



RSA-Verschlüsselung

Informatikseminar WS 08/09 – Codierung

Daniel Wischer - inf6501



- Einleitung und Geschichte der RSA-Verschlüsselung
- Asymmetrische vs. Symmetrische Verschlüsselung
- Mathematische Grundlage: Einwegfunktionen
- Bestandteile von RSA und Erzeugung der Schlüssel
- Ablauf der Verschlüsselung
- Sicherheit und Korrektheit von RSA
- Digitale Signaturen mit RSA
- Praktische Anwendungen



- Entwickelt im Jahr 1977 von Ron **R**ivest, Adi **S**hamir und Len **A**dleman
- Erstes veröffentlichtes asymmetrisches Verschlüsselungsverfahren
- 1983 zum Patent angemeldet
- 2000 ist das Patent ausgelaufen. Implementierungen des RSA-Verfahrens sind daher jetzt gebührenfrei



Asymetrische Verschlüsselung

- Öffentlicher und privater Schlüssel vorhanden
- Kaum Aufwand für den Austausch der Schlüssel
- Wenige Verfahren vorhanden
- Laufzeit: langsam

Symetrische Verschlüsselung

- Nur 1 Schlüssel zum Verschlüsseln und Entschlüsseln
- Hoher Aufwand nötig für das Verteilen der Schlüssel
- Viele Verfahren vorhanden
- Laufzeit: schnell



- Einwegfunktionen sind spezielle Funktionen, deren inverse Funktion nur schwer errechenbar ist
- Für einen Wert x lässt sich der Funktionswert y effizient berechnen, die Umkehrfunktion ist nicht in Polynomialzeit berechenbar
- Beispiel: Produkt von 2 Primzahlen leicht berechenbar. Die Faktorisierung einer großen Zahl nicht effizient lösbar → Faktorisierungsproblem
- Spezialfall Trapdoor-Einwegfunktionen: Einwegfunktionen bei denen mit Hilfe einer Trapdoor die Umkehrfunktion berechenbar wird



- **n** RSA-Modul, Bestandteil des öffentlichen und des privaten Schlüssels
- **e** Encrypt, bildet zusammen mit dem RSA-Modul den öffentlichen Schlüssel
- **d** Decrypt, bildet zusammen mit dem RSA-Modul den privaten Schlüssel
- **p, q** 2 Primzahlen zum Berechnen der Schlüssel

Public-Key: (e, n)

Private-Key: (d, n)



1. Schritt: Wahl von 2 zufälligen, möglichst großen Primzahlen p, q
 - Berechnung z.B. mit Sieb des Eratosthenes
2. Schritt: Berechnung des RSA-Moduls n mit $n = p * q$
3. Schritt: Berechnung der Eulerzahl von n mit $\Phi(n) = (p-1)(q-1)$
 - Eulerzahl von x ist die Anzahl der Zahlen die zu x teilerfremd sind
 - Beispiel: $\Phi(8) = 4$, weil 1, 3, 5, 7 teilerfremd sind
 - Wenn x eine Primzahl, dann $\Phi(x) = x-1$



4. Schritt: Auswählen einer zu $\Phi(n)$ teilerfremden Zahl e
 - Zur Prüfung: Berechnen des $\text{ggT}(\Phi(n), e) = 1$

 - Mögliche Strategie: Wählen einer bel. Zahl, wenn nicht teilerfremd \rightarrow Teilen durch berechneten ggT , dies führt immer zu einer teilerfremden Zahl

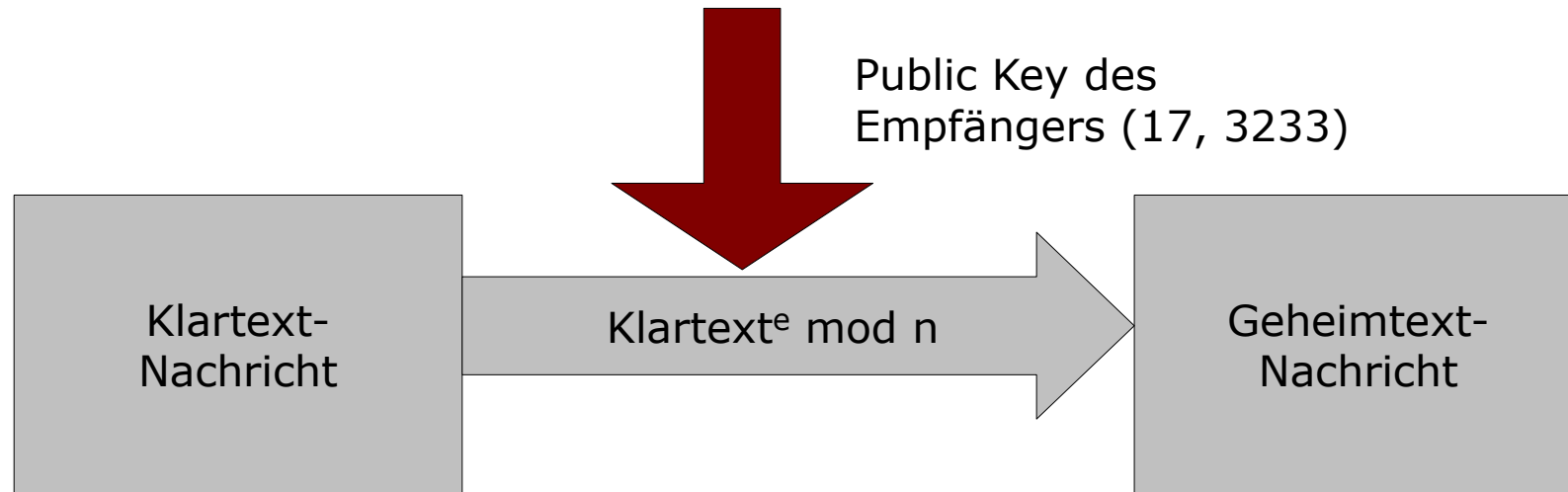
5. Schritt: Berechnen der modular Inversen von e zu $\Phi(n)$
 - Bedingung: $e * d \bmod \Phi(n) = 1$

 - Berechnung erfolgt mit dem Erweiterten Euklidischen Algorithmus



- Verschlüsselung mit der Funktion: Geheimnachricht = $m^e \bmod n$
- Voraussetzung:
 - Nachricht m wird zum Verschlüsseln als Zahl betrachtet
 - Empfänger hat seinen öffentlichen Schlüssel bei einer Schlüsselstelle hinterlegt
- e und n sind dabei der öffentliche Schlüssel des Empfängers
- Bedingung: Nachricht m muss vom Zahlenwert echt kleiner als n sein
→ Immer erreichbar durch Blockung der Nachricht
- Verschlüsselung für den Absender nicht umkehrbar. Einmal verschlüsselt kann nur der Empfänger die Nachricht entschlüsseln

Verschlüsselung einer Nachricht



Nachricht:

$$m = 123$$

Verschlüsselung:

$$123^{17} \bmod 3233 = 855$$

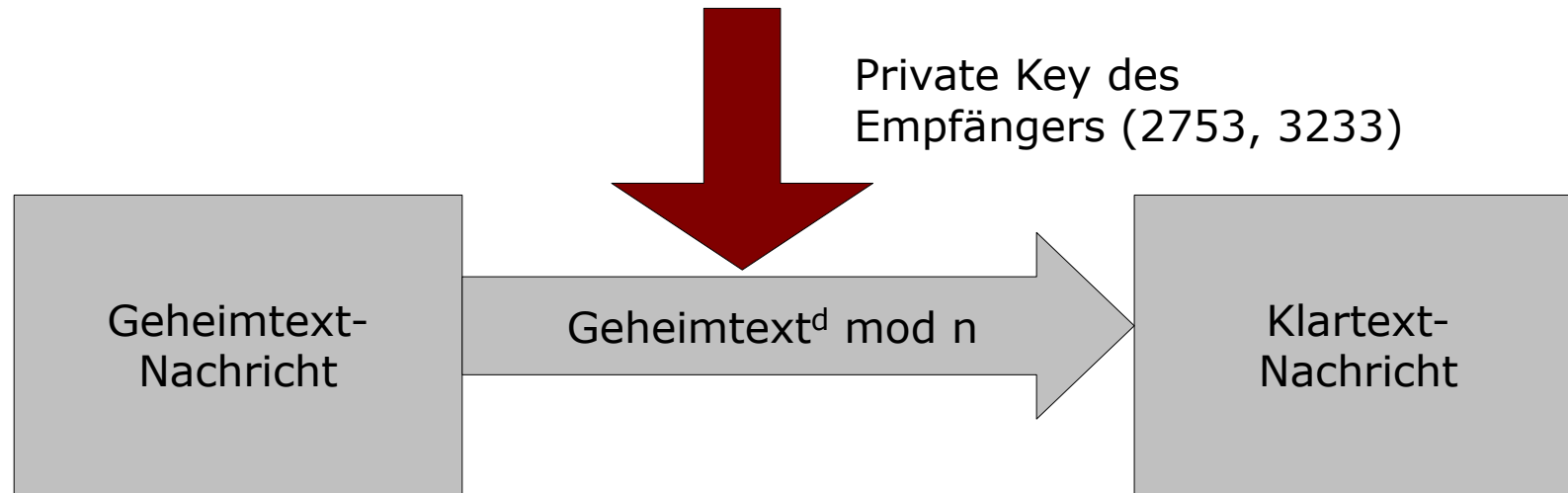
Geheimtext:

$$G = 855$$



- Entschlüsselung mit der Funktion: Klartext = $G^d \bmod n$
- d und n sind der private Schlüssel des Empfängers
- Gleiche Formel wie beim Verschlüsseln → Nur 1 Algorithmus für Ver- und Entschlüsselung nötig
- d ist dabei der Falltürschlüssel für die Einwegfunktion „Potenzieren modulo n “
- Warum funktioniert dies immer? Beweis später im Vortrag

Entschlüsselung einer Nachricht



Geheimtext:

$$m = 855$$

Verschlüsselung:

$$855^{2753} \bmod 3233 = 123$$

Klartext:

$$G = 123$$



1. Max möchte mit RSA verschlüsselte Nachrichten erhalten und lässt sich von einer Schlüsselstelle die Schlüssel berechnen
2. Anne möchte Max eine verschlüsselte Nachricht schicken. Sie ruft den öffentlichen Schlüssel von Max von der Schlüsselstelle ab und verschlüsselt die Nachricht
3. Max erhält die verschlüsselte Nachricht von Anne und entschlüsselt diese wieder mit seinem privaten Schlüssel

Problem?: Es kann hier nur sichergestellt werden, dass keiner die verschlüsselte Nachricht verfälscht hat. Allerdings kann Max nicht sicherstellen, dass die Nachricht tatsächlich von Anne stammt.

Lösung: Digitale Signaturen → Hierzu später mehr



- Um die Korrektheit des RSA-Verfahrens zu beweisen, muss folgendes gelten: $(m^e)^d \bmod n = m$
- Laut Schlüsselerstellungsvorschrift gilt: $e * d \bmod (p-1)(q-1) = 1$
- Einführen eines Faktors j um $\bmod n$ aufzuheben und Umformen der Gleichung: $e * d = 1 + j * (p-1)(q-1)$
- Ersetzen von $e * d$: $(m^e)^d = m^{ed} = m * (m^{(p-1)})(q-1)^j$
- Daraus folgt folgende Kongruenzgleichung, die bewiesen werden muss:

$$\text{Zu Zeigen: } (m^e)^d \equiv m * (m^{(p-1)})(q-1)^j \equiv m \bmod p$$

Beweis durch Fallunterscheidung

Zu Zeigen: $(m^e)^d \equiv m * (m^{(p-1)})^{(q-1)j} \equiv m \pmod{p}$

1. Fall: p ist ein Teiler von m :

Wenn p m teilt, ist die Kongruenz auf beiden Seiten 0

→ Kongruenzgleichung gilt

2. Fall: p ist kein Teiler von m :

Wenn p m nicht teilt, folgt die Kongruenz aus dem Satz von Fermat:

$$a^{(p-1)} \pmod{p} = 1 \rightarrow m * 1^{(q-1)j}$$

Da folgende Äquivalenz gilt: $x \equiv m \pmod{y} \iff x = m \pmod{y} * z$

gilt demnach auch $(m^e)^d \pmod{n} = m$, da p und q Primfaktoren von n sind

→ q.e.d



- Sicherheit von RSA basiert auf dem Faktorisierungsproblem
- Faktorisierungsproblem: Zerlegung einer Zahl in seine Primfaktoren
- Faktorisierung des RSA-Moduls n würde sofort zum geheimen Schlüssel d führen
- RSA Factoring Challenge hat als bisher größte Zahl RSA-640 faktorisiert. Aufwand: 30 2,2GHz Prozessoren brauchten über 5 Monate um RSA-640 zu faktorisieren
(Quelle: RSA Laboratories)
- Sobald ein Verfahren zur effizienten (in Polynomialzeit) Berechnung der Faktorisierung gefunden wird ist RSA wertlos
- Weitere Angriffsmöglichkeiten vorhanden (Brute-Force, Wurzelziehen modulo n), allerdings alle ineffizienter als Faktorisierung

Sicherheit der RSA-Verschlüsselung (Beispiel)



Bekannt: Öffentlicher Schlüssel (299, 5)

Der Angreifer faktorisiert 299 \rightarrow $p=13$ $q=23$ und berechnet damit $\Phi(299) = (13-1)*(23-1) = 264$

Nun kann er mit dem Erweiterten Euklidischen Algorithmus d berechnen:

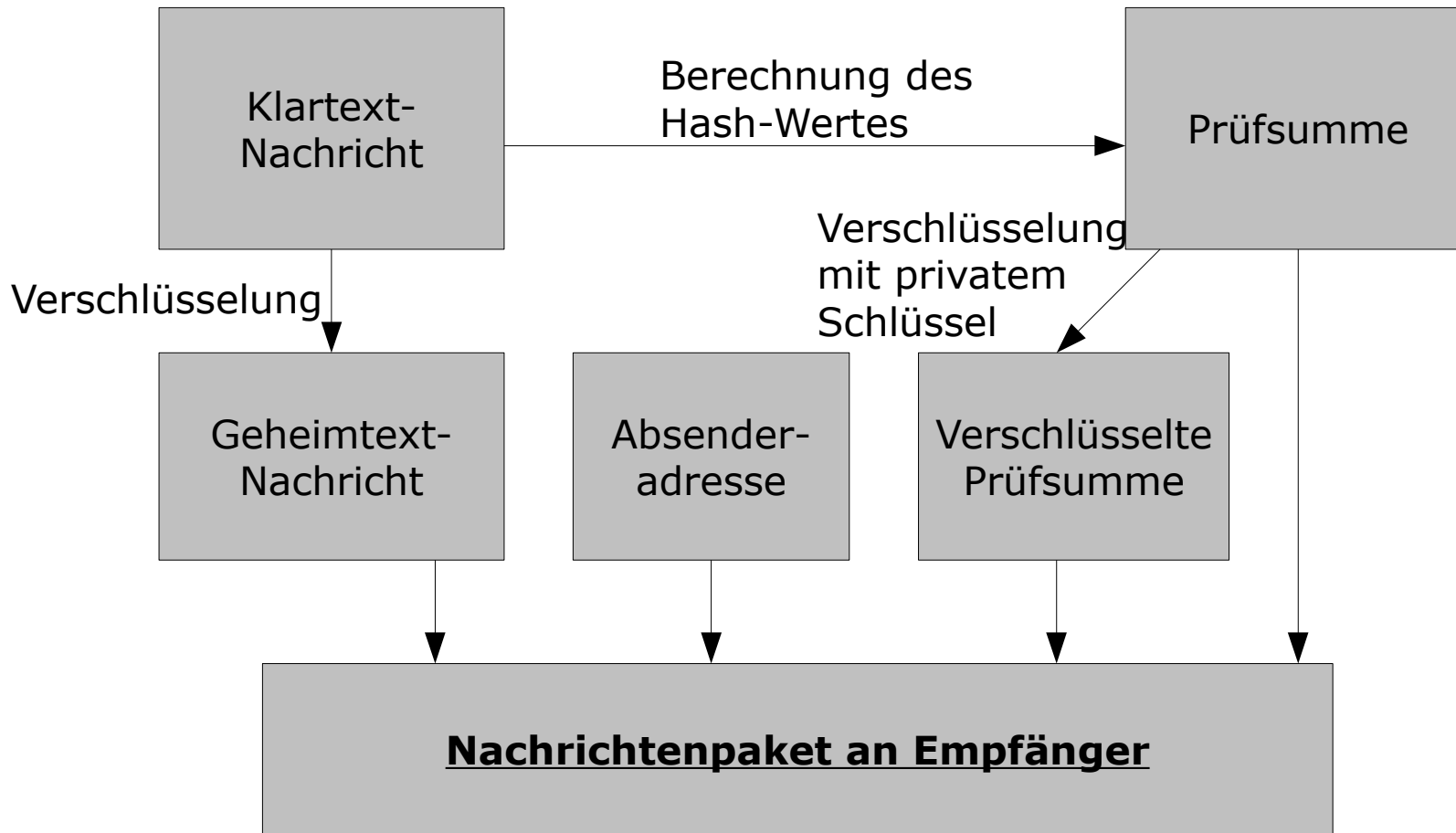
a	b	q	s	t
264	5	52	-1	53
5	4	1	1	-1
4	1	4	0	1
1	0		1	0

Damit hat der Angreifer den privaten Schlüssel errechnet und erhält Zugriff auf den Geheimtext



- Für das Standard-RSA-Verfahren musste nur der Empfänger einen RSA-Schlüssel besitzen
- Für eine digitale Signatur einer Nachricht müssen beide, Sender und Empfänger, einen RSA-Schlüssel besitzen
- Für die Signatur wird die Austauschbarkeit der Exponenten bei der Verschlüsselungsfunktion genutzt
- $(m^e)^d \bmod n = m \iff (m^d)^e \bmod n = m$
- Zusätzlich zum RSA-Algorithmus wird eine Hash-Funktion benötigt (z.B.: md5, sha, etc.)

Digitale Signatur mit dem RSA-Verfahren





- Absender:
- Erzeugt zusätzlich eine Prüfsumme der Nachricht
 - Verschlüsselt diese mit seinem eigenen privaten Schlüssel
 - Verschickt verschlüsselte Nachricht, Prüfsumme, Absenderadresse und verschlüsselte Prüfsumme
- Empfänger:
- Holt den öffentlichen Schlüssel des Absenders anhand der Absenderadresse aus dem Schlüsselverzeichnis
 - Entschlüsselt die verschlüsselte Prüfsumme mit dem öffentlichen Schlüssel des Empfängers
 - Vergleich der beiden Prüfsummen: Sind beide gleich wurde die Nachricht nicht verändert und kann nur vom angegebenen Absender stammen



- Digitale Signatur ist derzeit das Haupteinsatzgebiet der RSA-Verschlüsselung
- Einfach zu nutzen, da keine zusätzlichen Algorithmen entwickelt werden müssen
- Wenn 2 Kommunikationspartner schon Teilnehmer der RSA-Verschlüsselung sind, kann automatisch die digitale Signatur verwendet werden
- Zusätzlich zur Signatur erhält man eine Error-Detection für die versendete Nachricht. Stimmen die Prüfsummen nicht überein, ist die Nachricht verfälscht worden



- Haupteinsatzgebiete: - Digitale Signatur
 - Schlüsselaustausch für symmetrische Verschlüsselung
- RSA wird in der heutigen Zeit nicht mehr zur Verschlüsselung großer Datenmengen genutzt, da bisher keine effiziente Hardwareimplementierung gefunden wurde
- Anwendungen: - EMV (Europay MasterCard Visa)
 - PGP (Pretty Good Privacy)
 - X.509 Zertifikate
 - S-HTTP



- Gutes, einfaches und sicheres Verschlüsselungsverfahren
- Einfache Implementierung des Verfahrens möglich
- Leider keine Hardwareimplementierung vorhanden, daher nicht für große Datenmengen geeignet

Danke für ihre Aufmerksamkeit