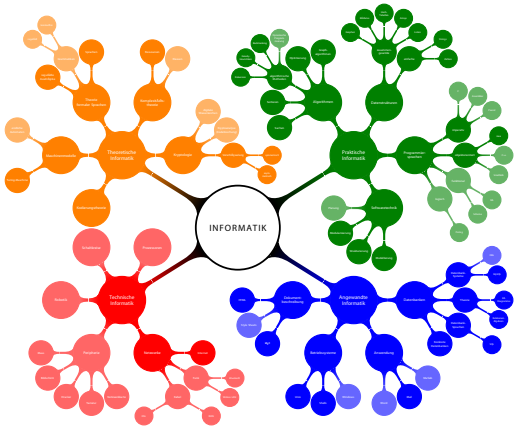
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

# Kapitel 43

# Kommunikations-Sicherheit

## Verschlüsselung: Der digitale Briefumschlag

Vorlesung Einführung in die Informatik 2 vom 8. Juli 2014 von Till Tantau



# Lernziele von Kapitel 43

1. Konzept der Verschlüsselung verstehen
2. Unterschied zwischen symmetrischen und asymmetrischen Verfahren kennen
3. Konzept der digitalen Unterschrift verstehen
4. Programme zur Verschlüsselung einsetzen können
5. Zertifikate erstellen und installieren können

# Gliederung von Kapitel 43

# Worum geht es bei IT-Sicherheit?





Bei *IT-Sicherheit* geht es um folgende Anliegen:

1. Schutz vor und die Aufrechterhaltung des Betriebs bei
   - Ausfall von Teilen des Systems (Stromausfall, Absturz)
   - Angriffen auf das System (durch Hacker, korrumpierte Mitarbeiter)

   Diese *Systemsicherheit* wird uns im nächsten Kapitel interessieren.
2. Schutz von Daten und Kommunikation vor
   - Spionage
   - Fälschung

   Diese *Daten- und Kommunikationssicherheit* wird uns in diesem Kapitel interessieren.

# Ziele der Verschlüsselung von Daten und Kommunikation







**Vertraulichkeit** Es muss sichergestellt werden, dass Daten und Kommunikation nur von »den Guten« gelesen werden können.
(Meine Daten gehen niemand etwas an.)

**Integrität** Es muss sichergestellt werden, dass Daten und Kommunikation nicht verfälscht werden können.
(Aus 1000 Euro dürfen nicht 10000 Euro werden.)

**Authentizität** Es muss sichergestellt werden, dass Daten und Kommunikation wirklich von den behaupteten Personen stammen.
(Die Email mit Alices Absenderadresse wurde tatsächlich von Alice versandt;
der Online-Banking-Server muss wirklich der Server meiner Bank sein.)

## Zur Übung

Beurteilen Sie Postkarten, versiegelte Briefe und die Eröffnung eines Bankkontos bei einer Online-Bank in Bezug auf die drei Kriterien.

# Symmetrisch Verschlüsselung mittels eines Schlüssels





- Zum Schutz vor unbefugtem Lesen kann man Daten und Kommunikation *verschlüsseln*.
  Dazu benutzt man ein *Verschlüsselungsverfahren* sowie einen *geheimen Schlüssel*.
- Beim *Verschlüsseln* (encryption) wird ein *Klartext* $m$ (message) zusammen mit dem *Schlüssel* $k$ (key) in ein *Chiffretext* $c = e(m, k)$ verwandelt.
  Dies entspricht dem »Abschließen« mit dem Schlüssel.
- Beim *Entschlüsseln* (decryption) wird der Chiffretext zusammen mit dem Schlüssel in den Klartext zurückverwandelt, d.h. $m = d(c, k)$.
  Dies entspricht dem »Aufschließen« mit dem Schlüssel.
- Da man zum »Auf- und Zuschließen« denselben Schlüssel verwendet, spricht man von *symmetrischen Verfahren*.

# Verfahren I: Die Cäsar-Chiffre







- Bereits in der Antike nutzte Julius Cäsar Verschlüsselungen, wenn er Befehle an seine Feldherren übermittelte.
- Das als *Cäsar-Chiffre* bekannte Verfahren funktioniert wie folgt: Jeder Buchstabe der Nachricht wird durch den Buchstaben ersetzt, der *drei Buchstaben später im Alphabet* kommt.
- Allgemeiner kann man statt »drei Buchstaben später« auch »$k$ Buchstaben später« benutzen.
- Mathematisch ist also $e$ die Funktion, die eine Nachricht und eine Zahl $k$ nimmt und jeden Buchstaben der Nachricht um $k$ viele Stellen im Alphabet vorwärts schiebt. Entsprechend schiebt $d$ jeden Buchstaben um $k$ viele Stellen zurück.

## Zur Diskussion

Betätigen Sie sich als *Kryptoanalytiker*! Wie kann man – ohne Kenntnis des Schlüssels $k$ – eine mit einer Cäsar-Chiffre kodierte Nachricht entschlüsseln?

# Beispiel einer Cäsar-Verschlüsselung

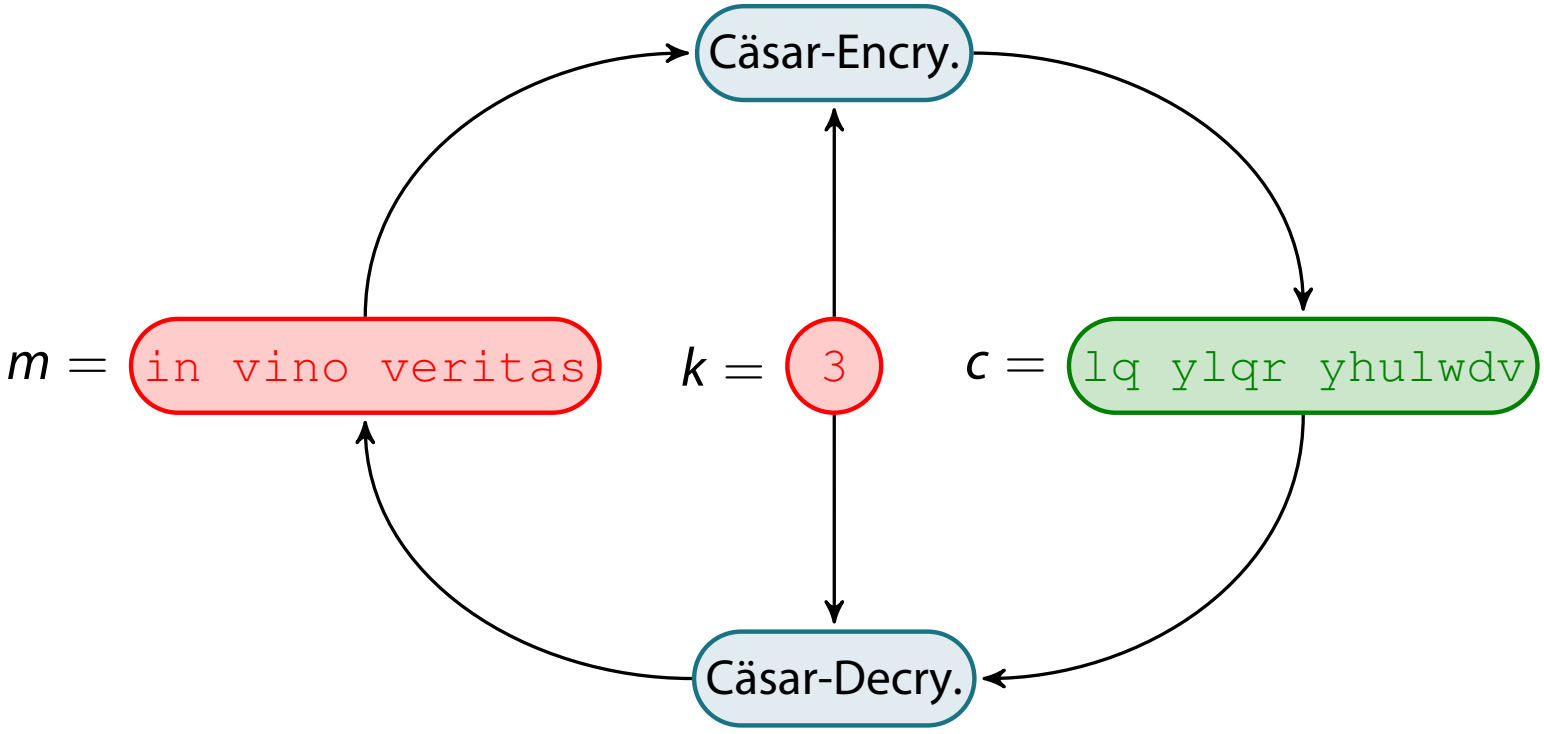






Rot = geheim, Grün = öffentlich

Public domain



# Verfahren II: Der One-Time-Pad







Da die Cäsar-Chiffre offenbar nicht sonderlich sicher ist, benötigen wir ein besseres Verfahren:

## One-Time-Pad-Algorithmus (Ver- und Entschlüsselung)

Eingaben seien die Nachricht $m$ und der Schlüssel $k$ *gleicher Länge*

1. Schreibe Nachricht und Schlüssel als Bitstrings auf (wie im ersten Kapitel).
2. Bilde nun das bitweise XOR von Nachricht und Schlüssel. Dies bedeutet: Flippe das $i$-te Bit der Nachricht, wenn das $i$-te Bit des Schlüssels eine 1 ist.

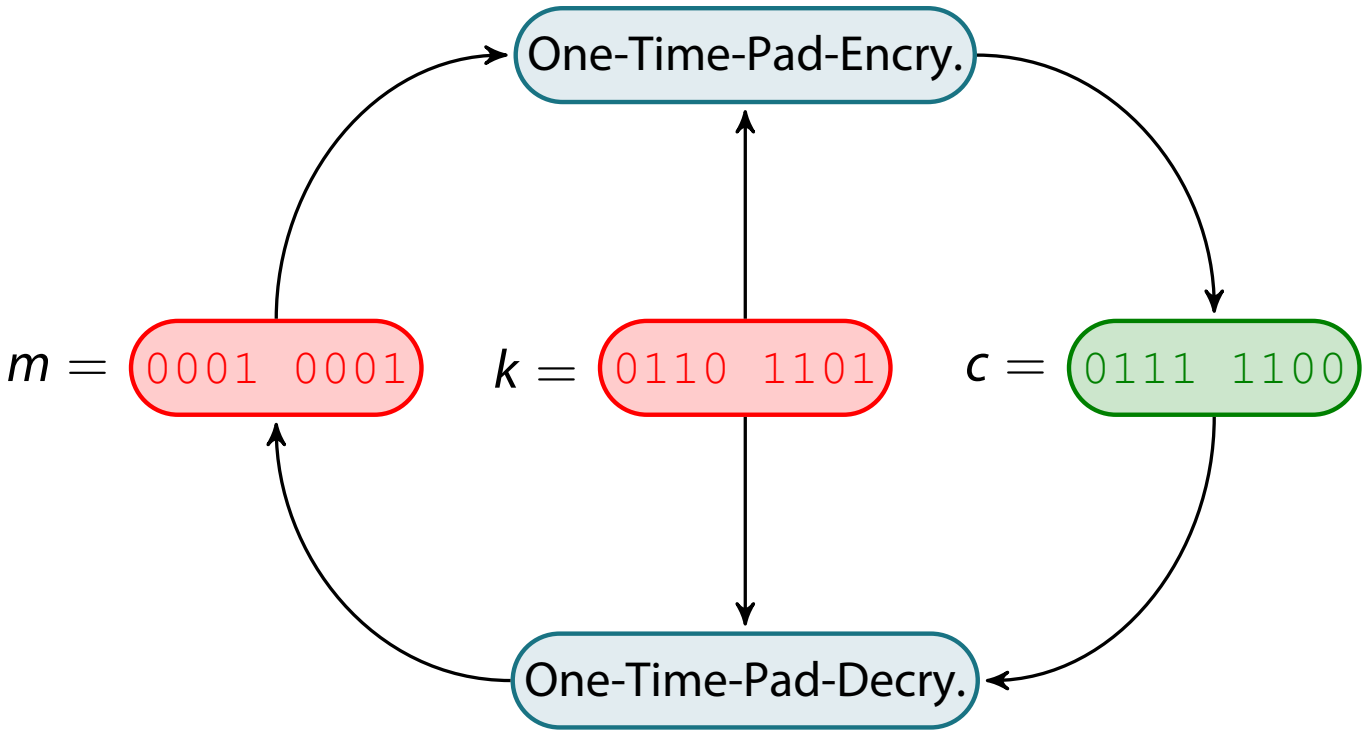
- Wie der Name schon sagt, kann man das Verfahren leider nur einmal pro Schlüssel (sicher) verwenden.
- Außerdem sind die Schlüssel schrecklich lang.

# Beispiel einer One-Time-Pad-Verschlüsselung





Rot = geheim, Grün = öffentlich



# State-of-the-Art in Sachen symmetrische Verschlüsselung.







- Moderne symmetrische Verschlüsselungverfahren kann man sich als eine *sehr clevere Mischung* aus Cäsar-Verfahren und One-Time-Pad vorstellen.
- Lange Zeit war das amerikanische DES (data encryption standard) das wichtigste Verfahren. Wegen Problemen wie Ausfuhrverbot, zu kurze Schlüssellänge und Patenten wurde es vor gut 10 Jahren durch ein moderneres und sicheres Verfahren ersetzt.
- Der neue internationale Standard nennt sich AES (advanced encryption standard).

# Sicherheit von Verschlüsselungsverfahren.





## Wann ist ein Verfahren »sicher«?

- *Informationstheoretisch sicher* heißt ein Verfahren, wenn man Chiffretexte ohne Kenntnis des Schlüssels nicht entschlüsseln kann.
- Leider kann man zeigen, dass nur One-Time-Pad-Verfahren in diesem Sinn sicher sind.
- *Komplexitätstheoretisch sicher* heißt ein Verfahren, wenn es für die Entschlüsselung kein wesentlich schnelleres Verfahren gibt, als alle Schlüssel durchzuprobieren.
- Bei Schlüssellängen ab 500 Bits ist solch eine Sicherheitsstufe dann *in diesem Universum nicht zu brechen*.

# Eine ungewöhnliche Idee.

- Symmetrische Verschlüsselungsverfahren haben den *Nachteil*, dass Kommunikationspartner erstmal einen *Schlüssel sicher austauschen müssen*.
- Rivest, Shamir und Adleman haben 1977 ein Verfahren vorgeschlagen, *bei dem man keinen Schlüssel auszutauschen braucht* – das RSA-Verfahren. (Es gab vorher auch schon ein schlechteres, geheimgehaltenes Verfahren.)
- Heute ist dies ein Spezialfall der *Klasse der asymmetrischen Verschlüsselungsverfahren*.

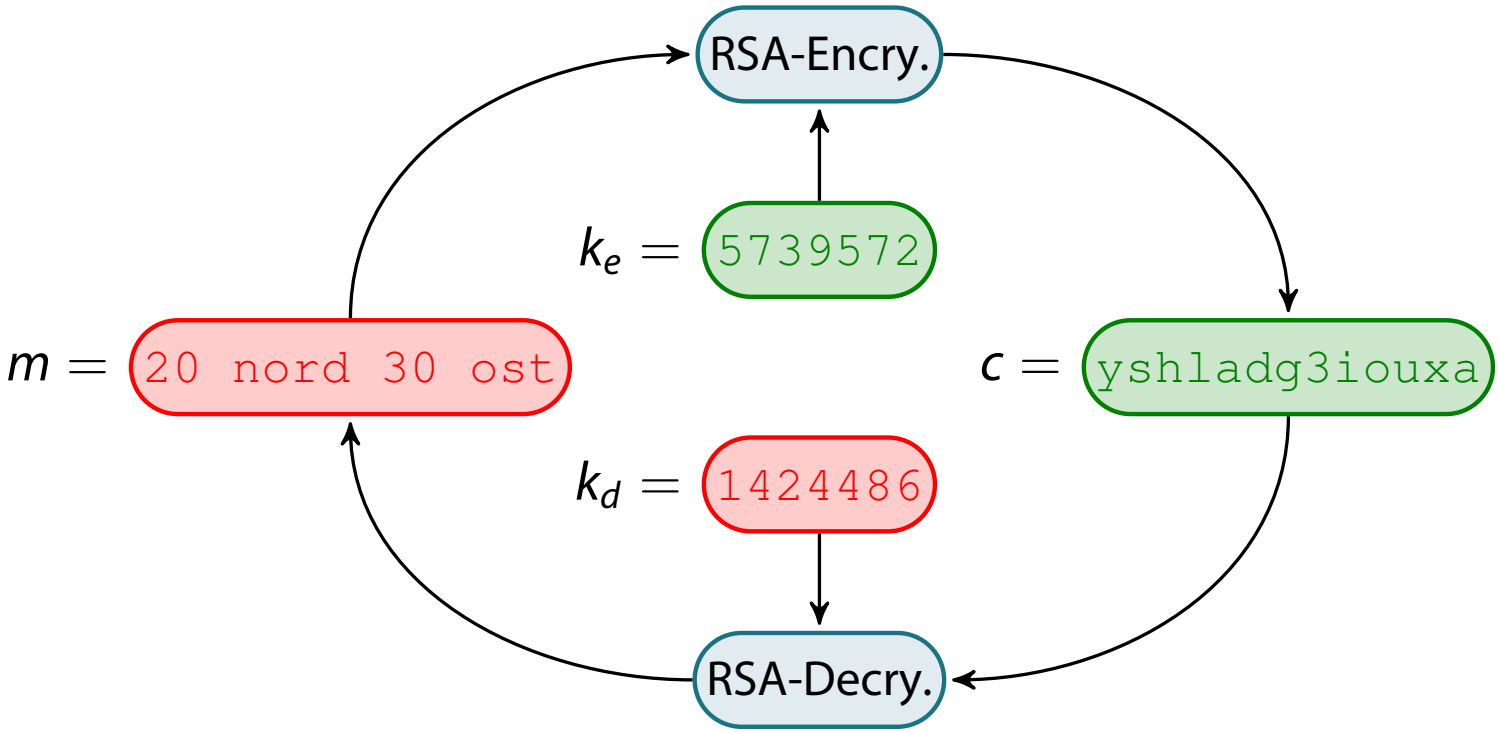
## Idee der asymmetrischen Verfahren

- Man benutzt *zwei Schlüssel*.
- Wenn man eine Nachricht mit einem ersten Schlüssel »abschließt«, kann man sie *nur mit dem zweiten Schlüssel »aufschließen«*.
- Kennt man einen der Schlüssel, so kann man daraus nicht mit vertretbarem Aufwand den anderen Schlüssel generieren.

# Ablauf einer asymmetrischen Verschlüsselung.



- Die Royal Airforce möchte den Aufenthaltsort von Prinz Harry an den Buckingham Palace schicken.
- Dazu braucht sie nur den *öffentlichen Schlüssel 5739572 des Buckingham Palace* zu kennen.
- Nur im Palast kennt man den *privaten Schlüssel 1424486 des Palastes* und kann die Nachricht entschlüsseln.



# Eine Nachricht soll vertraulich sein.





## Ziel: Vertraulichkeit

- Eine Nachricht soll einer Person zugestellt werden.
- Nur diese Person soll die Nachricht lesen können.

Dies nennt man auch einen *digitalen Briefumschlag*.

## Methode

- Die Person erzeugt ein RSA-Paar $(k_e, k_d)$.
- Der Schlüssel $k_e$ wird »allgemein bekanntgegeben« und heißt nun *öffentlicher Schlüssel*.
- Nachrichten werden mit dem öffentlichen Schlüssel der Person verschlüsselt.
- Effekt: Nur diese Person kann die Nachricht (mit dem nur ihr bekannten privaten Schlüssel $k_d$) entschlüsseln.

# Praktische Umsetzung: Erzeugen des Schlüsselpaares.

- Zur Erzeugung eines Schlüsselpaares benutzt man ein *geeignetes Programm* wie beispielsweise `openssl`.
- Bei MacOS kann man auch komfortabel den *Schlüsselbundverwaltung* nutzen.

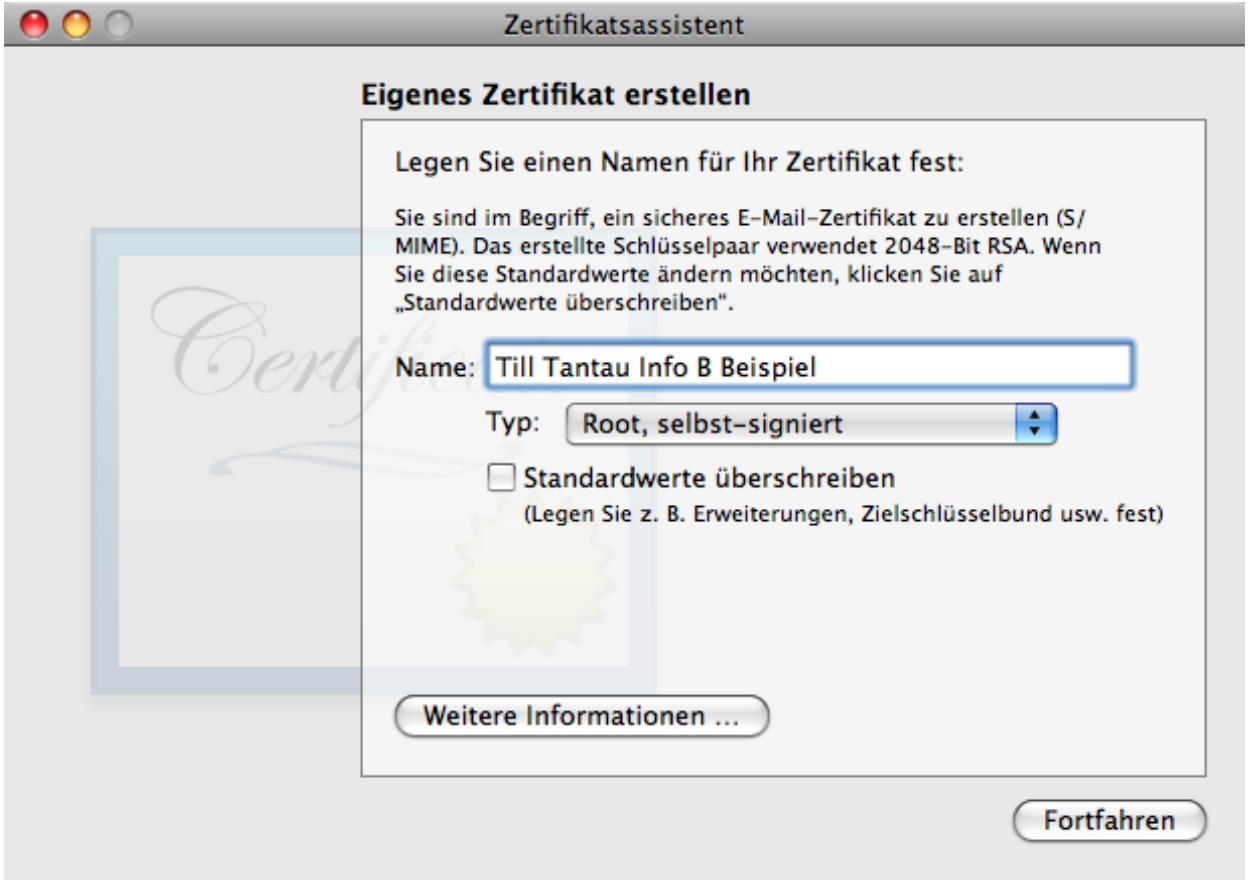
(Das Verfahren wird gleich noch etwas komplizierter werden, aber kümmern wir uns erstmal um den einfachen Fall.)

# Praktische Umsetzung: Erzeugen des Schlüsselpaares.



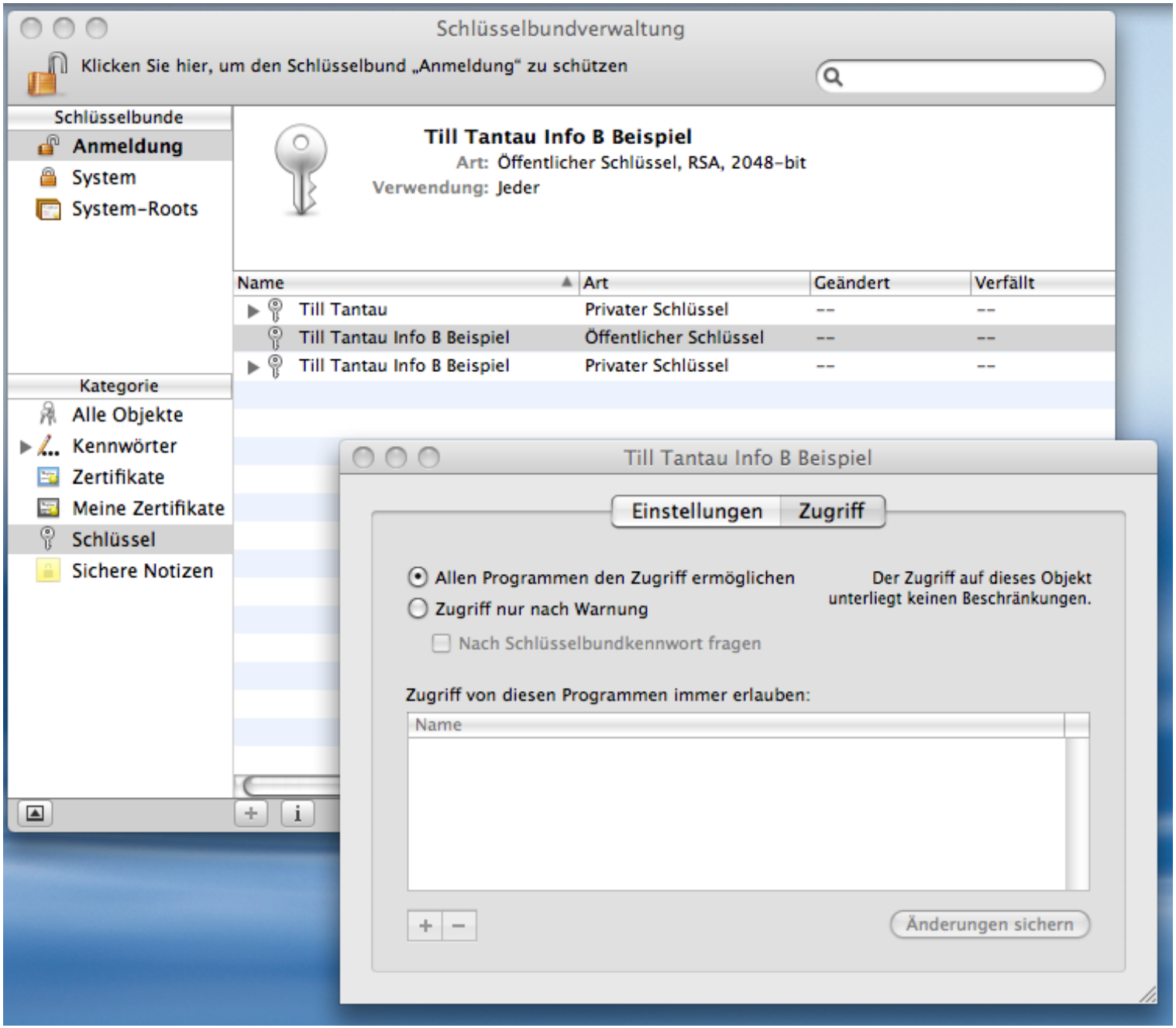






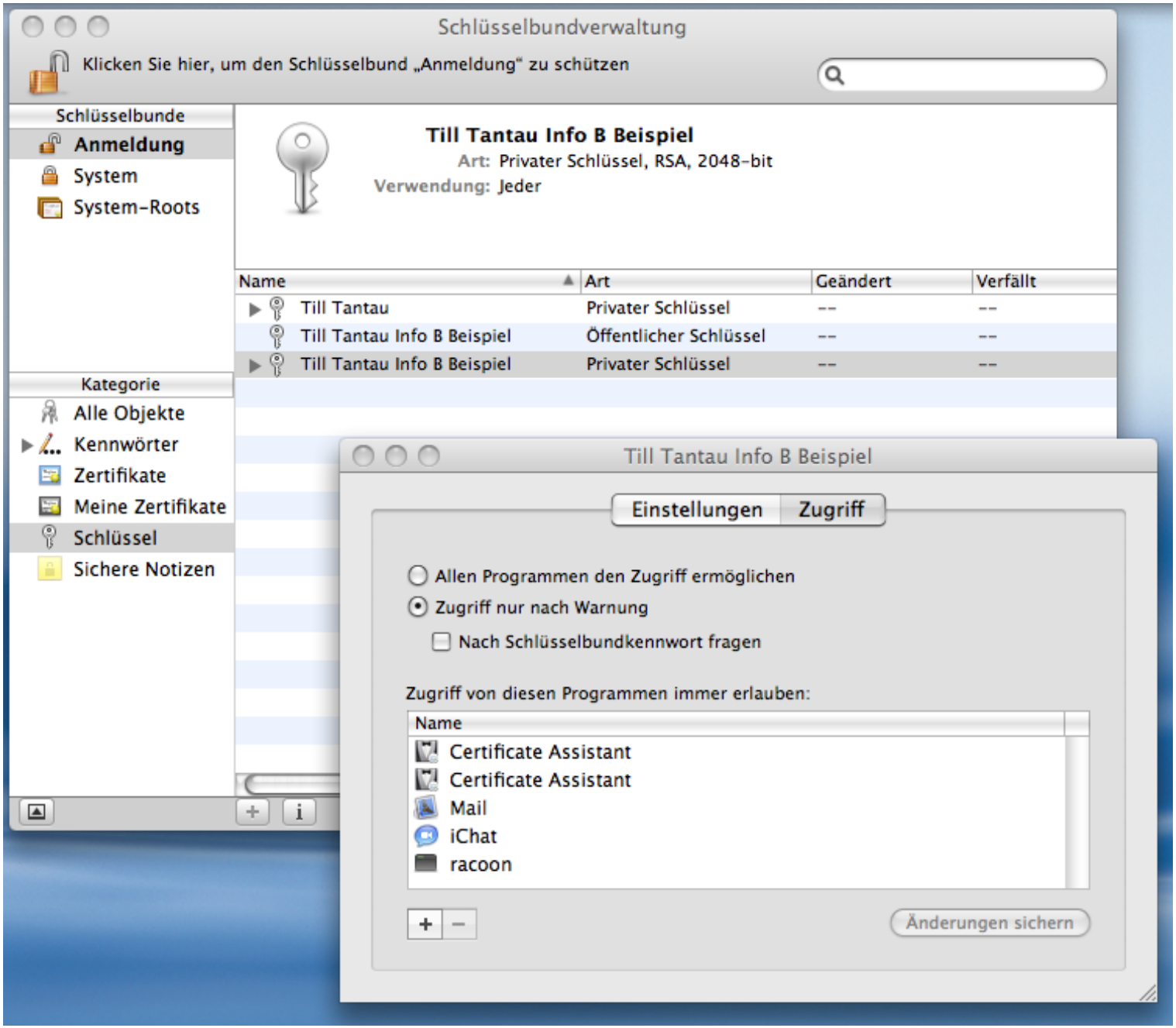

Screenshot by Till Tantau

# Praktische Umsetzung: Erzeugen des Schlüsselpaares.




Screenshot by Till Tantau

# Praktische Umsetzung: Erzeugen des Schlüsselpaares.




Screenshot by Till Tantau

# Praktische Umsetzung: E-Mail an Johannes.

## Schritt 1: Wir brauchen Johannes öffentlichen Schlüssel.

- Um eine Mail zu verschlüsseln, muss das E-Mail-Programm den *öffentlichen Schlüssel der Person kennen, der man eine E-Mail schreiben möchte*.
- Diesen öffentlichen Schlüssel kann einem die Person beispielsweise vorher geschickt haben.
- E-Mail-Programme oder die Schlüsselverwaltung erlaubt es, solche öffentliche Schlüssel zu *importieren* (ein geeigneter Menüpunkt).

# Praktische Umsetzung: E-Mail an Johannes.

## Schritt 1: Wir brauchen Johannes öffentlichen Schlüssel.







So sieht der öffentliche Schlüssel von Johannes beispielsweise wie folgt aus:

```
-----BEGIN CERTIFICATE-----
MIIFSTCCBDGgAwIBAgIECz1arTANBgkqhkiG9w0BAQUFADCBqzELMAkGA1UEBhMC
REUxIDAeBgNVBAoTF1VuaXZlcnNpdGFldCB6dSBMdWViZWNrMS4wLAYDVQQLEyVJ
bnN0aXR1dCBmdWVyIE1lZGl6aW5pc2NoZSBJbmZvcm1hdGlrMScwJQYDVQQDEx5D
QSBkZXIgVW5pdmVyc2l0YWV0IHp1IEx1ZWJlY2sxITAfBgkqhkiG9w0BCQEWEnBr
aUB1bmktbHVlYmVjay5kZTAeFw0wNzEwMjMxMTM4MDdaFw0xMDEwMjIxMTM4MDda
... 20 ausgelassene Zeilen ...
Q8RGGPHGKcAocX3kGB3VTWZptDCACiJ9E5Q5pD4mWMRPYAgYnjJwWv4KzF5Moboe
29IKgSvifR5ttcdqCFn5gfwVYrHWOLxxSZkbUpqGIsAhoGeWd65Z9u4FArI87UIw
3KCso+ohahGUeVqZQk6ZXU6zJX/hw6K4y2l1QfiBwVQuNjvudADM3Q6RSedECK4m
MsUhRqoUqvXegCZ7mA==
-----END CERTIFICATE-----
```

- Er kann auf
  `https://pki.pca.dfn.de/uzl-ca/cgi-bin/pub/pki`
  heruntergeladen werden.
- Alternativ kann Johannes den Schlüssel zunächst von seinem Mail-Programm oder von der Schlüsselverwaltung *exportieren* und dann verschicken (zur Not per Post).

# Praktische Umsetzung: E-Mail an Johannes.

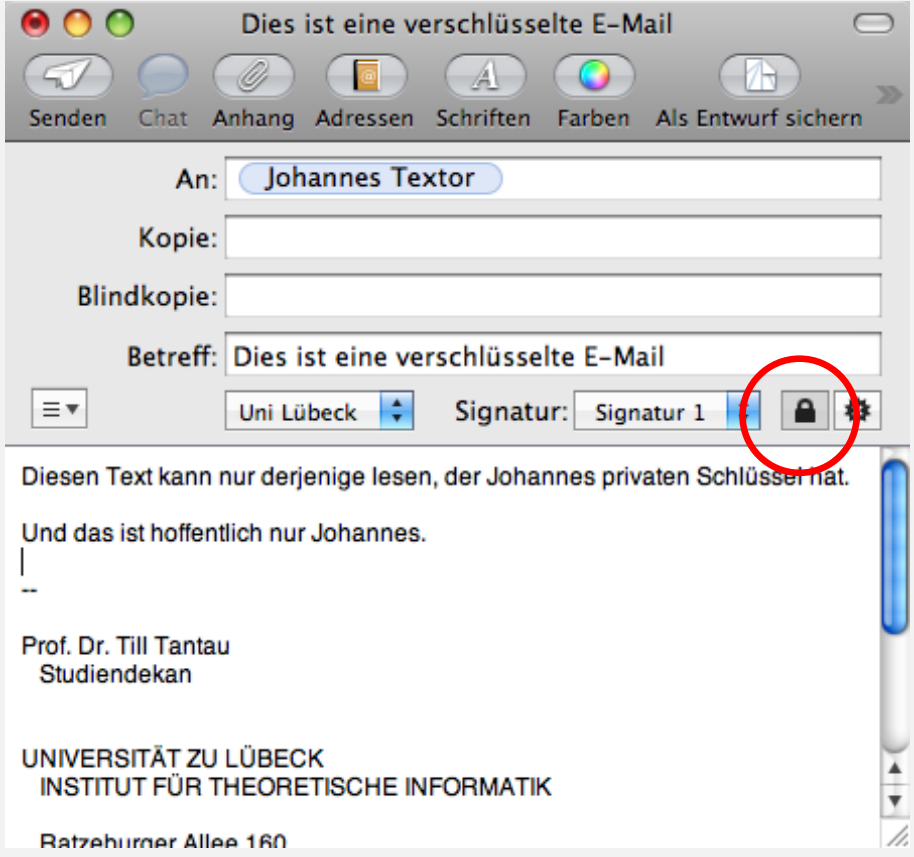
## Schritt 2: Schreiben und abschicken der E-Mail.







- Hat man dem eigenen System erstmal Johannes öffentlichen Schlüssel beigebracht, so kann man ihm eine verschlüsselte E-Mail schreiben.
- Bei modernen E-Mail-Programmen muss man dazu einfach auf einen Ver-schlüsselungsknopf drücken.




Screenshot by Till Tantau

# Praktische Umsetzung: E-Mail an Johannes.

## Schritt 2: Schreiben und abschicken der E-Mail.

- An Johannes wird dann ein mit dem *öffentlichen Schlüssel von Johannes verschlüsselter* Text geschickt.
- Um den Text zu entschlüsseln, braucht man den privaten Schlüssel von Johannes.
- Folglich kann niemand außer Johannes – *nichtmal der Autor der Mail* – den Text wieder entschlüsseln.

# Praktische Umsetzung: E-Mail an Johannes.

## Schritt 2: Schreiben und abschicken der E-Mail.







```
Message-Id: <AF944753-2276-487B-9BB1-995DA5D2058F@tcs.uni-luebeck.de>
From: Till Tantau <tantau@tcs.uni-luebeck.de>
To: Johannes Textor <textor@tcs.uni-luebeck.de>
Content-Type: application/pkcs7-mime;
name=smime.p7m;
smime-type=enveloped-data
Content-Transfer-Encoding: base64
X-Smtp-Server: theogate.tcs.uni-luebeck.de:tantau
Content-Disposition: attachment;
filename=smime.p7m
Mime-Version: 1.0 (Apple Message framework v936)
Subject: =?ISO-8859-1?Q?Dies_ist_eine_verschl=FCsselte_E-Mail?=
Date: Fri, 11 Jun 2010 16:05:31 +0200

MIAGCSqGSIb3DQEHA6CAMIACAQAxggOiMIIBzQIBADCBtDCBqzELMAkGA1UEBhMCREUxIDAeBgNV
BAoTF1VuaXZlcnNpdGFldCB6dSBMdWViZWNrMS4wLAYDVQQLEyVJbnN0aXR1dCBmdWVyIE1lZGl6
aW5pc2NoZSBJbmZvcm1hdGlrMScwJQYDVQQDEx5DQSBkZXIgVW5pdmVyc2l0YWV0IHp1IEx1ZWJl
Y2sxITAfBgkqhkiG9w0BCQEWEnBraUB1bmktbHVlYmVjay5kZQIECz1arTANBgkqhkiG9w0BAQEF
AASCAQDbXJGvqKCu+Qd1xfTLSJKW6I8T2Eds7ks1qtvBVoD6RFzcHF7XOTllzcOmxgC2GS8frzNr
pVIC5VBssy/BEnm3BCpD8sw8HCxE0pcIm96/p5oHp9Qyk0FYs97JF3GLHVXPAs8AoHEMoVBXAuku
... 66 ausgelassene Zeilen ...
FbiGph3to8Q/xaLEGHplQNN245YGcAY3xPAKibQfs3725ctGA9bBUXliZnejMXR15DMWcd8VozAA
3MAQYwRIfJprLx7pYpUMEbSUN1Zdqi4WhTYS+6MvVEbj5ItZ68PWDx2OFsqm5enql58XViNLuS58
aTzsJ6uw86lgEgqbbiD8wpMExZxpBCBxX07+jmfobYe16/gFjInEB7XOV+VC95GEryi87pu3GAQI
qJB1HZWcdHQAAAAAAAAAAAA
```

# Zusammenfassung: Wer bekommt welchen Schlüssel?



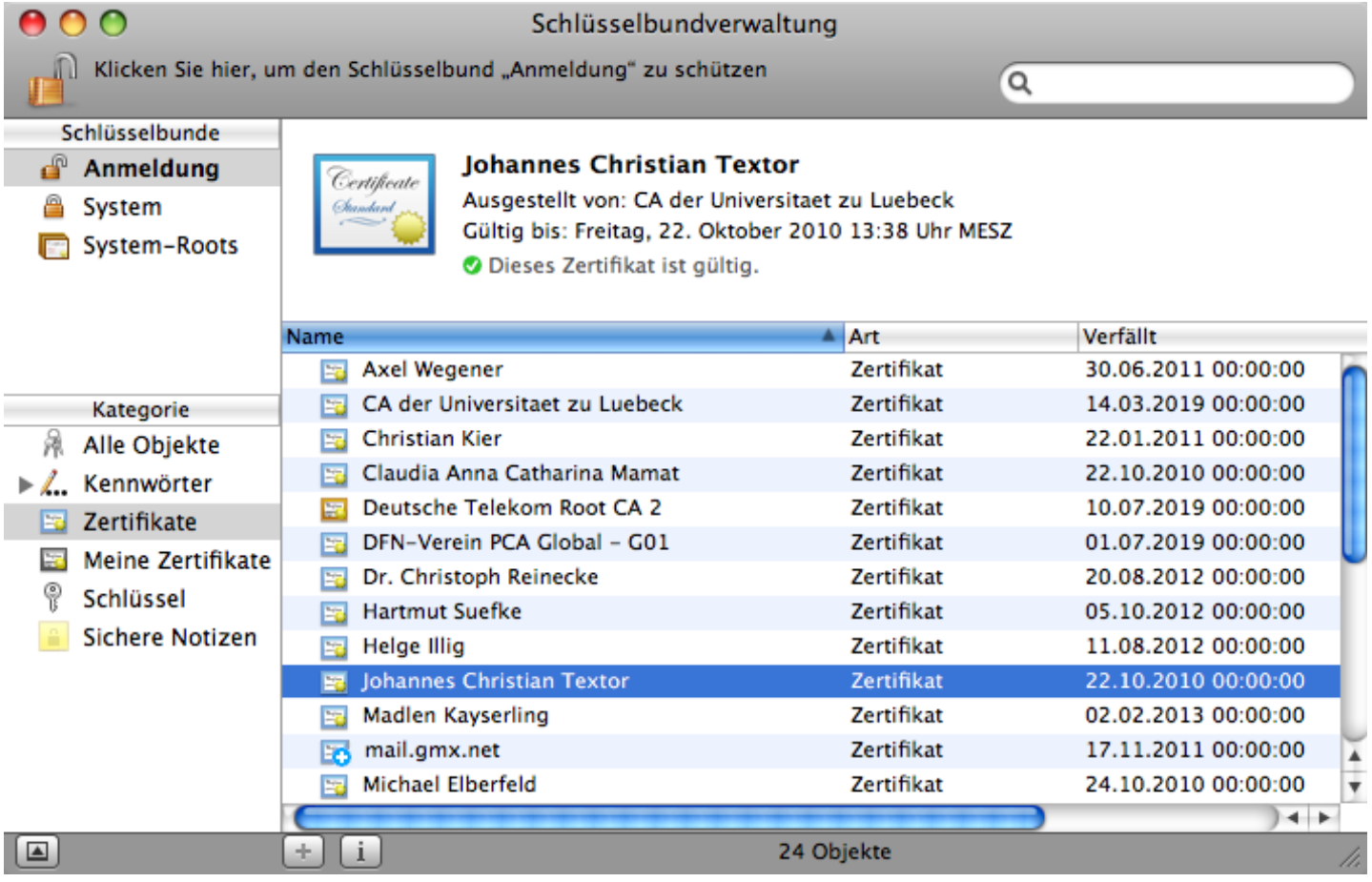


- Generell gilt: Ihr *privater* Schlüssel ist nur Ihnen bekannt (auch nicht Ihrem Lover!) und möglichst gut auf Ihrem Computer geschützt.
  - Am sichersten ist es, wenn der Schlüssel auf einer separaten Karte liegt, die ihrerseits durch eine PIN geschützt ist – dies ist mit dem neuen Personalausweis möglich.
  - In der Praxis liegt Ihr privater Schlüssel aber auf Ihrer Festplatte.
  - Dort ist er selbst wieder durch ein Master-Passwort symmetrisch verschlüsselt und damit selbst vor Diebstahl des Computers oder vor bösartigen Hackern geschützt.
- Generell gilt: *Öffentliche* Schlüssel sind möglichst breit gestreut und überall verfügbar. Sie sollten auf Ihrem System die öffentlichen Schlüssel aller Ihnen bekannter Personen haben.

# Zusammenfassung: Wer bekommt welchen Schlüssel?








Screenshot by Till Tantau

# Eine Nachricht soll digital unterschrieben werden.





## Ziel: Authentizität

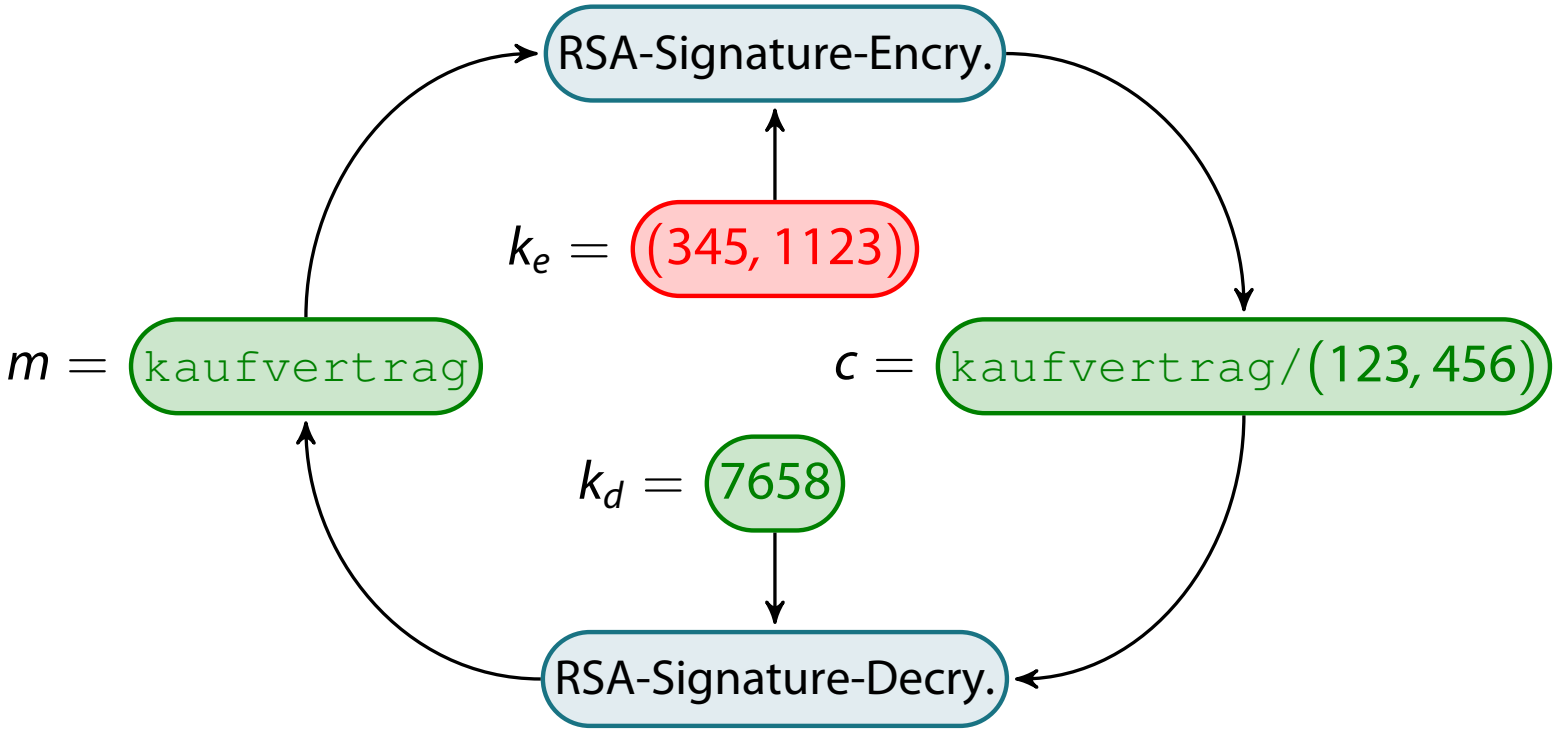
Es soll garantiert werden, dass eine Nachricht von einer bestimmten Person stammt.
Dies nennt man auch eine *digitale Unterschrift*.

## Methode

Man benutzt dieselben Schlüssel wie beim Verschlüsseln von Nachrichten, *nur umgekehrt*:

- Die zu unterschreibende Nachricht wird mit dem *privaten Schlüssel* $k_e$ verschlüsselt und dieser Text an die Originalnachricht angehängt.
- Überprüfung: Man entschlüsselt den verschlüsselten Teil mit dem öffentlichen Schlüssel $k_d$ und vergleicht ihn mit dem behaupteten Text.
- Effekt: Nur die Person, die $k_e$ kennt, kann die unterschriebene Nachricht erzeugen.

# Ablauf einer digitalen Unterschrift.

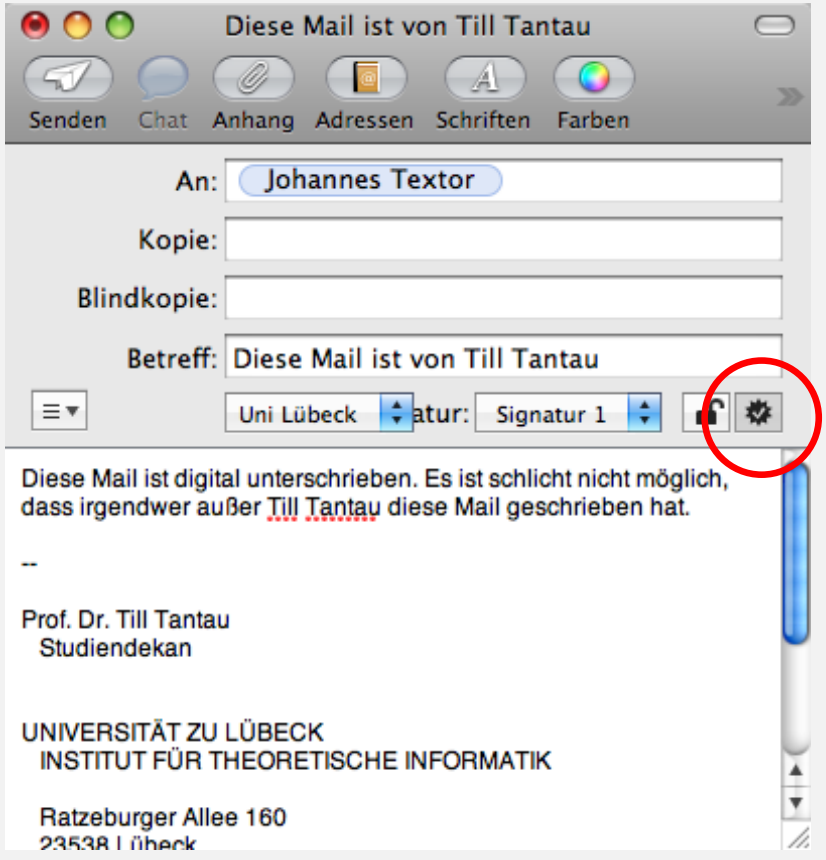


# Praktische Umsetzung: Unterschriebene E-Mail an Johannes.





- Um eine E-Mail zu unterschreiben, braucht man lediglich den eigenen privaten Schlüssel – und den haben wir ja schon erzeugt.
- Damit jemand die E-Mail überprüfen kann, braucht er den öffentlichen Schlüssel – diese sind ja aber frei zugänglich.



Screenshot by Till Tantau

*Achtung:* Mit »Signatur« wird auch manchmal der Standard-Text am Ende einer Mail bezeichnet.
Dieser ist *keine* digitale Unterschrift.

# Praktische Umsetzung: Unterschriebene E-Mail an Johannes.







```
Message-Id: <A15605B5-9389-442D-BD9E-FC65D76A18DD@tcs.uni-luebeck.de>
From: Till Tantau <tantau@tcs.uni-luebeck.de>
To: Johannes Textor <textor@tcs.uni-luebeck.de>
Content-Type: multipart/signed;
boundary=Apple-Mail-11--309560619;
micalg=sha1;
protocol="application/pkcs7-signature"
X-Smtp-Server: theogate.tcs.uni-luebeck.de:tantau
Mime-Version: 1.0 (Apple Message framework v936)
Subject: Diese Mail ist von Till Tantau
Date: Mon, 14 Jun 2010 08:44:26 +0200

... 13 ausgelassene Zeilen ...
Diese Mail ist digital unterschrieben. Es ist schlicht nicht m=F6glich, =20=

dass irgendwer au=DFer Till Tantau diese Mail geschrieben hat.
... 62 ausgelassene Zeilen ...

--Apple-Mail-11--309560619
Content-Disposition: attachment;
filename=smime.p7s
Content-Type: application/pkcs7-signature;
name=smime.p7s
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAoIIOnzCCBCEw
ggMJoAMCAQICAgDHMA0GCSqGSIb3DQEBBQUAMHExCzAJBgNVBAYTAkRFMRwwGgYDVQQKExNEZXV0
c2NoZSBUZWxla29tIEFHMR8wHQYDVQQLExZULVRlbGVTZWMgVHJ1c3QgQ2VudGVyMSMwIQYDVQQD
... 79 ausgelassene Zeilen ...
DfpFZundteQqc6R/FdbTj67j23Y5h/8+qCIewT//LLWh4hvW/kEs7bi1Ibcm3yniFd4PzjkKI9gi
WuPJkqz3VrcuGYfjCHT3RGyAANNQOF6fiJQ5tipmN4dHkfoxWQ7nPJBobS13MLLd+fIvBrI78pTp
8mFDaOsObxdlyOXhm5RTBor2U/4uDGtZJddGOCNLpNjnCqEti3PWAHD/IAAAAAAAAA==

--Apple-Mail-11--309560619--
```

# Das Henne-Ei-Problem bei öffentlichen Schlüsseln

- Damit ich Johannes eine sichere Mail schreiben kann, muss er mir seinen öffentlichen Schlüssel zukommen lassen.
- Nun könnte auch irgendjemand anderes mir einen Schlüssel schicken und behaupten, er sei Johannes und dies sei sein Schlüssel.
- Um das auszuschließen, würde ich gerne verlangen, dass die Mail von Johannes unterschrieben ist – aber dazu brauche ich ja gerade den öffentlichen Schlüssel, um den es gerade geht.

# Certificate-Authorities zertifizieren die Echtheit von Schlüsseln.



## Ziel: Echtheit von Schlüsseln bezeugen

A will sichergehen, dass ein vermeintlicher öffentliche Schlüssel von B tatsächlich von B stammt.

## Methode

- Eine *vertrauenswürdige Instanz*, »Certificate Authority (CA)« oder »Trust-Center« genannt, legt *Root-Schlüssel* $(k_d, k_e)$ für digitale Unterschriften an.
- Der öffentliche Schlüssel $k_d$ ist allgemein bekannt (er ist zum Beispiel in Ihren Browser schon fest eingebaut).
- Benutzer B lässt sich von der vertrauenswürdigen Instanz den folgenden Text unterschreiben: »Person B hat den öffentlichen Schlüssel 1234567.«
- Wenn A mit B zum ersten Mal redet, schickt B diesen unterschriebenen Text.
- A kann die Unterschrift der CA überprüfen (A kennt ja das Root-Zertifikat) und kann dann den öffentlichen Schlüssel von B benutzen.

# Certificate-Authorities zertifizieren die Echtheit von Schlüsseln.





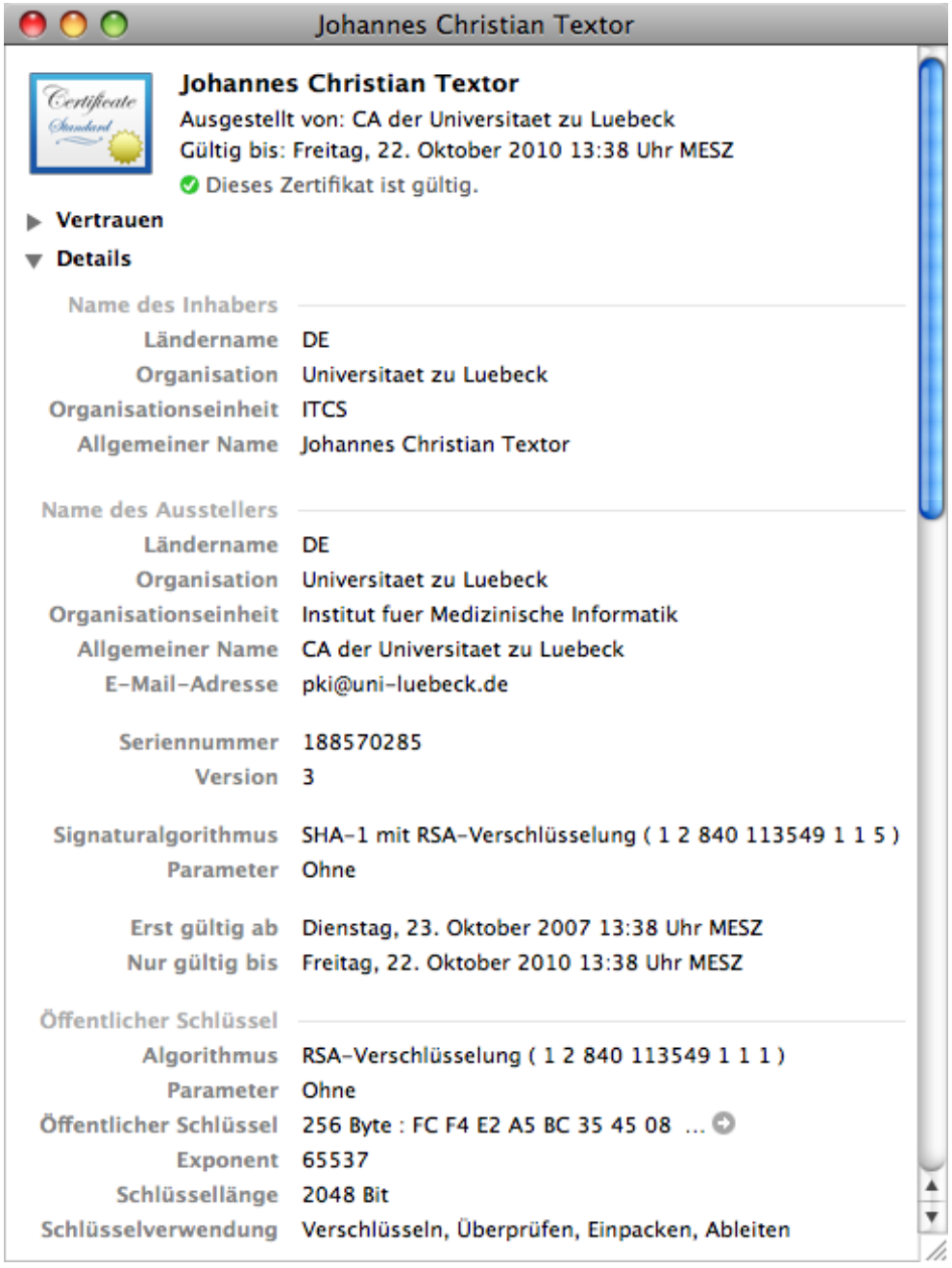


- Die privaten Schlüssel der Root-CAs sind *extrem gut gesichert*.
- Das darf man sich in etwa wie bei Mission Impossible vorstellen – inklusive gepanzerter Räume mit Servern, die vom Netz und auch von sonst allem getrennt sind.
- Selbst wenn Sie mit Ihrer Privatarmee das Gebäude stürmen, die wackeren Systemadministratoren überwältigen bevor diese die Schlüssel löschen können und den Schlüssel stehlen, würde das nicht viel nützen.
- Schlüssel können in zentralen Verzeichnissen als ungültig erklärt werden, was einige Minuten nach Ihrem Angriff der Fall wäre.
- Merke: Wenn man den privaten Schlüssel einer Root-CA klaut, so darf dies niemand merken. Ich empfehle, sich beispielsweise an Herrn Hunt zu wenden.

# Kaskaden von Unterschriften

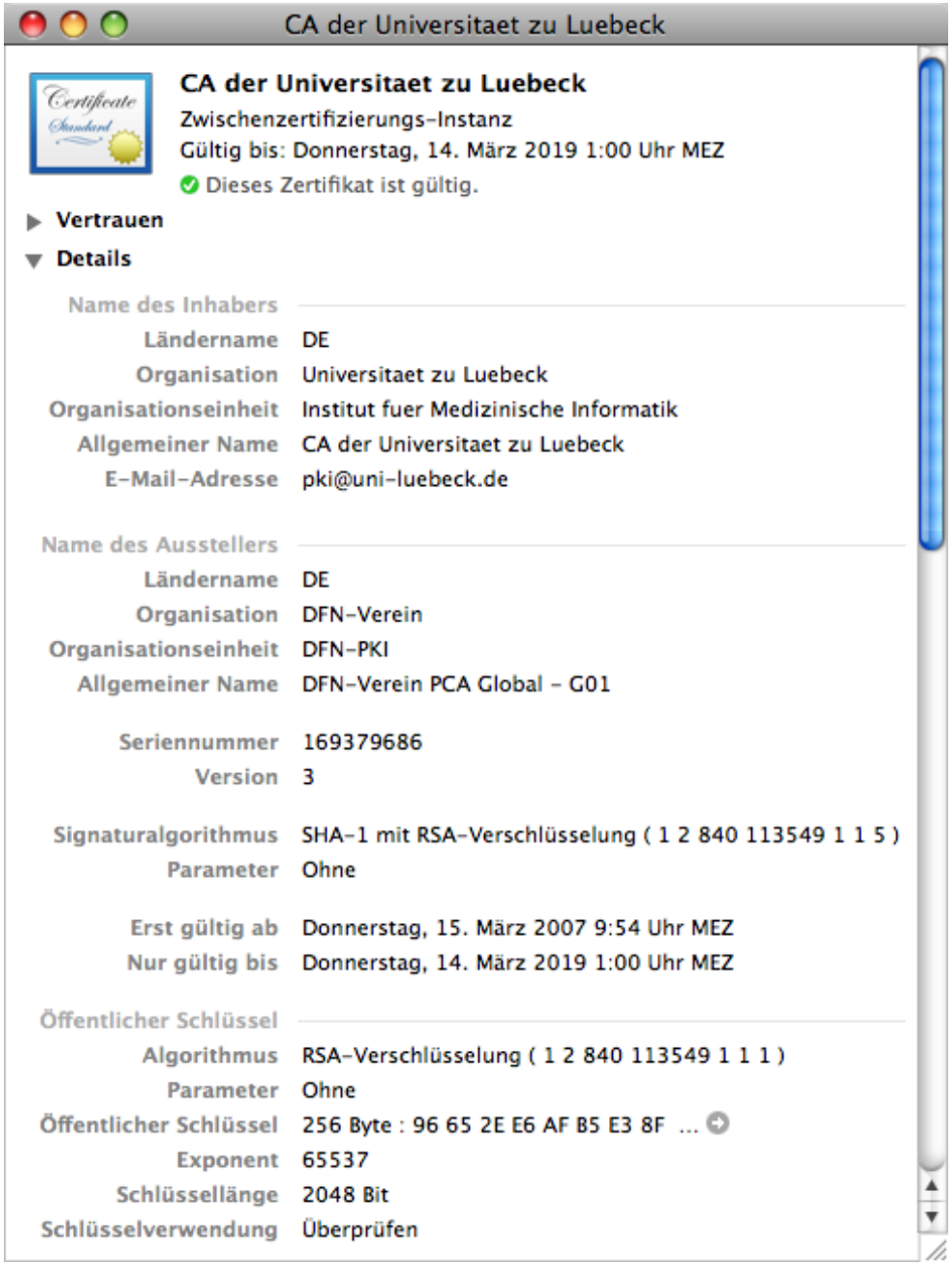
- In der Praxis kann die Root-CA nicht die Schlüssel beispielsweise aller Deutschen unterschreiben.
- Stattdessen gibt es »Zwischen-CAs«.

# Kaskaden von Unterschriften



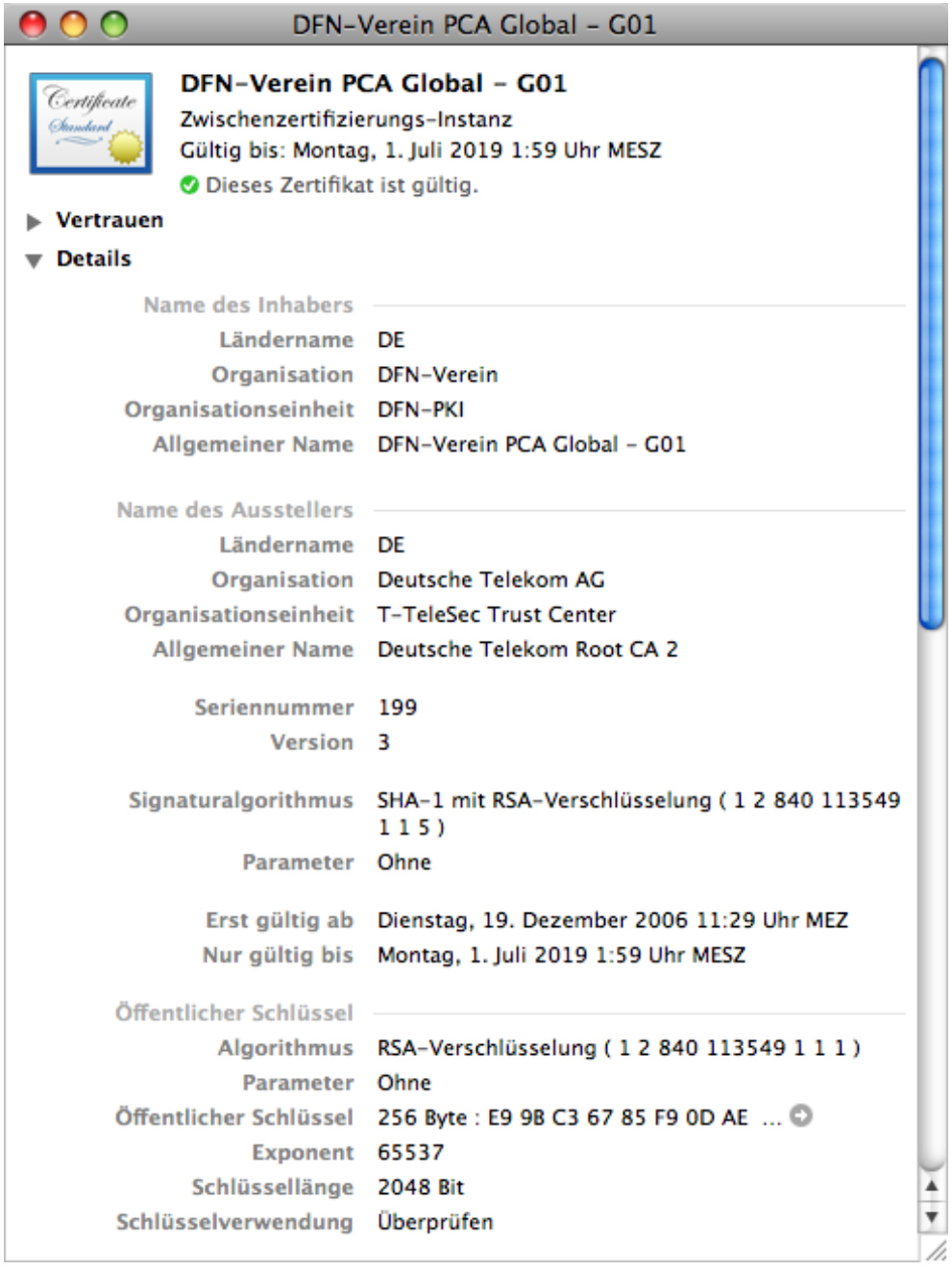

Screenshot by Till Tantau

# Kaskaden von Unterschriften



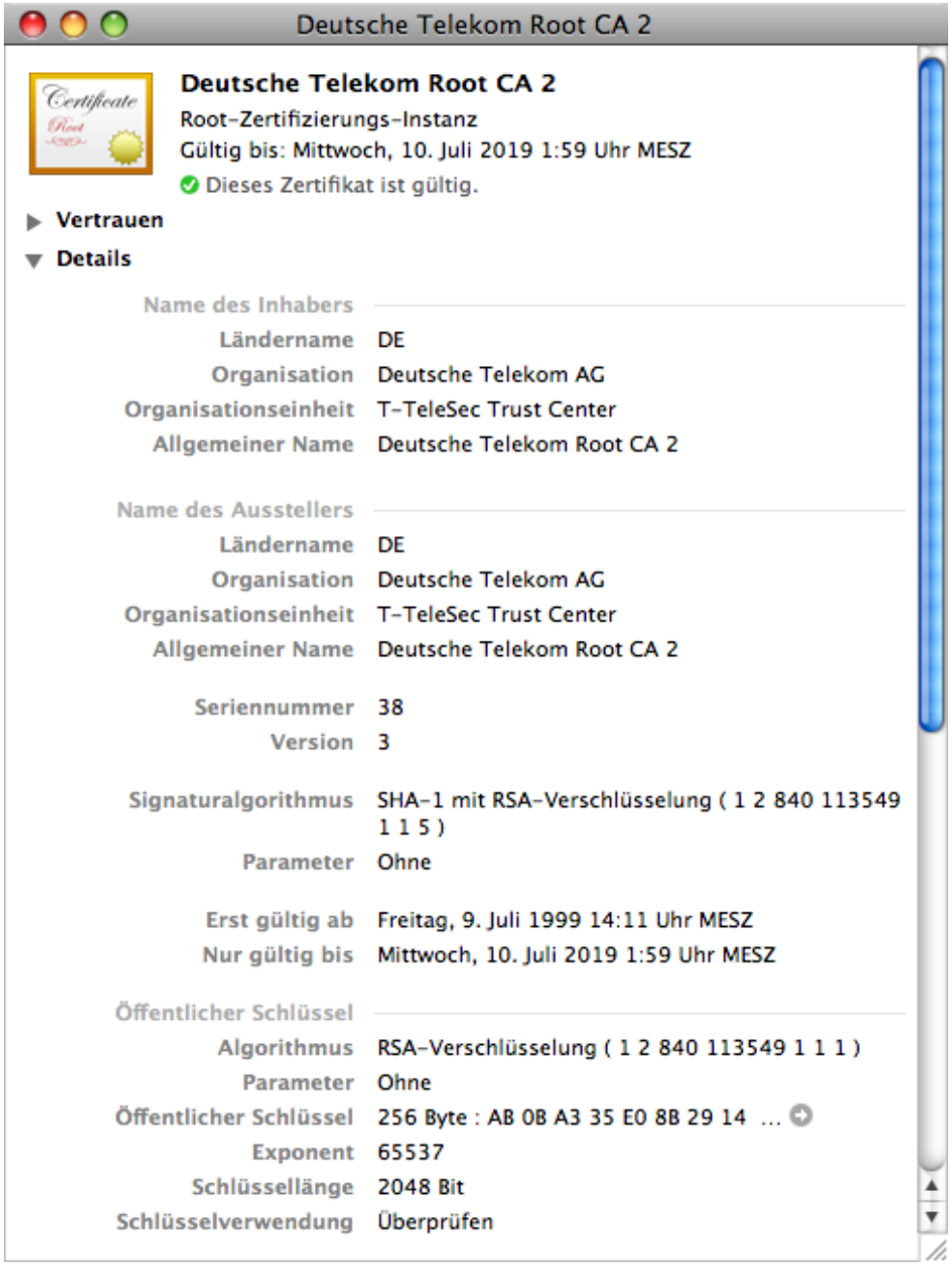

Screenshot by Till Tantau

# Kaskaden von Unterschriften




Screenshot by Till Tantau

# Kaskaden von Unterschriften

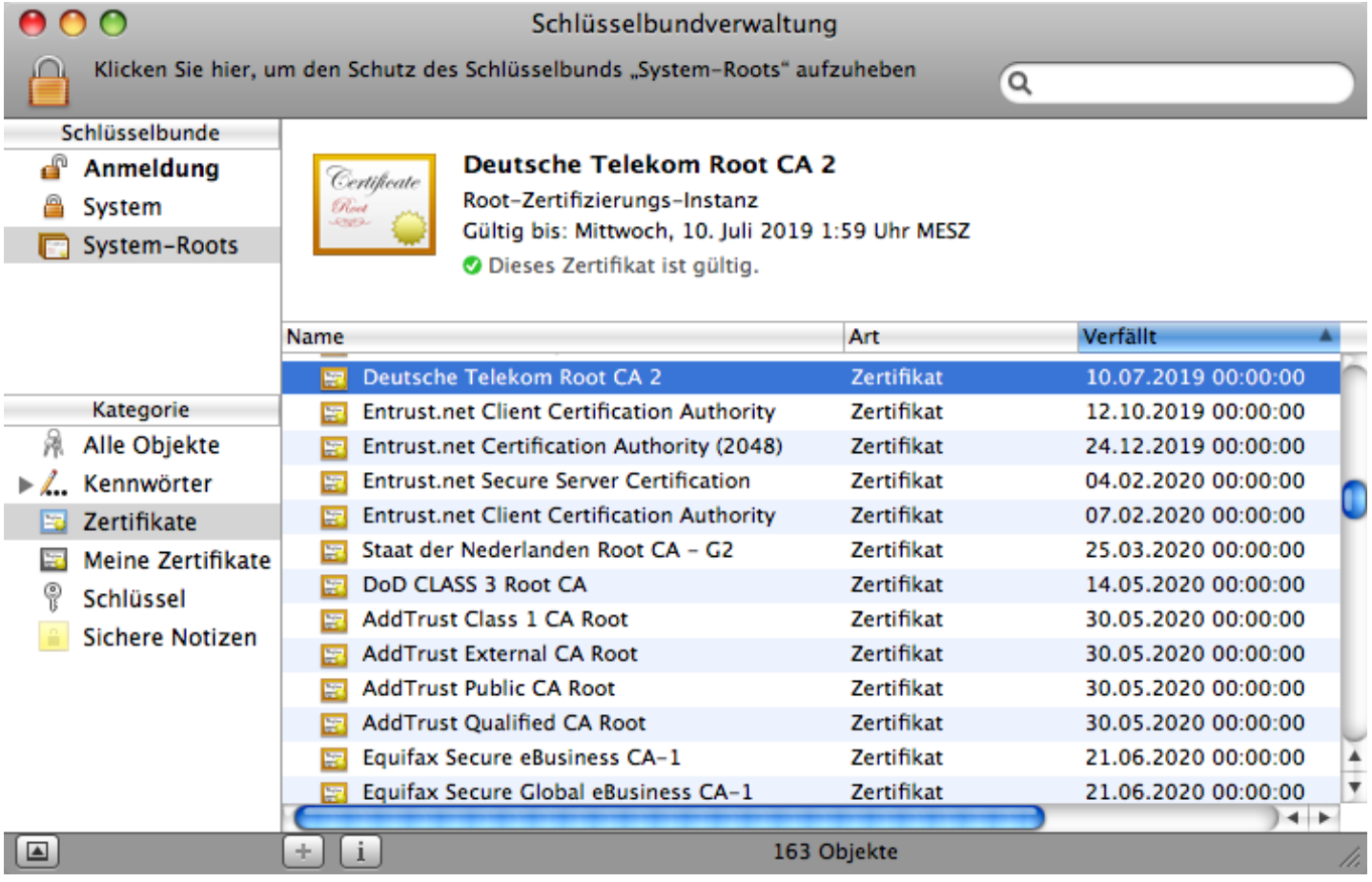



Screenshot by Till Tantau

# Kaskaden von Unterschriften








Screenshot by Till Tantau

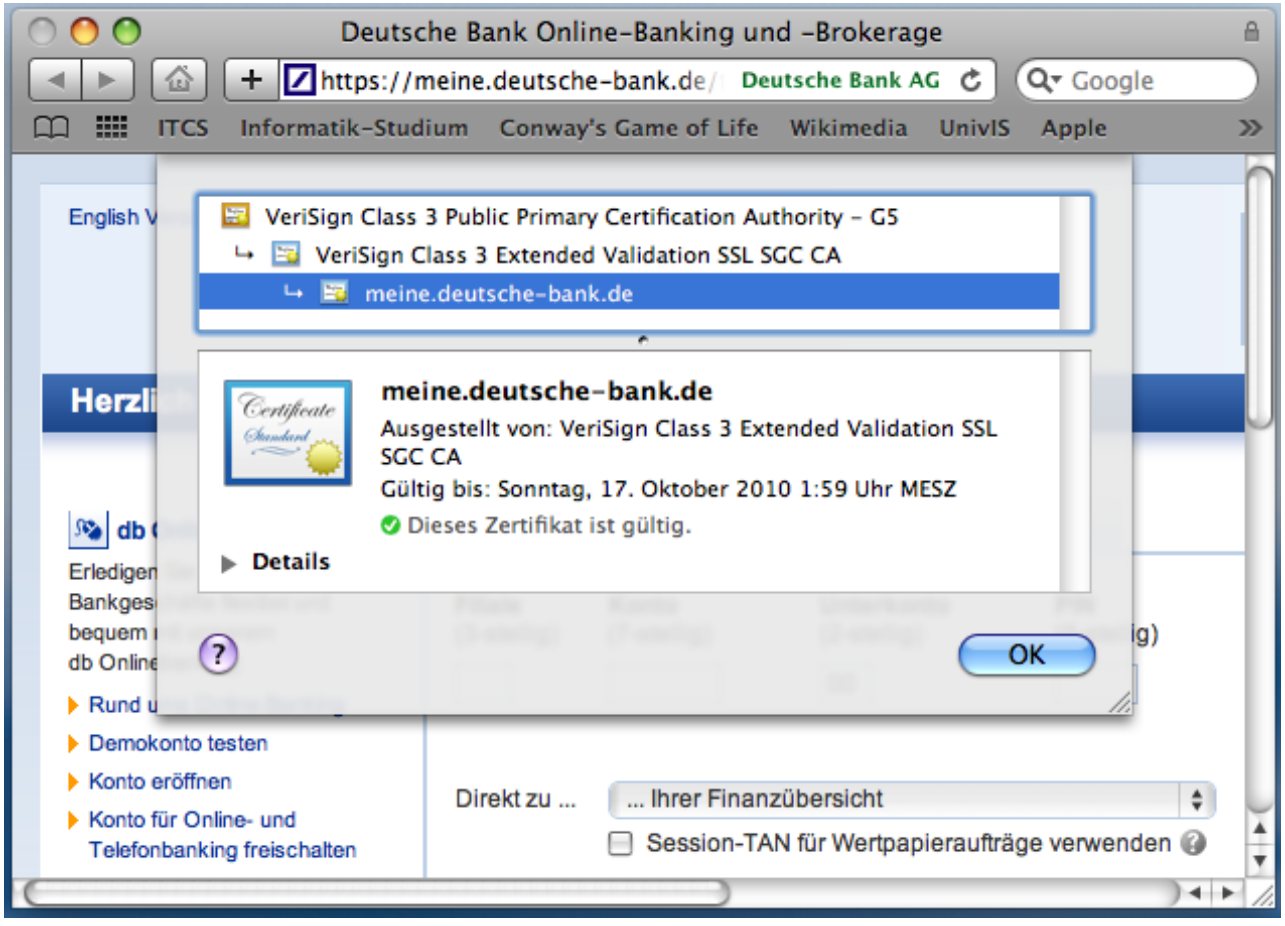
# Sicheres Surfen funktioniert wie sichere E-Mail.





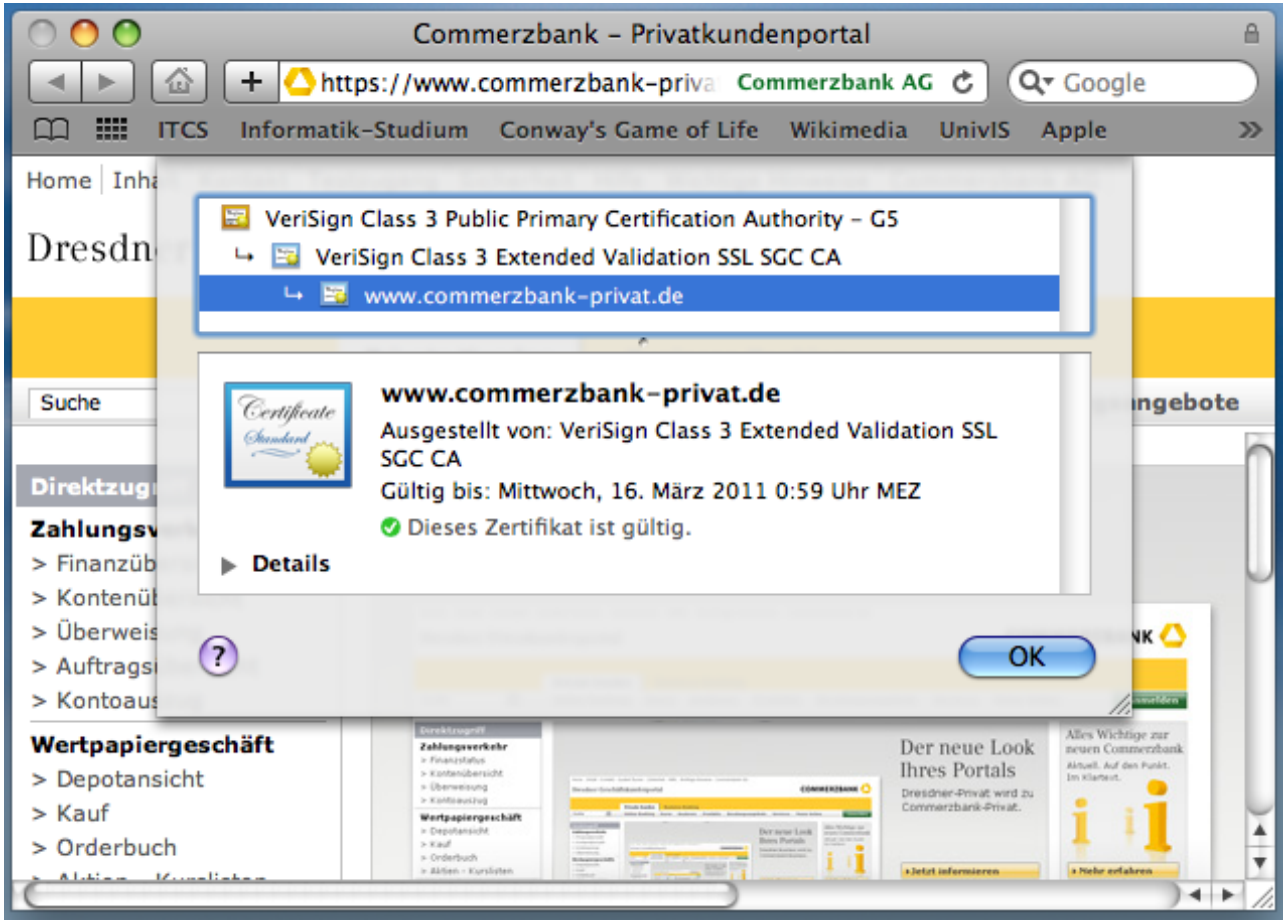
- Wenn Sie mit einem Online-Shop oder Ihrer Bank kommunizieren, haben Sie *dieselben Problem wie bei E-Mail*:
  - Niemand soll die Kommunikation abhören können.
  - Sie müssen sicher sein können, dass Ihre Bank auch wirklich Ihre Bank ist.
- Diese Probleme werden auch genauso gelöst:
  - Die Bank oder der Online-Shop hat ein Schlüsselpaar, dass nun aber *keine Person* identifiziert sondern *eine Webseite*.
  - Das Schlüsselpaar ist über eine Kette von CAs unterschrieben.
- Die Protokolle *https* und *ssh* bauen auf diese Art sichere Kanäle auf.

# Sicheres Surfen funktioniert wie sichere E-Mail.




Screenshot by Till Tantau

# Sicheres Surfen funktioniert wie sichere E-Mail.




Screenshot by Till Tantau

# Zusammenfassung





## Symmetrische Verschlüsselung

Eine Nachricht $m$ wird mit einem Schlüssel $k$ zu einem Chiffrat $c = e(m, k)$ verarbeitet. Mit *demselben* Schlüssel $k$ lässt sich dann $m = d(c, k)$ berechnen. Das Standardverfahren heißt AES.

## Asymmetrische Verschlüsselung

Eine Nachricht $m$ wird mit einem *öffentlichen Schlüssel* $k_e$ zu einem Chiffrat $c = e(m, k_e)$ verarbeitet. Mit einem *ganz anderen* Schlüssel $k_d$ lässt sich dann $m = d(c, k_d)$ zurückgewinnen. Das Standardverfahren heißt RSA.

# Zusammenfassung







## Vertraulichkeit

Die *Vertraulichkeit* von E-Mails und der Kommunikation mit Webservern (https) wird sichergestellt, indem mit dem *öffentlichen Schlüssel des Empfängers* verschlüsselt wird.

## Authentizität

Die *Authentizität* einer Nachricht wird sichergestellt, indem mit dem *privaten Schlüssel des Absenders* verschlüsselt wird. Dies nennt man *digitale Unterschrift*.

## Zertifikate

Ein *digitales Zertifikat* ist ein von einer *Certificate Authority* unterschriebener Text, der bezeugt, dass ein öffentlicher Schlüssel zu einer bestimmten Person gehört.