Vorlesung Nr. 5

Kryptologie II - Datum: 15.10.2018

- Modulare Arithmetik
- Mathematische Grundlagen
- Euklidischer Algorithmus
- Kleiner Satz von Fermat
- Euler’sche Phi-Funktion

Buch der Woche

Titel: Cryptography and Network Security (7th Edition)

Autor(en): William Stallings

Verlag: Pearson

Umfang: ca. 760 Seiten

Hinweise zum Inhalt:

- trotz des allgemeinen Titels fast nur Kryptographie
- auch Protokolle für Netzwerksicherheit, Wireless Networks, E-Mail etc.
- praktisch orientiert, gut verständliche Darstellung
- viele Beweise, doch diese sind ausführlich und anschaulich
- enthält oft auch konkrete Rechenbeispiele
- insgesamt eines der besten Bücher für einen umfassenden Einstieg

Alte Hausaufgabe

Python Crypto Libraries: Bewertung und Vergleich:

Freiwillige vor...

➔ Tafelanschrieb

Alte Hausaufgabe: Inhalte (1)

Wünschenswerte Bestandteile der Analyse:

- als Einleitung: Aufgabenstellung wiedergeben, was soll getan werden, was waren die Ziele?
- Tipp 1 für später: Hinweis geben auf investierte bzw. zur Verfügung stehende Zeit für die Bearbeitung; nur dann läst sich die Ergebnisqualität von unbeteiligten Dritten richtig einschätzen.
- Tipp 2 für später: stets eine Art “Management Summary” an den Beginn stellen, für eilige Leser. Ein Absatz, ca. 5-10 Zeilen.
- Anforderungskatalog / Bewertungskriterien: was war wichtig, was unwichtig, was wurde betrachtet, was nicht
- Bewertungsschema festlegen, z.B. Schulnoten, oder --/-/0/+/++, etc.
- Übersicht in Tabellenform
- Links zur Entwicklerseite
- Hinweis zu den jeweiligen Urhebern/Entwicklern/Ownern

Alte Hausaufgabe: Inhalte (2)

Wünschenswerte Bestandteile der Analyse:

- Hinweise zur eigenen Vorgehensweise: wo wurden Tests bzw. Angaben Dritter zitiert, wo wurden eigene Untersuchungen angestellt?
- Funktionsumfang: was kann die Library, was nicht
- gibt es Support, falls ja: wo, von wem, unter welchen Bedingungen
- Lizenzbedingungen
- für den Leser sichtbar machen, was persönliche Präferenzen sind und was objektive Fakten sind, bei der Bewertung
- keine Marketing-Begriffe übernehmen wie “modern” o.ä.; oder zumindest erläutern
- insgesamt deutlich ausführlicher, nicht eine Seite mit Stichworten, sondern besser 5 – 10 Seiten

Krypto Wunschkonzert

Frage in die Runde: diskutieren Sie, welche Fragestellungen und Inhalte rund um die Kryptographie Sie noch nicht in der Vorlesung behandelt haben, die Sie jedoch interessant fänden und gerne in einer zukünftigen Vorlesung kennenlernen würden.

Primzahldarstellung (1)

Eine natürliche Zahl > 1 ist genau dann Primzahl, wenn sie nur durch 1 und durch sich selbst teilbar ist.

Fundamentalsatz der Arithmetik: Jede natürliche Zahl a > 1 besitzt eine eindeutige Darstellung als Produkt von Primzahlen wie folgt:

$a = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$

wobei $p_1 < p_2 < \ldots < p_k$ Primzahlen sind und $a_i \in \mathbb{N}$.

Bezeichnen wir mit P die Menge aller Primzahlen, so lässt sich obiges a auch wie folgt schreiben:

$a = \prod_{p_i \in P} p_i^{a_i}$, wobei alle $a_i \geq 0$.

Da wir das Produkt über alle Primzahlen bilden, sind hier alle bis auf endlich viele a_i gleich Null.

Primzahldarstellung (2)

Die Multiplikation zweier natürlicher Zahlen a und b können wir wie folgt darstellen:

Seien $a = \prod_{p_i \in P} p_i^{a_i}$, $b = \prod_{p_j \in P} p_j^{b_j}$, wobei alle a_i , $b_j \geq 0$. Dann gilt:

$a \cdot b = \prod_{p_k \in P} p_k^{c_k}$, mit $c_k = a_k + b_k$

Ist a Teiler von b, so gilt: $a_k \leq b_k$ für alle k.

Ist d = ggt(a, b), und $d = \prod_{p_i \in P} p_i^{d_i}$, so gilt für alle i: $d_i = \min(a_i , b_i)$

Bei der Berechnung größter gemeinsamer Teiler hilft diese Darstellung noch nicht. Hierzu verwenden wir beispielsweise den Euklidischen Algorithmus.

Euklidischer Algorithmus (1)

- Teilbarkeit und größter gemeinsamer Teiler spielen in der Kryptographie eine wesentliche Rolle. Die Berechnung des ggt ist für uns daher von großer Bedeutung.
- Standardverfahren zur Berechnung des ggt ist der Euklidische Algorithmus. Diesen lernt man oftmals schon in der Schule kennen, dann in der Mathematik-Vorlesung, spätestens in Krypto 1 erneut, und auch wir nehmen uns hier in Krypto 2 nochmals Zeit dafür, da der Euklidische Algorithmus in verallgemeinerter Form von uns immer wieder benötigt wird.
- Eigentlich handelt es sich um einen mathematischen Satz. Dessen Beweis erfolgt sodann konstruktiv in Form eines Algorithmus zur Berechnung des ggt.
- Wir werden anschließend auch den sogenannten Erweiterten Euklidischen Algorithmus nochmals betrachten.
- Hierauf aufbauend können wir dann weitere für uns wichtige Algorithmen einführen und besser verstehen.

Euklidischer Algorithmus (2)

Wir möchten den größten gemeinsamen Teiler zweier ganzer Zahlen a und b bestimmen. Wegen ggt(a, b) = ggt(|a|,|b|) nehmen wir ohne Einschränkung an, dass $a \geq b > 0$.

Division mit Rest erlaubt uns die Darstellung

$a = q_1 b + r_1$, mit $0 \leq r_1 < b$

Nehmen wir zunächst an, dass $r_1 = 0$. Dann ist b Teiler von a und natürlich auch (größter) Teiler von b. Somit gilt ggt(a, b) = b.

Nun betrachten wir den Fall $r_1 \neq 0$. Setzen wir ggt(a, b) =: g, so gilt:

Wegen $r_1 = a - q_1 b$ folgt $g \mid r_1$.

Wegen $g \mid b$ gilt daher $g \mid ggt(b, r_1)$.

Als nächstes zeigen wir, dass $ggt(b, r_1) \mid g$:

Zunächst sehen wir: $ggt(b, r_1)$ ist auch Teiler von a.

Insbesondere ist $ggt(b, r_1)$ also gemeinsamer Teiler von a und b.

Jeder gemeinsame Teiler von a und b ist zugleich Teiler von ggt(a, b).

Folglich: $ggt(b, r_1) \mid g$

$\Rightarrow ggt(b, r_1) = ggt(a, b)$.

Euklidischer Algorithmus (3)

Betrachten wir nochmals die Gleichung $a = q_1 b + r_1$, mit $0 \leq r_1 < b \leq a$.

Wir wissen nun, dass $ggt(b, r_1) = ggt(a, b)$. In gleicher Weise fahren wir fort, bis wir erstmals einen Rest $r_{n+1} = 0$ erhalten:

$a = q_1 b + r_1$, mit $0 < r_1 < b$

$b = q_2 r_1 + r_2$, mit $0 < r_2 < r_1$

$r_1 = q_3 r_2 + r_3$, mit $0 < r_3 < r_2$

$r_2 = q_4 r_3 + r_4$, mit $0 < r_4 < r_3$

...

$r_{n-2} = q_n r_{n-1} + r_n$, mit $0 < r_n < r_{n-1}$

$r_{n-1} = q_{n+1} r_n + 0$

In jedem Schritt wenden wir die oben gezeigte Gleichung an und erhalten:

$ggt(a, b) = ggt(b, r_1) = ggt(r_1, r_2) = ... = ggt(r_{n-2}, r_{n-2}) = ggt(r_{n-1}, r_n) = r_n$.

Sobald unsere Division mit Rest also erstmals einen Rest Null ergibt, haben wir den größten gemeinsamen Teiler gefunden: $ggt(a, b) = r_n$.

Euklidischer Algorithmus (4)

Verwenden wir modulare Arithmetik, erhalten wir eine andere, elegante Darstellung des obigen Sachverhalts:

Wir betrachten erneut die Gleichung $a = qb + r$, mit $0 \leq r < b \leq a$, wobei $q = \lfloor \frac{a}{b} \rfloor$

Wir haben weiter oben gezeigt, dass ggt(a, b) = ggt(b, r)

Aus der Definition modularer Restklassen folgt zudem: (a mod b) = r

Insgesamt also: ggt(a, b) = ggt(b, r) = ggt(b, a mod b)

Diese Gleichung liefert uns sofort eine einfache rekursive Funktion zur Berechnung größter gemeinsamer Teiler:

```
ggt(a, b)
   if b = 0 then return a
   else return ggt(b, a mod b)
```

Euklidischer Algorithmus (5)

Beispielrechnung für ggt zweier größerer Zahlen

➔ Tafelanschrieb

Erweiterter Euklidischer Algorithmus

Erweitern wir den Euklidischen Algorithmus, so liefert uns dieser zudem stets zwei ganze Zahlen x und y, so dass gilt:

$xa + yb = ggt(a, b)$

Für die Berechnung passender x und y schauen wir nochmals die Gleichungen aus dem Beweis zum Euklidischen Algorithmus an:

$a = q_1 b + r_1$, mit $0 < r_1 < b$ $\quad \Rightarrow r_1 = a - q_1 b = x_1 a + y_1 b$, mit $x_1 = 1$, $y_1 = q_1$

$b = q_2 r_1 + r_2$, mit $0 < r_2 < r_1$ $\quad \Rightarrow r_2 = b - q_2 r_1 = b - q_2(x_1 a + y_1 b) =$

$= (-q_2 x_1)a + (-q_2 y_1 + 1)b =$

$= x_2 a + y_2 b$, mit $x_2 = (-q_2 x_1)$, $y_2 = (-q_2 y_1 + 1)$

Für r_1 und r_2 haben wir also schon die gewünschte Darstellung.

Anmerkung: dies gilt natürlich oben auch für den Fall $r_1 = 0$ oder $r_2 = 0$.

Wenn die Darstellung jedoch für r_{i-2} und r_{i-1} existiert, $i \geq 2$, dann existiert sie auch für r_i:

$r_{i-3} = q_{i-1} r_{i-2} + r_{i-1}$

$r_{i-2} = q_i r_{i-1} + r_i \quad \Rightarrow r_i = r_{i-2} - q_i r_{i-1} = (x_{i-2} a + y_{i-2} b) - q_i(x_{i-1} a + y_{i-1} b) =$

$= (x_{i-2} - q_i x_{i-1})a + (y_{i-2} - q_i y_{i-1})b$

Somit haben wir (vollst. Ind.) auch für $ggt(a, b) = r_n$ eine Darstellung als $r_n = x_n a + y_n b$

Ganzzahlige Linearkombinationen

Definition: für a, b ∈ $\mathbb{Z}$ sei $a\mathbb{Z} + b\mathbb{Z} = \{ai + bj : i,j \in \mathbb{Z}\}$ die Menge aller ***ganzzahligen Linearkombinationen*** von i und j.

Satz: die Menge aller ganzzahligen Linearkombinationen von a und b ist gleich der Menge aller ganzzahligen Vielfachen von ggt(a,b).

Beweis:

(i) eine Richtung folgt unmittelbar aus dem erweiterten Euklidischen Algorithmus: ggt(a,b) ist darstellbar als Linearkombination xa + yb.

(ii) sei $a = a' \cdot ggt(a,b)$, $b = b' \cdot ggt(a,b)$, dann ist

$ai + bj = a' \cdot ggt(a,b) \cdot i + b' \cdot ggt(a,b) \cdot j = (a' \cdot i + b' \cdot j)\ ggt(a,b)$.

Modulare Arithmetik (1)

Die wichtigsten Eigenschaften beim Rechnen in modularer Arithmetik sind schon bekannt. Nachfolgend nochmals ein paar einfache Fakten:

Sei $a = qn + r$, mit $0 \leq r < n$. Dann ist $q = \lfloor \frac{a}{n} \rfloor$, beziehungsweise:

$a = \lfloor \frac{a}{n} \rfloor \cdot n + (a \bmod n)$

Es gilt:

i. $a = b \pmod n \Leftrightarrow n \mid (a - b)$

ii. $a = b \pmod n$ und $b = c \pmod n \Rightarrow a = c \pmod n$

iii. $(a \bmod n) \cdot (b \bmod n) = (a \cdot b) \pmod n$

iv. $a + b \pmod n = a + c \pmod n \Rightarrow b = c \pmod n$

Modulare Arithmetik (2)

Die folgende Aussage gilt nur, wenn a und n teilerfremd sind (dann hat a ein multiplikatives Inverses modulo n):

$a \cdot b \pmod n = a \cdot c \pmod n \Rightarrow b = c \pmod n$

Beispiel: sei a = 5, b = 4, c = 8, und n = 10. Dann gilt

$a \cdot b \pmod n = 5 \cdot 4 \pmod{10} = 0 = 5 \cdot 8 \pmod{10} = a \cdot c \pmod n$

Aber es gilt nicht $4 = 8 \pmod{10}$

Betrachten wir die Restklassen mod 10 und multiplizieren jedes Element mit 5:

0	1	2	3	4	5	6	7	8	9
0	5	0	5	0	5	0	5	0	5

In der zweiten Zeile erhalten wir nicht etwa alle zehn Elemente der Restklassenmenge, sondern lediglich zwei...

Kleiner Satz von Fermat (1)

Kleiner Satz von Fermat: sei p Primzahl und sei die natürliche Zahl a teilerfremd zu p. Dann gilt:

$a^{p-1} = 1 \pmod p$

Beweis: sei M = {1, 2, ..., p-1} die Menge aller natürlichen Zahlen < p.

Ferner sei M' = {a·1 (mod p), a·2 (mod p), ..., a·(p-1) (mod p)} = {a·i (mod p)}, $i \in$ M, die Menge der Elemente aus M nach Multiplikation modulo p jedes Elementes mit a.

Wir wollen zunächst zeigen, dass M' = M. Hierfür genügt es zu zeigen, dass alle Elemente in M' verschieden sind.

Anderenfalls gibt es i, j mit $1 \leq i < j \leq p-1$, so dass a·i (mod p) = a·j (mod p).

Da a und p teilerfremd, können wir a aus dieser Gleichung herauskürzen:

$\Rightarrow$ i = j mod p $\Rightarrow$ i = j , im Widerspruch zur Voraussetzung. Folglich ist M' = M.

Kleiner Satz von Fermat (2)

Als nächstes multiplizieren wir alle Elemente aus M miteinander und erhalten

$1 \cdot 2 \cdot 3 \cdot \ldots \cdot (p-1) = (p-1)!$

Dasselbe tun wir mit den Elementen der Menge M' und erhalten:

$(a \cdot 1) \pmod p \cdot (a \cdot 2) \pmod p \cdot (a \cdot 3) \pmod p \cdot \ldots \cdot a \cdot (p-1) \pmod p = a^{p-1}(p-1)! \pmod p$

Wegen M' = M folgt: $a^{p-1}(p-1)! = (p-1)! \pmod p$

Wir können aus beiden Seiten (p-1)! herauskürzen, da (p-1)! teilerfremd zu p.

Daraus folgt: $a^{p-1} = 1 \pmod p$

Bemerkung: aus obigem Satz folgt insbesondere, dass $a^p = a \pmod p$. Es lässt sich zeigen, dass diese Aussage für alle a gilt, nicht nur für teilerfremde a und p.

Eulersche Φ-Funktion (1)

In der asymmetrischen Kryptographie arbeiten wir viel mit Primzahlen. Bei Berechnungen von/mit Primzahlen hilft uns eine Funktion sehr: die sogenannte ***Eulersche Phi-Funktion*** bzw. ***Φ-Funktion*** (nach dem griechischen Buchstaben).

Anmerkung: Im englischsprachigen Raum heisst diese ***Euler's Totient Function***.

Für n $\in \mathbb{N}$ definieren wir ***Φ(n)*** als die Anzahl teilerfremder ganzer Zahlen zwischen 1 und n-1. Per Definition setzt man Φ(1) = 1.

Beispiel:

Φ(23) = |{1, 2, 3, ..., 22}| = 22

Φ(24) = |{1, 5, 7, 11, 13, 17, 19, 23}| = 8

Wie man sofort sieht, gilt allgemein für eine Primzahl p: Φ(p) = p – 1.

Eulersche Φ-Funktion (2)

Satz: Sei n = pq für zwei Primzahlen p ≠ q. Dann gilt:

$\Phi(n) = \Phi(pq) = (p - 1)(q - 1) = \Phi(p) \cdot \Phi(q)$.

Beweis:

Betrachten wir, welche Zahlen zwischen 1 und pq-1 teilerfremd sind zu pq:

Nicht teilerfremd sind 1, p, 2p, ..., (q-1)p. Ebenso 1, q, 2q, ..., (p-1)q.

Die erste Menge besteht aus q Elementen, die zweite aus p Elementen, wobei wir die 1 doppelt gezählt haben und folglich einmal abziehen müssen:

Nicht teilerfremd sind also p + q – 1 Elemente.

Alle anderen Zahlen zwischen 1 und n sind offenbar teilerfremd, da weder Vielfache von p noch von q.

Wir erhalten also p + q -1 nicht teilerfremde Elemente. Ziehen wir diese ab von n = pq, so ehalten wir:

$\Phi(n) = \Phi(pq) = pq - (p + q - 1) = pq - p - q + 1 = (p - 1)(q - 1) = \Phi(p) \cdot \Phi(q)$.

Satz von Euler (1)

Satz von Euler: Für teilerfremde, positive Zahlen a und n gilt: $a^{\Phi(n)} = 1 \pmod n$.

Beweis:

Vorbemerkung: ist n eine Primzahl, so erhalten wir den kleinen Satz von Fermat und die Aussage ist bereits bewiesen. Der Beweis für den allgemeinen Fall verwendet dieselbe Beweisidee wie beim kleinen Satz von Fermat.

Sei F(n) die Menge aller zu n teilerfremden positiven Zahlen zwischen 1 und n-1.

$F(n) = \{x_1, x_2, \dots, x_{\Phi(n)}\}$

Weiter sei $F'(n) = \{a \cdot x_1 \pmod n, a \cdot x_2 \pmod n, \dots, a \cdot x_{\Phi(n)} \pmod n\}$

die Menge der Elemente aus F(n) nach Multiplikation modulo n jedes Elementes mit a.

Zunächst gilt $F'(n) \subseteq F(n)$, denn wenn sowohl a als auch alle x_i teilerfremd sind zu n, so gilt dies auch für alle $a \cdot x_i \pmod n$.

Nehmen wir an, zwei Elemente aus F'(n) seien identisch: $a \cdot x_i \pmod n = a \cdot x_j \pmod n$. Dann folgt wieder (wie im kleinen Satz von Fermat), dass $x_i = x_j$.

Folglich ist F'(n) = F(n).

Satz von Euler (2)

Aus F’(n) = F(n) erhalten wir:

$\prod_{i=1}^{\Phi(n)} a \cdot x_i = \prod_{i=1}^{\Phi(n)} x_i \pmod n$

$\Rightarrow a^{\Phi(n)} \cdot \prod_{i=1}^{\Phi(n)} x_i = \prod_{i=1}^{\Phi(n)} x_i \pmod n$

$\Rightarrow a^{\Phi(n)} = 1 \pmod n$.

Bemerkung: aus obigem Satz folgt insbesondere, dass $a^{\Phi(n)+1} = a \pmod n$.

Wie schon beim kleinen Satz von Fermat lässt sich auch hier wieder zeigen, dass diese Aussage für alle a gilt, nicht nur für teilerfremde a und p.

Primzahleigenschaften (1)

Primzahlen benötigen wir für verschiedene Public Key Verfahren wie beispielsweise RSA. Doch wie können wir feststellen, ob eine (pseudo-)zufällig erzeugte große Zahl Primzahl ist oder nicht?

Für Zahlen mit mehreren tausend Bit Länge ist es nicht machbar, alle möglichen Teiler durchzuprobieren. Wir benötigen daher ein effizienteres Verfahren für einen Test auf Primzahleigenschaft.

Eines davon ist der sogenannte ***Miller Rabin Test***. Wendet man diesen Test an, so lässt sich von einer Zahl sagen, ob sie mit sehr hoher Wahrscheinlichkeit eine Primzahl ist oder nicht. Absolute Gewissheit im Sinne von 100% Wahrscheinlichkeit lässt sich allerdings nicht erzielen. Mathematisch ist das unschön, für die Praxis allerdings kein Hinderungsgrund.

Es mag überraschen, dass bis vor wenigen Jahren kein praktikabler Primzahltest für große Zahlen bekannt war, der die Primzahleigenschaft zu 100% zusichert.

Bevor wir den Miller Rabin Test einführen, benötigen wir einige Vorüberlegungen.

Primzahleigenschaften (2)

Vorbemerkung: ein ***Lemma*** ist ein Sachverhalt bzw. ein Hilfssatz, dessen Aussage für sich betrachtet keine besondere Bedeutung hat, der jedoch im Beweis eines später zu beweisenden Satzes verwendet bzw. zitiert werden kann.

Lemma 1: sei p Primzahl und $a \in \mathbb{N}, 0 < a < p$. Dann gilt

$$a^2 \bmod p = 1 \quad \Leftrightarrow \quad a \bmod p = \pm 1$$

Beweis:

"$\Leftarrow$": sei $a \bmod p = \pm 1$, dann folgt $1 = (\pm 1)^2 = (a \bmod p)^2 = a^2 \bmod p$.

"$\Rightarrow$": sei $a^2 \bmod p = 1$, dann folgt $1 = a^2 \bmod p = (a \bmod p)^2 \bmod p$.

Aus $1 = |1| = |a \bmod p| \cdot |a \bmod p|$ folgt $a \bmod p = \pm 1$.