KAPITEL 9

Digitale Signaturen

1. Einführung

Hier sind einige Eigenschaften, die eine eigenhändige Unterschrift hat bzw. haben sollte (entnommen aus B. Schneier, Angewandte Kryptographie, 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. Allgemeine Verfahren

Wir geben im folgenden einige mögliche Verfahren an:

Unterschreiben mit einem Public-Key-Datenverschlüsselungssystem:

(1) 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.
(2) A will eine Nachricht/Datei M an B übermitteln und B soll sicher sein, dass die Nachricht wirklich von A stammt.
   (a) A wendet f_A^{-1} auf M an und schickt $f_A^{-1}(M)$ an B.
   (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:

(1) Das beschriebene Verfahren ist natürlich nicht besonders effizient, wenn es sich um längere Nachrichten oder Dateien handelt.
(2) 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 Hash-Funktion:

(1) Man legt ein Public-Key-Verfahren mit öffentlichen Schlüsseln f_A zugrunde. Außerdem einigt man sich auf eine Hashfunktion h.
(2) A will eine Nachricht M mit Unterschrift an B senden.

(a) A berechnet den Hashwert $h(M)$, wendet f_A^{-1} darauf an, erhält also $f_A^{-1}(h(M))$.
(b) A schickt M und $f_A^{-1}(h(M))$ an B.
(c) 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) gibt es eine Variante, die getrennt zum Klartext eine Signatur erstellt (`gpg --clearsign datei`), die z.B. wie folgt aussehen kann:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

In diesem Kapitel der Vorlesung 'Kryptographie I' werden digitale
Signaturen behandelt.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.9 (GNU/Linux)

iEYEARECAAYFAkhze8QACgkQTUYI5o+0h3p32wCgg/SeAO7mY1PCUPKvpxYON7+z
CqsAn1Z45Q6GANJsNSFSo2PhLhkP1tUJ
=kxOv
-----END PGP SIGNATURE-----
```

Die nächste Variante:

Digitale Signatur mit Verschlüsselung:

(1) Wieder legen wir ein Public-Key-Kryptosystem mit öffentlichen Verschlüsselungsfunktionen f_A zugrunde.
(2) 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.
(3) 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-----
Version: GnuPG v2.0.9 (GNU/Linux)

hQEOAy+GDlkj9DUVEAP7BgPEThCSI+SCYFIqN7Zjw03l/3B5Vp/rZjMQjjuF0bGo
MlElw57I87emhKXmZLrDZYpqZ+S7LIRlWcixvZGOI1yYpYzvnKUa4nm2WA68kOmP
hyVDyhQw64cVmZjSWY7HachJ++Fk+7ZQVx1TSO2GEXQir01CsVBMrWTQMGRZ0WoE
AN2Q4U4lHpuIUAd82J38bsaJMnVVbgRw/SEMN/+LGZKiXc7A9eXF45zDPeVVlD8s
KdF3veQkwu1pi7BvMFaKnR/toyFRtPdYykshep1BPUHUlN3W649b6FcSWDdnS5zc
zNBPlsEuHdwl94at1i0d8sWF6C+VKV1GWDRuOHsGOsKKOsAlAVvEAWx2RMfWK+Y5
iE11mE2qksoMfJv5YQWL7q8fPznCmISHUKKw5G4WfdSLNtfmDezTymajiQLx7JOY
PyMJJJgXyWxO3QGnvsmBb3U6ykPoCc4tZZyP8Zz701tY+GhwfmOJrbMwLEA4FAMA
SLOZSmH3n2AJ5vIWvjbxoS6FOILndthCYuEIZfJNrGwfmQe8cB0ki/mvYeTbk4DT
FGUUlO1n65NuO9bhlaaQ8EZZKlAzhcjnNSy/73p4dIXSa1aCdOZFTpwD5j7EFCny
SqMM4SXInVT7tbFCES4WGl83uXMhagBsuw==
=b8fl
-----END PGP MESSAGE-----
```

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

3. Die RSA-Signatur

Die RSA-Signatur:

(1) **Schlüsselerzeugung:**
   (a) Jeder Teilnehmer A besorgt sich einen öffentlichen und einen privaten RSA-Schlüssel (N_A, e_A) und (N_A, d_A).
   (b) Außerdem muss A sich auf eine Hashfunktion H festlegen.

(2) **Signieren einer Nachricht M:**
   (a) A bestimmt den Hashwert
   $$h = H(M)$$
   der Nachricht M.
   (b) 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.

(3) **Signaturüberprüfung:** Erhält jemand die Nachricht M und die zugehörige Signatur s, so geht er so vor:
   (a) Er bestimmt den Hashwert
   $$h = H(M)$$
   der Nachricht M.
   (b) 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:

(1) 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.

(2) Wie kann ein Angreifer die Unterschrift von A fälschen? Er braucht, eine Zahl $\widetilde{s}$, das die Signaturtestgleichung
$$h \equiv \widetilde{s}^{e_A} \bmod N_A$$
erfüllt. Dies ist aber genau das RSA-Problem, wobei hier $\widetilde{s}$ dem Klartext und h dem Chiffretext entspricht. Wenn die RSA-Parameter passend gewählt sind, sollte ein Angreifer keine Zahl $\widetilde{s}$ 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

$N = 114213506014571077565586332408463598421487556397007695495062582712427758620336881480839412413695815 1,$
$e = 65537,$
$d = 113638404057130113597192242204536153047437921936907933639164255329034999684304245367031236480082210 5.$

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

$s = 114110934925547531289487525537403817378868189129919071264192931746497625724599659628931783603742461 8,$

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

$$\begin{aligned} h &= (\text{ac1ab5b1d953e429505a6af13a02f4c00eaa4b3ff90cd3cc67a85fda1527f0a1})_{16} = \\ &= 778450019908926641233365319656511299755335739054989365055347730777947017012 81 \end{aligned}$$

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. Die ElGamal-Signatur

Die ElGamal-Signatur:

(1) **Schlüsselerzeugung:** Jeder Teilnehmer A hat Folgendes zu tun:
   (a) Jeder Teilnehmer A wählt sich eine (große) Primzahl p_A und dazu eine Primitivwurzel g_A modulo p_A mit $3 \leq g_A \leq p_A - 2$ (und $g_A \nmid p_A - 1$). (Die Zahlen p_A und g_A können auch von anderen Teilnehmern verwendet werden.)
   (b) A wählt geheim und zufällig eine Zahl e_A mit $2 \leq e_A \leq p_A - 2$. Dann berechnet A die Zahl $f_A = g_A^{e_A} \bmod p_A$. Der öffentliche Schlüssel von A ist das Tripel (p_A, g_A, f_A), der geheime Schlüssel ist der Exponent e_A.
   (c) Dazu muss noch eine kryptographische Hashfunktion H festgelegt werden.

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

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

   wobei $0 \leq b \leq p_A - 1$ und $0 \leq c \leq p_A - 2$ gilt.
   Die Signatur/Unterschrift von A für die Nachricht M ist das Paar (b, c).

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

$$1 \leq b \leq p_A - 1 \quad \text{und} \quad g_A^h \equiv f_A^b b^c \bmod p_A$$

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

Bemerkungen:

(1) Wurde die Signatur richtig erstellt, also

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

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

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

d.h. auch die zweite Testbedingung ist erfüllt. (Später werden wir überlegen, warum diese Bedingungen auch sinnvoll sind.)

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

$$f_A^b b^c \equiv g_A^h \bmod p_A$$

erfüllt, nämlich

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

Denn es ist

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

sodass sich ergibt

$$f_A^b b^c \equiv f_A^0 g_A^h \equiv g_A^h \bmod p_A,$$

da man im Exponenten modulo $p_A - 1$, in der Basis modulo p_A 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:

$$\begin{aligned} p &= 168600185061638396245742195839459005247443255833991318598501882715850529490973 47, \\ g &= 5, \\ e &= 369169129043646567067125161792381129791709565880436594040003898112036131894112 8, \\ f &= 124332664308681423601810239583804090220703658325449996385213524664828449463089 15. \end{aligned}$$

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

$$\begin{aligned} h &= (17\text{adf8ade0072f41f86616c1a0d55fc880ed0da549d24cf8785abbafa95b97a3353b670346799d1ab \\ &\quad \text{e67733ca4a2b850})_{16} = \\ &= 364462028180302507889265834775526384671148403138359535732315409573993971482686 48 \\ &\quad 797369739510859285000306670784164 64, \end{aligned}$$

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

$$z = 135963229140014016031133419106260776727845546943263278997040754502529745428285 7$$

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

$$\begin{aligned} b &= 937665324758983664079535901031019504241628102681467219403676401836615486320 3985, \\ c &= 551213266400382076120235194517664950027283154533856692218091385088612094968 8410. \end{aligned}$$

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?

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

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

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 $g_A^{e_A} \equiv f_A \bmod p_A$ erhalten.

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

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

Dies ist eine Gleichung mit den beiden Unbekannten e_A und z. Damit ist klar: z muss auf jeden Fall geheim gehalten werden!

(3) 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 $p_A - 1$) die gleiche Zufallszahl z vor.

(a) Nach Definition erhalten wir dann ein Gleichungssystem

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

Subtraktion liefert

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

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

$$z_0 \equiv \frac{h_1 - h_2}{c_1 - c_2} \bmod \frac{p_A - 1}{\mathrm{ggT}(p_A - 1, c_1 - c_2)},$$

so sind

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

die anderen Lösungen.

(b) Nun probiert man durch, für welches z_i die Gleichung $b \equiv g_A^{z_i} \bmod p_A$ gilt. Dies ist dann das richtige z.

(c) Nun bleibt die Gleichung

$$be_A \equiv h_1 - c_1 z \bmod (p_A - 1)$$

mit der Unbekannten e_A, die man wie zuvor untersuchen kann. Die Gleichung

$$bx \equiv h_1 - c_1 z \bmod (p_A - 1)$$

hat $\mathrm{ggT}(p_A - 1, b)$ Lösungen modulo $p_A - 1$. 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.

(4) Diese Konstruktion wird nicht mehr benötigt. Lässt man die Testbedingung $1 \leq b \leq p_A - 1$ weg, so lassen sich leicht Unterschriften fälschen, wenn man bereits eine Unterschrift (b, c) von A (eines Dokuments M mit Hashwert h) kennt:

(a) Es gibt ein z mit $\mathrm{ggT}(z, p_A - 1) = 1$ und

$$b = g_A^z \bmod p_A, \quad c = \frac{1}{z}(h - be_A) \bmod (p_A - 1).$$

(b) Sei M' ein Dokument mit Hashwert h'. Wir nehmen an, es dass ein u existiert mit $h' \equiv hu \bmod (p_A - 1)$. Wir wollen (b', c') suchen, sodass die Gleichung

$$g_A^{h'} \equiv f_A^{b'} b'^{c'} \bmod p_A$$

erfüllt ist.

(c) Bei einer Gleichung wie eben darf man die Zahlen in der Basis modulo p_A, im Exponenten modulo $p_A - 1$ abändern. Wir schränken die Möglichkeiten für b' ein, indem wir nur solche mit $b' \equiv b \bmod p_A$ betrachten, d.h. die die Form $b' = b + kp_A$ mit $k \in \mathbb{N}_0$ haben. Dann haben wir die Äquivalenzen:

$$\begin{aligned} g_A^{h'} \equiv f_A^{b'} b'^{c'} \bmod p_A \quad &\iff \quad g_A^{h'} \equiv f_A^{b'} b^{c'} \bmod p_A \quad \iff \quad g_A^{h'} \equiv g_A^{e_A b' + zc'} \bmod p_A \\ &\iff \quad h' \equiv e_A b' + zc' \bmod (p_A - 1). \end{aligned}$$

Nun hatten wir vorausgesetzt, dass $h' \equiv hu \bmod (p_A - 1)$ gilt. Da auch noch $h \equiv e_A b + zc \bmod (p_A - 1)$ gilt, sehen wir, dass bei Wahl von

$$b' \equiv bu \bmod (p_A - 1), \quad c' \equiv cu \bmod (p_A - 1)$$

die Gleichung

$$g_A^{h'} \equiv f_A^{b'} b'^{c'} \bmod p_A$$

erfüllt ist.

(d) Damit haben wir an b' die beiden Forderungen

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

gestellt. Mit dem chinesischen Restsatz lässt sich dann ein solches b' finden. Allerdings wird im Allgemeinen $b' \geq p_A$ sein. Dies zeigt, dass man auf die Forderung $1 \leq b \leq p_A - 1$ nicht verzichten kann.

(e) Eliminiert man den chinesischen Restsatz, so erhält man die explizite Darstellung

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

(5) 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 ganze Zahlen 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 Hash-Funktion ist.

(6) In einem nachfolgendem Satz wird gezeigt, dass es im Fall $g_A \mid p_A - 1$ sehr oft möglich ist, die Signatur zu fälschen. Die Bedingung $g_A \mid p_A - 1$ ist natürlich für $g_A = 2$ erfüllt. Daher muss man $g_A \mid p_A - 1$ ausschließen, insbesondere also $g_A = 2$.

(7) 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. Wir skizzieren dies in nachfolgendem Beispiel, weil die nötige mathematische Theorie in der Vorlesung nicht behandelt wurde.

Beispiel: (Nguyen-Angriff) Wir starten mit

$$\begin{aligned} p &= 23458236458623485623894562837465289374658923648957265738251286794345739475345401, \\ g &= 19, \\ e &= 63734653423421159 27 \approx p^{0.24}, \\ f &= 11507488626769142308413284037555938190497252706592087846017214552305987359855232. \end{aligned}$$

g ist eine Primitivwurzel modulo p, (p, g, f) ein öffentlicher ElGamal-Schlüssel, e der zugehörige Exponent. Wir wollen „`Kryptographie`“ signieren unter Verwendung der Hashfunktion SHA-256. Der Hashwert ist

$$h = 30584352687306778985061702547129159697739851556770210393195060469523595677885.$$

Als Zufallszahl wählen wir

$$z = 5334756384753193469 \approx p^{0.24}$$

(mit $\mathrm{ggT}(p-1, z) = 1$) und berechnen dann

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

mit dem Ergebnis

$$\begin{aligned} b &= 10292382095472353680527348817496378768110853897270130461749420516809493619965431, \\ c &= 97405931348055240911025136743686994149988743810589939059395824812750808252318 92. \end{aligned}$$

Die Signatur ist also (b, c).
Öffentlich bekannt sind möglicherweise $p, g, f, h, b, c)$. Wir definieren die Matrix

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

und betrachten die Menge

$$\Lambda = \{zM \in \mathbb{R}^4 : z \in \mathbb{Z}^4\},$$

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 $\mathbb{R}^4$). Aus $cz \equiv h - be \bmod (p-1)$, also $be + cz - h \equiv 0 \bmod (p-1)$ folgt die Existenz einer Zahl $m \in \mathbb{Z}$ mit

$$be + cz - h + (p-1)m = 0,$$

also ist

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

Dies ist ein recht kurzer Gittervektor. Man findet ihn durch sogenannte LLL-Reduktion als ersten Vektor - bis aufs Vorzeichen - einer sogenannten LLL-reduzierten Gitterbasis. (Sage bestimmt zur Matrix M mit `M.LLL()` eine LLL-reduzierte Gitterbasis.) Hier findet man

$$(-6373465342342115927, -5334756384753193469, 1, 0),$$

sodass man (bis aufs Vorzeichen) e und z bestimmt hat.

Der folgende Satz zeigt, warum die ElGamal-Signatur im Fall $g_A \mid p_A - 1$, insbesondere im Fall $g_A = 2$ nicht sicher ist.

Satz. *Sei (p_A, g_A, f_A) ein öffentlicher ElGamal-Signatur-Schlüssel, e_A der zugehörige geheime Schlüssel. Wir setzen voraus, dass außerdem*

$$g_A \mid p_A - 1$$

gilt. Wir definieren

$$b = \frac{p_A - 1}{g_A}.$$

Es gibt ein $\widetilde{e} \in \{0, 1, \ldots, g_A - 1\}$ mit

$$f_A^b \equiv g_A^{b\widetilde{e}} \bmod p_A.$$

Für beliebiges $h \in \mathbb{Z}$ definieren wir

$$c \equiv \frac{p_A - 3}{2}(h - b\widetilde{e}) \bmod (p_A - 1).$$

Dann gilt

$$f_A^b b^c \equiv g_A^h \bmod p_A \quad \Longleftrightarrow \quad p_A \equiv 1 \bmod 4 \text{ oder } (p_A \equiv 3 \bmod 4 \text{ und } h \equiv b\widetilde{e} \bmod 2).$$

Ist die Bedingung erfüllt, so haben wir zum Hashwert h die gültige ElGamal-Signatur (b, c).

Beweis:

(1) Aus $g_A b = p_A - 1$ folgt $g_A b \equiv -1 \bmod p_A$. Da g_A Primitivwurzel sein sollte, gilt $g_A^{\frac{p_A-1}{2}} \equiv -1 \bmod p_A$ und damit

$$g_A b \equiv g_A \cdot g_A^{\frac{p_A-1}{2}} \bmod p_A, \quad \text{also} \quad b \equiv g_A^{\frac{p_A-3}{2}} \bmod p_A.$$

(2) Sei e_A mit $f_A \equiv g_A^{e_A} \bmod p_A$. Für $\widetilde{e} \in \mathbb{Z}$ gilt

$$f_A^b \equiv g_A^{b\widetilde{e}} \bmod p \quad \Longleftrightarrow \quad g_A^{e_A b} \equiv g_A^{b\widetilde{e}} \bmod p_A \quad \Longleftrightarrow \quad e_A b \equiv b\widetilde{e} \bmod p_A - 1 \quad \Longleftrightarrow$$
$$\Longleftrightarrow \quad e_A \equiv \widetilde{e} \bmod g_A.$$

Wählt man also $\widetilde{e} \in \{0, 1, \ldots, g_A - 1\}$ mit $\widetilde{e} \equiv e_A \bmod g_A$, so folgt die Behauptung.

(3) Es gilt

$$f_A^b b^c \equiv g_A^{b\widetilde{e}} (g_A^{\frac{p_A-3}{2}})^c \equiv g_A^{b\widetilde{e}+\frac{p_A-3}{2}c} \equiv g_A^{b\widetilde{e}+(\frac{p_A-3}{2})^2(h-b\widetilde{e})} \bmod p_A.$$

Daher:

$$\begin{aligned}
f_A^b b^c \equiv g_A^h \bmod p_A \quad &\Longleftrightarrow \quad g_A^{b\widetilde{e}+(\frac{p_A-3}{2})^2(h-b\widetilde{e})} \equiv g_A^h \bmod p_A \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad b\widetilde{e} + (\frac{p_A-3}{2})^2(h-b\widetilde{e}) \equiv h \bmod (p_A-1) \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad (p_A-1) \mid \left((\frac{p_A-3}{2})^2 - 1\right)(h-b\widetilde{e}) \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad (p_A-1) \mid \frac{(p_A-1)(p_A-5)}{4}(h-b\widetilde{e}) \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad 4(p_A-1) \mid (p_A-1)(p_A-5)(h-b\widetilde{e}) \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad 4 \mid (p_A-5)(h-b\widetilde{e}) \quad \Longleftrightarrow \\
&\Longleftrightarrow \quad (p_A \equiv 1 \bmod 4) \text{ oder } (p_A \equiv 3 \bmod 4 \text{ und } h \equiv b\widetilde{e} \bmod 2).
\end{aligned}$$

Damit ist die Behauptung bewiesen. ■

Bemerkung: Ist $g_A = 2$, also 2 eine Primitivwurzel modulo p_A, so ist $g_A \mid (p_A - 1)$ trivialerweise erfüllt. Im Fall $p_A \equiv 1 \bmod 4$ kann man dann zu jedem Hashwert eine Signatur erstellen, im Fall $p_A \equiv 3 \bmod 4$ für die Hälfte der Hashwerte. Daher muss man $g_A = 2$ bei der ElGamal-Signatur ausschließen.

5. Von der ElGamal-Signatur zu DSA

Überlegungen:

(1) 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.

(2) 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:

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

(3) 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.

(4) Bisher haben wir angenommen, dass g eine Primitivwurzel modulo p ist, d.h. dass $\mathrm{ord}_p(g) = p - 1$ 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. $\mathrm{ord}_p(g) = q$. 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:

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

(5) 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.

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

(6) 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. 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 und dessen letzte Aktualisierung vom Juli 2013 in FIPS PUB 186-4 ((Federal Information Processing Standards Publication) beschrieben wird. Dieses Signierverfahren soll im Folgenden vorgestellt werden.

DSA - Digital Signature Algorithm:

(1) **Schlüsselerzeugung:** Jeder Teilnehmer A hat Folgendes zu tun:
   (a) A wählt Parameter l und n aus folgender Menge:
   $$(l, n) \in \{(1024, 160), (2048, 224), (2048, 256), (3072, 256)\}.$$
   (b) A wählt sich eine Primzahl q mit $2^{n-1} < q < 2^n$, d.h. q hat n Bits.
   (c) 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.
   (d) 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.)
   (e) A wählt eine (zuverlässige) Hashfunktion H. (Hier soll angenommen werden, dass die Hashwerte mindestens n Bits haben. Sonst muss man die Hashwerte „kürzen“.)
   (f) Die Zahlen p, q, g kann eine ganze Benutzergruppe gemeinsam haben, weswegen auf einen Index, der die Abhängigkeit von A zeigt, verzichtet wird.)
   (g) 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.

(2) **Signieren einer Nachricht bzw. eines Dokuments:** Wie signiert A eine Nachricht M?
   (a) A berechnet den Hashwert $h = H(M)$ der Nachricht M.
   (b) A wählt eine Zufallszahl z mit $0 < z < q$. (Die Zufallszahl muss geheim bleiben.)
   (c) A berechnet
   $$b = (g^z \bmod p) \bmod q,$$
   d.h. zuerst $\widetilde{b}$ mit $1 \leq \widetilde{b} \leq p - 1$ und $\widetilde{b} \equiv g^z \bmod p$, sodann b mit $0 \leq b \leq q - 1$ und $b \equiv \widetilde{b} \bmod q$.
   (d) Dann berechnet A sich c mit
   $$c \equiv \frac{1}{z}(h + be_A) \bmod q.$$

(e) Ist $b = 0$ oder $c = 0$ wählt A eine andere Zufallszahl z. Dieser Fall ist allerdings sehr unwahrscheinlich.
(f) Die DSA-Signatur von A der Nachricht M ist dann das Zahlenpaar (b, c) (mit $0 \leq b, c \leq q-1$). Kurz:

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

(3) **Signaturüberprüfung:** Wie kann ein Empfänger B sehen, dass die Unterschrift (b, c) der Nachricht M tatsächlich von A stammt?
(a) B besorgt sich den öffentlichen Schlüssel (p, q, g, f_A) von A.
(b) B berechnet den Hashwert $h = H(M)$ der Nachricht M.
(c) Ist eine der Bedingungen $0 < b < q$, $0 < c < q$ verletzt, wird die Unterschrift nicht akzeptiert.
(d) Dann berechnet B

$$u_1 = c^{-1}h \bmod q \quad \text{und} \quad u_2 = c^{-1}b \bmod q$$

und damit

$$v = (g^{u_1} f_A^{u_2} \bmod p) \bmod q.$$

(e) Gilt nun $v = b$, so akzeptiert B die Unterschrift. Ist $v \neq b$, so wird die Unterschrift nicht als gültig anerkannt.

Bemerkungen:

(1) 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 ($\operatorname{ord}(g) = q$)

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

was dann sofort $b = v$ liefert.
(2) 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 $p-1$.
(3) 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:

(1) 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 $p \equiv 1 \bmod q$:

$$\begin{aligned} p &= 2^{2048} - 387q - 1628150074335205280 = \\ &= 32317006071311007300714876688669951960444102669715484032130345427524655138867890 \\ &\quad 89319720141152291346368871796092189801949411955915049092109508815238644828312063 \\ &\quad 08773673009960917501977503896521067960576383840675682767922186426197561618380943 \\ &\quad 38476170470581645852036305042887575891541065808607552399123930385521914333389668 \\ &\quad 34242068497478656456949485617603532632205807780565933102619270846031415025859286 \\ &\quad 41771167259436037184618573575983511523016459044036976132328723122712568471082020 \\ &\quad 97251571017269313234696785425806566979350459972683529985934039866315480697066621 \\ &\quad 63093707100926542983441300205386492346921439860409044328 7. \end{aligned}$$

Mit

$$\begin{aligned} g &= 2^{\frac{p-1}{q}} \bmod p = \\ &= 13340502274063705829532404390593100236939283273101880740483962836166896750453332 \\ &\quad 74956146174827690044806580576454049517155531431585485336121367869861499662074016 \\ &\quad 26468550604769785397820995962847440597700422749651505138553546786042250731986900 \\ &\quad 14875716554894210197497845888328667877534598166950779319914738511907374779753122 \\ &\quad 71405252178720296338130120635659500098223836954519480964979363457835592998017956 \\ &\quad 34862954090214646479550423510170041200258217183662927096301076119981205165731766 \\ &\quad 23632740260039246612533216127992613704369614213663975060858604908043704952502019 \\ &\quad 17228932076374004311511339269716105055818174210288745257 6 \end{aligned}$$

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

$$\begin{aligned} e &= 252284163794657618775625294467563627045100669073155065792345747291457689161 82, \\ f &= g^e \bmod p = \\ &= 23915546739763525259384007533187869693474281706026714725759465718719824210463398 \\ &\quad 64137044664867330735563434177131278110802128519096255788862032268186815371870461 \\ &\quad 79008827307900882169804134346348298737400176942535082352800366862582532301205304 \\ &\quad 90667194060601445826745173484084766632136622274565393671815275222733596370153035 \\ &\quad 39563241565567808759923969554699332344713564100058534841986973348316388927199835 \\ &\quad 46351885601028116628541857210313513025761527463697243298028147633592141215132804 \\ &\quad 59027006081512277714728427646511611076135737455612292103955920843566667665882641 \\ &\quad 48759180566974887664611540410299913749512009328615946467 3. \end{aligned}$$

Als Hashfunktion wählen wir SHA-256.

(2) 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 $c = \frac{1}{z}(h + be) \bmod q$:

$$\begin{aligned} b &= 73198383411316939674814395008998562481172671170804942863080997636936019486565, \\ c &= 59712016806597402105447586244809340250443265629723613183104019959564093472526. \end{aligned}$$

(b, c) ist nun die zugehörige DSA-Signatur.

(3) Um die Gültigkeit der Signatur zu überprüfen berechnen wir

$$\begin{aligned} u_1 &= \frac{1}{c}h \bmod q = \\ &= 61933813807326198013256523569049430346821751676807540742457236325506005034532 \\ u_2 &= \frac{1}{c}b \bmod q = \\ &= 52046673039709411360603988507821220067162532188095771596865179474284745382831 \\ v &= (g^{u_1} f^{u_2} \bmod p) \bmod q = \\ &= 73198383411316939674814395008998562481172671170804942863080997636936019486565 \end{aligned}$$

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

Bemerkungen:

(1) 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.

(2) 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 $b = (g^z \bmod p) \bmod q$.

(3) 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.

(4) 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

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

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

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

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

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. Wie verallgemeinert man die ElGamal-Signatur?

Überlegungen:

(1) 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 $\mathbb{F}_p^*$, 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

$$\ell : \mathbb{F}_p^* \to \mathbb{Z}, \quad \overline{b} \mapsto b \text{ mit } 1 \le r \le p-1.$$

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

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

(2) Sei G eine multiplikativ geschriebene Gruppe $g \in G$ mit $\operatorname{ord}(g) \mid n$. Außerdem sei $\ell : G \to \mathbb{Z}$ eine Abbildung.
   (a) Als privater Schlüssel wird eine Zahl e_A mit $0 \le e_A \le n-1$ verwendet, der zugehörige öffentliche Schlüssel ist dann $f_A = g^{e_A}$.
   (b) 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).

(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

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

(3) 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 $\ell : G \to \mathbb{Z}$.
   (a) Privater Schlüssel e_A mit $0 \leq e_A \leq n-1$. Öffentlicher Schlüssel $Q_A = e_A \cdot P \in G$.
   (b) 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.
   (c) 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.

(4) 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.)