor: *Die E-Business-(R)Evolution* [Galileo Press GmbH], 2001 (ISBN:3-89842-185-6)

[Buchmann 2001] J. Buchmann: *Einführung in die Kryptographie* [Springer] 2001 (ISBN: 3-540-41283-2)

[CT Nr. 26 2002] Heise Verlag, CT-Magazin, Nr. 26 vom 16. Dez. 2002. "Virtueller Ladendiebstahl" Seite 92

[iX Nr. 10 2003] Heise Verlag, iX Magazin für professionelle Informationstechnik, Nr. 10: Oktober 2003 "Sonderangebote" Seite 62

[Merz 2001] H. Merz: *Praxis-Lexikon e-Business* [verlag moderne industrie] 2003 (ISBN: 3-478-24940-6)

[Modul 150] *Modulidentifikation M150* [i-zh] 2003

[Tages Anzeiger 2003 11. 17.] Tages-Anzeiger, Digital, Montag 17. Nov. 2003 (111. Jahrgang, Nr. 267), Seite 27

[Uhr 2003] W. Uhr: *E-Business* [http://www.tu-dresden.de] 2003

[Wahrig] *WAHRIG* Deutsches Wörterbuch

[Zwerger/Paulus 2002] F. Zwerger, S. Paulus: *E-Business Projekte* [Galileo Business] 2003 (ISBN: 3898421953)

E Schlagwortverzeichnis