Public-Key-Kryptographie und RSA

Dozent: Prof. Dr. Michael Eichberg

Kontakt: michael.eichberg@dhbw.de

Version: 1.1.2

Basierend auf: *Cryptography and Network Security - Principles and Practice, 8th Edition, William Stallings*

Folien: [HTML] https://delors.github.io/sec-public-key-kryptographie/folien.de.rst.html
[PDF] https://delors.github.io/sec-public-key-kryptographie/folien.de.rst.html.pdf

Fehler melden: https://github.com/Delors/delors.github.io/issues

1. Grundlagen

Terminologie bzgl. asymmetrischer Verschlüsselung

- Asymmetrische Schlüssel
- Public-Key-Zertifikat
- Public-Key (asymmetrischer) kryptografischer Algorithmus
- Public-Key-Infrastruktur (PKI)

Asymmetrische Schlüssel:

Zwei zusammengehörige Schlüssel, ein öffentlicher und ein privater Schlüssel, die zur Durchführung komplementärer Operationen verwendet werden, z. B. Ver- und Entschlüsselung oder Signaturerstellung und Signaturprüfung.

Public-Key-Zertifikat:

Ein digitales Dokument, das mit dem privaten Schlüssel einer Zertifizierungsstelle (eng:*Certification Authority*) ausgestellt und digital signiert wird und den Namen eines Teilnehmers an einen öffentlichen Schlüssel bindet. Das Zertifikat gibt an, dass der im Zertifikat genannte Teilnehmer die alleinige Kontrolle und den Zugriff auf den entsprechenden privaten Schlüssel hat.

Public-Key (asymmetrischer) kryptografischer Algorithmus:

Ein kryptographischer Algorithmus, der zwei zusammengehörige Schlüssel verwendet, einen öffentlichen und einen privaten Schlüssel. Die beiden Schlüssel haben die Eigenschaft, dass die Ableitung des privaten Schlüssels aus dem öffentlichen Schlüssel rechnerisch nicht machbar ist bzw. sein sollte (vgl. Quantenkryptografie).

Public-Key-Infrastruktur (PKI):

Eine Reihe von Richtlinien, Prozessen, Serverplattformen, Software und Workstations, die für die Verwaltung von Zertifikaten und öffentlich-privaten Schlüsselpaaren verwendet werden, einschließlich der Möglichkeit, Public-Key-Zertifikate auszustellen, zu pflegen und zu widerrufen.

Prinzipien von Public-Key-Kryptosystemen

- Das Konzept der *Public-Key-Kryptographie* (d. h. der Kryptografie mit öffentlichen Schlüsseln) entstand aus dem Versuch, zwei der schwierigsten Probleme im Zusammenhang mit der symmetrischen Verschlüsselung zu lösen:

  Schlüsselverteilung

  Wie kann man generell sicher kommunizieren, ohne einem "Key Distribution Center" (KDC) seinen Schlüssel anvertrauen zu müssen?

  Digitale Signaturen

  Wie kann man überprüfen, ob eine Nachricht unversehrt vom angegebenen Absender stammt?

KDC = Key Distribution Center

Prinzipien von Public-Key-Kryptosystemen

Whitfield Diffie und **Martin Hellman** von der Stanford University erzielten 1976 einen Durchbruch, indem sie eine Methode entwickelten, die beide Probleme löste und sich radikal von allen bisherigen Ansätzen der Kryptografie unterschied.

Bestandteile von Public-Key-Kryptosystemen

Klartext (*Plaintext*):
: Die lesbare Nachricht oder Daten, die dem Algorithmus als Eingabe dienen.

Verschlüsselungsalgorithmus:
: Führt verschiedene Umwandlungen des Klartextes durch.

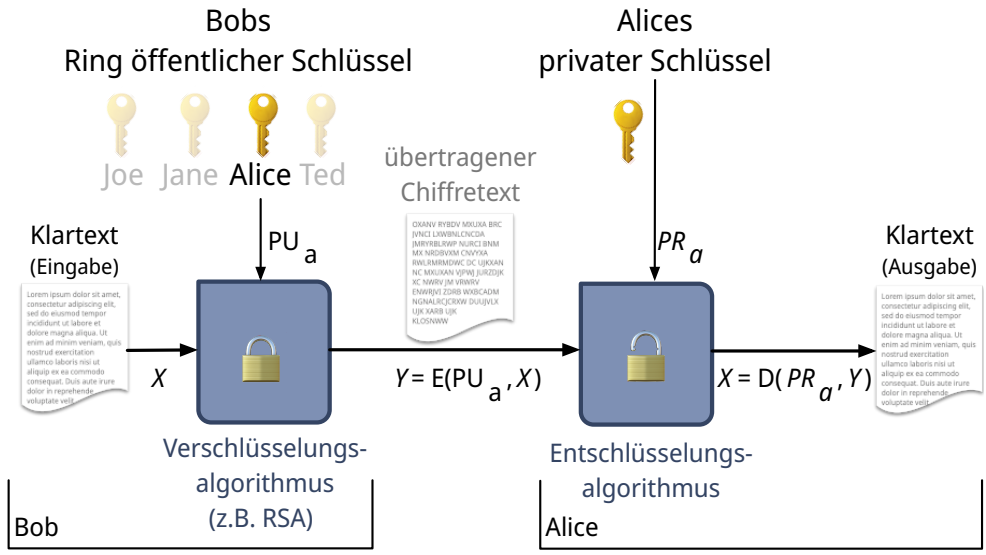
Öffentlicher Schlüssel:
: Wird für *Verschlüsselung* oder *Entschlüsselung* verwendet.

Privater Schlüssel: Verwendet für *Verschlüsselung* oder *Entschlüsselung*.

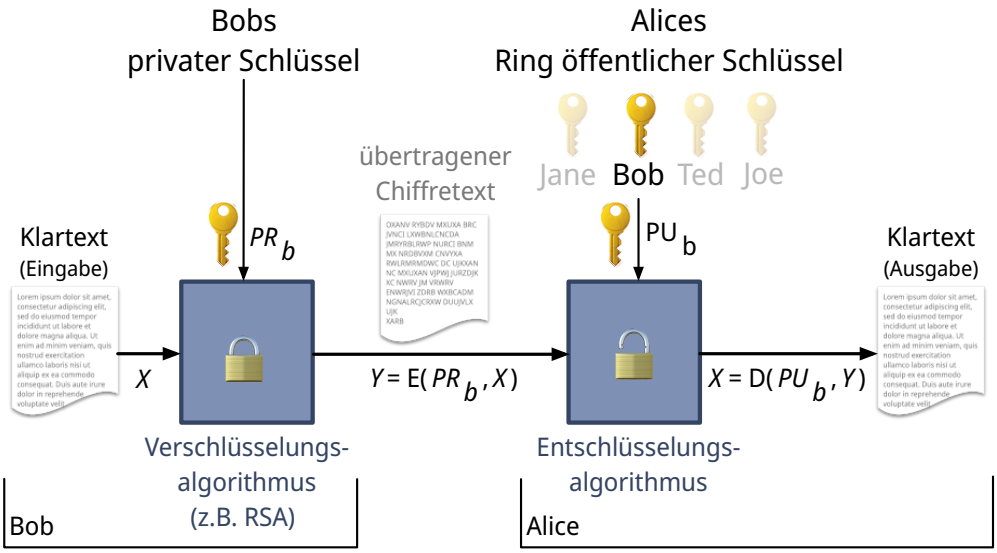
Chiffretext (*Ciphertext*):
: Die verschlüsselte Nachricht, die als Ausgabe produziert wird.

Entschlüsselungsalgorithmus:
: Nimmt den Geheimtext und den passenden Schlüssel entgegen und erzeugt den ursprünglichen Klartext.

Verschlüsselung mit öffentlichem Schlüssel



Verschlüsselung mit privatem Schlüssel



Konventionelle und Public-Key-Verschlüsselung

Konventionelle Verschlüsselung

Benötigt zur Anwendung

1. Es wird derselbe Algorithmus mit demselben Schlüssel für die Ver- und Entschlüsselung verwendet.
2. Der Sender und der Empfänger müssen den Algorithmus und den Schlüssel kennen bzw. besitzen.

Notwendig für die Sicherheit

1. Der Schlüssel muss geheim gehalten werden.
2. Es muss „*unmöglich*“ sein, eine Nachricht zu entschlüsseln, wenn der Schlüssel geheim gehalten wird.
3. Die Kenntnis des Algorithmus und von (ggf. vielen) Geheimtexten ist nicht ausreichend, um den Schlüssel zu ermitteln.

Public-Key Verschlüsselung

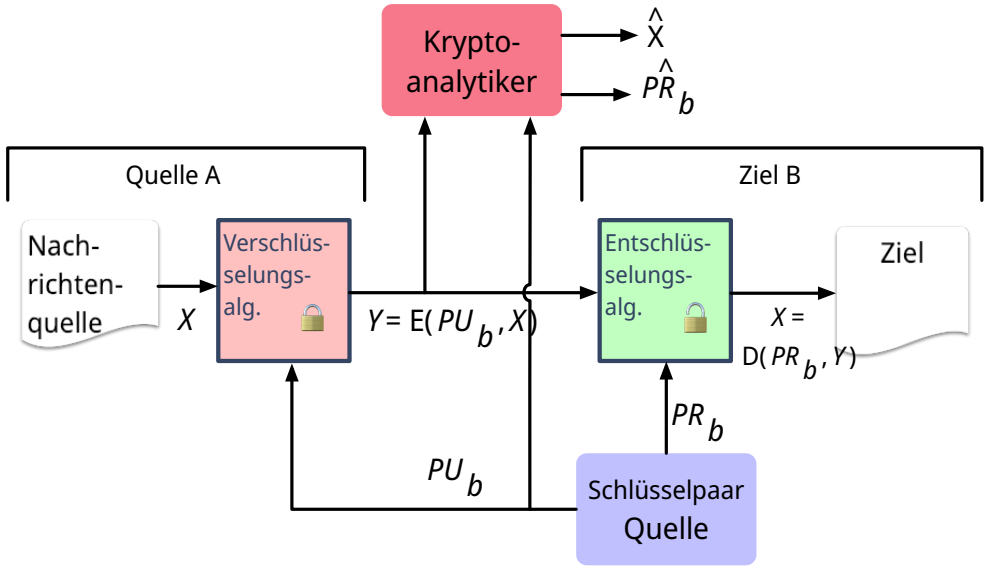
Benötigt zur Anwendung

1. Zwei Algorithmen: je einer für die Ver-/Entschlüsselung. Weiterhin ein Paar von Schlüsseln; je einer für die Ver-/Entschlüsselung.
2. Der Absender und der Empfänger müssen jeweils einen der passenden Schlüssel besitzen (nicht den gleichen).

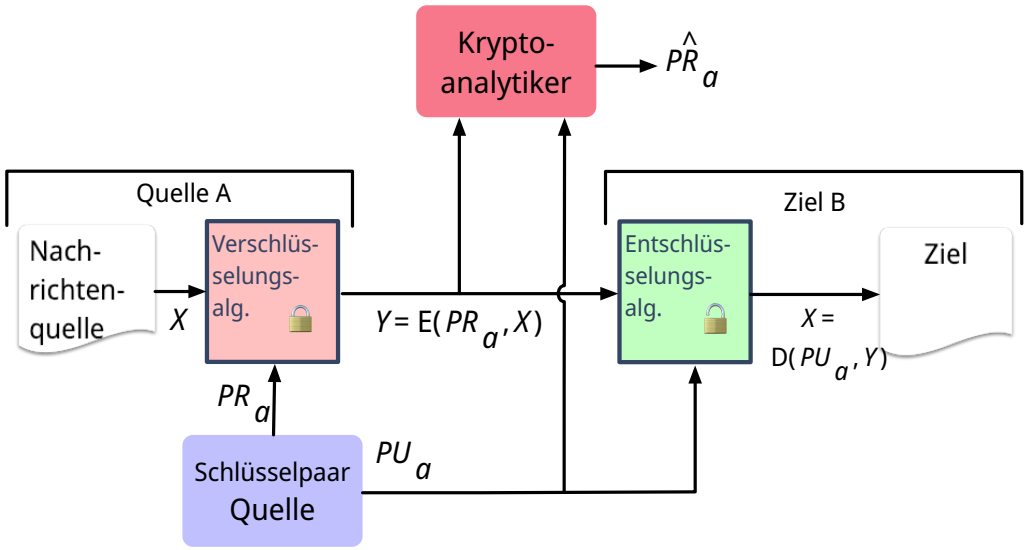
Notwendig für die Sicherheit

1. Einer der Schlüssel muss geheim bleiben.
2. Es muss „*unmöglich*“ sein, eine Nachricht zu entschlüsseln, wenn ein Schlüssel geheim gehalten wird.
3. Die Kenntnis des Algorithmus und eines Schlüssels sowie von Geheimtexten ist nicht ausreichend, um den anderen Schlüssel zu ermitteln.

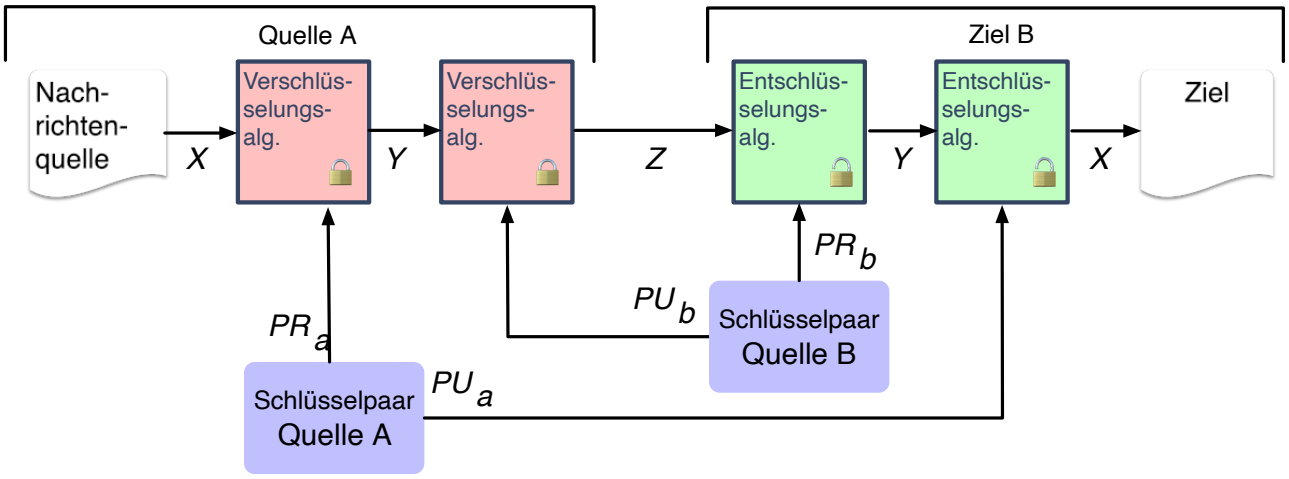
Public-Key-Kryptosystem: Vertraulichkeit



Public-Key-Kryptosystem: Authentifizierung



Public-Key-Kryptosystem: Authentifizierung und Geheimhaltung



Anwendungen für Public-Key-Kryptosysteme

Kryptosysteme mit öffentlichen Schlüsseln lassen sich in drei Kategorien einteilen:

1. *Ver-/Entschlüsselung*: Der Absender verschlüsselt eine Nachricht mit dem öffentlichen Schlüssel des Empfängers.
2. *Digitale Unterschriften*: Der Absender „unterschreibt“ eine Nachricht mit seinem privaten Schlüssel.
3. *Schlüsselaustausch*: Zwei Seiten arbeiten zusammen, um einen Sitzungsschlüssel (d. h. einen symmetrischen Schlüssel) auszutauschen.

Einige Algorithmen eignen sich für alle drei Anwendungen, während andere nur für eine oder zwei verwendet werden können:

Algorithmus	Ver-/ Entschlüsselung	Digitale Signaturen	Schlüssel-austausch
RSA	✓	✓	✓
Elliptic Curve	✓	✓	✓
Diffie-Hellman	✗	✗	✓
DSS[1]	✗	✓	✗

[1] DSS = Digital Signature Standard*, entwickelt von der NSA (National Security Agency)

Anforderungen an Public-Key-Algorithmen

- Für eine Partei B ist es rechnerisch einfach, ein Schlüsselpaar (bestehend aus öffentlicher Schlüssel PU_b und privater Schlüssel PR_b) zu erzeugen.
- Für einen Absender A ist es rechnerisch einfach, bei Kenntnis des öffentlichen Schlüssels von B und der zu verschlüsselnden Nachricht den entsprechenden Chiffretext zu erzeugen.
- Für den Empfänger B ist es rechnerisch einfach, den resultierenden Chiffretext mit Hilfe des privaten Schlüssels zu entschlüsseln, um die ursprüngliche Nachricht wiederherzustellen.
- Für einen Angreifer, der den öffentlichen Schlüssel kennt, ist es *rechnerisch unmöglich*, den privaten Schlüssel zu ermitteln.
- Für einen Angreifer, der den öffentlichen Schlüssel und einen Chiffretext kennt, ist es *rechnerisch unmöglich*, die ursprüngliche Nachricht wiederherzustellen.
- Die beiden Schlüssel können in beliebiger Reihenfolge verwendet werden.

Anforderungen an Public-Key-Algorithmen

- Benötigt wird eine Falltürfunktion (🇺🇸 *Trapdoor one-way function*)

  Eine Einwegfunktion ist im Allgemeinen eine Funktion, bei der jeder Funktionswert eine eindeutige Umkehrung hat, wobei die *Berechnung der Funktion einfach* ist, während die *Bestimmung der Umkehrfunktion praktisch undurchführbar ist*.

  - $Y = f(X)$ einfach
  - $X = f^{-1}(Y)$ „unmöglich“
- Eine Einwegfunktion mit Falltür ist eine Familie invertierbarer Funktionen f_k, für die gilt:
  - $Y = f_k(X)$ einfach, wenn k und X bekannt sind.
  - $X = f_k^{-1}(Y)$ einfach, wenn k und Y bekannt sind.
  - $X = f_k^{-1}(Y)$ unmöglich, wenn Y bekannt ist, aber k nicht.
- Ein praktisches Public-Key-Verfahren hängt von einer geeigneten Trapdoor-Einwegfunktion ab.

Ein Falltürfunktion lässt sich nicht trivial umkehren; bzw. die Umkehrung erfordert spezielle (weitergehende) Informationen; d. h. die Falltür.

Public-Key-Kryptoanalyse

Ein Verschlüsselungsverfahren mit öffentlichem Schlüssel ist anfällig für einen Brute-Force-Angriff.

- Gegenmaßnahme: große Schlüssel verwenden!
- Die Schlüsselgröße muss klein genug sein, um eine praktische Ver- und Entschlüsselung zu ermöglichen.
- Vorgeschlagene Schlüsselgrößen führen zu Verschlüsselungs-/Entschlüsselungsgeschwindigkeiten, die für den allgemeinen Gebrauch zu langsam sind.
- Die Verschlüsselung mit öffentlichen Schlüsseln ist derzeit auf die Schlüsselverwaltung und Signaturanwendungen beschränkt.

Eine andere Form des Angriffs besteht darin, einen Weg zu finden, den privaten Schlüssel anhand des öffentlichen Schlüssels zu berechnen.

Bislang konnte nicht mathematisch bewiesen werden, dass diese Form des Angriffs für einen bestimmten Public-Key-Algorithmus nicht durchführbar ist.

Schließlich gibt es noch einen Angriff mit wahrscheinlicher Nachricht.

Dieser Angriff kann vereitelt werden, indem einige zufällige Bits an einfache Nachrichten angehängt werden.

Bei einem Angriff mit „wahrscheinlicher Nachricht“, verschlüsselt der Angreifer eine Reihe von Nachrichten (z. B. alle DES Schlüssel mit dem öffentlichen Schlüssel des Adressaten) und analysiert die resultierenden Chiffretexte, um den privaten Schlüssel zu ermitteln.

Missverständnisse bei der Verwendung von Public-Key-Kryptosystemen

- Public-Key-Verschlüsselung ist sicherer vor Kryptoanalyse als die symmetrische Verschlüsselung.
- Public-Key-Kryptografie (d. h. die Verschlüsselung mit öffentlichen Schlüsseln) ist eine Allzwecktechnik, die die symmetrische Verschlüsselung überflüssig gemacht hat.
- Man hat das Gefühl, dass die Schlüsselverteilung bei der Verschlüsselung mit öffentlichen Schlüsseln trivial ist, verglichen mit dem mühsamen Handshaking, das bei der symmetrischen Verschlüsselung mit Schlüsselverteilungszentren verbunden ist.

2. Public-Key Algorithmen

Rivest-Shamir-Adleman (RSA) Algorithm

- Entwickelt 1977 am MIT von Ron Rivest, Adi Shamir und Len Adleman.
- Universeller Ansatz zur Verschlüsselung mit öffentlichen Schlüsseln.
- Ist eine Chiffre, bei der Klartext und Chiffretext ganze Zahlen zwischen 0 und $n - 1$ für ein bestimmtes n sind.
- Eine typische Größe für n waren 1024 Bits oder 309 Dezimalziffern.

  Solch kleine Zahlen werden heute als äußerst unsicher angesehen, insbesondere angesichts der bevorstehenden Quantencomputer und der Entwicklung von Quantenalgorithmen (vgl. Shors Algorithmus (1994)), die Zahlen effizient faktorisieren können, wenn genügend QBits in hinreichender Qualität[2] zur Verfügung stehen.

---

[2] Aktuell sind Quantencomputer nicht in der Lage, die für RSA verwendeten Schlüssel zu brechen und es ist auch (noch) nicht geklärt, ob die aktuellen Technologien entsprechend skaliert werden können. Es besteht aber die Möglichkeit!

RSA Algorithmus

- RSA verwendet einen Ausdruck mit Exponentialen
- Der Klartext wird in Blöcken verschlüsselt, wobei jeder Block einen Binärwert hat, der kleiner als eine bestimmte Zahl n ist[3].
- Die Ver- und Entschlüsselung erfolgt für einen Klartextblock M und einen Chiffretextblock C in der folgenden Form:

$$C = M^e \ mod \ n \qquad M = C^d \ mod \ n \qquad (M^e)^d \ mod \ n = M^{ed} \ mod \ n$$

- Sowohl der Sender als auch der Empfänger müssen den Wert von n kennen.
- Der Absender kennt den Wert von e, und nur der Empfänger kennt den Wert von d
- Dies ist ein Public-Key-Verschlüsselungsalgorithmus mit dem öffentlichen Schlüssel $PU = \{e, n\}$ und dem privaten Schlüssel $PR = \{d, n\}$.

$$M = C^d \ mod \ n \Rightarrow M = (M^e \ mod \ n)^d \ mod \ n = (M^e)^d \ mod \ n$$

---

[3] Basierend auf der Zahl n ergibt sich die maximale Größe des Blocks in Bit. Sei, hypothetisch, $n = 4.294.967.296 + 1$, dann kann der Block maximal 32 Bit groß sein ($2^{32} = 4.294.967.296$).

Anforderungen an den RSA Algorithmus

Damit dieser Algorithmus für die Verschlüsselung mit öffentlichen Schlüsseln geeignet ist, müssen die folgenden Anforderungen erfüllt sein:

1. Es ist möglich, Werte für e, d, n so zu finden, dass $M^{ed} \, mod \, n = M$ für alle $M < n$.
2. Es ist relativ einfach, $M^e \, mod \, n$ und $C^d \, mod \, n$ für alle Werte von $M < n$ zu berechnen.
3. Es ist nicht möglich, d zu bestimmen, wenn e und n gegeben sind.

The RSA Algorithm

Schlüsselgenerierung von Alice

Wähle p, q	p und b beide prim, $p \neq q$
Berechne n	$n = p \times q$
Berechne $\phi(n)$	$\phi(n) = (p-1)(q-1)$
Wähle e	$GGT(\phi(n), e) = 1; \quad 1 < e < \phi(n)$
Berechne d	$d \equiv e^{-1} \ (mod\ \phi(n)) \Leftrightarrow ed\ mod\ \phi(n) = 1$
Public-Key	$PU = \{e, n\}$
Private-Key	$PR = \{d, n\}$

Verschlüsselung von Bob mit Alices öffentlichen Schlüssel

Klartext $M < n$
Chiffretext $C = M^e\ mod\ n$

Entschlüsselung von Alice mit ihrem privaten Schlüssel

Chiffretext C
Klartext $M = C^d\ mod\ n$

Berechnung von d

Der Wert von d wird mit Hilfe des erweiterten Euklidischen Algorithmus[4] berechnet.

Wir wissen dass $GGT(\phi(n), e) = 1$ gilt; d. h. e und $\phi(n)$ sind teilerfremd/*coprime*.

$$\begin{aligned} ex + \phi(n)y &= GGT(e, \phi(n)) \\ &= 1 \\ \text{Umgestellt:} & \\ ex &= -\phi(n)y + 1 \\ \Rightarrow & \\ ex \ mod \ \phi(n) &= 1 \end{aligned}$$

$$\text{somit } x \hat{=} d$$

Jupyter Notebook zur Berechnung

---

[4] Zur Erinnerung: der erweiterte Euklidische Algorithmus berechnet den größten gemeinsamen Teiler von zwei Zahlen (a, b) und zusätzlich zwei Koeffizienten (x, y), so dass gilt: $ax + by = ggt(a, b)$.

Beispiel für den RSA-Algorithmus

p und q: $p = 11; \quad q = 17; \quad n = 187 \qquad (\phi(n) = 10 \times 16 = 160)$

Klartext: 88

Verschlüsselung: $PU = \{e = 7, n = 187\}$:
$88^7 \; mod \; 187 = 11 = C$

Entschlüsselung: $PR = \{d = 23, n = 187\}$:
$11^{23} \; mod \; 187 = 88 = P$

Alternativer Exponent:
$e = 137 \Rightarrow d = 153$
$88^{137} \; mod \; 187 = 99 = C \qquad 99^{153} \; mod \; 187 = 88$

Potenzierung in der Modularen Arithmetik

- Sowohl bei der Verschlüsselung als auch bei der Entschlüsselung in RSA wird eine ganze Zahl potenziert mit einer weiteren ganzen Zahl $mod\ n$.

  Weiterhin haben wir es mit potenziell großen Exponenten zu tun, so dass die Effizienz der Potenzierung eine wichtige Rolle spielt.

- Eine Eigenschaft der modularen Arithmetik kann genutzt werden:

  $[(a\ mod\ n) \times (b\ mod\ n)]\ mod\ n = (a \times b)\ mod\ n$

Beispiel

$[11 = 1011_b]$ $\quad 2^{11} = 2^1 \times 2^2 \times 2^8 = 2 \times 4 \times 256$

$[09 = 1001_b]$ $\quad 2^9\ mod\ 13 = [(2^1\ mod\ 13) \times (2^8\ mod\ 13)]\ mod\ 13$

$$\begin{array}{rcccc} & 2^3 = 8 & 2^2 = 4 & 2^1 = 2 & 2^0 = 1 \\ 11 = & 1_b & 0_b & 1_b & 1_b \end{array}$$

Algorithmus zur Berechnung von $a^b \ mod \ n$

Quadrieren und Multiplizieren (*Square and Multiply*)

Die Ganzzahl b wird als Binärzahl `b[k]b[k-1]...b[0]` ausgedrückt:

```
1  c := 0; f := 1
2  for i := k downto 0
3      do c := 2 * c
4         f := (f * f) mod n
5      if b[i] = 1
6          then c := c + 1
7               f := (f * a) mod n
8  return f
```

Hinweis

`c` stellt lediglich die Komponente dar.

Jupyter Notebook mit Implementierung

Ergebnis des schnellen modularen Exponierungsalgorithmus für $a^b \ mod \ n$

$a = 7; b = 560 = 10\,0011\,0000_b$, und $n = 561$

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
f	7	49	157	526	160	241	298	166	67	1

Effiziente Verschlüsselung mit dem öffentlichen Schlüssel

Um den RSA-Algorithmus bei Verwendung des öffentlichen Schlüssels zu beschleunigen, wird in der Regel eine bestimmte Wahl von e getroffen:

- Die häufigste Wahl ist 65537 ($2^{16} + 1$).
- Zwei weitere beliebte Wahlmöglichkeiten sind $e = 3$ und $e = 17$.
- Jede dieser Möglichkeiten hat nur zwei 1-Bits, so dass die Anzahl der Multiplikationen, die für die Potenzierung erforderlich sind, minimiert wird.
- Mit einem sehr kleinen öffentlichen Schlüssel, wie $e = 3$, wird RSA jedoch anfällig für einen einfachen Angriff.

Effiziente Entschlüsselung mit dem privaten Schlüssel

- Die Entschlüsselung verwendet die Potenzierung mit d.
- Ein kleiner Wert von d ist jedoch anfällig für einen Brute-Force-Angriff und für andere Formen der Kryptoanalyse.
- Der Chinesischen Restsatz (CRT) kann verwendet werden, um Berechnungen zu beschleunigen:
  Die Größen $d \ mod \ (p-1)$ und $d \ mod \ (q-1)$ können vorberechnet werden.
  Das Ergebnis ist, dass die Berechnung etwa viermal so schnell ist wie die direkte Berechnung von $M = C^d \ mod \ n$.

Schlüsselgenerierung

Vor der Anwendung des Public-Key-Kryptosystems muss jeder Teilnehmer ein Schlüsselpaar erzeugen:

- Bestimmung der Primzahlen p und q.
- Wahl von e oder d und Berechnung der anderen.
- Da der Wert von $n = pq$ jedem potenziellen Gegner bekannt sein wird, müssen die Primzahlen aus einer ausreichend großen Menge ausgewählt werden.
- Die Methode, die zum Finden großer Primzahlen verwendet wird, muss einigermaßen effizient sein.

  Es kann z. B. der Miller-Rabin-Algorithmus verwendet werden.

Die Sicherheit von RSA - Fünf mögliche Ansätze für einen Angriff

Brute-Force: Dabei werden alle möglichen privaten Schlüssel ausprobiert.

Mathematische Angriffe:
Es gibt mehrere Ansätze, die vom Aufwand her alle dem Faktorisieren des Produkts aus zwei Primzahlen entsprechen.

Zeitliche Angriffe: Diese hängen von der Laufzeit des Entschlüsselungsalgorithmus ab.

Hardware-Fehler-basierter Angriff:
Hier geht es darum, Hardware-Fehler in den Prozessor zu induzieren, der digitale Signaturen erzeugt.

Gewählte Chiffretext-Angriffe:
Ziel ist es Eigenschaften des RSA-Algorithmus auszunutzen.

Faktorisierungsproblem

Es gibt drei Ansätze für einen mathematischen Angriff auf RSA:

1. Faktorisierung von n in seine beiden Primfaktoren. Dies ermöglicht die Berechnung von $\phi(n) = (p-1) \times (q-1)$, was wiederum die Bestimmung von $d = e^{-1} (mod\ \phi(n))$ ermöglicht.
2. Direkte Bestimmung von $\phi(n)$, ohne vorher p und q zu bestimmen. Dies ermöglicht wiederum die Bestimmung von $d = e^{-1} (mod\ \phi(n))$.
3. Direkte Bestimmung von d, ohne vorher $\phi(n)$ zu bestimmen.

Timing-Angriffe

- Paul Kocher, ein IT-Sicherheits-Berater, demonstrierte, dass ein Schnüffler einen privaten Schlüssel ermitteln kann, indem er verfolgt, wie lange ein Computer braucht, um Nachrichten zu entschlüsseln.
- Diese Angriffe sind nicht nur auf RSA, sondern auch auf andere Verschlüsselungssysteme mit öffentlichen Schlüsseln anwendbar.
- Solche Angriffe sind aus zwei Gründen alarmierend:
  - Es kommt aus einer völlig unerwarteten Richtung.
  - Es handelt sich um einen reinen Chiffretext-Angriff.

Gegenmaßnahmen gegen Timing-Angriffe

Konstante Potenzierungszeit:
Es gilt sicherzustellen, dass alle Potenzierungen die gleiche Zeit benötigen, bevor ein Ergebnis zurückgegeben wird; dies ist eine einfache Lösung, die jedoch die Leistung beeinträchtigt.

Zufällige Verzögerung:
Eine bessere Leistung könnte erreicht werden, indem man dem Potenzierungsalgorithmus eine zufällige Verzögerung hinzufügt, um den Zeitangriff zu verwirren.

Verschleierung: Multiplikation des Chiffriertextes mit einer Zufallszahl vor der Potenzierung; dieser Vorgang verhindert, dass der Angreifer erfährt, welche Bits des Chiffriertextes im Computer verarbeitet werden, und verhindert somit die für den Timing-Angriff erforderliche Bit-für-Bit-Analyse.

Fehlerbasierter Angriff

- Ein Angriff auf einen Prozessor, der digitale RSA-Signaturen erzeugt.
  - Verursacht Fehler in der Signaturberechnung, indem er die Leistung des Prozessors reduziert.
  - Diese Fehler führen dazu, dass die Software ungültige Signaturen erzeugt, die dann vom Angreifer analysiert werden können, um den privaten Schlüssel wiederherzustellen.
- Der Angriffsalgorithmus besteht darin, Ein-Bit-Fehler zu erzeugen und die Ergebnisse zu beobachten.
- Obwohl dieser Angriff eine Überlegung wert ist, scheint er in vielen Anwendungen keine ernsthafte Bedrohung für RSA darzustellen.
  - Er setzt voraus, dass der Angreifer physischen Zugriff auf den Zielcomputer hat und in der Lage ist, die Eingangsleistung des Prozessors direkt zu kontrollieren.

(*Fault-based attack*)

Gewählter Chiffretext-Angriff

( *Chosen Ciphertext Attack (CCA)*)

- Der Angreifer wählt eine Reihe von Chiffretexten aus und erhält dann die entsprechenden Klartexte, die mit dem privaten Schlüssel des Ziels entschlüsselt wurden.
  - Der Angreifer könnte also einen Klartext auswählen, ihn mit dem öffentlichen Schlüssel des Ziels verschlüsseln und dann den Klartext zurückerhalten, indem er ihn mit dem privaten Schlüssel entschlüsselt.
  - Der Angreifer macht sich die Eigenschaften von RSA zunutze und wählt Datenblöcke aus, die, wenn sie mit dem privaten Schlüssel des Ziels verarbeitet werden, die für die Kryptoanalyse benötigten Informationen liefern.
- Um solche Angriffe abzuwehren, empfiehlt RSA Security Inc., den Klartext mit einem Verfahren zu modifizieren, das als optimales asymmetrisches Verschlüsselungs-Padding (OAEP) bekannt ist.

Die Idee bei OAEP ist, dass der Klartext vor der Verschlüsselung mit dem öffentlichen Schlüssel des Empfängers mit einem *zufälligen* Padding versehen wird, um ein Element des Zufalls in den sonst deterministischen Verschlüsselungsvorgang einzuführen.

Zusammenfassung - Hashes, Macs und digitale Signaturen

- Hashes dienen der Gewährleistung der Integrität von Daten.
- Macs dienen der Authentifizierung von Daten. Da jedoch ein gemeinsamer Schlüssel vom Sender und Empfänger verwendet wird, können beide Seiten Nachrichten fälschen. Sie bieten keine Nichtabstreitbarkeit.
- Digitale Signaturen bieten Integrität, Authentizität und Nichtabstreitbarkeit. Sie basieren auf asymmetrischen Verschlüsselungsalgorithmen und sind daher langsamer als Macs.

Übung

2.1. Square-and-Multiply

Führen Sie den Square-and-Multiply Algorithmus Schritt-für-Schritt für $3^{17}\ mod\ 23$ aus.

2.2. Nachrichtenentschlüsselung

Entschlüsseln sie die folgende mit RSA verschlüsselte Nachricht

$$C = 70789294130501 \qquad (\text{Verschlüsselte Nachricht})$$
$$n = 2000557908870247 \qquad \phi(n) = 2000557818857736$$
$$e = 65537$$

Berechnen Sie d und wandeln Sie die (Klartext)zahl in Text (ASCII 7-Bit pro Zeichen) um. (Nutzen Sie ggf. das Jupyter Notebook als Hilfestellung.)

Um einen Integer-Wert ($\mathbb{m}$) in einen String umzuwandeln, können Sie den folgenden Pyhton-Code verwenden:

```
bstr = bin(m) # the string will start with '0b'
chars = [bstr[i:i+7] for i in range(2, len(bstr)-1, 7)] # Segmentierung in 7-Bit-Blöcke
"".join(list(map(lambda x : chr(int(x,2)), chars))) # Umwandlung in ASCII-Zeichen und Konkatenation
```

Übung

2.3. Verschlüsselung mit RSA

Verschlüsseln Sie eine Nachricht mit RSA mit selber gewählten Parametern.

D. h., wählen Sie 2 kleine Primzahlen, berechnen Sie dann e, d, n. Verschlüsseln Sie dann die Nachricht (d. h. einen (eher) kleinen Wert) mit dem öffentlichen Schlüssel einer anderen Person und senden Sie der Person die verschlüsselte Nachricht. Die Zielperson soll Ihre Nachricht entschlüsseln.