Kryptologie

 RSA & Digitale Signaturen

Cyril Blum

Fachschaft Informatik
Kantonsschule Im Lee

Symmetrische vs. Asymmetrische Kryptosysteme

Symmetrische Kryptosysteme (z.B. Caesar, Vigenère, OTP)

Verwenden **demselben Schlüssel** zur Ver- und Entschlüsselung. Problem: Wie tauscht man den Schlüssel sicher aus?

Symmetrische vs. Asymmetrische Kryptosysteme

Symmetrische Kryptosysteme (z.B. Caesar, Vigenère, OTP)

Verwenden **demselben Schlüssel** zur Ver- und Entschlüsselung. Problem: Wie tauscht man den Schlüssel sicher aus?

Idee (Asymmetrische Kryptosysteme (Public-Key Kryptologie))

Jede Person generiert ein Schlüsselpaar:

- **Öffentlicher Schlüssel** (🔓): Wird allen bekannt gemacht. Jeder kann damit Nachrichten für den Empfänger verschlüsseln.

Symmetrische vs. Asymmetrische Kryptosysteme

Symmetrische Kryptosysteme (z.B. Caesar, Vigenère, OTP)

Verwenden **demselben Schlüssel** zur Ver- und Entschlüsselung. Problem: Wie tauscht man den Schlüssel sicher aus?



Idee (Asymmetrische Kryptosysteme (Public-Key Kryptologie))

Jede Person generiert ein Schlüsselpaar:

- **Öffentlicher Schlüssel** (🔓): Wird allen bekannt gemacht. Jeder kann damit Nachrichten für den Empfänger verschlüsseln.
- **Privater Schlüssel** (🔑): Bleibt absolut geheim. Nur der Empfänger kann damit die Nachrichten entschlüsseln.

Prinzip der Public-Key-Verschlüsselung

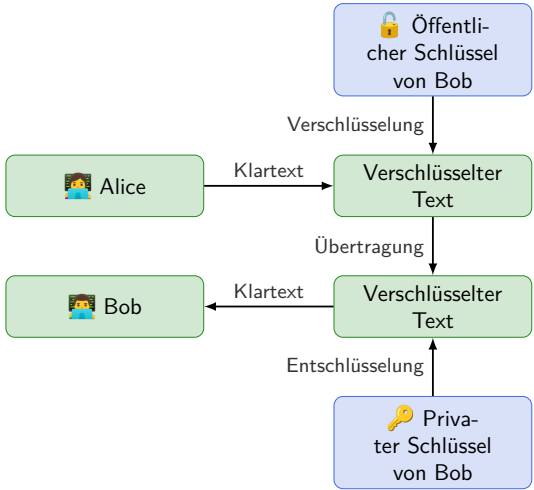


Abbildung: Prinzip der Public-Key-Verschlüsselung

Das RSA-Verfahren

1977 von Ron Rivest, Adi Shamir und Leonard Adleman erfunden.

Definition (Mathematisches Fundament von RSA)

Es basiert auf der Schwierigkeit der **Primfaktorzerlegung**:

- Zwei grosse Primzahlen p und q miteinander zu multiplizieren ($n = p \cdot q$) ist extrem einfach.
- Aus dem Produkt n wieder die Primfaktoren p und q zu finden, ist für ausreichend grosse Zahlen praktisch unmöglich!

RSA Schlüsselerzeugung: Theorie & Beispiel

Theorie / Algorithmus

1. Wähle zwei Primzahlen p und q.
2. Modul: $n = p \cdot q$
3. Phi-Wert: $\varphi(n) = (p-1)(q-1)$
4. Wähle e mit $\text{ggT}(e, \varphi(n)) = 1$.
5. Berechne d mit:

$$(e \cdot d) \equiv 1 \pmod{\varphi(n)}$$

6. **Public**: (e, n), **Private**: (d, n).

Zahlenbeispiel

- Wähle $p = 3,\ q = 11$.
- Modul: $n = 3 \cdot 11 = 33$
- $\varphi(33) = (3-1)(11-1) = 20$
- Wähle $e = 3$ (da $\text{ggT}(3, 20) = 1$).
- Bestimme d:

$$3 \cdot d \equiv 1 \pmod{20} \implies d = 7$$

- **Public**: $(3, 33)$, **Private**: $(7, 33)$.

RSA Ver- und Entschlüsselung: Theorie & Beispiel

Theorie

Public Key (e, n), Private Key (d, n).

1. Verschlüsselung (Absender):

- Klartext als Zahl $m < n$.
- Chiffrat berechnen:

$$c = m^e \bmod n$$

2. Entschlüsselung (Empfänger):

- Mit privatem Schlüssel d:

$$m = c^d \bmod n$$

Garantie: $m \equiv (m^e)^d \pmod{n}$ (Euler).

Zahlenbeispiel

Schlüssel: Public $(3, 33)$, Private $(7, 33)$.

1. Verschlüsselung von Klartext $m = 3$:

$$c = 3^3 \bmod 33 = 27 \bmod 33 = \mathbf{27}$$

2. Entschlüsselung von Chiffrat $c = 27$:

$$m = 27^7 \bmod 33 = \mathbf{3} \quad \checkmark$$

Auftrag: RSA-Verfahren

Skript

- Aufgabe 4.1 (RSA Schlüsselerzeugung & Verschlüsselung)
- Aufgabe 4.2 (Sicherheit)
- Challenge: 4.3–4.5: RSA zu zweit und mit Python für eigene Nachrichten
- Challenge: 4.6: Satz von Euler nachrechnen

Digitale Signaturen mit RSA

Neben der Verschlüsselung können asymmetrische Verfahren auch für **digitale Signaturen** genutzt werden.

Digitale Signaturen mit RSA

Neben der Verschlüsselung können asymmetrische Verfahren auch für **digitale Signaturen** genutzt werden.

Definition (Ziele digitaler Signaturen)

Digitale Signaturen mit RSA

Neben der Verschlüsselung können asymmetrische Verfahren auch für **digitale Signaturen** genutzt werden.

Definition (Ziele digitaler Signaturen)

- **Authentizität**: Garantiert die Identität der absendenden Person (nur der Besitzer des passenden privaten Schlüssels konnte die Signatur erzeugen).

Digitale Signaturen mit RSA

Neben der Verschlüsselung können asymmetrische Verfahren auch für **digitale Signaturen** genutzt werden.

Definition (Ziele digitaler Signaturen)

- **Authentizität**: Garantiert die Identität der absendenden Person (nur der Besitzer des passenden privaten Schlüssels konnte die Signatur erzeugen).
- **Integrität**: Garantiert, dass die Nachricht auf dem Übertragungsweg nicht verändert oder manipuliert wurde.

Digitale Signaturen mit RSA

Neben der Verschlüsselung können asymmetrische Verfahren auch für **digitale Signaturen** genutzt werden.

Definition (Ziele digitaler Signaturen)

- **Authentizität**: Garantiert die Identität der absendenden Person (nur der Besitzer des passenden privaten Schlüssels konnte die Signatur erzeugen).
- **Integrität**: Garantiert, dass die Nachricht auf dem Übertragungsweg nicht verändert oder manipuliert wurde.
- **Verbindlichkeit (Nicht-Abstreitbarkeit)**: Die Urheberschaft kann im Nachhinein nicht geleugnet werden.

RSA Digitale Signatur: Theorie & Beispiel

Theorie: RSA-Signatur

Rolle der Schlüssel vertauscht:

1. Signatur erstellen (Alice):

$$s = m^{d_A} \bmod n_A$$

$\rightarrow$ Alice sendet (m, s) an Bob.

2. Signatur prüfen (Bob):

$$m' = s^{e_A} \bmod n_A$$

- $m' = m \implies$ Authentisch & echt! ✓
- $m' \neq m \implies$ Manipulation erkannt! ✗

Zahlenbeispiel (Skript)

Alice: $(d, n) = (9677, 12317)$, $(e, n) = (5, 12317)$.

1. Signaturerstellung $(m = 42)$:

$$s = 42^{9677} \bmod 12317 = \mathbf{161}$$

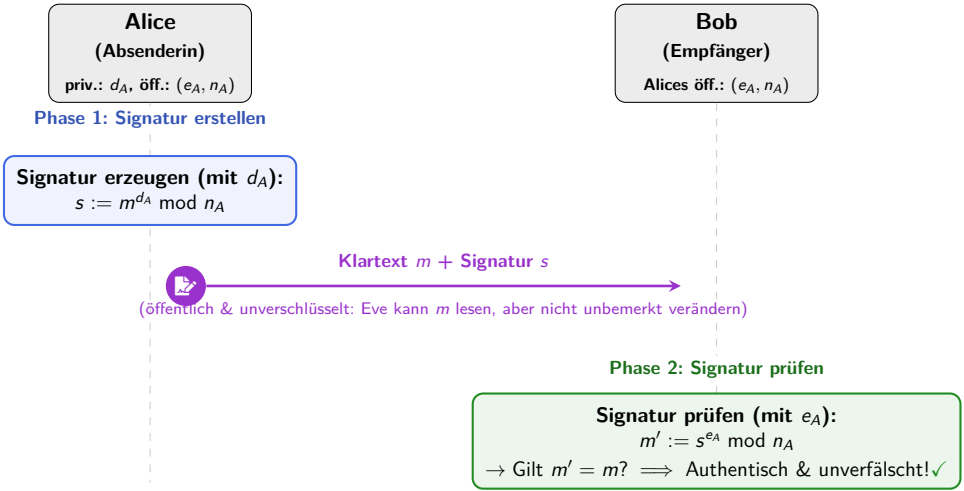
2. Signaturprüfung durch Bob:

$$m' = 161^5 \bmod 12317 = \mathbf{42} = m \quad ✓$$

Erkennung von Manipulation:
Eve ändert Nachricht zu $m = 43$ $(s = 161)$:

$161^5 \bmod 12317 = 42 \neq 43 \implies$ Abgelehnt!

Ablauf: Digitale Signatur



Auftrag: Digitale Signatur

Aufgabe 4.7 im Skript (RSA Digitale Signatur)

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von** m:

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von m:**
  1. RSA kann nur Zahlen $m < n$ verarbeiten. Reale Dokumente (PDFs, Verträge, E-Mails) sind viele Megabytes gross.

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von m:**
  1. RSA kann nur Zahlen $m < n$ verarbeiten. Reale Dokumente (PDFs, Verträge, E-Mails) sind viele Megabytes gross.
  2. Modulare Potenzierung mit grossen Exponenten ist rechenintensiv und langsam.

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von m**:
  1. RSA kann nur Zahlen $m < n$ verarbeiten. Reale Dokumente (PDFs, Verträge, E-Mails) sind viele Megabytes gross.
  2. Modulare Potenzierung mit grossen Exponenten ist rechenintensiv und langsam.
- **Lösung: Hashwert signieren**:

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von m:**
  1. RSA kann nur Zahlen $m < n$ verarbeiten. Reale Dokumente (PDFs, Verträge, E-Mails) sind viele Megabytes gross.
  2. Modulare Potenzierung mit grossen Exponenten ist rechenintensiv und langsam.
- **Lösung: Hashwert signieren**:
  - Eine **kryptographische Hashfunktion** (z. B. **SHA-256**) komprimiert ein beliebig grosses Dokument auf einen kurzen, eindeutigen Fingerabdruck fester Länge (z. B. 256 Bit).

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von m:**
  1. RSA kann nur Zahlen $m < n$ verarbeiten. Reale Dokumente (PDFs, Verträge, E-Mails) sind viele Megabytes gross.
  2. Modulare Potenzierung mit grossen Exponenten ist rechenintensiv und langsam.
- **Lösung: Hashwert signieren**:
  - Eine **kryptographische Hashfunktion** (z. B. **SHA-256**) komprimiert ein beliebig grosses Dokument auf einen kurzen, eindeutigen Fingerabdruck fester Länge (z. B. 256 Bit).
  - **Eigenschaften**: Einwegfunktion (nicht umkehrbar), kollisionsresistent, Lawineneffekt.

Praxis-Problem & Lösung: Kryptographische Hashfunktionen

- **Problem beim direkten Signieren von m:**
  1. RSA kann nur Zahlen $m < n$ verarbeiten. Reale Dokumente (PDFs, Verträge, E-Mails) sind viele Megabytes gross.
  2. Modulare Potenzierung mit grossen Exponenten ist rechenintensiv und langsam.
- **Lösung: Hashwert signieren**:
  - Eine **kryptographische Hashfunktion** (z. B. **SHA-256**) komprimiert ein beliebig grosses Dokument auf einen kurzen, eindeutigen Fingerabdruck fester Länge (z. B. 256 Bit).
  - **Eigenschaften**: Einwegfunktion (nicht umkehrbar), kollisionsresistent, Lawineneffekt.
- **Ablauf in der Praxis**:

$$\text{Alice signiert:} \quad s := H(m)^{d_A} \bmod n_A$$

$$\text{Bob prüft:} \quad s^{e_A} \bmod n_A \stackrel{?}{=} H(m)$$

Vertraulichkeit vs. Authentizität: Sign-then-Encrypt

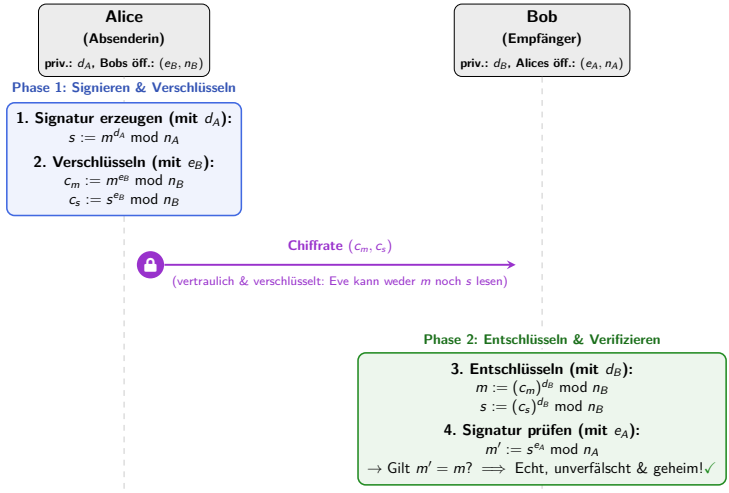
Nur Signatur

- Alice sendet Klartext m + Signatur s.
- **Authentizität:** Bob weiss, dass m von Alice stammt.
- **Integrität:** m wurde nicht verändert.
- **Keine Vertraulichkeit:** Jede Person auf dem Kanal kann m lesen.

Sign-then-Encrypt

- 1. Alice signiert m mit $d_A \rightarrow s$.
- 2. Alice verschlüsselt (m, s) mit Bobs Schlüssel $e_B \rightarrow (c_m, c_s)$.
- 3. Bob entschlüsselt mit d_B.
- 4. Bob verifiziert Signatur mit e_A.
- $\rightarrow$ **Vertraulich & Authentisch!**

Ablauf: Sign-then-Encrypt



Auftrag: Sign-then-Encrypt

Aufgabe 4.8 im Skript (Sign-then-Encrypt)