9. Kapitel: Digitale Signaturen

Eine digitale Signatur soll für ein Dokument, das als Datei vorliegt, Ähnliches leisten wie eine gewöhnliche Unterschrift. Es ist keineswegs offensichtlich, wie man so etwas erreichen kann.

Bemerkung: Bachelor-, Zulassungs- und Masterarbeiten müssen eine Eigenständigkeitserklärung mit persönlicher Unterschrift enthalten.

1. Abschnitt: Einführung

Hier sind einige Eigenschaften, die eine eigenhändige Unterschrift hat bzw. haben sollte: (Die Eigenschaften wurden dem Buch „Angewandte Kryptographie“ von B. Schneier entnommen (S.41).)

- Eine Unterschrift ist authentisch. Sie überzeugt den Empfänger des Dokuments davon, dass der Unterzeichner das Dokument willentlich unterschrieben hat.
- Eine Unterschrift ist fälschungssicher. Sie beweist, dass der Unterzeichner und kein anderer das Dokument unterschrieben hat.
- Eine Unterschrift ist nicht wiederverwendbar. Sie ist Bestandteil des Dokuments und kann in kein anderes Dokument übertragen werden.
- Das unterzeichnete Dokument ist unveränderbar. Nachdem das Dokument unterschrieben ist, kann es nicht mehr geändert werden.
- Die Unterschrift kann nicht zurückgenommen werden. Unterschrift und Dokument liegen physisch vor. Der Unterzeichner kann später nicht behaupten, dass er das Dokument nicht unterschrieben hat.

Eine digitale Signatur soll für ein Dokument, das als Datei vorliegt, Ähnliches bewirken, was eine eigenhändige Unterschrift für ein gewöhnliches Dokument tut. Wie kann man das erreichen?

2. Abschnitt: Allgemeine Verfahren

Wir geben im Folgenden einige mögliche Verfahren an:

Unterschreiben mit einem Public-Key-Verschlüsselungsverfahren:

- Man legt ein Public-Key-Kryptosystem zugrunde, wo jeder Teilnehmer A die öffentliche Verschlüsselungsfunktion f_A bekannt gibt, die private Entschlüsselungsfunktion f_A^{-1} geheim hält.
- A will eine Nachricht/Datei M an B übermitteln und B soll sicher sein, dass die Nachricht wirklich von A stammt.
  - A wendet f_A^{-1} auf M an und schickt $f_A^{-1}(M)$ an B.
  - B wendet f_A auf die erhaltene Nachricht an und erhält mit $f_A(f_A^{-1}(M)) = M$ eine sinnvolle Nachricht, die natürlich nur von A stammen kann, da nur A die Funktion f_A^{-1} kennt.

Man überzeugt sich schnell, dass die gewünschten Forderungen an eine Unterschrift erfüllt sind.

Bemerkungen:

- Das beschriebene Verfahren ist natürlich nicht besonders effizient, wenn es sich um längere Nachrichten oder Dateien handelt.
- Verschlüsselung ist bisher noch nicht im Spiel: Jeder kann $f_A^{-1}(M)$ entschlüsseln durch Anwenden von f_A.

Das folgende Verfahren beschreibt eine effizientere Version:

Unterschreiben mit einem Public-Key-Verfahren unter Verwendung einer Hashfunktion:

- Man legt ein Public-Key-Verfahren mit öffentlichen Schlüsseln f_A zugrunde. Außerdem einigt man sich auf eine Hashfunktion h.
- A will eine Nachricht M mit Unterschrift an B senden.
  - A berechnet den Hashwert $h(M)$, wendet f_A^{-1} darauf an, erhält also $f_A^{-1}(h(M))$.
  - A schickt M und $f_A^{-1}(h(M))$ an B.
  - B besorgt sich den öffentlichen Schlüssel f_A von A, berechnet aus M den Hashwert $h(M)$ und mit f_A den Wert $f_A(f_A^{-1}(h(M)))$. Stimmen beide Werte überein, dann akzeptiert B die Unterschrift, da f_A^{-1} nur von A stammen kann.

Beispiel: Bei dem Programmpaket GnuPG (gpg), das unter Ubuntu bereits installiert ist, gibt es eine Variante, die getrennt zum Klartext eine Signatur erstellt (`gpg --clear-sign datei`), die z.B. wie folgt aussehen kann:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

In diesem Kapitel der Vorlesung 'Kryptographie I' werden digitale
Signaturen behandelt.
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEEW1yrx4qUz22m9e0BuOOJySp57wFAmeIr3cACgkQuOOJySp
57zIzgv+MuolRxUrTcRGRoQG1tZkaFT6kfI8iCIGzXEjHPqtGTXesWpjRdOFNei/
kkIUkjyxIjyqJ/O2q5d/h2wtqw1hqeuh5L9A8TZrE2m89D1v9nKLjI9ayrG/bumt
VQpMq+SAmb1wti5C7EkCwmQW+nuNldNVh4Qnv6LTzZHUZ+ugudFlySbvvrGzKTsE
kaNg9hkCdm16ejpTerywiBkSVwYH+iyX4GCWfYTb3L+Qv5XnB9TrZLjFbL24HKbX
AyYmgr2eObey0GMLZVvIpyP76+8P82xJxl5KBY4q8blo/PVbBU/odXoUWc4DIS/q
A+OncfJi4/CKnpPrK0GhtQ34KN3N2xciUXTvI/GwRN9yBr3dcj8lcGYVjnBvWVC5
Qe4THpWgJ/QBtnYIkHGrSmRqoAV09ZRCYtDYrPjevmXKcspRrrau97DX4zthBVC6
5K2LhM9msJSqucTLBlvrwtgWNqWaVIVy9PS+LG08up5LbHPUy3oEug7PApHVNJC9
RWQeUkbm
=c1D6
-----END PGP SIGNATURE-----
```

Die nächste Variante:

Digitale Signatur mit Verschlüsselung:

- Wieder legen wir ein Public-Key-Kryptosystem mit öffentlichen Verschlüsselungsfunktionen f_A zugrunde.
- Will A eine Nachricht M unterschrieben und verschlüsselt an B senden, bildet er $h(M)$, dann $f_A^{-1}(h(M))$ und hängt dies an das Ende von M, d.h. man hat $Mf_A^{-1}(h(M))$. Darauf wendet A jetzt den öffentlichen Schlüssel von B an und erhält $f_B(Mf_A^{-1}(h(M)))$. Dies wird an B verschickt.
- B wendet seine private Entschlüsselungsfunktion f_B^{-1} auf $f_B(Mf_A^{-1}(h(M)))$ an und hat dann $Mf_A^{-1}(h(M))$. Auf den Schluss $f_A^{-1}(H(M))$ wendet er f_A an, erhält $h(M)$, was mit dem Hashwert $h(M)$ des ersten Teils M übereinstimmt.

Beispiel: Auch bei GnuPG (gpg) gibt es die Möglichkeit, eine Datei zu verschlüsseln und zu unterschreiben
(`gpg -sea -r empfaenger datei`). Das Ergebnis sieht dann z.B. so aus:

```
-----BEGIN PGP MESSAGE-----

hQGMA4/i75c+TFdyAQv/aNr09iwJnrE33FS8NgWAUQ6LYkyW+KCzmMHVDglceiXS
Tl/ip3SCcPioKfdHTptReLoh4Z8Fs8JvOwkqPyDnLLvXNYn1DWpNK8a8c9H8APYV
Od+ffkoyZLy6DwrCaxi7/fzA5MhzsRy1s3bdWcEH3UjmhEhTiNW2XiwbbvV5Hus9
tgm21T0wdaOWvdf1CCzoSY55KcFfa1fPhSbzj7xPRkOBp+WkyAaZxxghctQLUENE
bLOZgKbDQ54CUUXL4n/TXtPJNZq9nxY0aPG7m0mwp2+d565ZW/ftCNGAR22foQD0
4PQSzUMjOZEb7mrp2SsoZk422Pr2qhuq0bNJe7D9BZs8doWSTRDHL+NwqapJ/mKJ
110+DcA4Q9Gi6KqHnLDKek/bmwP+IeeI/AMR6TLQCnbfIDbew4u1EO8k7dGrPvgy
+bsV3VGbIPm7pI8Oht/vJB//FF7edGC6TUlZpvClePDYaA2uqhVhSNbA//veg1Hb
xVhHxyeVtdOnxSLTfb951OkBCQIQJWfImS5U6saKZuzoyTJzT2CEVGtAZ5VavK0m
1YP7aA5QrPgaSV6+6TOYfU2TwI7SiiYFODf+V6p5FTrmVdAUk5wbkWsBIrJv5ocz
pjHsVFwICBWkzfTuLIFQoy9WaS2MnxE8zpc41/AGTLcID+NJ3oUPl0eMzNXQivgI
DeM1KUjXBm5DTWyto9H6DXCUogB1CP8+9BpyISMWCRQrWhKQcz+DJCfY/YVSRLr4
l4Swut/YxE/dyWKjPU6ZWRAuTgWV99p/coztB6+ZN9zHy8D+4r0XxqEgY6Ct/XZk
SQX0xj23fc2v08FKG7tAbHWd+w3EerZcxfkQiGCmWurmGnE6Cbo4Eub2ueXxR6hp
SmODTS1YFp3k2eui/Ij8oIOaHKRPKMcPtXiHlGygAqhKnsaXFqRAUmt1qoJhB6pm
V3UkSED38t10pbjTXDzILv7JvopkpIvxJnk3YEHGYevK7cJuF11jZfjMu4yUbfa/
r0OZa8HxLr4afN8R6TlQBfJWJk/90VWns4QEGVgL/8f8O0wMm89dhlBEJmmRsIab
pf2WJ8Z/aNkdD5FY52OAxu5IdfFpItA9psoaaPqpa28LA0HAVE0wC3UFzSInY836
EQogQTTcSkGSXL3vwzJjxi59ygiFR9zoedqXmRN8L8fmU+AWxpyK0FxT4ORRtlKK
o2rpyVohiA/SheUKYW5JOIYTMwKu8Csrf+IzGh6Ytn37T89al64XnzSqK9BAecEw
X+v4d+5QYqBwih2WzUxoVVxsnucO1f5m0Q+wSmOQ2aP7uMp1osC958aCBfGBolVf
Tm4zTKQOdpchR1o3
=joy7
-----END PGP MESSAGE-----
```

Wir geben jetzt explizite Algorithmen an, mit denen man signieren kann.

3. Abschnitt: Die RSA-Signatur

Die RSA-Signatur

- **Schlüsselerzeugung:**
  - Jeder Teilnehmer A besorgt sich einen öffentlichen und einen privaten RSA-Schlüssel (N_A, e_A) und (N_A, d_A).
  - Außerdem muss A sich auf eine Hashfunktion H festlegen.
- **Signieren einer Nachricht M:**
  - A bestimmt den Hashwert

$$h = H(M)$$

    der Nachricht M.
  - A berechnet mit seinem privaten Schlüssel

$$s = h^{d_A} \bmod N_A.$$

    Die RSA-Signatur von A für die Nachricht M ist dann die Zahl s.
- **Signaturüberprüfung:** Erhält jemand die Nachricht M und die zugehörige Signatur s, so geht er so vor:
  - Er bestimmt den Hashwert

$$h = H(M)$$

    der Nachricht M.
  - Er berechnet mit dem öffentlichen Schlüssel (N_A, e_A) von A

$$h' = s^{e_A} \bmod N_A.$$

  - Gilt $h' = h \bmod N_A$, so wird die Signatur als gültig akzeptiert. (Kurz: Gilt $H(M) \equiv s^{e_A} \bmod N_A$?)
  - Gilt $h' \neq h \bmod N_A$, so wird die Signatur nicht akzeptiert.

Bemerkungen:

- Wurde die Signatur s zum Hashwert h von A richtig erzeugt, d.h. $s = h^{d_A} \bmod N_A$, so gilt

$$h' \equiv s^{e_A} \equiv (h^{d_A})^{e_A} \equiv h^{d_A e_A} \equiv h \bmod N_A,$$

  also $h' = h \bmod N_A$, die Signatur besteht den Test.

- Wie kann ein Angreifer die Unterschrift von A fälschen? Er braucht, eine Zahl $\tilde{s}$, das die Signaturtestgleichung

$$h \equiv \tilde{s}^{e_A} \bmod N_A$$

  erfüllt. Dies ist aber genau das RSA-Problem, wobei hier $\tilde{s}$ dem Klartext und h dem Chiffretext entspricht. Wenn die RSA-Parameter passend gewählt sind, sollte ein Angreifer keine Zahl $\tilde{s}$ mit der angegebenen Eigenschaft praktisch bestimmen können.

Beispiel: Anna verwendet die RSA-Signatur und hat den öffentlichen RSA-Schlüssel (N, e) und den privaten Exponenten d mit

$$N = 1142135060145710775655863324084635984214875563970076954950625827124277586203368814808394124136958151$$
$$e = 65537,$$
$$d = 1136384040571301135971922422045361530474379219369079336391642553290349996843042453670312364800822105$$

Birgit erhält die Nachricht „`Ich stimme dem Vorschlag zu.`“ zusammen mit der Signatur

$$s = 1141109349255475312894875255374038173788681891299190712641929317464976257245996596289317836037424618,$$

die angeblich von Anna stammt. Birgit bestimmt den SHA-256-Hashwert der Nachricht

$$h = (5279e2f02af7f6a75f7cba6719b42b26db0a80500369e99c892b37320d651979)_{16} =$$
$$= 37305008348252725647931874747794067669287737688890315252165654044175455558009$$

und dann $h' = s^e \bmod N$ (mit dem öffentlichen Schlüssel (N, e) von Anna):

$$h' = 107562882556879572095594648943625591765216068227760592751218674422617077998922553355072598231972 3.$$

Da $h \neq h'$ ist, glaubt Birgit nicht, dass die Nachricht von Anna stammt.
Nach kurzer Zeit erhält Birgit die Nachricht
„`Ich stimme dem Vorschlag nicht zu.`“ zusammen mit der Signatur

$$s = 628084768488422084567369741138871721204232143047118292452055897657568533449323255487887592562316956,$$

die auch von Anna stammen soll. Birgit berechnet den SHA-256-Hashwert

$$h = (ac1ab5b1d953e429505a6af13a02f4c00eaa4b3ff90cd3cc67a85fda1527f0a1)_{16} =$$
$$= 778450019908926641233365319656511299755335739054989365055347730777947017012 81$$

und dann $h' = s^e \bmod N$ mit dem öffentlichen Schlüssel (N, e) von Anna mit dem Ergebnis

$$h' = 778450019908926641233365319656511299755335739054989365055347730777947017012 81.$$

Da $h = h'$ ist, glaubt Birgit, dass diese Nachricht von Anna stammt.

4. Abschnitt: Die ElGamal-Signatur

Die ElGamal-Signatur beruht nicht auf der ElGamal-Verschlüsselung.

Die ElGamal-Signatur:

- **Schlüsselerzeugung:** Jeder Teilnehmer A hat Folgendes zu tun:
  - Jeder Teilnehmer A wählt sich eine ungerade Primzahl p und dazu eine Primitivwurzel g modulo p.
    - Diskrete Logarithmen zur Basis g modulo p sollten sich praktisch nicht berechnen lassen.
    - Für den Fall $g \mid p-1$ gibt es einen Angriff von Bleichenbacher. Deshalb sollte man auch auf $g \nmid p-1$, insbesondere $g \neq 2$ achten.
    - Die Zahlen p und g können auch von anderen Teilnehmern verwendet werden.
  - A wählt geheim und zufällig eine Zahl e_A mit $0 \leq e_A \leq p-2$. Dann berechnet A die Zahl $f_A = g^{e_A} \bmod p$. Der öffentliche Schlüssel von A ist das Tripel (p, g, f_A), der geheime Schlüssel ist der Exponent e_A.
  - Dazu muss noch eine kryptographische Hashfunktion H festgelegt werden.

- **Signieren einer Nachricht:** Will A eine Nachricht M unterschreiben, so geht er folgendermaßen vor:
  - A bestimmt den Hashwert h der Nachricht M: $h = H(M)$.
  - A wählt eine zufällige Zahl z mit $0 \leq z \leq p-2$ und $\mathrm{ggT}(z, p-1) = 1$. (Wegen $\mathrm{ggT}(z, p-1) = 1$ ist z invertierbar modulo $p-1$.)
  - A berechnet nacheinander

$$b = g^z \bmod p \quad \text{und} \quad c = \frac{1}{z}(h - be_A) \bmod (p-1),$$

wobei $0 \leq b \leq p-1$ und $0 \leq c \leq p-2$ gilt.

Die Signatur/Unterschrift von A für die Nachricht M ist das Paar (b, c).

- **Signaturüberprüfung:** Wie überprüft B, der die Nachricht M erhält zusammen mit der Unterschrift (b, c), ob die Unterschrift gültig ist?
  - B berechnet den Hashwert h der Nachricht M: $h = H(M)$.
  - B testet, ob

$$0 \leq b \leq p-1 \quad \text{und} \quad g^h \equiv f_A^b b^c \bmod p$$

gilt. Wenn ja, akzeptiert B die Unterschrift. Andernfalls ist die Unterschrift ungültig.

Bemerkungen:

- Wurde die Signatur richtig erstellt, also

$$b = g^z \bmod p \quad \text{und} \quad c = \frac{1}{z}(h - be_A) \bmod (p-1),$$

so gilt natürlich $0 \leq b \leq p-1$, die erste Testbedingung. Da weiter $cz \equiv h - be_A \mod (p-1)$, also $h \equiv cz + be_A \mod (p-1)$ gilt, da man im Exponenten von g modulo $p-1$ rechnen darf, folgt

$$g^h \equiv g^{be_A + cz} \equiv (g^{e_A})^b (g^z)^c \equiv f_A^b b^c \bmod p,$$

d.h. auch die zweite Testbedingung ist erfüllt.

- Warum wird explizit $0 \leq b \leq p-1$ gefordert? Der Grund ist der, dass man sonst zu beliebigem Hashwert h ein Zahlenpaar (b, c) angegeben kann, das die Signaturtestgleichung

$$f_A^b b^c \equiv g^h \bmod p$$

erfüllt, nämlich

$$b = (p-1)(p-g) \quad \text{und} \quad c = h \bmod (p-1).$$

Denn es ist

$$b \equiv g \bmod p \quad \text{und} \quad b \equiv 0 \bmod (p-1),$$

sodass sich ergibt

$$f_A^b b^c \equiv f_A^0 g^h \equiv g^h \bmod p,$$

da man im Exponenten modulo $p-1$, in der Basis modulo p rechnen darf.

Beispiel: Andreas verwendet die ElGamal-Signatur, sein Schlüssel besteht aus p, g, e, f mit einer Primzahl p, einer Primitivwurzel g modulo p, einem privaten Exponenten e und $f = g^e \bmod p$; öffentlich sind (p, g, f) zugänglich:

$$
\begin{aligned}
p &= 168600185061638396245742195839459005247443255833991318598501\\
g &= 5,\\
e &= 369169129043646567067125161792381129791709565880436594040003\\
f &= 124332664308681423601810239583804090220703658325449963852135
\end{aligned}
$$

(Die Primzahl wurde so gewählt, dass $p - 1 = 2q$ mit einer Primzahl q gilt. Dadurch ist die Bestimmung einer Primitivwurzel einfach.) Andreas verwendet als Hashfunktion SHA-384.

(Fortsetzung des Beispiels) Andreas will die Nachricht „`Ich stimme dem Vorschlag zu.`“ signieren, bestimmt dafür den SHA-384-Hashwert der Nachricht:

$$\begin{aligned} h &= (\text{17adf8ade0072f41f86616c1a0d55fc880ed0da549d24cf8785abbafa95b97a} \\ &\quad \text{e67733ca4a2b850})_{16} = \\ &= 364462028180302507889265834775526384671148403138359535732315 \\ &\quad 797369739510859285000306670784164 64, \end{aligned}$$

wählt eine Zufallszahl z mit $\mathrm{ggT}(z, p-1) = 1$:

$z = 135963229140014016031133419106260776727845546943263278997040754$

und berechnet dann $b = g^z \bmod p$ und $c = \frac{1}{z}(h - be) \bmod (p-1)$ mit dem Ergebnis

$$\begin{aligned} b &= 937665324758983664079535901031019504241628102681467219403676 \\ c &= 551213266400382076120235194517664950027283154533856692218091 \end{aligned}$$

Die Signatur ist (b, c). (Man testet, dass tatsächlich $g^h \equiv f^b b^c \bmod p$ gilt.)

Bemerkungen: Wie sicher ist die ElGamal-Signatur?

- Wie kann ein Außenstehender C die Unterschrift von A fälschen? Natürlich kann C sich ein zufälliges z wählen, damit $b \equiv g^z \bmod p$ berechnen, dann aber braucht C noch c, sodass gilt $g^h \equiv f_A^b b^c \bmod p$ (oder $g^h \equiv g^{be_A+cz} \bmod p$). Dies ist aber gleichwertig mit

$$h \equiv be_A + cz \bmod (p-1).$$

  Eine gültige Signatur ist also äquivalent mit der Kenntnis des privaten Schlüssels e_A von A. Diesen kann man aber im allgemeinen nur durch Logarithmenberechnung aus $g^{e_A} \equiv f_A \bmod p$ erhalten.

- Wenn nicht verschlüsselt wird, sind eventuell h und (b, c) zugänglich. Dann weiß man, dass gilt

$$h \equiv b \cdot e_A + c \cdot z \bmod p - 1.$$

Was passiert, wenn man die „Zufallszahl“ z erraten kann? Ob man das richtige z hat, kann man mit der Gleichung

$$b = g^z \bmod p$$

testen. Obige Gleichung wird zu

$$b \cdot e_A \equiv h - cz \bmod (p - 1)$$

mit der Unbekannten e_A (und den bekannten Größen b, c, z, h, p). Wie kann man diese Gleichung lösen? (Dies ist eine Gleichung vom Typ $ax \equiv b \bmod m$, die wir im Grundlagenteil behandelt haben.)

- Zusammenfassung: Wie findet man eine Lösung der Gleichung $b \cdot e_A \equiv h - cz \bmod (p-1)$, wenn b, c, h, p, z gegeben sind und e_A gesucht wird?
  Sei $\tilde{b}$ ein Inverses von $\frac{b}{\mathrm{ggT}(p-1,b)}$ modulo $\frac{p-1}{\mathrm{ggT}(p-1,b)}$, d.h.

$$\frac{b}{\mathrm{ggT}(p-1,b)} \cdot \tilde{b} \equiv 1 \bmod \frac{p-1}{\mathrm{ggT}(p-1,b)}.$$

  Weiter sei

$$e_0 = \tilde{b} \cdot \frac{h - cz}{\mathrm{ggT}(p-1,b)} \bmod \frac{p-1}{\mathrm{ggT}(p-1,b)}.$$

  Dann ist

$$e_A = e_0 + i \cdot \frac{p-1}{\mathrm{ggT}(p-1,b)} \text{ für ein } i \text{ mit } 0 \leq i \leq \mathrm{ggT}(p-1,b) - 1.$$

  Das richtige e_i findet man durch die Bedingung $g^{e_i} \equiv f_A \bmod p$. Also: Kann man z erraten, kann man (im Allgemeinen) e_A bestimmen.

- *Beweis:* Da $\frac{b}{\mathrm{ggT}(p-1,b)}$ und $\frac{p-1}{\mathrm{ggT}(p-1,b)}$ teilerfremd sind, ist $\frac{b}{\mathrm{ggT}(p-1,b)}$ invertierbar modulo $\frac{p-1}{\mathrm{ggT}(p-1,b)}$, es existiert also eine Zahl $\tilde{b}$ mit den angegebenen Eigenschaften. Da die Gleichung $b \cdot e_A \equiv h - cz \bmod (p-1)$ lösbar ist, gilt weiter $\mathrm{ggT}(p-1,b) \mid h - cz$, also $\frac{h-cz}{\mathrm{ggT}(p-1,b)} \in \mathbb{Z}$. Wir formen äquivalent um:

$$
\begin{aligned}
b \cdot e_A \equiv h - cz \bmod (p-1) \quad &\Longleftrightarrow \quad \frac{b}{\mathrm{ggT}(p-1,\,b)} \cdot e_A \equiv \frac{h-cz}{\mathrm{ggT}(p-1,\,b)} \bmod \frac{p-1}{\mathrm{ggT}(p-1,\,b)} \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad e_A \equiv \tilde{b} \cdot \frac{h-cz}{\mathrm{ggT}(p-1,\,b)} \bmod \frac{p-1}{\mathrm{ggT}(p-1,\,b)} \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad e_A \equiv e_0 \bmod \frac{p-1}{\mathrm{ggT}(p-1,\,b)} \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad \frac{p-1}{\mathrm{ggT}(p-1,\,b)} \mid e_A - e_0 \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad e_A - e_0 = i \cdot \frac{p-1}{\mathrm{ggT}(p-1,\,b)} \text{ für ein } i \in \mathbb{Z} \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad e_A = e_0 + i \cdot \frac{p-1}{\mathrm{ggT}(p-1,\,b)} \text{ für ein } i \in \mathbb{Z}.
\end{aligned}
$$

Aus $0 \leq e_A \leq p-2$ erhält man dann noch $0 \leq i \leq \mathrm{ggT}(p-1,b) - 1$. ■

- Angenommen wir sehen, dass jemand 2 Nachrichten signiert mit Unterschriften (b, c_1) und (b, c_2) und Hashwerten h_1, h_2, also gleicher erster Komponente in den Signaturen, aber $c_1 \neq c_2$. Damit liegt auch bei beiden Signaturen (modulo $p-1$) die gleiche Zufallszahl z vor.
  - Nach Definition erhalten wir dann ein Gleichungssystem

$$\begin{aligned} c_1 z &\equiv h_1 - b e_A \bmod (p-1), \\ c_2 z &\equiv h_2 - b e_A \bmod (p-1). \end{aligned}$$

    Subtraktion liefert

$$(c_1 - c_2) z \equiv h_1 - h_2 \bmod (p-1).$$

    Diese Gleichung hat $\mathrm{ggT}(c_1 - c_2, p-1)$ Lösungen modulo $p-1$, die wir explizit angegeben können: Ist

$$z_0 = \frac{h_1 - h_2}{\mathrm{ggT}(p-1, c_1 - c_2)} \cdot \left(\frac{c_1 - c_2}{\mathrm{ggT}(p-1, c_1 - c_2)} \right)^{-1} \bmod \frac{p-1}{\mathrm{ggT}(p-1, c_1 - c_2)},$$

    so sind

$$z_i \equiv z_0 + i \frac{p-1}{\mathrm{ggT}(c_1 - c_2, p-1)} \bmod (p-1) \text{ für } i = 0, \ldots, \mathrm{ggT}(c_1 - c_2, p-1) - 1$$

    die anderen Lösungen.
  - Nun probiert man durch, für welches z_i die Gleichung $b \equiv g^{z_i} \bmod p$ gilt. Dies ist dann das richtige z.
  - Nun bleibt die Gleichung

$$b e_A \equiv h_1 - c_1 z \bmod (p-1)$$

    mit der Unbekannten e_A, die man wie zuvor lösen kann. Die Gleichung hat $\mathrm{ggT}(p-1, b)$ Lösungen modulo $p-1$. Ist diese Zahl nicht zu groß, kann man das richtige e_A durch Probieren finden.

  Man muss also darauf achten, dass der Zufallszahlengenerator gut ist, insbesondere sollte er lauter verschiedene Zahlen liefern.

- Die folgenden Ausführungen sind eigentlich nicht mehr nötig, weil wir schon zuvor gesehen habe, dass die Signaturtestbedingung $0 \leq b \leq p-1$ wichtig ist.
  Sei (b, c) eine gültige ElGamal-Signatur für ein Dokument mit Hashwert h zum öffentlichen Schlüssel (p, g, f_A). Es gilt also

$$f_A^b b^c \equiv g^h \bmod p.$$

  Sei h' ein Hashwert. Gilt $\mathrm{ggT}(h, p-1) \mid h'$, dann gibt es ein $u \in \mathbb{N}$ mit

$$h' \equiv hu \bmod (p-1).$$

  Mit dem chinesischen Restsatz finden wir ein b' mit

$$b' \equiv b \bmod p \quad \text{und } b' \equiv bu \bmod (p-1).$$

  Wir wählen ein c' mit

$$c' \equiv cu \bmod (p-1).$$

  Da man an der Basis modulo p, im Exponenten modulo $p-1$ rechnen darf, ergibt sich

$$f_A^{b'} {b'}^{c'} \equiv f_A^{b'} b^{c'} \equiv f_A^{bu} b^{cu} \equiv (f_A^b b^c)^u \equiv (g^h)^u \equiv g^{h'} \bmod p.$$

  (b', c') erfüllt also die Signaturtestgleichung zum Hashwert h' für den öffentlichen Schlüssel (p, g, f_A). Man kann b', c' auch explizit angeben:

$$b' = b(up - p + 1) \bmod p(p-1), \quad c' = cu \bmod (p-1).$$

  Im Allgemeinen wird hier $b' \geq p$ gelten, d.h. die erste Signaturtestbedingung ist verletzt.

- Um die Unterschrift von A für ein Dokument M zu fälschen, braucht man Zahlen b, c mit $0 \leq b, c \leq p-1$ und

$$g^h \equiv f_A^b b^c \bmod p \quad \text{und} \quad h = h(M).$$

Es ist leicht, viele Lösungen (h, b, c) der Gleichung $g^h \equiv f_A^b b^c \bmod p$ anzugegeben, wenn man die Bedingung $h = h(M)$ weglässt: Seien $u, v \in \{0, 1, \ldots, p-2\}$ mit $\mathrm{ggT}(v, p-1) = 1$ und

$$b \equiv g^u f_A^v \bmod p, \quad c \equiv -\frac{b}{v} \bmod (p-1), \quad h \equiv uc \bmod (p-1).$$

Dann gilt

$$f_A^b b^c \equiv f_A^b (g^u f_A^v)^c \equiv g^{uc} f_A^{b+vc} \equiv g^{uc} \equiv g^h \bmod p.$$

Da man nach Voraussetzungen an die Hash-Funktion h allerdings keine Nachrichten M mit $h = h(M)$ finden kann, kann man auf diese Weise die Unterschrift auch nicht fälschen. Es wird aber klar, wie wichtig hier die Hashfunktion ist.

Bleichenbacher hat einen Angriff auf die ElGamal-Signatur im Fall $g \mid p-1$ beschrieben, der für „viele“ Hashwerte h eine gültige Signatur zu einem öffentlichen Schlüssel (p, g, f_A) liefert. Deswegen muss man bei Verwendung der ElGamal-Signatur darauf achten, dass

$$g \nmid p-1, \quad \text{insbesondere} \quad g \neq 2$$

gilt. Der folgende Satz macht eine genaue Aussage.

Satz

(Bleichenbacher) Sei p eine ungerade Primzahl, g eine Primitivwurzel modulo p, $e_A \in \mathbb{N}$ und $f_A \equiv g^{e_A} \bmod p$. Wir setzen voraus, dass gilt

$$g \mid p-1.$$

Wir definieren

$$b = \frac{p-1}{g}.$$

Es gibt genau ein $\tilde{e} \in \{0, 1, \ldots, g-1\}$ mit

$$f_A^b \equiv g^{b\tilde{e}} \bmod p,$$

das man durch Probieren bestimmen kann, wenn g nicht zu groß ist. Sei $h \in \mathbb{N}_0$ beliebig gegeben. Wir unterscheiden zwei Fälle:

- **Fall** $p \equiv 1 \bmod 4$**:** *Definiert man*

$$c = \frac{p-3}{2} \cdot (h - b\tilde{e}) \bmod (p-1),$$

  so gilt

$$f_A^b b^c \equiv g^h \bmod p.$$

- **Fall** $p \equiv 3 \bmod 4$ **und** $h \equiv b\tilde{e} \bmod 2$**:** *Wählt man $c \in \mathbb{N}_0$ mit*

$$c \equiv b\tilde{e} - h \bmod \frac{p-1}{2},$$

  so gilt

$$f_A^b b^c \equiv g^h \bmod p.$$

- **Fall** $p \equiv 3 \bmod 4$ **und** $h \not\equiv b\tilde{e} \bmod 2$**:** *Für alle $c \in \mathbb{N}_0$ gilt $f_A^b b^c \not\equiv g^h \bmod p$.*

Beweis:

- Die Voraussetzung $g \mid p-1$ impliziert, dass

$$b = \frac{p-1}{g}$$

  eine natürliche Zahl ist.
- Da g eine Primitivwurzel modulo p sein soll, gilt $g^{\frac{p-1}{2}} \equiv -1 \bmod p$. ($-1$ ist das einzige Element der Ordnung 2 modulo p, das sich auch in der Form $g^{\frac{p-1}{2}}$ schreiben lässt.) Aus $b = \frac{p-1}{g}$ folgt $gb = p-1$, also modulo p

$$gb \equiv -1 \equiv g^{\frac{p-1}{2}} \bmod p, \quad \text{und damit} \quad b \equiv g^{\frac{p-3}{2}} \bmod p.$$

- Es ist $f_A \equiv g^{e_A} \bmod p$ mit dem privaten Exponenten e_A. Vorbereitend überlegen wir, dass für $\tilde{e} \in \mathbb{N}_0$ folgende Äquivalenzen gelten:

$$\begin{aligned} f_A^b \equiv g^{b\tilde{e}} \bmod p &\iff g^{e_A b} \equiv g^{b\tilde{e}} \bmod p \iff e_A b \equiv b\tilde{e} \bmod (p-1) \iff \\ &\iff p-1 \mid b(e_A - \tilde{e}) \iff p-1 \mid \frac{p-1}{g}(e_A - \tilde{e}) \iff \\ &\iff g \mid e_A - \tilde{e} \iff \tilde{e} \equiv e_A \bmod g. \end{aligned}$$

  Dies sollte gezeigt werden.

- Wir brauchen jetzt noch ein c, das die Bedingung $f_A^b b^c \equiv g^h \bmod p$ erfüllt. Wir formen äquivalent um:

$$\begin{aligned} f_A^b b^c \equiv g^h \bmod p \quad &\Longleftrightarrow \quad g^{b\tilde{e}} \cdot \left(g^{\frac{p-3}{2}}\right)^c \equiv g^h \bmod p \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad g^{b\tilde{e}+\frac{p-3}{2}\cdot c} \equiv g^h \bmod p \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad b\tilde{e} + \frac{p-3}{2} \cdot c \equiv h \bmod (p-1) \quad \Longleftrightarrow \\ &\Longleftrightarrow \quad \frac{p-3}{2} \cdot c \equiv h - b\tilde{e} \bmod (p-1). \end{aligned}$$

Die letzte Gleichung ist vom Typ $ax \equiv b \bmod n$ und kann gut untersucht werden. Schreiben wir

$$p = 1 + 2m, \quad \text{so ist} \quad \frac{p-3}{2} = m - 1 \quad \text{und} \quad p - 1 = 2m.$$

Die Gleichung wird dann zu

$$(m-1) \cdot c \equiv h - b\tilde{e} \bmod 2m.$$

Es ist

$$\mathrm{ggT}(m-1, 2m) = \mathrm{ggT}(m-1, 2m-2(m-1)) = \mathrm{ggT}(m-1, 2) = \begin{cases} 1, & \text{falls } m \equiv 0 \bmod 2, \\ 2, & \text{falls } m \equiv 1 \bmod 2. \end{cases}$$

Wir unterscheiden zwei Fälle:

Zu lösende Gleichung: $(m-1)\cdot c \equiv h - b\tilde{e} \bmod 2m$.

- **Fall $m \equiv 0 \bmod 2$, also $p \equiv 1 \bmod 4$:** Dann ist $\mathrm{ggT}(m-1, 2m) = 1$, also $m-1$ invertierbar modulo $2m$. Das Inverse von $m-1$ modulo $2m$ ist $m-1$, was man mit dem erweiterten euklidischen Algorithmus finden, aber natürlich auch direkt nachrechnen kann:

$$(m-1)^2 = m^2 - 2m + 1 = 1 + 2m \cdot (\frac{m}{2} - 1) \equiv 1 \bmod 2m.$$

Damit kann man obige Gleichung weiter umformen:

$$(m-1)\cdot c \equiv h - b\tilde{e} \bmod 2m \quad \Longleftrightarrow \quad c \equiv (m-1)(h - b\tilde{e}) \bmod 2m.$$

Also gilt:

$$f_A^b b^c \equiv g^h \bmod p \quad \Longleftrightarrow \quad c \equiv \frac{p-3}{2} \cdot (h - b\tilde{e}) \bmod (p-1).$$

Zu lösende Gleichung: $(m-1)\cdot c \equiv h - b\tilde{e} \bmod 2m$.

- **Fall $m \equiv 1 \bmod 2$, also $p \equiv 3 \bmod 4$:** Dann ist $\mathrm{ggT}(m-1, 2m) = 2$. Obige Gleichung können wir wegen $\mathrm{ggT}(2, m) = 1$ umformen:

$$(m-1)\cdot c \equiv h - b\tilde{e} \bmod 2m \quad \Longleftrightarrow \quad \begin{cases} (m-1)\cdot c \equiv h - b\tilde{e} \bmod 2 \\ (m-1)\cdot c \equiv h - b\tilde{e} \bmod m \end{cases}$$

$$\Longleftrightarrow \quad \begin{cases} 0 \equiv h - b\tilde{e} \bmod 2 \\ c \equiv b\tilde{e} - h \bmod m \end{cases}$$

  Es gibt zwei Fälle:

  - **Fall $h \equiv b\tilde{e} \bmod 2$:** Die erste Gleichung ist trivialerweise erfüllt, also bleibt nur die zweite Gleichung, und damit

$$f_A^b b^c \equiv g^h \bmod p \quad \Longleftrightarrow \quad c \equiv b\tilde{e} - h \bmod \frac{p-1}{2}.$$

  - **Fall $h \not\equiv b\tilde{e} \bmod 2$:** Wegen $0 \not\equiv h - b\tilde{e} \bmod 2$ gibt es kein c, das die Gleichung $f_A^b b^c \equiv g^h \bmod p$ erfüllt.

  Damit ist alles gezeigt. ■

Bemerkungen:

- Der vorangegangene Satz liefert eine einfache Möglichkeit, Signaturen ohne Kenntnis des privaten Schlüssels zu fälschen, wenn $g \mid p-1$ gilt. Im Fall $p \equiv 1 \bmod 4$ kann man zu jedem Hashwert eine Signatur erstellen, im Fall $p \equiv 3 \bmod 4$ für die Hälfte der Hashwerte.
- Merkwürdigerweise wird die Bedingung $g \nmid p-1$ nicht in allen Büchern, die die ElGamal-Signatur behandeln, erwähnt. (Im Kryptographie-Buch von Buchmann steht nichts davon.)

Beispiel: Gerhard benutzt (p, g, f) als öffentlichen Schlüssel für die ElGamal-Signatur, wobei

$$\begin{aligned} p &= 75698236598763249875623874526348756238746563756479, \\ g &= 23, \\ f &= 336463817982345361571746674081113966758893373536 45. \end{aligned}$$

Als Hashfunktion verwendet er SHA3-224.
Hermann stellt fest, dass $23 \mid p-1$ gilt, also sollte es möglich sein, Signaturen von Gerhard zu fälschen. Nach Vorlesung berechnet Hermann

$$b = \frac{p-1}{g} = 3291227678207097820679298892449945923423763641586$$

und bestimmt $\tilde{e} \in \{0, 1, \ldots, 22\}$ mit $f^b \equiv g^{b\tilde{e}} \bmod p$ durch Probieren:

$$\tilde{e} = 9.$$

Es ist

$$p \equiv 3 \bmod 4 \quad \text{und} \quad b\tilde{e} \equiv 0 \bmod 2.$$

Ist h eine gerade Zahl, setzt man

$$c = (b\tilde{e} - h) \bmod \frac{p-1}{2},$$

so gilt $f^b b^c \equiv g^h \bmod p$.

(Fortsetzung des Beispiels) Hermann würde gerne den Satz `Ich werde Dir bei der Wahl meine Stimme geben.` mit der Unterschrift von Gerhard versehen. Er berechnet den Hashwert:

$$h = 176768966920926827017410531101410686220654780583252378492688998334 7.$$

Leider ist h ungerade. Hermann hängt an den Satz noch ein Leerzeichen an:
`Ich werde Dir bei der Wahl meine Stimme geben.` Jetzt ist der Hashwert

$$h = 208692092654440009326574242673307813998251753775069248690668101495 86$$

eine gerade Zahl. Hermann berechnet damit

$$c = (b\tilde{e} - h) \bmod \frac{p-1}{2} = 884244458474412237227965405121823935034737898455 4.$$

Tatsächlich gilt nun für das Paar (b, c)

$$f^b b^c \equiv g^h \bmod p = 572530021699251643413964854227276465883827034941 48$$

(b, c) ist also eine gültige ElGamal-Signatur von Gerhard für den Satz `Ich werde Dir bei der Wahl meine Stimme geben.` .

P. Q. Nguyen hat bemerkt, dass in GnuPG Version 1.2.3 die Parameter e und z bei der ElGamal-Signatur leichtsinnigerweise - um die Effizienz zu steigern - so klein gewählt wurden, dass ein Gitterangriff zur Auffindung des privaten Schlüssel führen kann. Da in der Vorlesung aber keine Gitter behandelt wurden, begnügen wir uns mit einer Skizze.

Eine Variante des Nguyen-Gitterangriffs auf die ElGamal-Signatur:

- Gegeben sei ein öffentlicher ElGamal-Signatur-Schlüssel (p, g, f), der Hashwert h einer Nachricht und eine zugehörige Unterschrift (b, c).
- Wenn die Unterschrift richtig erstellt wurde, gibt es einen privaten Exponenten e mit $g^e \equiv f \bmod p$ und eine Zahl z mit $\mathrm{ggT}(p-1, z) = 1$ und

$$b = g^z \bmod p \quad \text{und} \quad c = \frac{1}{z}(h - be) \bmod (p-1).$$

- Aus $c = \frac{1}{z}(h - be) \bmod (p-1)$ folgt $cz \equiv h - be \bmod (p-1)$, also $eb + zc - h \equiv 0 \bmod (p-1)$, es gibt also eine ganze Zahl $m \in \mathbb{Z}$ mit

$$eb + zc - h + m(p-1) = 0.$$

- Wir definieren eine 4×4-Matrix M (mit Einträgen aus $\mathbb{Z}$) durch

$$M = \begin{pmatrix} 1 & 0 & 0 & b \\ 0 & 1 & 0 & c \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & p-1 \end{pmatrix}.$$

  (Die Matrix M enthält nur die öffentlich bekannten Größen b, c, h, p.)
- Es gilt (mit $eb + zc - h + m(p-1) = 0$)

$$(e, z, -1, 0) = (e, z, -1, m) \begin{pmatrix} 1 & 0 & 0 & b \\ 0 & 1 & 0 & c \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & p-1 \end{pmatrix},$$

  also $(e, z, -1, 0) = (e, z, -1, m)M$.
- Nun betrachten wir die Menge

$$\Lambda = \{(z_1, z_2, z_3, z_4)M \in \mathbb{R}^4 : (z_1, z_2, z_3, z_4) \in \mathbb{Z}^4\},$$

  wobei wir die Elemente von $\mathbb{Z}^4$ als Zeilenvektoren auffassen. Λ besteht also aus den ganzzahligen Linearkombinationen der Zeilenvektoren von M. Die Menge Λ ist ein sogenanntes Gitter (in $\mathbb{R}^4$). Wegen $(e, z, -1, 0) = (e, z, -1, m)M$ gilt

$$(e, z, -1, 0) \in \Lambda.$$

- Bei der Betrachtung von Gittern lernt man im Zusammenhang mit dem LLL-Algorithmus folgende Aussagen[1]: Ist v_1 der erste Vektor einer LLL-reduzierten Basis von Λ (bzw. der erste Zeilenvektor der zugehörigen Matrix), so gilt für die euklidische Norm (in der 4-dimensionalen Situation)

$$\|v_1\| \leq 1.69 |\det M|^{\frac{1}{4}} \leq 1.69\, p^{\frac{1}{4}} \quad \text{und} \quad \|v_1\| \leq 2.83 \min_{v \in \Lambda \setminus \{0\}} \|v\|.$$

Ist nun $e \leq p^{\frac{1}{4}}$ und $z \leq p^{\frac{1}{4}}$, so ist

$$\|(e, z, -1, 0)\| = \sqrt{e^2 + z^2 + 1} \leq 1.73\, p^{\frac{1}{4}}$$

und man kann schauen, ob eventuell $(e, z, -1, 0) = \pm v_1$ gilt. Konkret: Man berechnet mit dem LLL-Algorithmus v_1 und testet, ob $\pm v_1 = (e, z, -1, 0)$ gelten kann. (Dies ist eine Variante des Nguyen-Angriffs.)

[1] J. Hoffstein, J. Pipher, J. H. Silverman, An Introduction to Mathematical

Beispiel: Gegeben ist ein öffentlicher ElGamal-Signatur-Schlüssel $(p, g, f) = (3745682638563756637 9, 10, 35221072137678344548)$, außerdem eine Signatur $(b, c) = (268025299251739382 1, 3007000086637061167)$ und ein zugehöriger Hashwert $h = 4900226293976985289$. Wir bilden die Matrix

$$M = \begin{pmatrix} 1 & 0 & 0 & 268025299251739382 1 \\ 0 & 1 & 0 & 3007000086637061167 \\ 0 & 0 & 1 & 4900226293976985289 \\ 0 & 0 & 0 & 3745682638563756637 8 \end{pmatrix}$$

LLL-Reduktion liefert die Matrix (Sage: `M.LLL()`)

$$\begin{pmatrix} 44444 & 55555 & -1 & 0 \\ -26534 & 47422 & -57370 & 2796 \\ 44425 & -48697 & -16209 & 42803 \\ -11147 & 36201 & 39874 & 78616 \end{pmatrix}$$

Setzt man jetzt $e = 44444$ und $z = 55555$, so findet man, dass tatsächlich $f = g^e \bmod p$ und $b = g^z \bmod p$ gilt. Der Angriff war also erfolgreich.

Experimente: Zum Experimenten haben wir mit folgender Python-Funktion zufällige Beispiele erzeugt:

```
# ng.py - 22.1.2025
# Erzeugung zufaelliger Beispiele
# p, e, z und h werden zufaellig gewaehlt
# Eingabe: st_p, st_e (st_p ist die Stellenzahl von p, st_e die Stellenzahl von
# Rueckgabe: [p,g,e,f,h,z,b,c]
def beispiel(st_p,st_e):
  from krypt import nxtprm, factor_kt, ord_p, sqrt, ggT, invmod, prim
  from random import randint
  while True:
    p=nxtprm(randint(10**(st_p-1),10**st_p))
    F=factor_kt(p-1,10**6)
    if F[-1]==1 or prim(F[-1]):
      g=2
      while (p-1)%g==0 or ord_p(g,p)!=p-1:
        g=g+1
      e=randint(10**(st_e-1),10**st_e)
      f=pow(g,e,p)
      h=randint(0,p-1)
      z=randint(10**(st_e-1),10**st_e)
      while ggT(p-1,z)>1:
        z=z+1
      b=pow(g,z,p)
      c=invmod(z,p-1)*(h-b*e)%(p-1)
      return [p,g,e,f,h,z,b,c]
```

(Fortsetzung)
Mit SAGE kann man dann den Nguyen-Angriff durchführen:

```
from ng import beispiel

def angriff_nguyen(bsp):
  p,g,e,f,h,z,b,c=bsp
  M=Matrix([[1,0,0,b],[0,1,0,c],[0,0,1,h],[0,0,0,p-1]])
  v1=M.LLL()[0]
  print("v1=",v1,sep="")
  ee=abs(v1[0])
  if pow(g,ee,p)==f:
    return 'erfolgreich'
  return 'nicht erfolgreich'
```

Ist die Stellenzahl von e und z etwas kleiner als $\frac{1}{4}$ der Stellenzahl von p, so scheint der Angriff erfolgreich zu sein.
(Leider kann ich diese Vermutung nicht beweisen.)

5. Abschnitt: Von der ElGamal-Signatur zu DSA

Überlegungen:

- Wir starten mit der ElGamal-Signatur. Wir haben einen öffentlichen Schlüssel (p, g, f), wobei g eine Primitivwurzel modulo p ist und $f = g^e \bmod p$ mit dem privaten Schlüssel e gilt.
- Zum Signieren einer Nachricht mit Hashwert h wird eine zufällige Zahl z mit $\mathrm{ggT}(z, p-1) = 1$ gewählt und dann

$$b = g^z \bmod p, \quad c = \frac{1}{z}(h - be) \bmod p - 1$$

berechnet. (b, c) ist die ElGamal-Signatur der Nachricht. Aus $cz \equiv h - be \bmod (p-1)$ folgt sofort die Signatur-Test-Gleichung:

$$g^h \equiv g^{eb+zc} \equiv (g^e)^b (g^z)^c \equiv f^b b^c \bmod p.$$

- Wir ändern die Vorgehensweise jetzt leicht ab. In der Definition von c ändern wir das Vorzeichen von be:

$$c = \frac{1}{z}(h + be) \bmod p - 1.$$

Dann ist $cz \equiv h + be \bmod p - 1$, sodass die Signatur-Test-Gleichung jetzt zu

$$b^c \equiv (g^z)^c \equiv g^{eb+h} \equiv (g^e)^b g^h \equiv f^b g^h \bmod p$$

wird.

- Bisher haben wir angenommen, dass g eine Primitivwurzel modulo p ist, d.h. dass $\operatorname{ord}_p(g) = p - 1$ gilt. Benutzt wurde diese Voraussetzung weder bei der Signaturerstellung noch beim Signaturtest. Jetzt nehmen wir die zweite Veränderung vor. Es gebe eine Primzahl q mit $q \mid p - 1$. Das Element g habe jetzt Ordnung q modulo p, d.h. $\operatorname{ord}_p(g) = q$. Im Exponenten dürfen wir dann modulo q rechnen. Wir wählen also z mit $0 < z < q$ und definieren

$$b = g^z \bmod p, \quad c = \frac{1}{z}(h + be) \bmod q.$$

(Da q eine Primzahl ist und $0 < z < q$ gilt, ist z invertierbar modulo q.) Die Signatur-Test-Gleichung bleibt gleich:

$$b^c \equiv f^b g^h \bmod p.$$

- Man sollte darauf achten, dass $c \neq 0$ ist, andernfalls könnte man aus $h + be \equiv 0 \bmod q$ den geheimen Schlüssel e berechnen. Dann ist c invertierbar modulo q, d.h. $\frac{1}{c} \bmod q$ existiert. Da wir im Exponenten modulo q rechnen dürfen, folgt durch Potenzieren mit $\frac{1}{c} \bmod q$

$$b \equiv f^{\frac{b}{c} \bmod q} g^{\frac{h}{c} \bmod q} \bmod p$$

bzw.

$$b = f^{\frac{b}{c} \bmod q} g^{\frac{h}{c} \bmod q} \bmod p.$$

- Durch Reduktion modulo q folgt aus der letzten Gleichung

$$b \bmod q = \left(f^{\frac{b}{c} \bmod q} g^{\frac{h}{c} \bmod q} \bmod p \right) \bmod q.$$

In diese Gleichung geht jetzt nur noch $b \bmod q$ ein. Ebenso geht in die Definition von c nur $b \bmod q$ ein. Wir schreiben daher jetzt für $b \bmod q$ einfach wieder b und erhalten folgende Vorgehensweise:

$$b = (g^z \bmod p) \bmod q, \quad c = \frac{1}{z}(h + be) \bmod q$$

und die zugehörige Test-Gleichung

$$b = \left(f^{\frac{b}{c} \bmod q} g^{\frac{h}{c} \bmod q} \bmod p \right) \bmod q.$$

Dies liefert nun genau das DSA-Verfahren, wie es im folgenden Abschnitt beschrieben wird.

6. Abschnitt: DSA - Digital Signature Algorithm

- Das NIST - National Institute of Standards and Technology der US-Regierung schlug im August 1991 einen Standard für digitale Signaturen - Digital Signature Standard (DSS) - vor, der inzwischen wiederholt aktualisiert wurde.
- Eine Aktualisierung steht in FIPS 186-4 und ist vom Juli 2013.
- In FIPS 186-5 (vom 3.2.2023) schreibt NIST: „This standard no longer approves the DSA for digial signature generation.“ (Weiterhin gültig: RSA-Signatur, ECDSA (Elliptic Curve Digital Signature Algorithmu, . . .)
- Das BSI schreibt in „Kryptographische Verfahren: Empfehlungen und Schlüssellängen“ (vom 2.2.2024) „Die Verwendung des Signaturverfahrens DSA wird aufgrund der geringen Verbreitung und der Abkündigung in [FIPS 186-5] in der vorliegenden Technischen Richtlinie nur noch bis 2029 empfohlen.“

Trotzdem soll hier DSA vorgestellt werden.

DSA - Digital Signature Algorithm:

- **Schlüsselerzeugung:** Jeder Teilnehmer A hat Folgendes zu tun:
  - A wählt Parameter l und n aus folgender Menge:

$$(l, n) \in \{(1024, 160), (2048, 224), (2048, 256), (3072, 256)\}.$$

  - A wählt sich eine Primzahl q mit $2^{n-1} < q < 2^n$, d.h. q hat n Bits.
  - A sucht sich eine weitere Primzahl p mit $p \equiv 1 \bmod q$ und $2^{l-1} < p < 2^l$, d.h. p hat l Bits.
  - A bestimmt ein g mit $1 < g < p$ und $\operatorname{ord}_p(g) = q$. (Ist $g_0 \in \mathbb{Z}$ mit $g_0^{\frac{p-1}{q}} \not\equiv 0, 1 \bmod p$, so liefert $g = g_0^{\frac{p-1}{q}} \bmod p$ ein Zahl mit dieser Eigenschaft.)
  - A wählt eine (zuverlässige) Hashfunktion H. (Hier soll angenommen werden, dass die Hashwerte mindestens n Bits haben. Sonst muss man die Hashwerte „kürzen“.)
  - Die Zahlen p, q, g kann eine ganze Benutzergruppe gemeinsam haben, weswegen auf einen Index, der die Abhängigkeit von A zeigt, verzichtet wird.)
  - A wählt zufällig eine Zahl e_A mit $0 < e_A < q$ und berechnet $f_A = g^{e_A} \bmod p_A$. Der öffentliche Schlüssel von A ist f_A zusammen mit p, q und g, der private Schlüssel von A ist e_A.

- **Signieren einer Nachricht bzw. eines Dokuments:** Wie signiert A eine Nachricht M?
  - A berechnet den Hashwert $h = H(M)$ der Nachricht M.
  - A wählt eine Zufallszahl z mit $0 < z < q$. (Die Zufallszahl muss geheim bleiben.)
  - A berechnet
    $$b = (g^z \bmod p) \bmod q,$$
    d.h. zuerst $\tilde{b}$ mit $1 \leq \tilde{b} \leq p-1$ und $\tilde{b} \equiv g^z \bmod p$, sodann b mit $0 \leq b \leq q-1$ und $b \equiv \tilde{b} \bmod q$.
  - Dann berechnet A sich c mit
    $$c \equiv \frac{1}{z}(h + be_A) \bmod q.$$
  - Ist $b = 0$ oder $c = 0$ wählt A eine andere Zufallszahl z. Dieser Fall ist allerdings sehr unwahrscheinlich.
  - Die DSA-Signatur von A der Nachricht M ist dann das Zahlenpaar (b, c) (mit $0 \leq b, c \leq q-1$). Kurz:
    $$b = (g^z \bmod p) \bmod q \quad \text{und} \quad c = \frac{1}{z}(h + be_A) \bmod q.$$
- **Signaturüberprüfung:** Wie kann ein Empfänger B sehen, dass die Unterschrift (b, c) der Nachricht M tatsächlich von A stammt?
  - B besorgt sich den öffentlichen Schlüssel (p, q, g, f_A) von A.
  - B berechnet den Hashwert $h = H(M)$ der Nachricht M.
  - Ist eine der Bedingungen $0 < b < q$, $0 < c < q$ verletzt, wird die Unterschrift nicht akzeptiert.
  - Dann berechnet B
    $$u_1 = c^{-1}h \bmod q \quad \text{und} \quad u_2 = c^{-1}b \bmod q$$
    und damit
    $$v = (g^{u_1} f_A^{u_2} \bmod p) \bmod q.$$
  - Gilt nun $v = b$, so akzeptiert B die Unterschrift. Ist $v \neq b$, so wird die Unterschrift nicht als gültig anerkannt.

Bemerkungen:

- Warum gilt $v = b$, wenn alles richtig gelaufen ist? Man hat

$$u_1 + e_A u_2 \equiv c^{-1}h + c^{-1}be_A = c^{-1}(h + be_A) \equiv z \bmod q$$

und damit $(\mathrm{ord}\,(g) = q)$

$$g^z \equiv g^{u_1 + e_A u_2} \equiv g^{u_1} g^{e_A u_2} \equiv g^{u_1} f_A^{u_2} \bmod p,$$

was dann sofort $b = v$ liefert.

- Da g Ordnung q in $(\mathbb{Z}/p\mathbb{Z})^*$, ist klar, dass gilt

$$x \equiv y \bmod q \quad \Longleftrightarrow \quad g^x \equiv g^y \bmod p.$$

Ein Ausdruck wie $(g^{u_1} f^{u_2} \bmod p) \bmod q$ ist allerdings nur dann sinnvoll, wenn man genau weiß, welchen Repräsentanten modulo p man zunächst nehmen muss, hier den zwischen 0 und $p - 1$.

- Ein Vorteil dieses Signaturverfahrens ist, dass die Signatur (b, c) recht kurz ist: Im Fall $(l, n) = (1024, 160)$ hat man 320 Bits, was einer 80-stelligen Hexadezimalzahl entspricht.

Beispiel: Wir wollen einen DSA-Schlüssel mit den Parametern $(l, n) = (2048, 256)$ erstellen. Als Primzahl q mit 256 Bits wählen wir

$$\begin{aligned} q &= 2^{256} - 189 = \\ &= 115792089237316195423570985008687907853269984665640564039457584007913129639747. \end{aligned}$$

Nun wählen wir eine 2048-Bit-Primzahl p mit $p \equiv 1 \bmod q$:

$$\begin{aligned} p &= 2^{2048} - 387q - 1628150074335205280 = \\ &= 32317006071311007300714876688669951960444102669715484032130345427524655138867890 \\ &\quad 89319720141152291346368871796092189801949411955915049092109508815238644828312063 \\ &\quad 08773673009960917501977503896521067960576383840675682767922186426197561618380943 \\ &\quad 38476170470581645852036305042887575891541065808607552399123930385521914333389668 \\ &\quad 34242068497478656456949485617603532632205807780565933102619270846031415025859286 \\ &\quad 41771167259436037184618573575983511523016459044036976132332872312271256847108202 \\ &\quad 09725157101726931323469678542580656697935045997268352998593403986631548069706621 \\ &\quad 63093707100926542983441300205386492346921439860409044328 7. \end{aligned}$$

Mit

$$\begin{aligned} g &= 2^{\frac{p-1}{q}} \bmod p = \\ &= 13340502274063705829532404390593100236939283273101880740483962836166896750453332 \\ &\quad 74956146174827690044806580576454049517155531431585485336121367869861499662074016 \\ &\quad 26468550604769785397820995962847440597700422749651505138553546786042250731986900 \\ &\quad 14875716554894210197497845888328667877534598166950779319914738511907374779753122 \\ &\quad 71405252178720296338130120635659500098223836954519480964979363457835592998017956 \\ &\quad 34862954090214646479550423510170041200258217183662927096301076119981205165731766 \\ &\quad 23632740260039246612533216127992613704369614213663975060858604908043704952502019 \\ &\quad 17228932076374004311511339269716105055818174210288745 2576 \end{aligned}$$

erhält man ein Element der Ordnung q. Wir wählen zufällig e mit $1 < e < q$ und berechnen $f = g^e \bmod p$:

$$\begin{aligned} e &= 252284163794657618775625294467563627045100669073155065792345747291457689161 82, \\ f &= g^e \bmod p = \\ &= 23915546739763525259384007533187869693474281706026714725759465718719824210463398 \\ &\quad 64137044664867330735563434177131278110802128519096255788862032268186815371870461 \\ &\quad 79008827307900882169804134346348298737400176942535082352800366862582532301205304 \\ &\quad 90667194060601445826745173484084766632136622274565393671815275222733596370153035 \\ &\quad 39563241565567808759923969554699332344713564100058534841986973348316388927199835 \\ &\quad 46351885601028116628541857210313513025761527463697243298028147633592141215132804 \\ &\quad 59027006081512277714728427646511611076135737455612292103955920843566667665882641 \\ &\quad 48759180566974887664611540410299913749512009328615946 4673. \end{aligned}$$

Als Hashfunktion wählen wir SHA-256.

Wir wollen `DSA-Signatur` unterschreiben. Der SHA-256-Hashwert ist

$$\begin{aligned} h &= (1515866a46d5979402fe0572df1534c7a3f886a218fdead2bb10afa979cdc993)_{16} = \\ &= 95366013078334839914818586014838642299017710218483557324757322748699741904 83. \end{aligned}$$

Nun wählen wir eine zufällige Zahl z mit $1 < z < q$

$$z = 252284163794657618775625294467563627045100669073155065792345747291457689161 82$$

und berechnen $b = (g^z \bmod p) \bmod q$ und $c = \frac{1}{z}(h + be) \bmod q$:

$$\begin{aligned} b &= 731983834113169396748143950089985624811726711708049428630809976369360194865 65, \\ c &= 597120168065974021054475862448093402504432656297236131831040199595640934725 26. \end{aligned}$$

(b, c) ist nun die zugehörige DSA-Signatur. Um die Gültigkeit der Signatur zu überprüfen berechnen wir

$$\begin{aligned} u_1 &= \frac{1}{c} h \bmod q = \\ &= 619338138073261980132565235690494303468217516768075407424572362532506005034532 \\ u_2 &= \frac{1}{c} b \bmod q = \\ &= 520466730397094113606039885078212200671625321880957715968651794742847453828 31 \\ v &= (g^{u_1} f^{u_2} \bmod p) \bmod q = \\ &= 731983834113169396748143950089985624811726711708049428630809976369360194865 65 \end{aligned}$$

Da nun $b = v$ gilt, ist die Signatur gültig.

Bemerkungen:

- Was passiert, wenn zwei Nachrichten zufällig mit der gleichen Zufallszahl z signiert werden? Man hat dann $b_i \equiv (g^z \bmod p) \bmod q$, also $b_1 = b_2 = b$. Weiter gilt $c_i z \equiv h_i + b_i e_A \bmod q$, ausgeschrieben:

$$\begin{aligned} c_1 \cdot z - b \cdot e_A &\equiv h_1 \bmod q, \\ c_2 \cdot z - b \cdot e_A &\equiv h_2 \bmod q. \end{aligned}$$

  Kennt jemand die Hashwerte h_1, h_2 und die Signaturen (b, c_1), (b, c_2), so ist dies ein lineares Gleichungssystem mit den 2 Unbekannten z und e, woraus man im Fall $c_1 \neq c_2$ sofort z und den privaten Schlüssel e_A berechnen kann. Es ist also wichtig, dass der Zufallszahlengenerator gut ist.
- Was passiert, wenn zwei Nachrichten mit Hashwerten h_1 und h_2 Signaturen (b, c_1) und (b, c_2) mit gleicher erster Komponente haben? Natürlich kann man dann probieren, ob die gleiche Zufallszahl z benutzt wurde, wie oben. Aber das muss nicht sein. Wir haben damit nur zwei Gleichungen mit drei Unbekannten e_A, z_1, z_2:

$$c_1 z_1 \equiv h_1 + b e_A \bmod q, \quad c_2 z_2 \equiv h_2 + b e_A \bmod q,$$

  was noch keine Lösung liefert. (Dies ist ganz anders als bei der ElGamal-Signatur.) Es ist auch nicht klar, wie man zu gegebenem b ein z findet mit $b = (g^z \bmod p) \bmod q$.

- Die Signatur kann man fälschen, wenn man e_A, also den Logarithmus von f_A zur Basis g modulo p berechnen kann. Nun scheint es so zu sein, dass die Rechenschwierigkeit und damit die Sicherheit in erster Linie von der Größe von p und nicht hauptsächlich von der Größe von $q = \operatorname{ord}_p(g)$ abhängt.
- Um zu testen, ob die Signatur (b, c) mit $0 < b, c < q$ für das Dokument M mit $h = H(M)$ von A stammt, wird getestet, ob

$$v = b \quad \text{mit} \quad v = (g^{u_1} f_A^{u_2} \bmod p) \bmod q \quad \text{mit} \quad u_1 \equiv \frac{h}{c} \bmod q, \quad u_2 \equiv \frac{b}{c} \text{ mo}$$

gilt. Es ist leicht, Lösungen (h, b, c) dieser Gleichung zu finden: Man wählt Zahlen x, y mit $\operatorname{ggT}(y, q) = 1$ und setzt

$$b = (g^x f_A^y \bmod p) \bmod q, \quad c \equiv \frac{b}{y} \bmod q, \quad h \equiv xc \equiv \frac{xb}{y} \bmod q,$$

denn dann ist

$$x \equiv \frac{h}{c} \bmod q, \quad y \equiv \frac{b}{c} \bmod q,$$

sodass die Gleichung mit $u_1 = x$, $u_2 = y$ erfüllt ist. Für viele Werte h können wir also die Unterschrift fälschen. Da aber die Hash-Funktion H eine Einwegfunktion sein soll, können wir dazu kein Dokument M mit $h = H(M)$ finden.

- A habe das Dokument M mit Hashwert $h = H(M)$ und Signatur (b, c) unterschrieben. Mit

$$u_1 \equiv \frac{h}{c} \bmod q, \quad u_2 \equiv \frac{b}{c} \bmod q, \quad v = (g^{u_1} f_A^{u_2} \bmod p) \bmod q \quad \text{gilt dann} \quad v =$$

Würden wir nur $v = (b \bmod p) \bmod q$ testen und keine Bedingung $0 < b < q$ stellen, so könnten wir leicht die Unterschrift fälschen. Sei jetzt M' ein weiteres Dokument mit Hashwert $h' = H(M')$. Wir wählen x, b', c' mit

$$x \equiv \frac{h'}{h} \bmod q, \quad b' \equiv \begin{cases} bx \bmod q, \\ b \bmod p, \end{cases} \qquad c' \equiv cx \bmod q$$

und erhalten dann

$$u_1' \equiv \frac{h'}{c'} \equiv \frac{h}{c} \equiv u_1 \bmod q, \quad u_2' \equiv \frac{b'}{c'} \equiv \frac{b}{c} \equiv u_2 \bmod q,$$

was sofort $v' = v$ liefert. Es folgt

$$(b' \bmod p) \bmod q = (b \bmod p) \bmod q = v = v'.$$

???

Beispiel: Um einen Eindruck von der Unregelmäßigkeit der Funktion

$$b(z) \equiv (g^z \bmod p) \bmod q$$

zu geben, betrachten wir die Größen

$$q = 19, \quad p = 9767, \quad g_0 = 2, \quad g = 2534$$

und listen sämtliche Urbilder auf (Es ist $b(z) = b(z \bmod 19)$):

$$\begin{aligned}
b^{-1}(0) &= \{2\},\\
b^{-1}(1) &= \{0, 16, 17\},\\
b^{-1}(2) &= \{5, 15\},\\
b^{-1}(3) &= \{3\},\\
b^{-1}(4) &= \emptyset,\\
b^{-1}(5) &= \{4, 13\},\\
b^{-1}(6) &= \{14\},\\
b^{-1}(7) &= \{1, 6, 8\},\\
b^{-1}(8) &= \emptyset,\\
b^{-1}(9) &= \emptyset,\\
b^{-1}(10) &= \emptyset,\\
b^{-1}(11) &= \emptyset,\\
b^{-1}(12) &= \emptyset,\\
b^{-1}(13) &= \emptyset,\\
b^{-1}(14) &= \{10, 12, 18\},\\
b^{-1}(15) &= \emptyset,\\
b^{-1}(16) &= \{7\},\\
b^{-1}(17) &= \emptyset,\\
b^{-1}(18) &= \{9, 11\}.
\end{aligned}$$

7. Abschnitt: Wie verallgemeinert man die ElGamal-Signatur?

Überlegungen:

- Wie kann man die ElGamal-Signatur verallgemeinern? Mit einer Zufallszahl z und dem Hashwert $h = H(M)$ haben wir

$$b = g^z \bmod p \quad \text{und dann} \quad c \equiv \frac{1}{z}(h - be_A) \bmod (p-1)$$

gebildet. Dann gilt $g^h \equiv f_A^b b^c \bmod p$. Das Problem ist, dass b einmal modulo p, das andere Mal modulo $p-1$ betrachtet wird. Bei der Betrachtung modulo p rechnet man in der multiplikativen Gruppe $(\mathbb{Z}/p\mathbb{Z})^*$, bei der Betrachtung modulo $p-1$ in der additiven zyklischen Gruppe $\mathbb{Z}/(p-1)\mathbb{Z}$. Wir wollen diese Funktionen von b unterscheiden und führen daher eine (hier künstliche) Abbildung ℓ ein mit

$$\ell : (\mathbb{Z}/p\mathbb{Z})^* \to \mathbb{Z}, \quad \overline{b} \mapsto b \text{ mit } 1 \leq b \leq p-1.$$

Dann haben wir jetzt

$$b = g^z \bmod p, \quad c = \frac{1}{z}(h - \ell(b)e_A) \bmod (p-1).$$

Da wir im Exponenten modulo $p-1$ rechnen dürfen, folgt

$$f_A^{\ell(b)} \cdot b^c \equiv (g^{e_A})^{\ell(b)} \cdot (g^z)^c \equiv g^{\ell(b)e_A + zc} \equiv g^h \bmod p.$$

- Sei G eine multiplikativ geschriebene Gruppe $g \in G$ mit $\operatorname{ord}(g) \mid n$. Außerdem sei $\ell : G \to \mathbb{Z}$ eine Abbildung.
  - Als privater Schlüssel wird eine Zahl e_A mit $0 \leq e_A \leq n-1$ verwendet, der zugehörige öffentliche Schlüssel ist dann $f_A = g^{e_A}$.
  - Zum Signieren des Dokuments M wird der Hashwert $h = H(M)$ berechnet, eine Zufallszahl z (mit $\operatorname{ggT}(z, n) = 1$) gewählt, dann

$$b = g^z \quad \text{und } c \equiv \frac{1}{z}(h(M) - \ell(b)e_A) \bmod n$$

    berechnet. Die Signatur ist dann (b, c).
  - Da man im Exponenten von g modulo n rechnen darf, folgt

$$f_A^{\ell(b)} \cdot b^c = (g^{e_A})^{\ell(b)} \cdot (g^z)^c = g^{\ell(b)e_A + zc} = g^h.$$

    Als Signatur-Test verwenden wir die Gleichung

$$g^h = f_A^{\ell(b)} \cdot b^c.$$

- Wir schreiben jetzt das Ganze für eine additive Gruppe G um. Ein Basiselement $P \in G$ werde gewählt mit $\operatorname{ord}(P) \mid n$. Außerdem gebe es eine Funktion $\ell : G \to \mathbb{Z}$.
  - Privater Schlüssel e_A mit $0 \leq e_A \leq n-1$. Öffentlicher Schlüssel $Q_A = e_A \cdot P \in G$.
  - Signieren des Dokuments M. Es wird eine Zufallszahl z (mit $\operatorname{ggT}(z, n) = 1$) gewählt, der Hashwert $h = H(M)$ berechnet, dann

    $$B = z \cdot P \in G, \quad c = \frac{1}{z}(h - \ell(B)e_A) \bmod n$$

    berechnet.
  - Es gilt

    $$\ell(B)\cdot Q_A + c\cdot B = \ell(B)\cdot e_A \cdot P + c\cdot z\cdot P = (\ell(B)e_A + zc)\cdot P = h\cdot P.$$

    Wir verwenden

    $$h \cdot P = \ell(B) \cdot Q_A + c \cdot B$$

    als Signaturtestgleichung. um die Signatur zu testen.
- Nach ein paar leichten Abänderungen erhält man diese Weise ECDSA - elliptic curve digital signature algorithm. (Da elliptische Kurven in der Vorlesung nicht behandelt wurden, wird auf eine genaue Ausführung verzichtet.)