Asymmetrische Kryptographie in Java

Sichere verteilte Anwendungen mit Java

A. H. W. Lindemann, N. Vetter

Institut für Informatik

10. Januar 2014

Gliederung

Übersicht

Ziele

- verbergen des Inhaltes einer Nachricht durch:
  - Transformieren von Klartext in Kryptotext
  - späteres Entschlüsseln von Kryptotext in Klartext

Kerckhoffs Prinzip:

„Sicherheit ist allein vom Schlüssel und nicht vom Verfahren abhängig.“

Asymmetrische Kryptographie

- entstand Mitte der 1970ger Jahre
- von Ralph Merkle sowie Diffie und Hellmann entwickelt

Grundidee:

- öffentlicher und privater Schlüssel (Schlüsselpaar)

Anforderungen:

- Schlüssel müssen leicht zu generieren sein
- privater Schlüssel darf nicht unter vertretbarem Aufwand aus öffentlichen Schlüssel zu berechnen sein
- Ver- und Entschlüsselung müssen effizient berechenbar sein

Ansatz

$\Rightarrow$ Einwegfunktionen. Definition:

- injektive Funktion $f : X \to Y$
- $\forall x \in X$ ist f(x) „effizient“ zu berechnen
- aus Bild y = f(x) darf Urbild x nicht effizient berechnet werden können
- Bsp.: Faktorisierungsproblem, Diskreter Logarithmus

Lösung

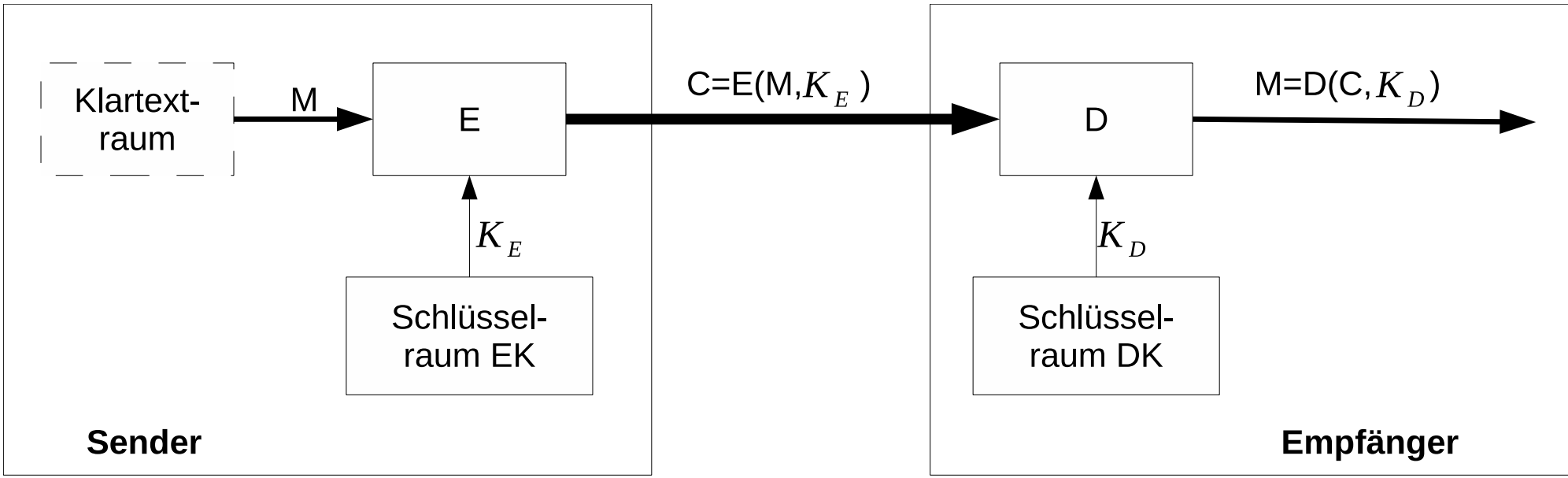
$\Rightarrow$ Einwegfunktionen mit Falltür.

- injektive Funktion $f : X \to Y$
- $\forall x \in X$ ist f(x) effizient zu berechnen
- aus Bild y = f(x) darf Urbild x nur „effizient“ berechnet werden können, wenn Zusatzinformationen verfügbar sind.
- Bsp.: h-te Potenz mod(n), Zusammengesetzter mod(n)

Bestandteile

- Tupel $= (M, C, EK, DK, E, D)$
- 2 endliche Alphabete (A_1, A_2)
- Klartext $(M \subseteq A_1^* \backslash \emptyset)$
- Kryptotext $(C \subseteq A_2^* \backslash \emptyset)$
- Verschlüsselungsschlüsselraum $(EK \backslash \emptyset)$
- Entschlüsselungsschlüsselraum
  $(DK/\emptyset \ mit \ f : EK \rightarrow DK \ und \ f(K_E) = K_D)$
- Verschlüsselungsverfahren $(E : \ MxEK \rightarrow C)$
- Entschlüsselungsverfahren $(D : \ CxDK \rightarrow M)$
- Es gilt: $\forall M : D(E(M, K_E), K_D) = M$

Erläuterung



(nach Eckert[1])

Vertreter

- RSA
- Diffie-Hellman
- DSA
- ElGamal

Findet Anwendung bei:

- PGP
- SSL/TLS (SSH, HTTPS, ...)

Vor-/Nachteile

Vorteil:

- kein Schlüsselaustausch notwendig

Nachteil:

- hohe Komplexität der Berechnungen
  $\rightarrow$ unter Umständen langsam
- Authentizität des öff. Schlüssels nicht garantiert
  $\rightarrow$ „man in the middle“

Übersicht

Erzeugung

2 Möglichkeiten zur Erzeugung:

- Nutzer erzeugt Schlüsselpaar selbst
  (oder lässt ein Token ein Schlüsselpaar erzeugen)
  - Nutzer hat volle Kontrolle über den Prozess
  - Verantwortung liegt allein beim Nutzer
    (Aufbewahrung, Sicherung, ...)
  - heterogene Umgebung zur Generierung
- zentrale Instanz stellt Schlüsselpaar bereit (z.B. CA)
  - Nutzer hat keine volle Kontrolle über den Prozess
  - Schlüsselmaterial kann gesichert werden
    (siehe Schlüsselwiederherstellung)
  - Transport des Schlüsselpaares kritisch

Speicherung und Verbreitung

Ansatz:

- keine Instanz außer dem Besitzer darf den priv. Schlüssel einsehen (am besten nicht mal dieser)
- öffentlicher Schlüssel soll frei verfügbar sein

Möglichkeiten:

- privater Schlüssel:
  - selbst sicher gespeichert
  - zentrale Instanz
  - Token (Bsp.: SmartCard)
- öffentlicher Schlüssel:
  - Verteilen per E-Mail oder ähnliches
  - zentrale Instanz

Wiederherstellung

Nur möglich durch Speicherung des priv. Schlüssels (oder Schlüsselmaterials)
Vorteile:

- keine neue Verteilung des öff. Schlüssels notwendig
- Wiederherstellung der verschlüsselten Daten

Nachteile:

- Schlüsselbackup ist zentraler Angriffspunkt

Übersicht

JCA vs JCE

- **J**ava **C**ryptography **A**rchitecture - Hashfunktionen, Schlüsselgeneratoren,...
- **J**ava **C**ryptography **E**xtension - Verschlüsselungsfunktionen
- Abspaltung aufgrund von Exportbeschränkungen der USA

Integrierte Provider

The SunJCE Provider

Fähigkeiten (unter anderem):

- AES, DES, ...
- RSA
- Diffie-Hellman

Externe Provider

Bouncy Castle

Aktivierung zur Laufzeit:

```
Security
  .addProvider( new BouncyCastleProvider() );
```

systemweite dauerhafte Aktivierung:

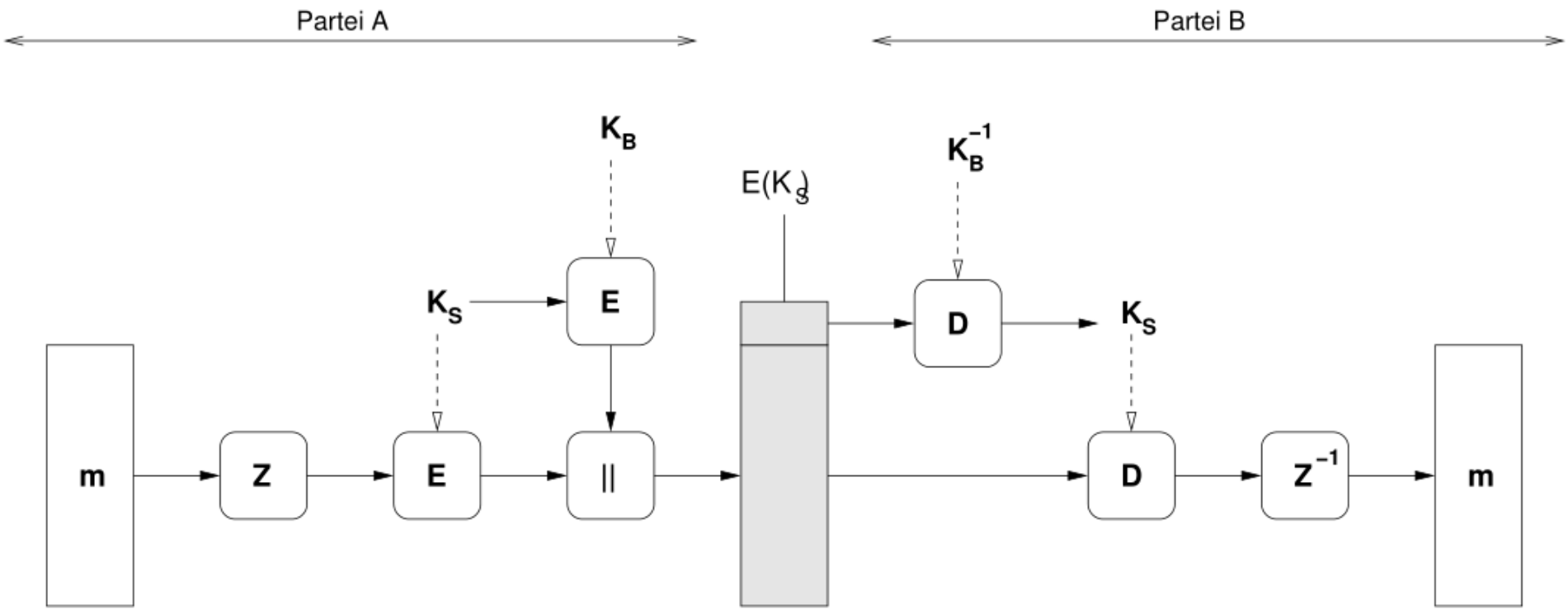
- Eintrag in die Datei: $JAVA/lib/security/java.security
  ($JAVA = Pfad zum Java Runtime Environment)

Fähigkeiten (unter anderem):

- AES, DES, ...
- RSA, ElGamal, NTRU
- Diffie-Hellman (verschiendene Varianten)

Übersicht

Hybride Kryptographie



(aus VL „Sicherheit in Rechnernetzen“ bei Prof. Dr. Bettina Schnor)

Hybride Kryptographie

Mischung aus symmetrischer und asymmetrischer Kryptographie:

- Nachricht wird symmetrisch verschlüsselt
- symmetrischer Schlüssel wird asymmetrisch verschlüsselt
- Nachricht und Schlüssel werden übertragen

Vorteile:

- benötigt weniger Rechenleistung (als vollständige asymmetrische Variante)
- brechen des Schlüssels einer Nachricht führt nicht zur Kompromittierung aller Nachrichten

Nachteil: schwieriger zu implementieren

Schlüsselgenerierung

java.security.KeyPairGenerator

```
KeyPairGenerator keyGen =
 KeyPairGenerator.getInstance( "RSA" );
keyGen.initialize( int keysize );
KeyPair keys = keyGen.generateKeyPair();
```

(Sitzungs-)Schlüsseleinigung

javax.crypto.KeyAgreement

Schlüsselaushandlung

```
KeyAgreement aKeyAgree =
   KeyAgreement.getInstance("DH", "JCE");
KeyAgreement bKeyAgree =
   KeyAgreement.getInstance("DH", "JCE");

KeyPair aPair = keyGen.generateKeyPair();
KeyPair bPair = keyGen.generateKeyPair();

aKeyAgree.init(aPair.getPrivate());
bKeyAgree.init(bPair.getPrivate());
```

(Sitzungs-)Schlüsseleinigung

javax.crypto.KeyAgreement Fortsetzung

```
aKeyAgree.doPhase(bPair.getPublic(), true);
bKeyAgree.doPhase(aPair.getPublic(), true);

SecretKey aSecret =
    aKeyAgree.generateSecret();
SecretKey bSecret =
   bKeyAgree.generateSecret();
```

(Sitzungs-)Schlüsseleinigung

javax.crypto.SecretKeyFactory

Schlüsselextraktion

```
SecretKeyFactory skf =
 SecretKeyFactory.getInstance( "AES" );
SecretKey =
 skf.generateSecret( keySpecObject );
```

Ver- und Entschlüsseln von Nachrichten

javax.crypto.Cipher

```
Cipher cipher =
 Cipher.getInstance( "RSA", "BC" );
cipher.init( Cipher.ENCRYPT_MODE, publicKey );
cipher.update( message );
byte[] crypt = cipher.doFinal();
```

Übersicht

Demo-Programm

Funktionalität

- Schlüsselpaar erzeugen
- Datei ver- und entschlüsseln (hybrid)
- symmetrischer Schlüssel wird beim Verschlüsseln erzeugt
  $\Rightarrow$ keine Schlüsseleinigung

Demo-Programm

Codereview

Codereview

[1] Claudia Eckert.
IT-Sicherheit : Konzepte - Verfahren - Protokolle.
Oldenburg, 2013.

[2] Michael Engelbrecht.
Entwicklung sicherer Software - Modellierung und Implementierung mit Java.
Spektrum, 2004.