Modul V4 – Public Key Kryptografie

Zeitrahmen
100 Minute

Der Modul V4 ist als selbständige Einheit konzipiert, die, sofern man sich auf die Verschlüsselung als solche konzentrieren kann, in 100 Minuten durchführbar ist, wenn man die in Schritt 2 dargestellten mathematischen Präliminarien als gegeben voraussetzen darf.
Wenn dies allerdings nicht vorausgesetzt werden kann, empfiehlt es sich, diese – gegebenenfalls gemeinsam mit den Erklärungen aus Schritt 1 als Motivation – in einer eigenen Doppelstunde zu wiederholen. In dieser Vorlaufstunde kann man auch ein Programm entwickeln lassen, mit dem die erforderlichen Rechnungen rasch ausgeführt werden können, um so Zeit für das Verständnis des RSA -Algorithmus zu gewinnen.

Zielgruppe
Sekundarstufe II

Inhaltliche Voraussetzungen
- Lehreinheit V2 Symmetrische Schlüssel
- Umrechnen Binär-Dezimal C4
- Verständnis der Modulo-Operation (siehe V3)

Lehrziel
Problem der Schlüsselverteilung bei der symmetrischen Verschlüsselung verstehen.
Prinzip und Vorteil der Nutzung asymmetrischer Verschlüsselung und des Prinzips digitaler Signaturen verstehen.

Motivation
Geheimhalten wird in der heutigen elektronischen Kommunikation immer wichtiger. Nicht jedes Verschlüsselungsverfahren ist bei jedem Anwendungsgebiet sicher. Einfache symmetrische Verschlüsselungsverfahren haben den Nachteil, dass sie von der Vertraulichkeit der Schlüsselübertragung abhängen. Diese Begrenzung wollen wir nun überwinden.
Durch die hier besprochene Verwendung öffentlicher Schlüssel wird auch die vertrauliche Kommunikation mit Zentralstellen gegenüber der Verwendung symmetrischer, bilateral vereinbarter Schlüssel deutlich erleichtert.
Weiters will man oft sicher sein, dass eine Nachricht tatsächlich von jenem Sender stammt, der sie angeblich abgesendet hat. Dies wird durch die elektronische Signatur erreicht. Sie fußt auf demselben Grundprinzip wie jenes der öffentlichen Schlüssel.

Requisiten
- Einfache ASCII Tabelle
- Verzeichnis für öffentliche Schlüssel
- Taschenrechner/Computer

Partizipanden
Gesamte Klasse, teilweise in Gruppen unterteilt

Unterlagen
V-AB 4.1, V-AB 4.2a und V-AB 4.2b, V-AB 4.3

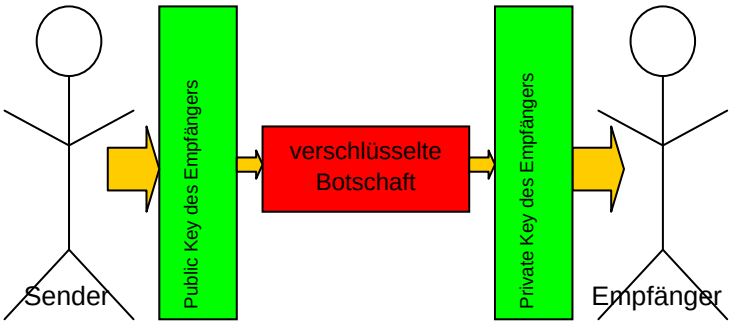
Vorgehensweise

1. Einheit V-3 zeigte, dass es möglich ist, sich auf einen Schlüssel zu einigen, ohne diesen persönlich zu übergeben. Allerdings hat dieses Verfahren noch den Nachteil, dass der Schlüssel vereinbart werden muss und erst dann kann die Nachricht kommuniziert werden. Denkt man an Kommunikationspartner aus unterschiedlichen Zeitzonen, also wenn sich A etwa in den USA befindet und B in Europa, dann ist es zeitlich nicht immer einfach, den Schlüssel zu vereinbaren. Das Verfahren ist also umständlich, auch wenn beim Schlüsselaustausch keine Gefahr mehr besteht.
   Weiters beruhen die bisher besprochenen Verschlüsselungsverfahren stets darauf, dass ein vereinbarter Schlüssel genau zwischen den beiden künftigen Kommunikationspartnern vereinbart wird. Wenn also eine Zentralstelle mit vielen externen Partnern vertraulich kommunizieren muss (z.B. die Krankenkasse mit den niedergelassenen Ärzten) müsste mit jedem Partner ein eigener Schlüssel vereinbart werden. Auch dies stößt offenbar rasch an praktische Grenzen.
   Ausgehend von einem von Diffie und Hellman 1976 postulierten Prinzip haben R.L. Rivest, A. Shamir und L. Adleman ein heute gebräuchliches, nach ihnen benanntes Verfahren (das RSA-Verfahren) publiziert, das erlaubt, Verschlüsselung so durchzuführen, dass der künftige Empfänger geheimer Nachrichten soviel von seiner Schlüsselkonstruktion veröffentlicht, dass Sender Nachrichten so verschlüsseln können, dass sie nur der künftige Empfänger entschlüsseln kann.

   Wenn wir wieder einen Vergleich mit der Kiste und dem Vorhängeschloss anstellen, so kann das Verfahren der öffentlichen Schlüssel wie folgt beschrieben werden: Jeder kann ein Vorhängeschloss schließen, aber nur der Besitzer des Schlüssels kann es öffnen. Das Verschließen (Verschlüsselung) ist also einfach, das Öffnen (Entschlüsselung) kann nur der Besitzer vornehmen. Weiters kann der Empfänger beliebig viele identische Schlösser an Personen oder Postämter verteilen. Sie eignen sich ja nur, die zu versendende Kiste zu verschließen. Den einzigen Schlüssel dazu behält der Verteiler der Schlösser.
   Ernst hat also ein Vorhängeschloss mit Schlüssel, er behält den Schlüssel und verteilt viele Vorhängeschlösser an Postämter. Wenn *Sandra* ihm eine Nachricht schicken möchte, geht sie zum Postamt, holt sich ein „Vorhängeschloss Ernst" und verschließt die Kiste damit. Jetzt kann niemand mehr die Kiste öffnen (auch nicht die Post oder andere Personen, die Kisten mit Ernst-Schlössern versenden möchten). Einzig *Ernst* kann die Kiste(n) öffnen, da er ja den zum Schloss passenden Schlüssel hat.
   Umgemünzt auf die Kryptografie spricht man hierbei von **asymmetrischer Verschlüsselung**. Der Schlüssel zum Chiffrieren entspricht nicht dem Schlüssel zum Dechiffrieren, wie dies bei der symmetrischen Verschlüsselung immer der Fall ist.
   In diesem Zusammenhang sind auch die Begriffe öffentlicher Schlüssel (***public key***) und privater Schlüssel (***private key***) von Bedeutung. Der öffentliche Schlüssel ist jedem bekannt, der eine Nachricht an Ernst schicken möchte, er hat also den Charakter eines Vorhängeschlosses und kann in einem öffentlichen Verzeichnis für jedermann einsichtig sein. Der private Schlüssel hingegen wird von Ernst ebenso wie einige Konstruktionsdetails des gesamten Schlüsselsystems geheim gehalten (er entspricht mithin dem Schlüssel zum Schloss).

   Es mag an dieser Stelle merkwürdig erscheinen, dass man sowohl zu dem von Ernst privat gehaltenen Schlüssel wie auch zu den von ihm verteilten Schlössern in der Fachsprache „Schlüssel" sagt. Der Grund dafür liegt darin, dass – im Unterschied zur Schloss-/Schlüssel-Analogie – das kryptologische Schloss und der kryptologische Schlüssel die selbe Bauweise haben. Wir werden dies in Kürze erkennen.

   Das folgende Tafelbild veranschaulicht diese Zusammenhänge



Wir empfehlen noch vor Beginn der Animation einige mathematische Grundlagen zu wiederholen bzw. zu ergänzen.

2. **Einschub**: Wiederholung einiger mathematischer Konzepte, die zum Verständnis des RSA-Verfahrens nötig sind:
   2.1. ***Primzahlen*** sind all jene Zahlen, die außer durch sich selbst durch keine andere Zahl ohne Rest dividiert werden können. Beispiele für Primzahlen sind 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, .., 97, .., 997, u.s.w.
   2.2. Zwei Zahlen sind ***relativ prim*** zueinander (sie sind ***teilerfremd***), wenn sie keinen gemeinsamen Teiler haben. Z.B. sind 13 und 17 klarerweise relativ prim zueinander, da beides Primzahlen sind. Allerdings ist auch 25 zu 33 relativ prim, obzwar 25 = 5*5 und 33 = 3 * 11, also beide keine Primzahlen sind. Aber diese Zerlegung der beiden Zahlen in ihre Primfaktoren liefert keine gemeinsame Zahl. Dies bedeutet, ihr ***kleinstes gemeinsames Vielfaches*** ist 1.
   2.3. Jede Zahl, die nicht selbst Primzahl ist, lässt sich in ihre ***Primfaktoren*** zerlegen. Das bedeutet, sie ist als Produkt von Primzahlen darstellbar
   2.4. Unter ***Faktorisierung*** versteht man die Zerlegung einer Zahl in das Produkt von Zahlen. Für das RSA-Verfahren ist die Faktorisierung in zwei Faktoren wesentlich. Letztlich können wir aber jede Zahl, die nicht selbst Primzahl ist, als Produkt von Potenzen von Primzahlen darstellen.
   2.5. In Einheit V-3 haben wir bereits die **Modulo-Operation** kennengelernt. *N mod Q = R.* Das bedeutet, wenn wir die Zahl *N* durch den Quotienten *Q* dividieren entsteht der Rest *R*.
   In Einheit V-3 kamen wir mit dieser uns aus der bekannten Division geläufigen Sichtweise aus. Nun müssen wir ein wenig tiefer in die Zahlentheorie eindringen, da *R* eine Fülle interessanter mathematischer Eigenschaften besitzt, auf die sich das zu besprechende RSA-Verfahren stützt. Im Moment ist für uns das wichtigste, dass man in der ***Restklasse*** Z(mod Q) nur Zahlen zwischen 0 und (Q - 1) vorfindet. Allerdings kann man innerhalb dieser Restklasse die uns bekannten mathematischen Operationen wie Addition, Subtraktion, Multiplikation und Division ausführen, bloß kommt man nie über die Grenze (1 – Q) hinaus.
   Wem dies merkwürdig vorkommt, erinnere sich an eine normale Uhr mit Zeigern und Zwölfer-Einteilung:
   9 Uhr + 2 Stunden = 11 Uhr aber 9 Uhr + 4 Stunden = 1 Uhr = 13(mod 12).
   Die normale Uhr (z.B. Turmuhr) zeigt die Stunden also in Stunde(mod 12), die Minuten in Minute(mod 60) an. Die digitale Bahnhofsuhr dagegen zeigt die Stunden über den ganzen Tag durchlaufend an. Auf ihr ist, für einen Zug der irgendwo um 9 h wegfährt und 4 Stunden unterwegs ist, zum Ankuftszeitpunkt 9 Uhr + 4 Stunden = 13 Uhr = 13(mod 24). Für den Nachtzug, der um 9 h Abend, also lt. Fahrplan um 21 Uhr wegfährt, gilt als Ankunftszeit nach 4-stündiger Reise 21 + 4 = 1(mod24).

Wir wollen nun noch die Berechnung des größten gemeinsamen Teilers (ggt) in Erinnerung rufen. Wenn hierbei im Rahmen des Informatikunterrichts bereits der Euklid'sche Algorithmus durchgenommen wurde, kann direkt auf diesen verwiesen werden. Wenn dies nicht der Fall ist, mag es sinnvoll sein, erst auf die im Mathematikunterricht übliche Berechnung des ggt durch Faktorisierung zu wiederholen und dann erst den Euklid'schen Algorithmus als kontrastierende Lösung.

2.6. Abschließend wollen wir noch den ***Euklid'schen Algorithmus*** zur Berechnung des ***größten gemeinsamen Teilers (ggt)*** vorstellen bzw. in Erinnerung rufen.
Allerdings müssen wir sehen, dass im Mathematikunterricht meist die Berechnung des ggt auf der Basis der Faktorisierung gelehrt wird. Dies ist für kleine Zahlen auch tatsächlich sehr effizient. Betrachten wir ein Beispiel der Berechnung des ggt durch **Faktorisierung**:

ggt (216, 168) = 216 , 168 | : 3
72 , 56 | : 2
36 , 28 | : 2
18 , 14 | : 2
9 , 7 also ggt(216, 168) = 3 * 2 * 2 * 2 = 24.

Wir sehen, dass hier systematisch durch Primfaktoren dividiert wird, solange bis zwei Zahlen übrig bleiben, die relativ prim zueinander (teilerfremd) sind. Der ggt ergibt sich sodann aus dem Produkt der Quotienten, durch die beide Zahlen dividiert werden konnten.

Faktorisierung ist allerdings für Computer ein relativ schwieriger Vorgang. Hier eignet sich der vom griechischen Mathematiker Euklid[1] auf der Basis der Kettendivision ganzer Zahlen beruhende Algorithmus besser.
Kettendivision von zwei ganzen Zahlen m und n basiert auf dem Prinzip der Herauslösung des gemeinsamen Anteils und Weiterverwendung des Restes, also dessen, was nach Modulo-Operation als Restklasse verbleibt.

$m = q_1 * n + r_1$ mit $0 < r_1 < n$
$n = q_2 * r_1 + r_2$ mit $0 < r_2 < r_1$
$r_1 = q_3 * r_2 + r_3$ mit $0 < r_3 < r_2$
$\ldots = \ldots$
$r_{k-2} = q_k * r_{k-1} + r_k$ mit $0 < r_k < r_{k-1}$
$r_{k-1} = q_{k+1} * r_k + 0$

Der ggt ist der letzte Rest ungleich 0, also r_k.

Wenn wir diesen Algorithmus an obigem Zahlenbeispiel ausführen, ergibt dies:

216 = 1 * 168 + 48
168 = 3 * 48 + 24 (3 * 48 = 144)
48 = 2 * 24 + 0
also ggt(216, 168) = 24

Wir wollen uns nach diesem Einschub wieder unserem eigentlichen Ziel, der Entwicklung eines asymmetrischen Schlüsselverfahrens mit öffentlichem Schlüssel zuwenden.

3. Das **Prinzip der RSA-Verschlüsselung** beruht so wie das Verfahren der öffentlichen Schlüsselkonstruktion auf Einwegfunktionen mit Falltüre (One-way trap-door function), also einer Funktion, die relativ leicht auszuführen, aber sehr schwer rückgängig zu machen (schwer invertierbar) ist, außer man kennt die „Falltüre", also einen zusätzlichen Wert, der die Rückführung erleichtert.
Wieder gehen wir, wie bei der Diffie-Hellman Verschlüsselung davon aus, dass der Schlüssel aus zwei Teilen besteht. Bezeichnen wir mit E die Verschlüsselung (encryption) und mit D die Entschlüsselung (decryption), so soll für eine numerisch interpretierte Botschaft gelten

D(E(M)) = M

Die Funktionen D und E müssen (für Computer) einfach berechenbar sein. Weiters darf durch die Veröffentlichung von E kein einfacher Weg zur Berechnung von D angedeutet werden. Dies bedeutet, dass

[1]) Euklid lebte im klassischen Griechenland etwa 300 v. Chr. Er war vermutlich ein Schüler Platons.

nur jener, der D kennt, die Entschlüsselung der verschlüsselten Botschaft C = E(M) , also die Decodierung D(C) = M, vornehmen kann.
Gilt weiters, dass man die Anwendung von D und E auch vertauschen kann, also wenn auch

$$\mathbf{E(D(M)) = M}$$

gilt, eignet sich dieses Verfahren auch als Verfahren, elektronische Unterschriften zu leisten.

Als Funktionen für E und D werden wiederum Exponentialfunktionen über Restklassen verwendet. Dies bedeutet, wir müssen Zahlen e und d berechnen für die gilt

$$\mathbf{C = E(M) = M^e \ (mod\ n)} \quad \text{und}$$
$$\mathbf{M = D(C) = C^d \ (mod\ n).}$$

Dass wir den „Text" einer Botschaft numerisch interpretieren können, sollte entweder aufgrund des Wissens über Binärcodierung (Einheit C4) oder aufgrund der Binärvariante von Einheit V2 unmittelbar einleuchtend sein.
Ob dies der Fall ist, sollte anhand von Fragen überprüft werden. Wenn dies nicht gegeben ist, kann anhand der vereinfachten ASCII-Tabelle (Arbeitsblatt V-AB 3.4) mit kurzen Beispieltexten gezeigt werden.

Gruppenarbeit: Die Klasse sollte nun in mindestens drei, idealer Weise jedoch in vier oder fünf Gruppen wie folgt geteilt werden:

- 1 *Empfängergruppe*, Gruppenname *Ernst*: Sie bekommt Arbeitsblatt V-AB 4.1 und erarbeitet den öffentlichen Schlüssel. Mit ihrem gleichzeitig erarbeiteten geheimen privaten Schlüssel wird sie anschließend empfangene Nachrichten entschlüsseln.
- 2 oder mehrere, jedoch mindestens 1 *Sendergruppe* mit Gruppennamen wie *Sandra, Siegfried, Susi, Sepp, …*: Diese Gruppe(n) bekommen das Arbeitsblatt V-AB 4.2(a, b, oder c). Anhand der Anweisungen am Arbeitsblatt verschlüsselt sie die kurze Nachricht, die sie an die Empfängergruppe senden möchte.
- 1 *Beobachtungsgruppe*, Gruppenname *Neugier*: Sie bekommt Arbeitsblatt V-AB 4.3

Alle Gruppen bekommen ihr gruppenspezifisches Arbeitsblatt einschließlich der ASCII-Tabelle.

Um den Jugendlichen Rechenarbeit zu ersparen, enthalten die Arbeitsblätter bereits voraus berechnete Ergebnisse für Teilschritte des Verfahrens.
Die Anweisungen am Arbeitsblatt sagen zwar „Gruppe … wählt". Allerdings machen wir einen Vorschlag für die Wahl. Dadurch verheddern sich die Gruppen nicht in mühsamen Rechnungen. In Schritt 5 sollten jedoch einige dieser Rechnungen exemplarisch auf der Tafel nachvollzogen werden.

Bei Public-Key-Kryptosystemen muss die Initiative vom künftigen Empfänger verschlüsselter Nachrichten ausgehen. Die Gruppenarbeit beginnt daher bei Gruppe *Ernst*:

3.1. Gruppe *Ernst* wählt zwei Primzahlen p und q, die sie geheim hält.
Normalerweise sollten dies sehr große Primzahlen sein. Damit unsere Rechnungen einfach nachvollziehbar sind, begnügen wir uns mit kleinen Zahlen. Um von den bereits ausgeführten Zwischenrechnungen zu profitieren, schlagen wir als Wahl für p = 17 und für q = 11 vor.

3.2. Aus diesen Zahlen bildet sie das Produkt N = p * q. Entscheidet sich die Gruppe für die vorgeschlagenen Werte für p und q ergibt dies N = 17 * 11 = 187.
Während die Werte für p und q, also 17 und 11 geheim bleiben müssen, wird Gruppe *Ernst* das Produkt N später veröffentlichen. Wir erinnern uns, dass Produktbildung eine Einwegfunktion ist. Es ist also nicht oder nur durch extremen Aufwand möglich, aus dem Produkt N auf die Faktoren, aus denen es gebildet wurde, zurückzuschließen, selbst wenn man weiß, dass es sich um Primzahlen handeln sollte.

3.3. Gruppe *Ernst* wählt nun eine weitere geheim zu haltende (in der Praxis relativ große) Zahl d, die zu (p-1) und (q-1) relativ prim sein sollte.
Um auf bestehenden Vorrechnungen aufbauen zu können, schlagen wir die Wahl von d = 7 vor.

3.4. Nun muss Gruppe *Ernst* aus den geheimen Werten von d, p und q mit Hilfe der Formel

$$e * d = 1 \pmod{((p-1)*(q-1))}$$

den Wert von e berechnen.
Dazu wird der Euklid'sche Algorithmus zur Berechnung des ggt verwendet. Wir wollen dies später mit der gesamten Klasse noch im Detail besprechen. Wenn sich die Gruppe an unsere bisherigen Vorschläge zur Wahl von Werten gehalten hat, wird das Ergebnis e = 23 lauten.

3.5. Nun kann Gruppe *Ernst* den von ihr erarbeiteten öffentlichen Schlüssel, das Zahlenpaar N, e, veröffentlichen. Sie erledigt dies, indem sie

Schlüssel der Gruppe Ernst: N = 187, e = 23

auf die Tafel schreiben. Das Tripel (p, q, d) = (17, 11, 7) verbleibt als Geheimnis der Gruppe *Ernst* vertraulich.

3.6. Nun können die Gruppen *Sandra, Siegfried, Susi, Sepp, Sabine, Sebastian*, … und natürlich auch die Gruppe *Neugier* den Schlüssel der Gruppe *Ernst* im öffentlichen Verzeichnis lesen. Damit besteht nun für die Sendergruppen die Möglichkeit, unter Zuhilfenahme von Arbeitsblatt V-AB 4.2 Nachrichten an *Ernst* zu senden.

Wir diskutieren dies hier für die Gruppe Sandra. Für die andere(n) Sendergruppen gilt Analoges. Sie müssen bloß einen anderen Text wählen, den sie verschlüsseln und sodann senden.
Um Rechenarbeit zu sparen, machen wir wieder einfache Vorschläge. Diese reduzieren sich bei dem extrem klein gewählten Wert für N auf einzelne Buchstaben.

3.7. Gruppe *Sandra* wird ersucht, den (geheimen) Text, den sie übertragen möchte, in Einzelbuchstaben zu zerhacken. Dies, damit gesichert ist, dass der Zahlenwert des Blocks einer übertragenen Nachricht kleiner als die Schlüsselkomponente N ist.
Nehmen wir an, Gruppe *Sandra* möchte die Botschaft XERXES übertragen, so ergäbe dies aufgrund der Umrechnung mit Hilfe der ASCII-Tabelle eine 6*7 = 42-stellige Binärzahl. Wir wollen jedoch der Einfachheit halber dezimal rechnen und erhalten daher 88, 69, 82, 88, 69, 83, also sechs Dezimalzahlen, die nun getrennt zu verschlüsseln und nacheinander zu senden sind.

3.8. Für den ersten Buchstaben X ergibt sich der dezimale Wert 88 als Botschaft M. Mit der Formel

$$C = M^e \pmod{N}$$

wird diese Botschaft nun verschlüsselt. $C = 88^{23} \pmod{187} = 88^1 * 88^2 * 88^4 * 88^{16} \pmod{187} =$
$= 88^1 \pmod{187} * 88^2 \pmod{187} * 88^4 \pmod{187} * 88^{16} \pmod{187} =$
$= 88 * 77 * 132 * 154 \pmod{187} = 44 * 132 * 154 \pmod{187} = 894.432 \pmod{187} =$
$= \mathbf{11} \pmod{187}$

3.9. Gruppe *Ernst* erhält nun die von *Sandra* und von anderen Sendergruppen gesendete Nachricht, also z.B. 11 von Gruppe *Sandra*.
Gruppe *Ernst* entschlüsselt diese mit ihrem Decodierungsschlüssel d anhand der Formel

$$M = C^d \pmod{N},$$

also $M = 11^7 \pmod{187} = 11^1 * 11^2 * 11^4 \pmod{187} =$
$= 11^1 \pmod{187} * 11^2 \pmod{187} * 11^4 \pmod{187} =$
$= 11 * 121 * 55 \pmod{187} = 73.205 \pmod{187} =$
$= 88$, und somit nach Rücktransformation mit Hilfe der ASCII-Tabelle der erste Buchstabe der geheimen Botschaft, X.

3.10. Gruppe *Ernst* verfährt mit der von der zweiten Gruppe (z.B. *Siegfried*) erhaltenen Botschaft analog. Wenn diese Gruppe etwa die codierte Botschaft c = 142 gesendet hat, lautet die Berechnung

$M = 142^7 \pmod{187} = 142^1 * 142^2 * 142^4 \pmod{187} =$
$= 142^3 * 142^4 \pmod{187} = 142^3 \pmod{187} * 142^4 \pmod{187} =$
$= 131 * 89 \pmod{187} = 11.659 \pmod{187} =$

= 65, und somit nach Rücktransformation mit Hilfe der ASCII-Tabelle der erste Buchstabe der geheimen Botschaft, A, der Anfangsbuchstabe von „Adam".

Wenn weitere Gruppen verschlüsselte Botschaften erarbeitet haben, sollte wohl an dieser Stelle nicht blind weiter entschlüsselt werden. Vielmehr empfiehlt es sich, unter Rückgriff auf das Tabellenkalkulationsprogramm im Anhang im Rahmen der Nachbesprechung noch einige Ver- und Entschlüsselungen durch die Klasse selbst berechnen zu lassen.
Man kann dazu ja nun Paare bilden, von denen jeweils eine Person verschlüsselt, die andere entschlüsselt. Aus Zeitgründen wird man dabei allerdings bei den Vorgeschlagenen Parametern (N, e, d) des Kryptosystems bleiben.

3.11. Nun kommt die Gruppe *Neugier* zu Wort. Sie kennt das Verfahren,

$$C = E(M) = M^e \pmod n \quad \text{und} \quad M = D(C) = C^d \pmod n$$

und den öffentlichen Schlüssel von Ernst, (N, e) = (187, 23). Weiters kennt sie die verschlüsselte Mitteilung, die Gruppe *Sandra* sandte, $C_{Sandra} = 11$ (und ggf. den Code der Gruppe Siegfried, $C_{Siegfried} = 142$).

Gruppe *Neugier* kann also die Gleichung $11 = M^{23} \pmod{187}$ mit der Unbekannten M aufstellen. Da diese Funktion jedoch eine Einwegfunktion ist und *Neugier* die Falltüre d nicht kennt und diese aus Unkenntnis von q und p auch nicht effektiv berechnen kann, ist dieses Verfahren relativ sicher. Auch die weitere Information $142 = M^{23} \pmod{187}$ reicht nicht aus, den Code zu brechen.

Nachbesprechung

Damit ein gemeinsames Verständnis über das Verfahren entsteht legen die beteiligten Gruppen nun die von ihnen gewählten Werte und die durchgeführten Algorithmen vor der Klasse dar.

Die Rolle der Übungsleitung beschränkt sich im Idealfall darauf, zu achten, dass die Erklärungen korrekt und für den Rest der Klasse nachvollziehbar sind. Gegebenenfalls wird jedoch einzuspringen und mit einigen Zusatzerklärungen, unter Rückgriff auf die Mathematikkenntnisse der Klasse auszuhelfen sein.
Die Berechnung des öffentlichen Schlüssels e ist jedoch in jedem Fall noch zu erläutern.

3.12. Gruppe *Ernst* erläutert, dass sie den für die Modulo-Berechnung nötigen Wert N als Produkt aus zwei geheimen Primzahlen, N = p * q gebildet hat.
Hier sollte nochmals darauf hingewiesen werden, dass Produktbildung eine Einwegfunktion ist. Es ist also nicht oder nur durch extremen Aufwand möglich, aus dem Produkt N auf die Faktoren, aus denen es gebildet wurde, zurückzuschließen, selbst wenn man weiß, dass es sich um Primzahlen handeln sollte.

3.13. Weiters wählte Gruppe *Ernst* einen zu (p-1) und (q-1) relativ prime Zahl d. Die Zahlen p, q, d bilden somit das Geheimnis von *Ernst*.
Da p und q (große) Primzahlen, also ungerade, sind, ist klar, dass (p-1) und (q-1) gerade Zahlen, also jedenfalls keine Primzahlen sind. (Für Interessierte: Für jede Primzahl *pr* gilt, dass es (*pr*-1) Zahlen gibt, die relativ Prim zu dieser Zahl sind.) Für die Korrektheit des Verfahrens ist jedoch wichtig, dass diese 3 Zahlen zueinander relativ prim sind.
Durch die geheim zu haltende Zahl d hat sich Gruppe Ernst ihre „Falltüre" konstruiert. Nun musste sie nur noch eine zu dieser Falltüre passende Zahl e entwickeln, die innerhalb der Restklasse N die Eigenschaft hat, dass sich d und e wechselseitig aufheben, also wechselseitig zueinander invers sind, bzw. innerhalb der Restklasse N die Gleichung e*d = 1 gilt.

3.14. Anschließend hat Gruppe *Ernst* mittels

$$e * d = 1 \pmod{((p-1)*(q-1))}$$

den zweiten Teil des öffentlichen Schlüssels e berechnet.
Dazu wurde der Euklidsche Algorithmus verwendet. Bei Wahl der Werte p = 17, q = 11 und d = 7 bedeutet dies:

$$7 * d = 1 \pmod{(17-1) * (q-1)))} = 1 \pmod{(16 * 10)} = 1 \bmod 160.$$

Wir berechnen das inverse Element von d, also e, daher als ggt von 160 und 7.

ggt (160, 7) : $160 = 22 * 7 + 6$ $(22 * 7 = 154)$
$7 = 1 * \mathbf{6} + 1$
$6 = 6 * 1 + 0.$

Obige Formel verlangte jedoch dass e*d (mod ((p-1)*(q-1)) den Wert (Rest) 1 liefert. Es interessiert uns daher nicht das ohnehin bekannte Endergebnis, nämlich dass ggt(160,7) =1 ist, 160 und 7 somit teilerfremd sind, sondern es interessiert uns jener Quotient, der genau den Rest 1 lieferte, also der in der vorletzten Zeile erhaltene Wert 6. Diesen setzen wir in die entsprechend umgeformte Gleichung ein.
Wir formen daher erst einmal die Gleichung um und setzen dann ein:

$$1 \pmod{((p-1)*(q-1))} = e * d$$
$$1 \pmod{160} = 7 * d \pmod{160}$$

Wir rechnen nun die ggt-Berechnung „rückwärts", um so die Glieder der Kette der Kettendivision sukzessive wieder einsetzen zu können

$$1 = 7 - 1*6$$
$$6 = 160 - 7 * 22$$

Wir können dies nun in die Ausgangsformel einsetzen und erhalten

$1 = 7 - 1 * (160 - 7 * 22)$ und rechnen den Klammerausdruck aus
$1 = 7 - 160 + 7* 22$

$$1 = 23 * 7 - 160,$$

also, e = 23.

Durch die Berechnung von e ist die Vorarbeit für geheime Kommunikation abgeschlossen.

3.15. Damit konnte Gruppe *Ernst* nun den vollständigen öffentlichen Schlüssel (N, e) = (187, 23) veröffentlichen.

3.16. Mit diesen beiden Werten konnte Gruppe *Sandra* die zu übermittelnde Botschaft X, also den zu X zugehörigen dezimalen ASCII Code 88 mit der Verschlüsselungsfunktion M^e, also $C = 88^{23} \pmod{187}$ verschlüsseln und in der Form C = 11 senden.

Die zugehörige Rechnung lautete:

$C = 88^{23} \pmod{187} = 88^1 * 88^2 * 88^4 * 88^{16} \pmod{187} =$
$= 88^1 \pmod{187} * 88^2 \pmod{187} * 88^4 \pmod{187} * 88^{16} \pmod{187} =$
$= 88 * 77 * 132 * 154 \pmod{187} = 44 * 132 * 154 \pmod{187}$
$= 894.432 \pmod{187} =$
$= \mathbf{11} \pmod{187}$

3.17. Dieser Wert wurde von Gruppe *Ernst* mit der Entschlüsselungsfunktion unter Rückgriff auf die geheime Schlüsselkomponente d, $M = M^d$, also $M = 11^7 \pmod{187}$ entschlüsselt. Gruppe *Ernst* erhält dadurch die Zahl 88, die sie mit der ASCII-Tabelle wieder als X interpretiert.

Die zugehörige Rechnung lautete:

$M = 11^7 \pmod{187} = 11^1 * 11^2 * 11^4 \pmod{187} =$
$= 11^1 \pmod{187} * 11^2 \pmod{187} * 11^4 \pmod{187} =$
$= 11 * 121 * 55 \pmod{187} = 73.205 \pmod{187} =$
$= \mathbf{88} \pmod{187}$
oder binär 1011000

Setzt man diesen Wert in die ASCII-Tabelle ein, erhält man den Buchstaben **X**, das erste Zeichen der verschlüsselten Botschaft

4. Ausblick: Digitale Signatur

Wenn wir ein Dokument unterschreiben, machen wir dies mit einem Schriftzug, den nur die unterschreibende Person zügig ausführen kann, den aber jede Person mehr oder weniger lesen kann. Jedenfalls lässt sich im Zweifelsfall prüfen, ob die Unterschrift von jener Person stammt, die angeblich unterschrieben hat (oder vorgibt, nicht unterschrieben zu haben, die vermeintliche Unterschrift also als Fälschung bezeichnet).

Diese Asymmetrie erinnert an das Public-Key-Kryptosystem. Allein, die Rollen sind vertauscht; nur der Unterschreibende kann unterschreiben, aber alle anderen sollen feststellen können, dass es sich um die Signatur der/des Unterschreibenden handelt.

Bauen wir also wieder ein Public-Key-Kryptosystem mit den Werten (p, q, d) und berechnen daraus wie eben beschrieben N und e. Nun können wir ein Dokument mit $S = M^d \pmod{N}$ „signiert" versenden.

Der Empfänger wendet die öffentlich bekannte Verschlüsselungsfunktion auf den „signierten" und damit eigentlich verschlüsselten Text S an und erhält dadurch $M = S^e \pmod{N}$.

Wir erkennen aus dieser merkwürdigen Anordnung, dass eigentlich erst einmal Klartext mit einer Entschlüsselung verschlüsselt wird und beim Empfänger mit Hilfe der Verschlüsselungsfunktion entschlüsselt wird, die Bedeutung der Forderung, dass im RSA-Verfahren nicht nur D(E(M)) = M sondern auch E(D(M)) = M gelten soll.

5. Abschlussdiskussion

- Wie sicher sind Public-Key-Kryptosysteme?

  Als RSA publiziert wurde (1978) meinte man, die Faktorisierung einer Zahl mit 200 Stellen sei nicht effektiv ausführbar. Durch leistungsfähigere Computer (und teils andere Basisverfahren) sind heute für N Werte in der Größenordnung von 1024 Binärstellen erforderlich. Weiters zeigte sich, dass es

nicht hilft, kleine Werte von e zu publizieren. Auch hier wählt man mindestens 5-stellige Dezimalzahlen.

- Mit welcher Art von Attacken können Public-Key-Kryptosysteme gebrochen werden?
  Public-Key-Kryptosysteme sind durch „falsche Empfänger" aushebelbar. Daher ist es wesentlich, sicher zu sein, dass der (publizierte) öffentliche Schlüssel tatsächlich vom beabsichtigten Empfänger stammt.

Quellen:

http://www.netplanet.org/kryptografie/verfahren.shtml (14. 1. 2009)
http://www.gnupp.de/verschluesselung/index.html, (14. 1. 2009)
http://einklich.net/etc/vigenere.htm, (14. 1. 2009)
Horster Patrick: *Kryptologie*; B.I. Wissenschaftsverlag, Mannheim, 1985.
Hromkovič, Juraj: *Sieben Wunder der Informatik. Eine Reise an die Grenze des Machbaren mit Aufgaben und Lösungen*. Wiesbaden: Teubner 2006.
Rivest R.L., Shamir A., Adlemann L.: *Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, in Communications of the ACM, Vol 21, Nr. 2, Feb. 1987, pp. 120 – 126.
Schulz, Ralph-Hardo: Codierungstheorie. Eine Einführung. Wiesbaden: Vieweg Verlag 2003.
Smart, Nigel: *Cryptography: An Introduction*, McGraw Hill, 2003-
Singh, Simon: *Geheime Botschaften. Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet*. München: dtv 2004.

Anhang: ASCII-Tabelle f. Großbuchstaben:

	Binär	Dezimal		Binär	Dezimal
A	1000001	65	N	1001110	78
B	1000010	66	O	1001111	79
C	1000011	67	P	1010000	80
D	1000100	68	Q	1010001	81
E	1000101	69	R	1010010	82
F	1000110	70	S	1010011	83
G	1000111	71	T	1010100	84
H	1001000	72	U	1010101	85
I	1001001	73	V	1010110	86
J	1001010	74	W	1010111	87
K	1001011	75	X	1011000	88
L	1001100	76	Y	1011001	89
M	1001101	77	Z	1011010	90

Pseudocode für Prozedur ggt

```
function GGT (m, n : integer) : integer
   {
    WENN n = 0 DANN GGT := m
    SONST GGT := GGT (n, m mod n)
   }.
```

Hinweise für die Exponentiation großer Zahlen (mod N)

Wenn wir (selbst gar nicht so große Zahlen) zu einer hohen Potenz erheben, besteht sehr rasch die Gefahr, dass wir in Zahlenbereiche jenseits der Rechengenauigkeit des Computers kommen.

> *Tabellenkalkulationsprogramme wandeln ab einer maximal darstellbaren Ganzzahl in die Gleitkomma-Darstellung um, dabei gehen uns die letzten Stellen verloren. Es stimmt nur mehr die Großenordnung der berechneten Zahl.*

Wir müssen daher von Anbeginn innerhalb der jeweiligen Restklasse nach N rechnen, also sofort die Modulo-Funktion anwenden. Wir können diese entweder selbst nachbilden durch die Berechnung

```
        B                C                   D
  1     <Wert für M>     =Ganzzahl(B1/N)     =B1-C1*N
```

oder wir errechnen dies direkt mit der Funktion „Rest“

```
        B                C                   D
  1     <Wert für M>                         =Rest(B1;N)
```

wobei N der Schlüsselanteil ist, nach dem die Restklasse gebildet werden soll. (Es empfiehlt sich, diesen Wert an einer fixen Stelle einzugeben, auf die sodann durch direkte Adressierung , etwa N1 (oder unten A1), verwiesen wird.

Nun erinnern wir uns, dass Exponenten aufgespaltet werden können, also

$x^1 = x$

$x^2 = x^1 * x^1 = x^1 * x$

$x^3 = x^1 * x^1 * x^1 = x^2 * x$

$x^4 = x^1 * x^1 * x^1 * x^1 = x^3 * x$ oder $= x^2 * x^2$

Damit ist auch

$x^8 = x^1 * x^1 * x^1 * \ldots * x^1 = x^7 * x$ oder $= x^4 * x^4$

$x^{16} = x^1 * x^1 * x^1 * x^1 * \ldots * x^1 = x^{15} * x$ oder $= x^4 * x^4 * x^4 * x^4$

Wir können somit Exponenten wie 7 aus der Summe der Exponenten 4+2+1 oder 4+3 bilden und auch den Exponenten 23 aus der Addition der Exponenten 16 + 4 + 3 durch Multiplikation der x-Potenzen erhoben zum jeweiligen Summanden, sodass letztlich der gewünschte Gesamtexponent entsteht, erhalten.
Allerdings werden, wenn dies direkt ausgeführt wird, so große Zahlen entstehen, dass wir den ganzzahligen Rechenbereich unseres Tagellenkalkulationsprogramms überschreiten. Daher müssen wir schrittweise die Modulo-Operation anwenden und dürfen nur mit dem jeweiligen Restklassenwert weiterrechnen. Wir berechnen also tatsächlich:

$x^1 \pmod N = x \pmod N$

$x^2 \pmod N = x^1 \pmod N * x^1 \pmod N$

$x^3 \pmod N = x^2 \pmod N * x^1 \pmod N$

$x^4 \pmod N = x^3 \pmod N * x^1 \pmod N$ oder $= x^2 \pmod N * x^2 \pmod N$

…

$x^{k+1} \pmod N = x^k \pmod N * x^1 \pmod N$

…

$x^{2k} \pmod N = x^k \pmod N * x^k \pmod N$

Wenn sodann in der Produktbildung

$x^r \pmod N = x^1 \pmod N * x^2 \pmod N * \ldots * x^k \pmod N * \ldots * x^j \pmod N$ zu große Werte auftreten, ist auch diese Multiplikation schrittweise, mit zwischengeschaltener Modulo-Bildung durchzuführen.

Dies bedeutet, wir können die Aufgabe mit folgendem Tabellenkalkulationsprogramm einfach und für alle, die mit Tabellenkalkulation umgehen können, nachvollziehbar lösen.

	A	B	C	D
1	<Wert für N>	<Wert für M(oder für C)		Ergebnis_(M23,C7)
2				
3	1	=B$1	=Rest(B3;A1)	
4	=A3+1	=C3*B$1	=Rest(B4;A1)	
5	=A4+1	=C4*B$1	=Rest(B5;A1)	
6	=A5+1	=C5*B$1	=Rest(B6;A1)	=Rest(C5*C6;A1)
7	=A6+1	=C6*B$1	=Rest(B7;A1)	
8	=A7+1	=C7*B$1	=Rest(B8;A1)	
	Verdopplung			
10	=A6*2	=C6*C6	=Rest(B10;A1)	
11	=A10*2	=B10*B10	=Rest(B11;A1)	=Rest(C11*C6*C5;$A1)

Abschließend noch eine Warnung: Es ist möglich, dass bei einigen Werten der ASCII-Tabelle relativ kurze Zyklen auftreten (nicht zuletzt, aufgrund der kleinen Werte, mit denen wir arbeiten). Dies ist an sich unerwünscht. Ein Beispiel dafür wäre etwa der Buchstabe „C" (= 67) für Caesar. Die Verschlüsselung und Entschlüsselung erfolgt aber auch in diesem Fall korrekt.