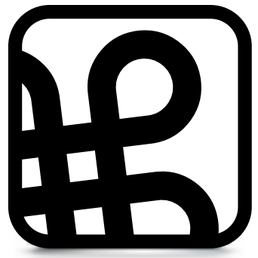


Macoun’11

QEEVEE

Kommunikativer Stabilbaukasten

Pascal Bihler

Ab, Lauf!

- Beziehungen spielen lassen
- Voll Konkret
- Privat werden

Beziehungen spielen lassen

(dt.: die Connections)

Verbindung zur Welt

- Das iPhone hat zwei unabhängige Daten-Kommunikationswege
  - WiFi (WLAN)
  - Mobilfunknetz
    - UMTS
    - EDGE
    - einfaches GPRS

Paketvermittelt
Asynchron
Ohne Garantien

Das Internet Protokoll

- IP-Adresse identifiziert Kommunikationspartner
  - Ändert sich bei Trägerwechsel!
- IP-Datenübertragung unsicher
  - User Datagram Protocol (UDP) gibt dies direkt weiter
  - Transmission Control Pr. (TCP) garantiert Datenübertragung
- Übertragungsabbrüche / -unterbrechungen immer möglich!

Network Link Conditioner

- Simulationstool
  - seit XCode 4/ Lion

`/Developer/Applications/Utilities/Network Link Conditioner`

Wahl der Übertragungstechnik

- Anwendungsfallabhängig:
  - UDP funktioniert auch!
  - Um die Übertragungsgarantien muss man sich bei Bedarf selbst kümmern

```
CFSocketRef newSocket = CFSocketCreate(
                        NULL,
                        PF_INET,
                        SOCK_DGRAM,
                        IPPROTO_UDP,
                        kCFSocketReadCallBack,
                        NetworkSocketCallBack,
                        &context);
…
CFSocketError err = CFSocketConnectToAddress(newSocket, address, 0.1);
```

```
CFSocketError err = CFSocketSendData(self.networkSocket,
                                     NULL, (CFDataRef) data, 0.1);
```

```
ssize_t bytesRead = recv(CFSocketGetNative(self.networkSocket),
                  buffer,
                  sizeof(buffer),
                  MSG_DONTWAIT);
```

```
CFSocketRef newSocket = CFSocketCreate(
                        NULL,
                        PF_INET,
                        SOCK_DGRAM,
                        IPPROTO_UDP,
                        kCFSocketReadCallBack,
                        NetworkSocketCallBack,
                        &context);
…
CFSocketError err = CFSocketConnectToAddress(newSocket, address, 0.1);
```

… weitere 115 Zeilen

UdpObjectiveC.m

```
CFSocketError err = CFSocketSendData(self.networkSocket,
                                     NULL, (CFDataRef) data, 0.1);
```

```
ssize_t bytesRead = recv(CFSocketGetNative(self.networkSocket),
                  buffer,
                  sizeof(buffer),
                  MSG_DONTWAIT);
```

```
final DatagramSocket socket = new DatagramSocket();
InetSocketAddress dest = new InetSocketAddress(…, UDP_LISTENER_PORT);

socket.connect(dest);
```


```
DatagramPacket packet = new DatagramPacket(text.getBytes("UTF-8"),
                                           text.length(), dest);
socket.send(packet);
```

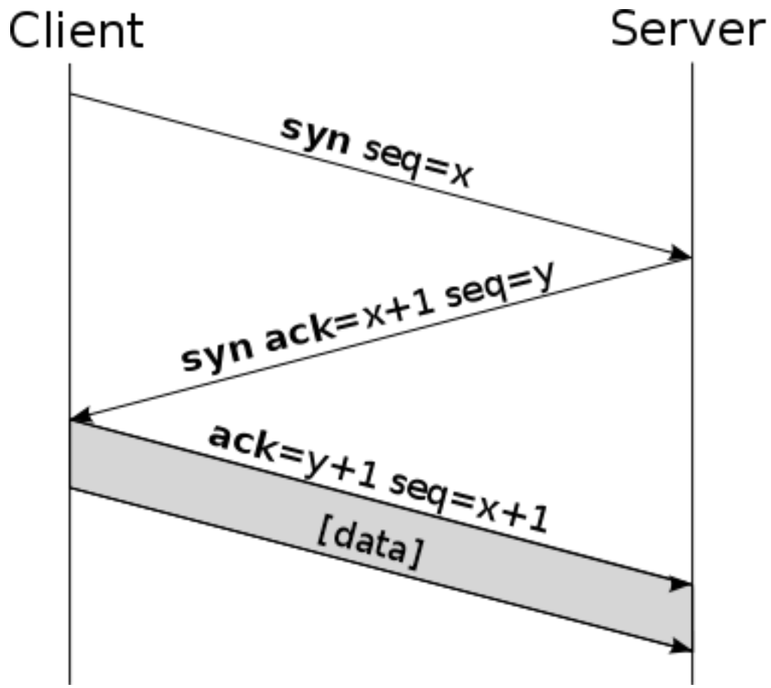
UdpSender.java


```
DatagramPacket inPacket =
             new DatagramPacket(new byte[MAX_UDP_PACKETLENGTH],
                                                MAX_UDP_PACKETLENGTH);
socket.receive(inPacket);
String payload =
             new String(inPacket.getData(), 0, inPacket.getLength());
```

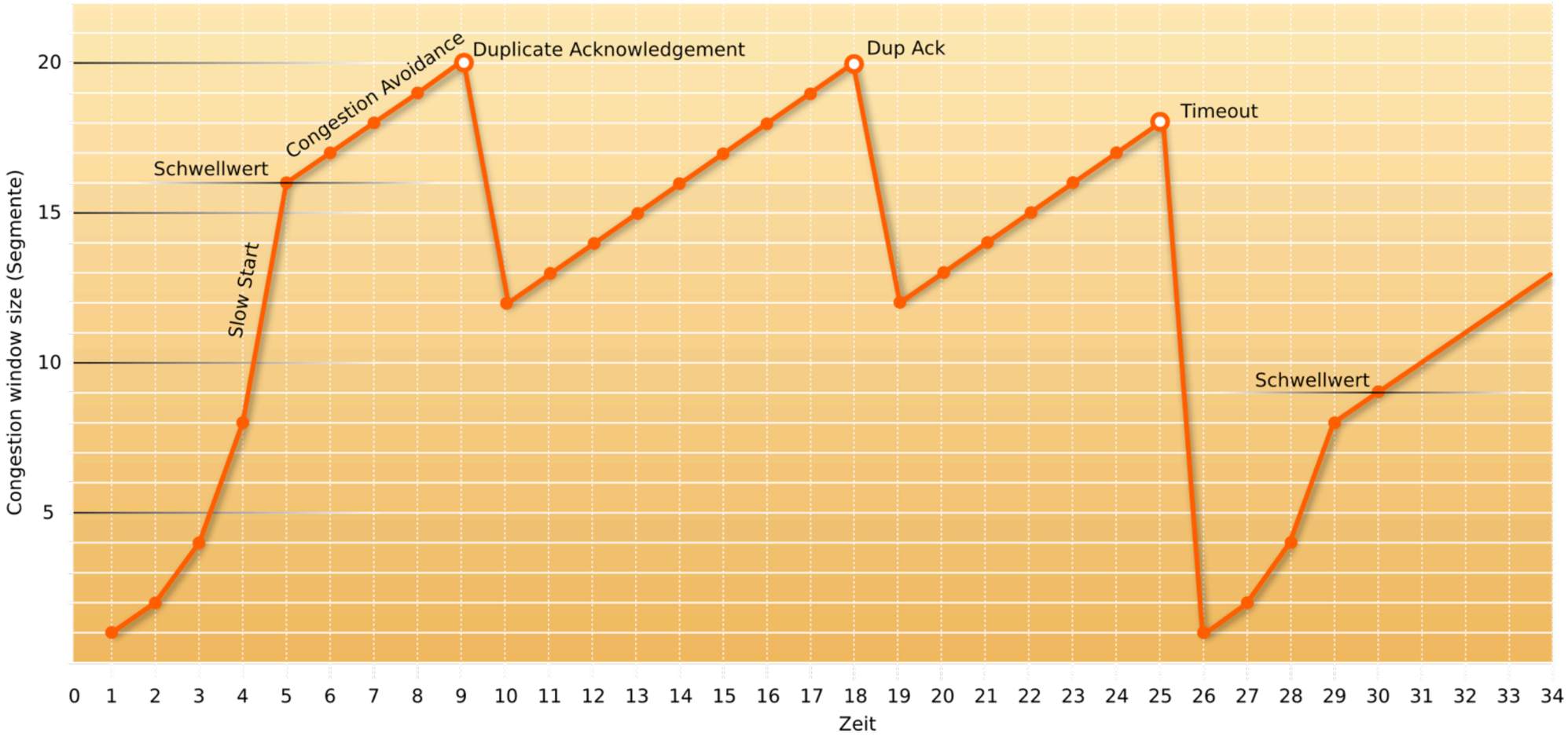
UdpListener.java

Transmission Control Protocol

- TCP verwendet Sequenznummern und Bestätigungen



TCP Slow-Start



Warum trotzdem HTTP?

- Gut gekapselt vom Betriebsystem
- Je nach Implementierung kleiner Overhead
- Meta-Daten-Übertragung spezifiziert
- Transparente Komprimierung möglich
- REST-Kompatibilität
  - Client-Bibliotheken

Resty für Objective-C

```
- (void)fetchSomething
{
  [[LRResty client] get:@"http://www.example.com"
   withBlock:^(LRRestyResponse *r) {
    NSLog(@"That's it! %@", [r asString]);
  }];
}
```

http://projects.lukeredpath.co.uk/resty/

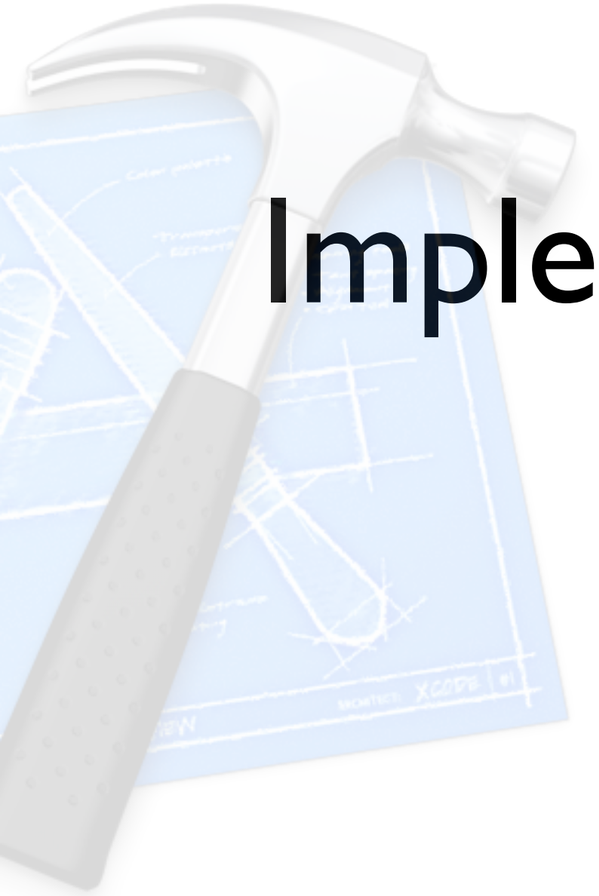
Voll Konkret

(weiter im Beispiel)

Anwendungs-Beispiel

- Beispiel 2009: „MacRunner“

- Erweiterung 2011: Wegpunkte online nachverfolgen können



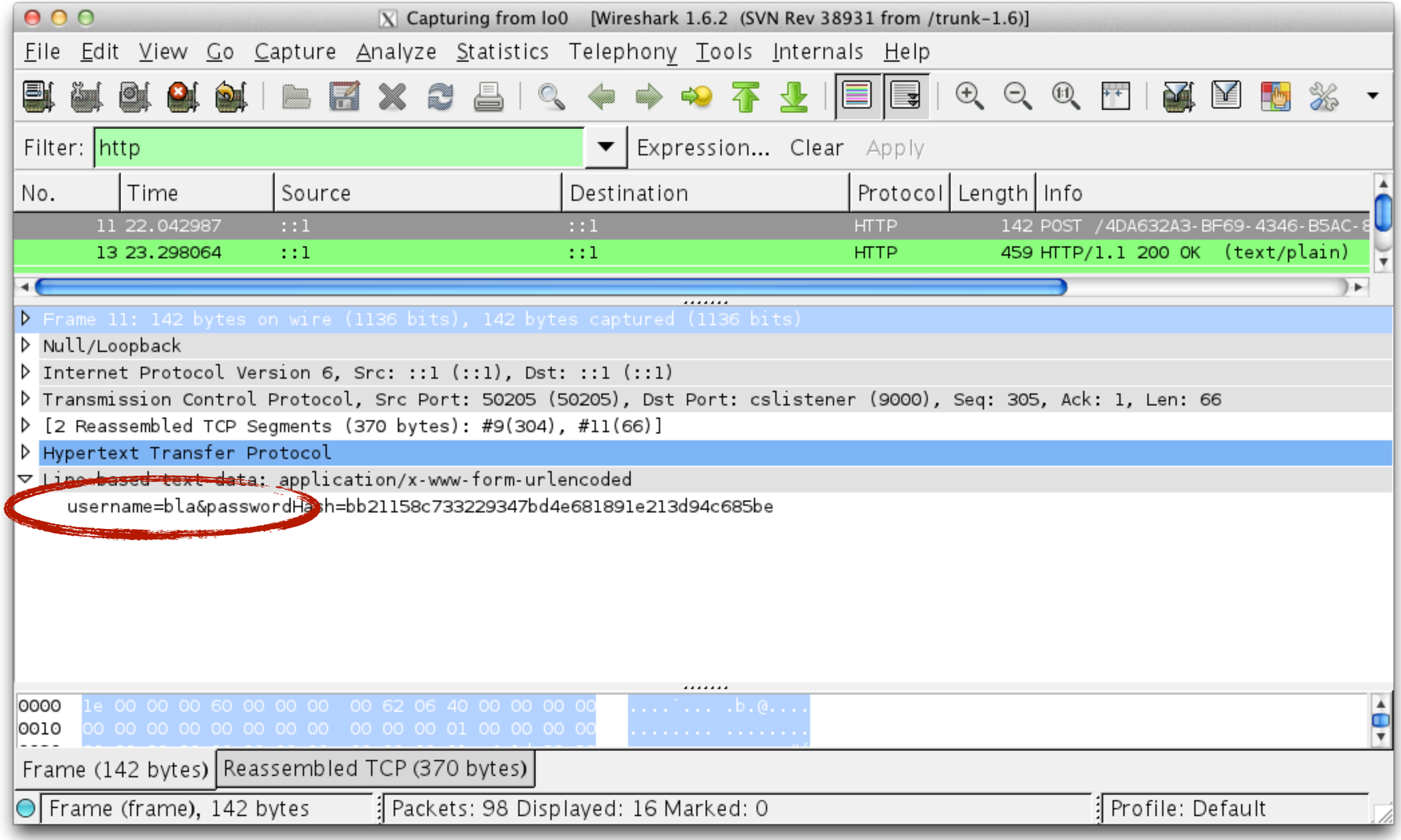
Implementierung…

Mitprogrammieren:

http://pascal-bihler.de/macoun2011.zip

Privat werden

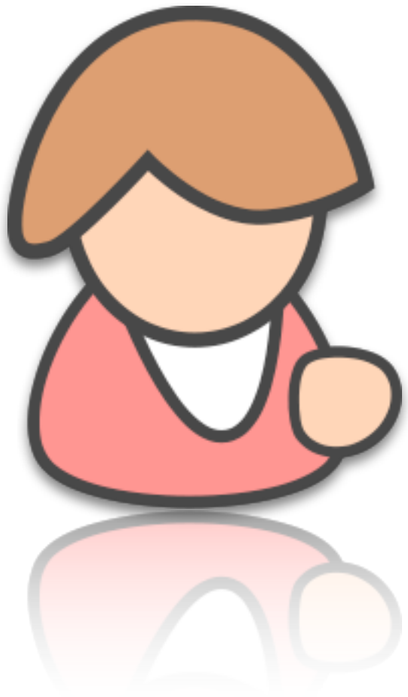
(You are not alone)

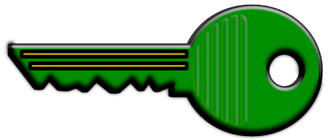

Capturing from lo0 [Wireshark 1.6.2 (SVN Rev 38931 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: http
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
11 22.042987 ::1 ::1 HTTP 142 POST /4DA632A3-BF69-4346-B5AC-8
13 23.298064 ::1 ::1 HTTP 459 HTTP/1.1 200 OK (text/plain)
Frame 11: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits)
Null/Loopback
Internet Protocol Version 6, Src: ::1 (::1), Dst: ::1 (::1)
Transmission Control Protocol, Src Port: 50205 (50205), Dst Port: cslistener (9000), Seq: 305, Ack: 1, Len: 66
[2 Reassembled TCP Segments (370 bytes): #9(304), #11(66)]
Hypertext Transfer Protocol
Line-based text data: application/x-www-form-urlencoded
username=bla&passwordHash=bb21158c733229347bd4e681891e213d94c685be
0000 1e 00 00 00 60 00 00 00 00 62 06 40 00 00 00 00 ....`.... .b.@....
0010 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 ....
Frame (142 bytes)
Reassembled TCP (370 bytes)
Frame (frame), 142 bytes
Packets: 98 Displayed: 16 Marked: 0
Profile: Default

Verschlüsselung

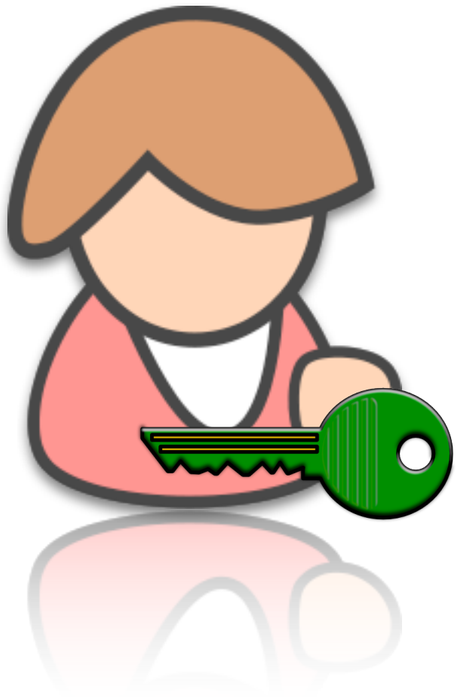
- HTTP ist ein Klartextprotokoll
- Verschlüsselung sensibler Daten notwendig
  - Symmetrische Verschlüsselung
  - Asymmetrische Verschlüsselung

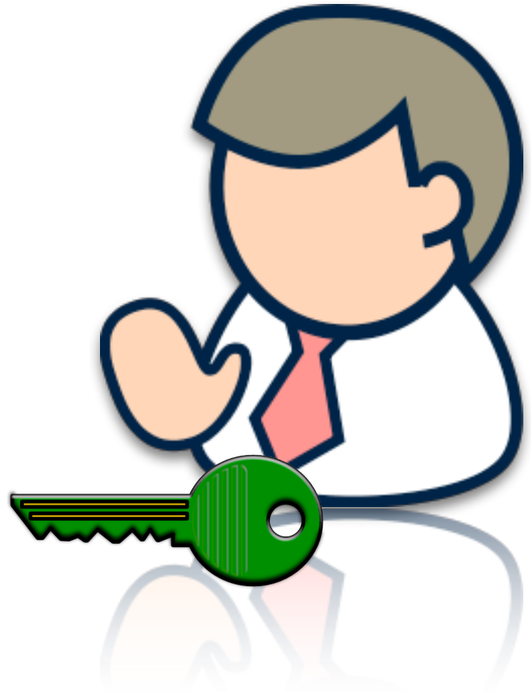
Symmetrische Verschlüsselung



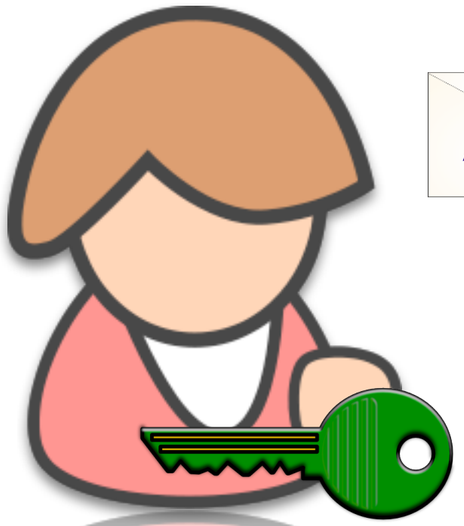


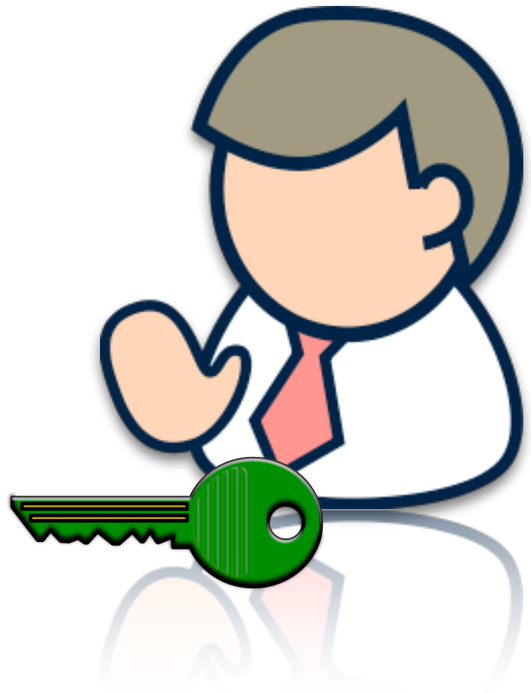
Symmetrische Verschlüsselung





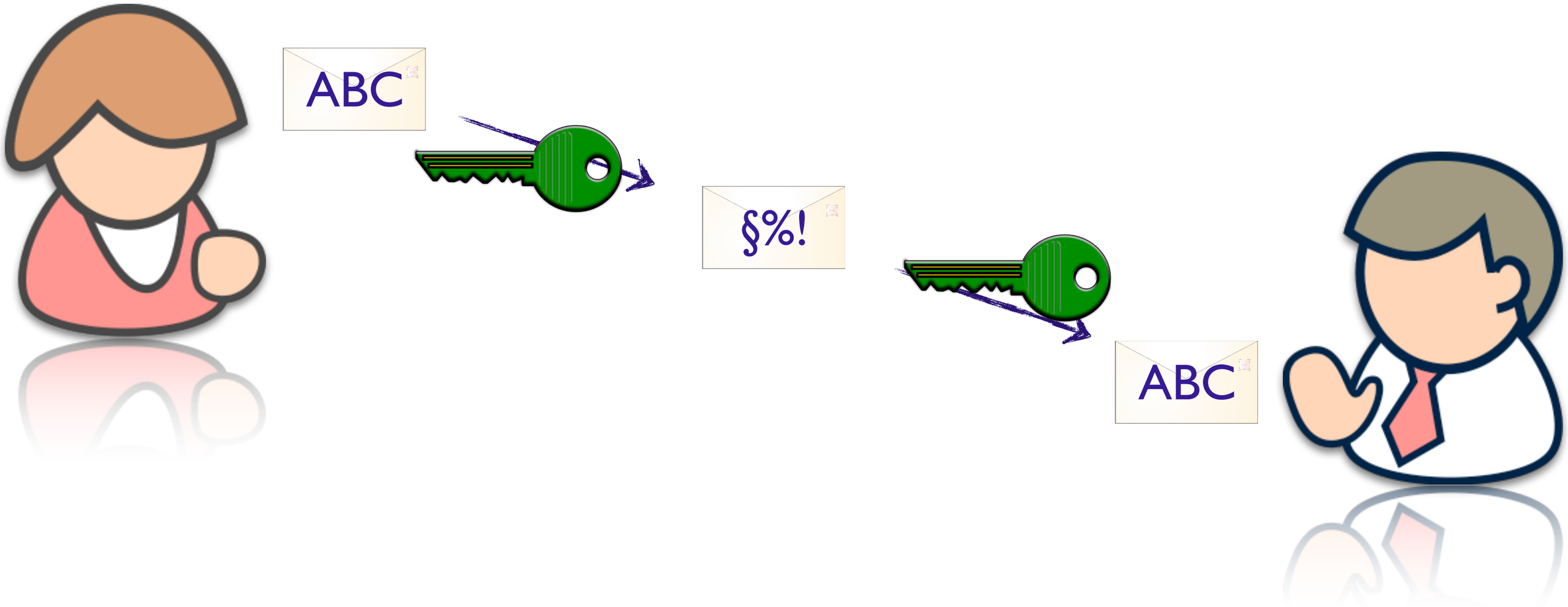
Symmetrische Verschlüsselung



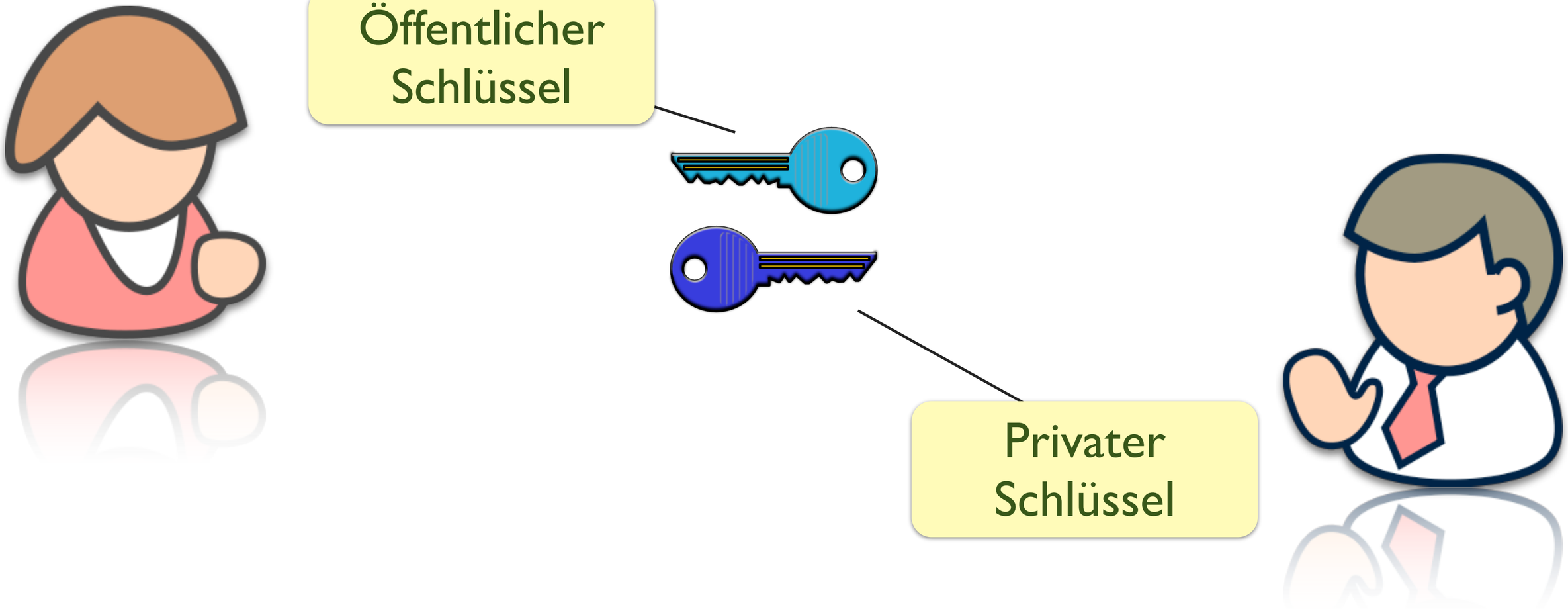


Symmetrische Verschlüsselung

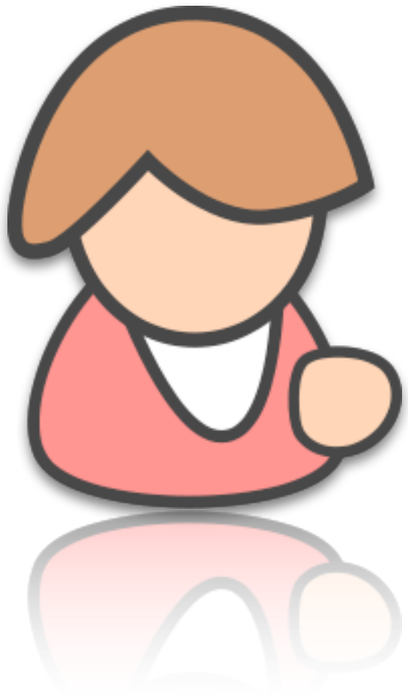
Symmetrische Verschlüsselung

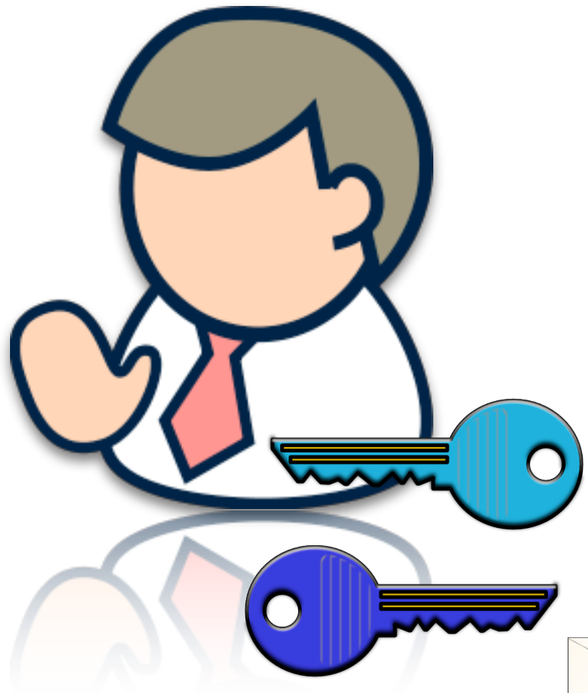


Asymmetrische Verschlüsselung

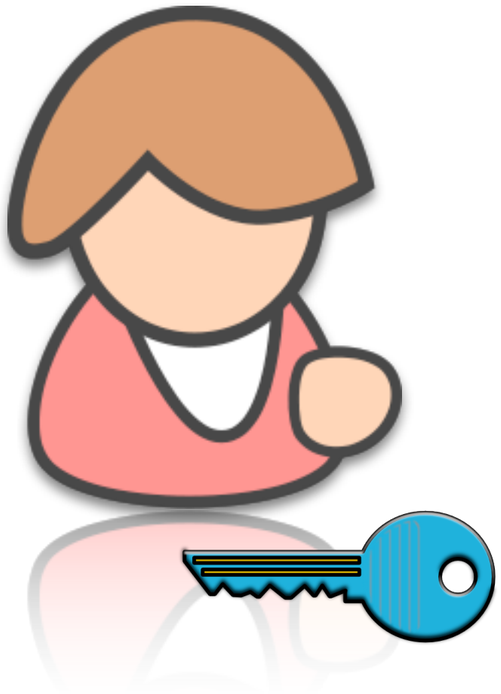


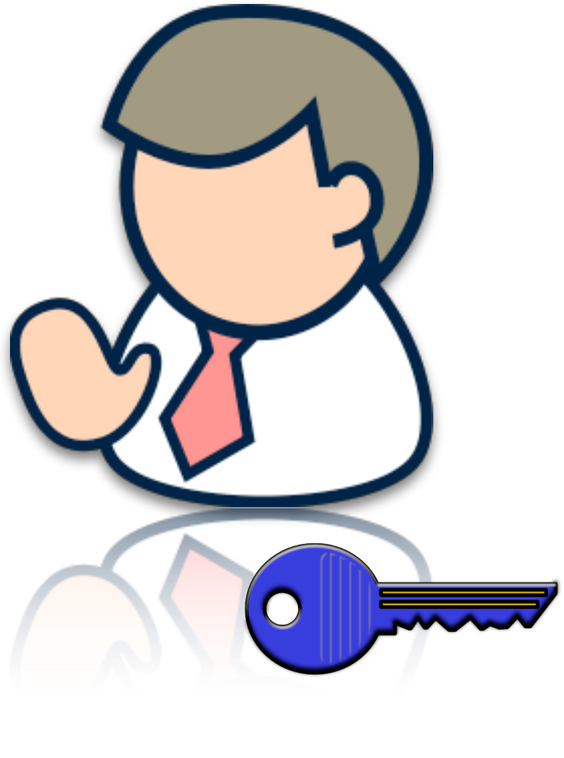
Asymmetrische Verschlüsselung



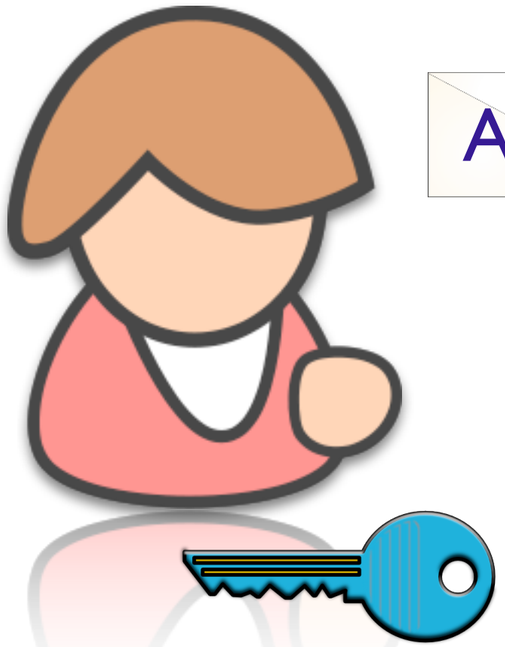


Asymmetrische Verschlüsselung

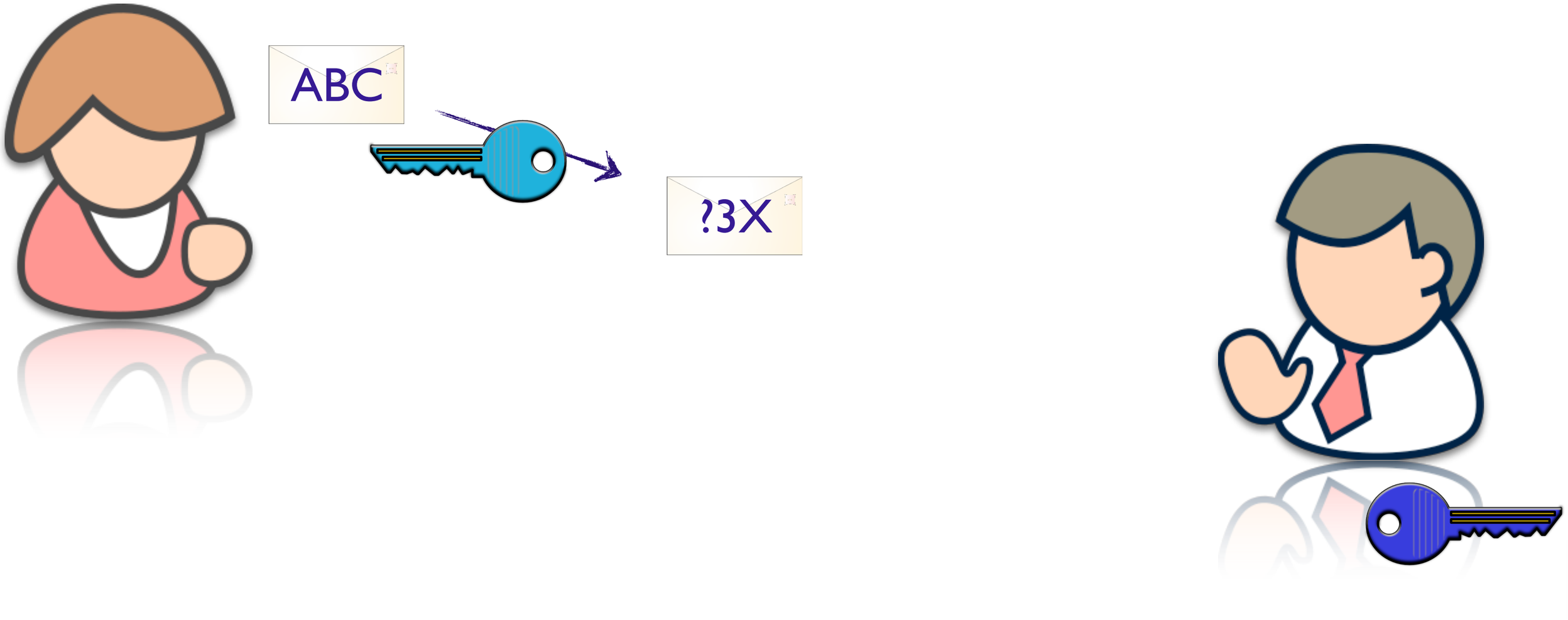




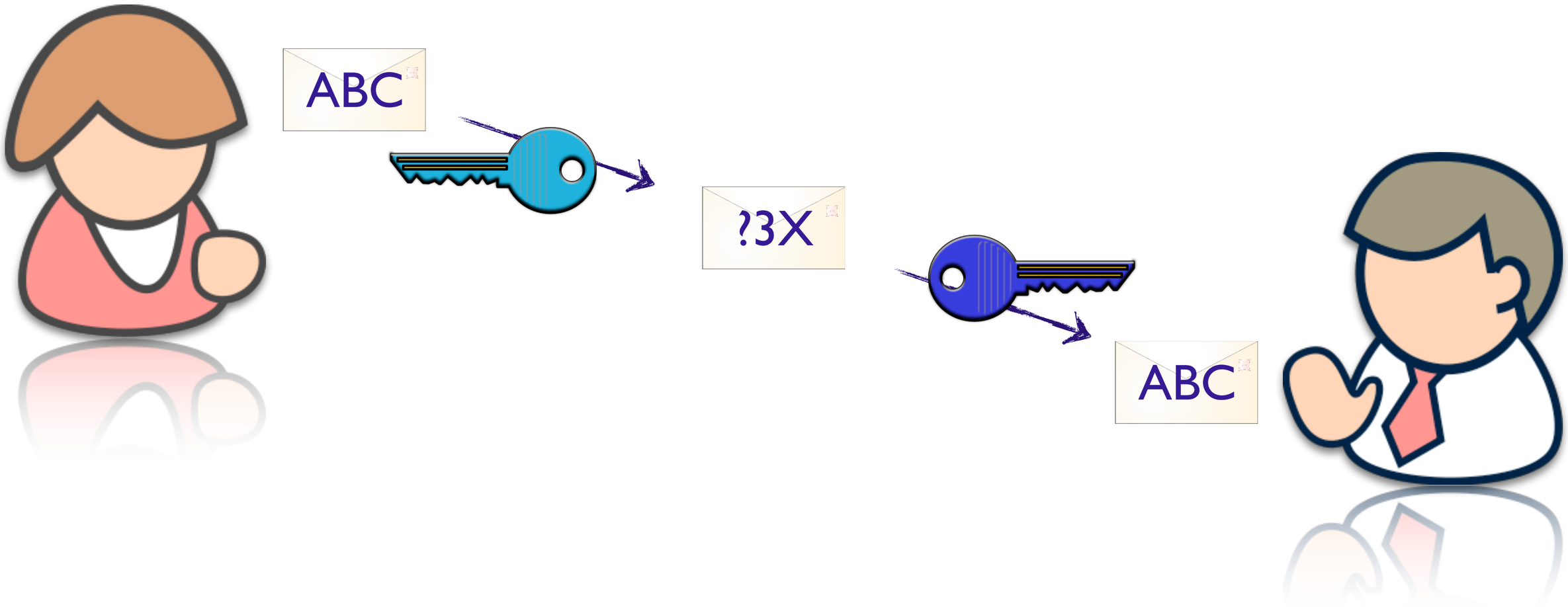
Asymmetrische Verschlüsselung



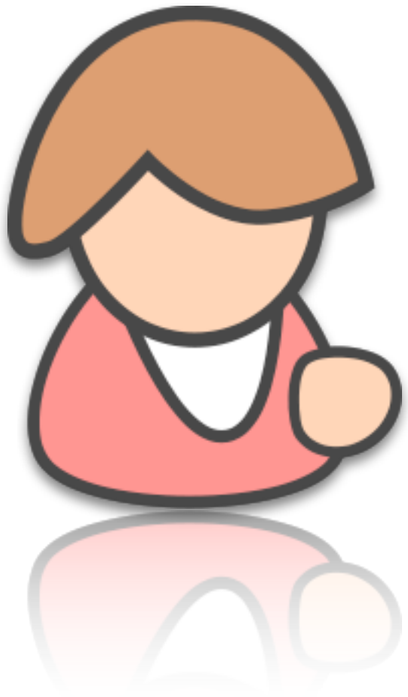
Asymmetrische Verschlüsselung

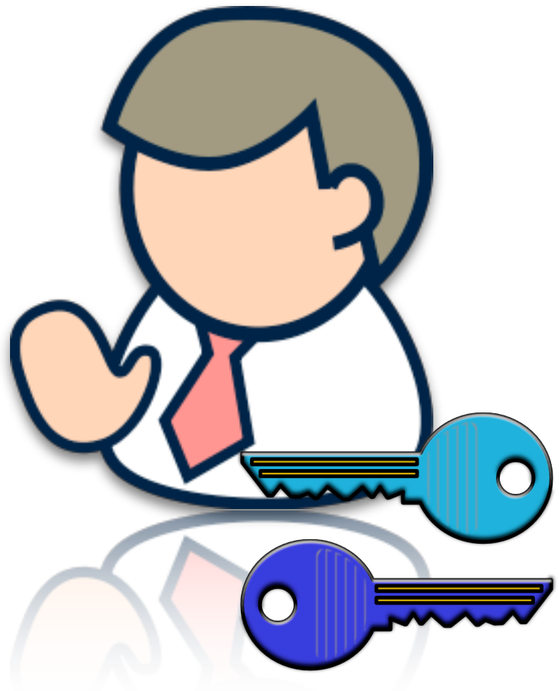


Asymmetrische Verschlüsselung

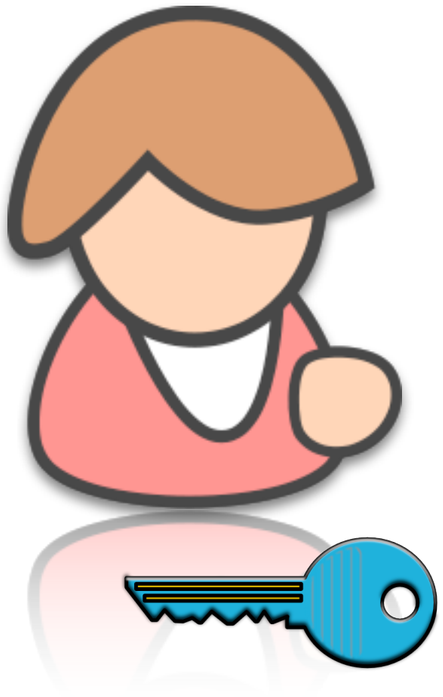


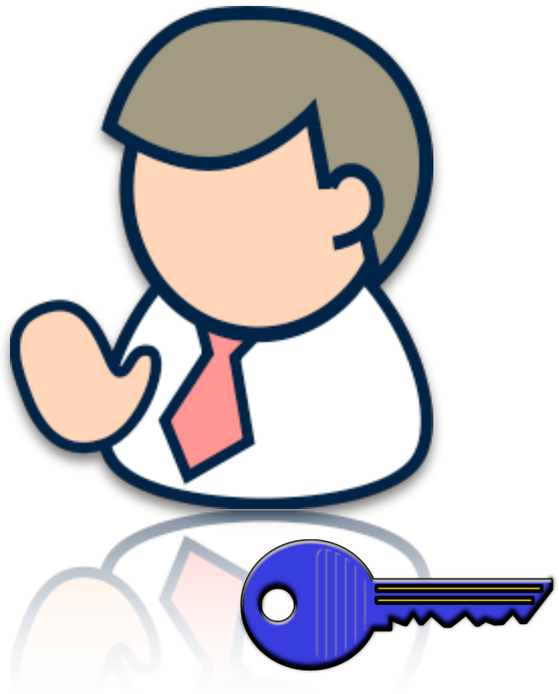
Kombiniertes Verfahren



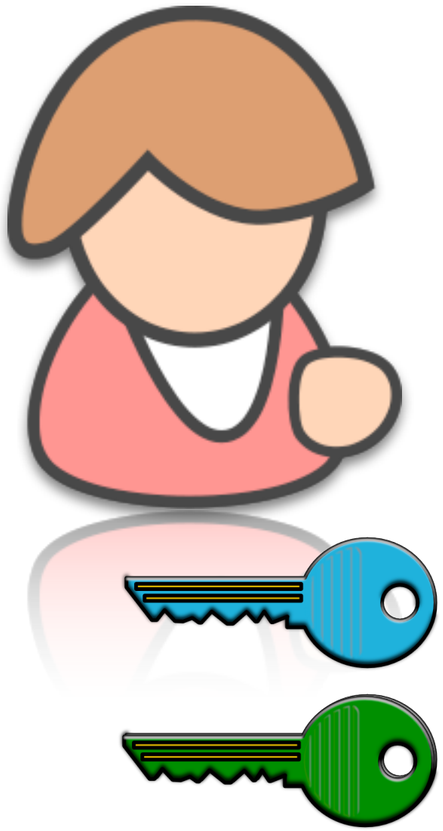


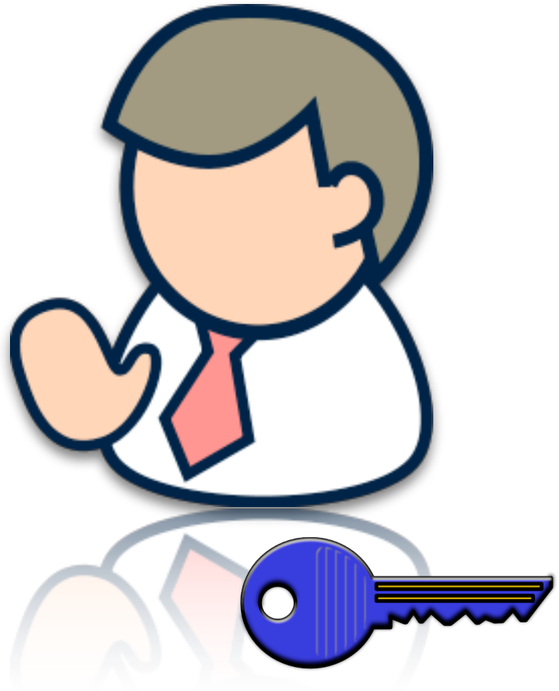
Kombiniertes Verfahren



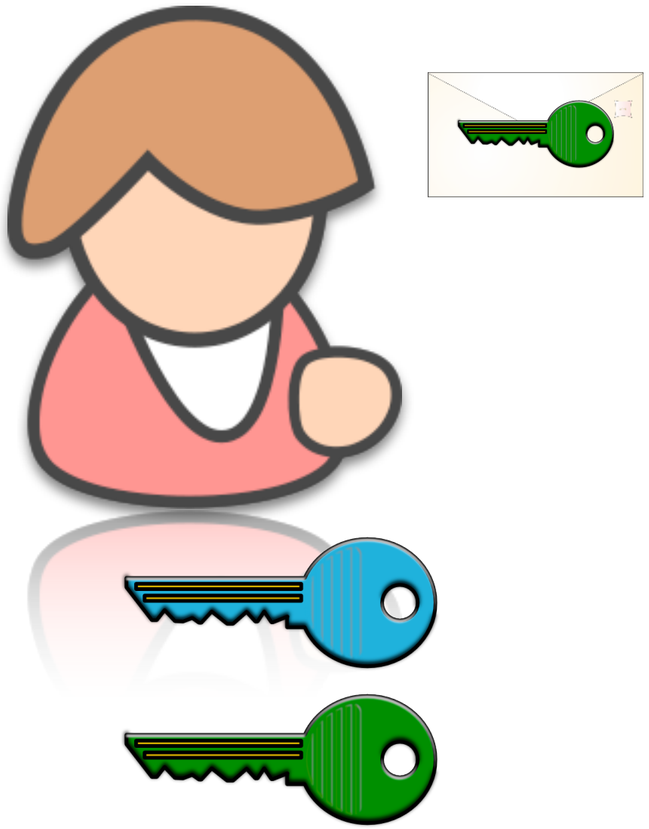


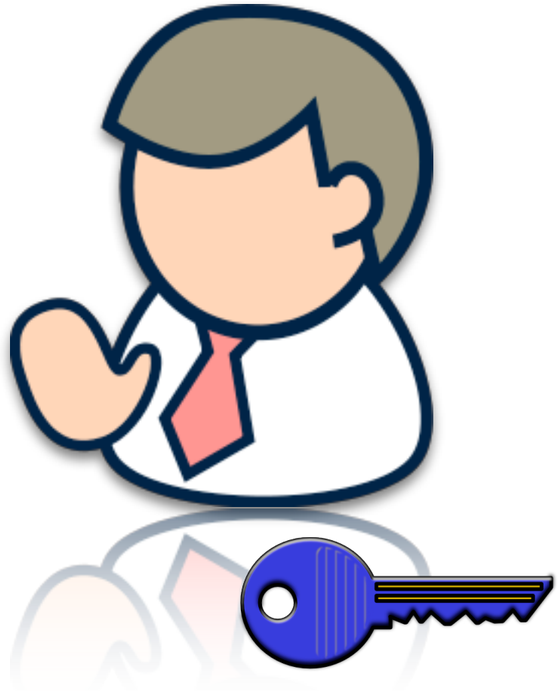
Kombiniertes Verfahren



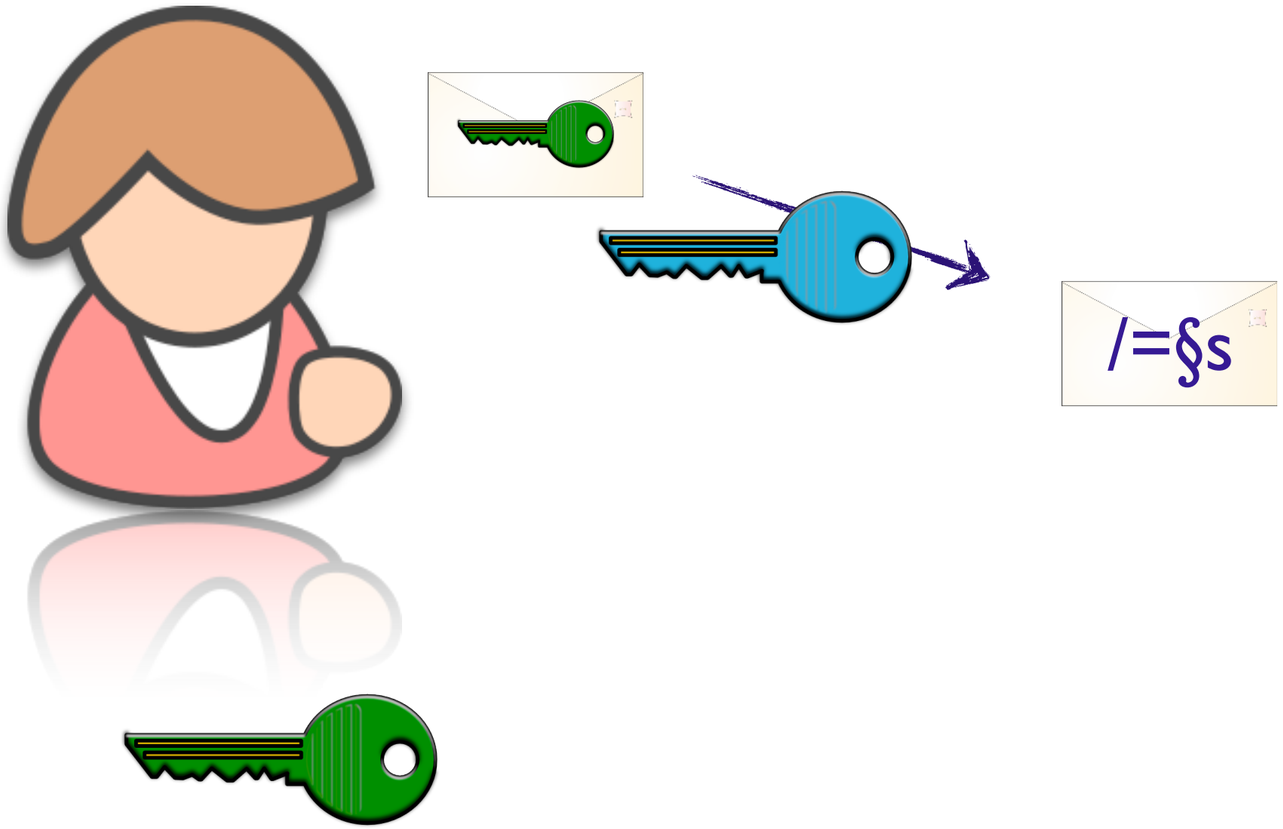


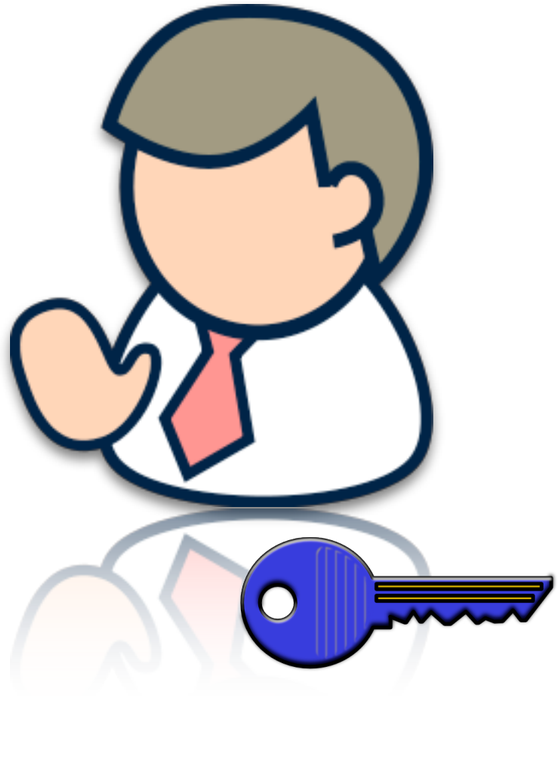
Kombiniertes Verfahren



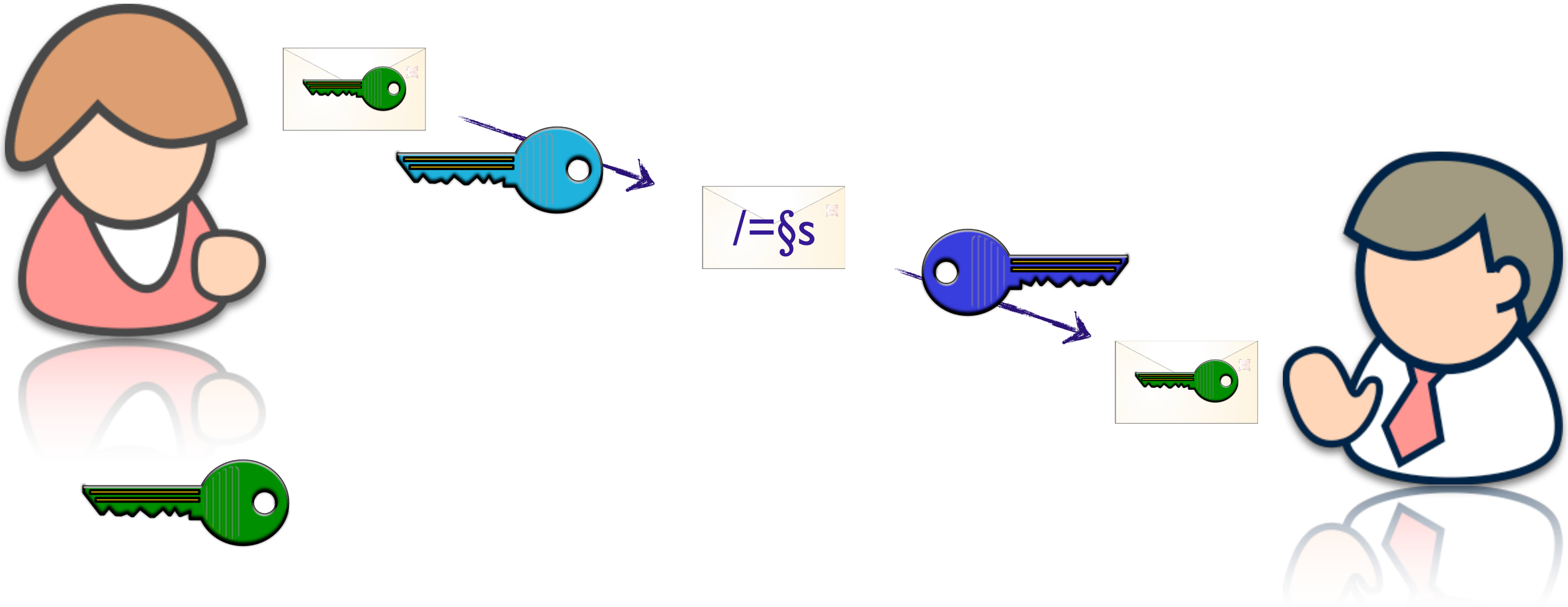


Kombiniertes Verfahren

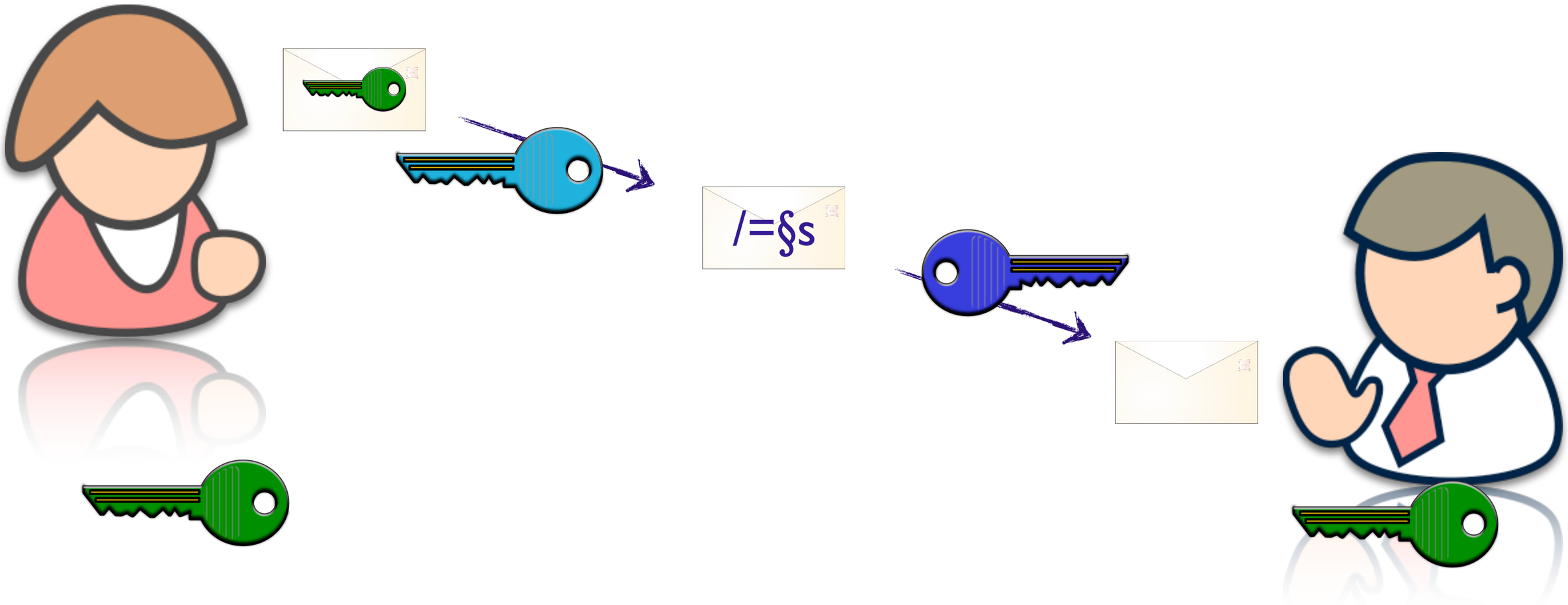




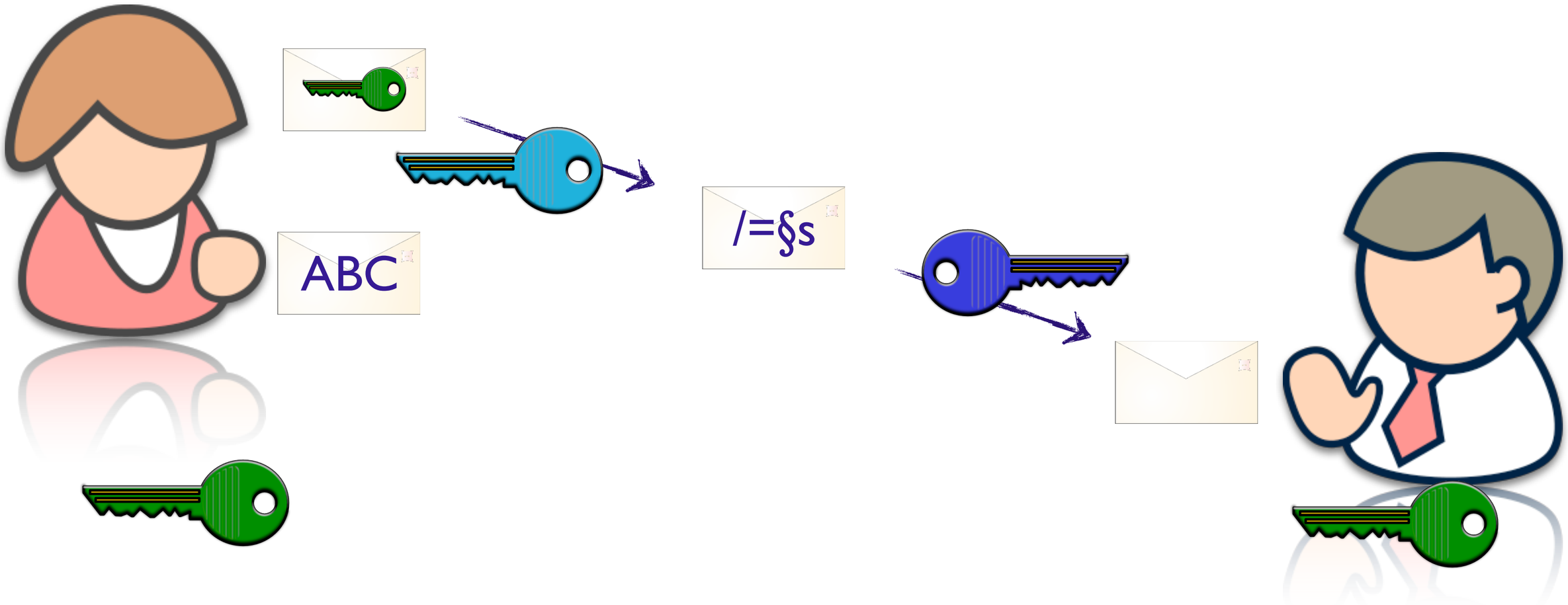
Kombiniertes Verfahren



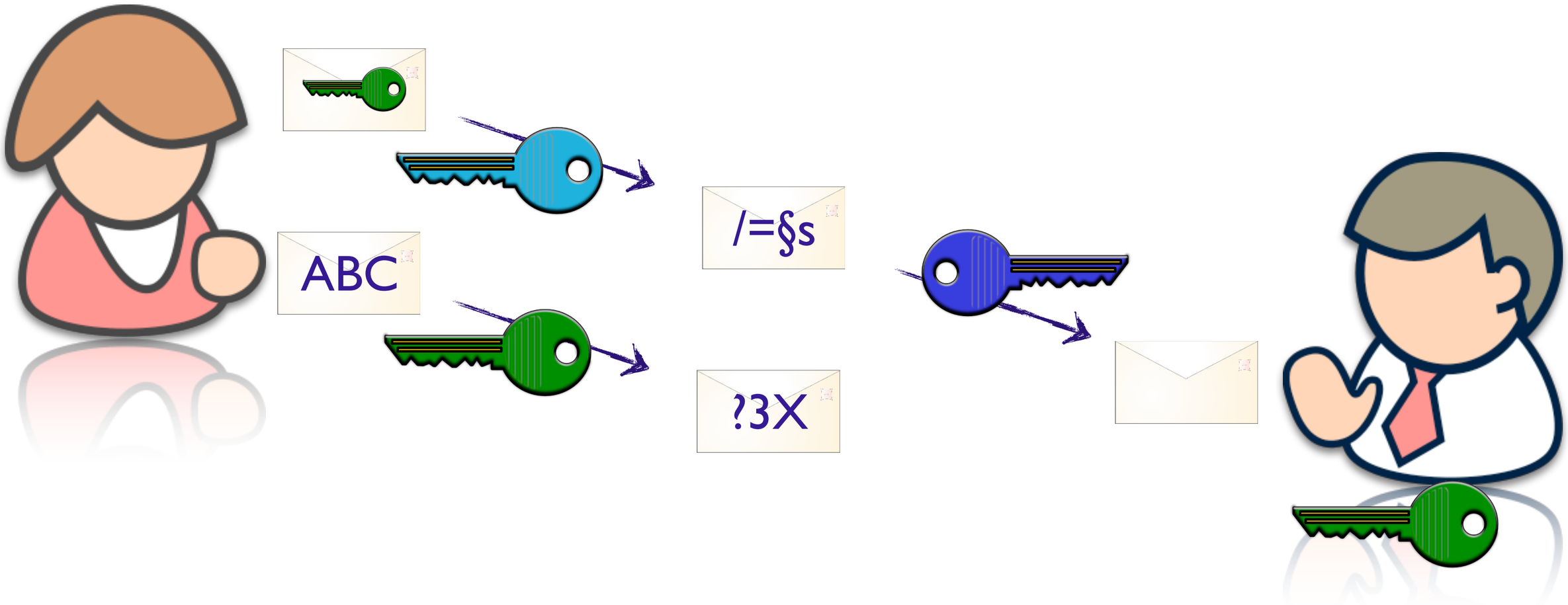
Kombiniertes Verfahren



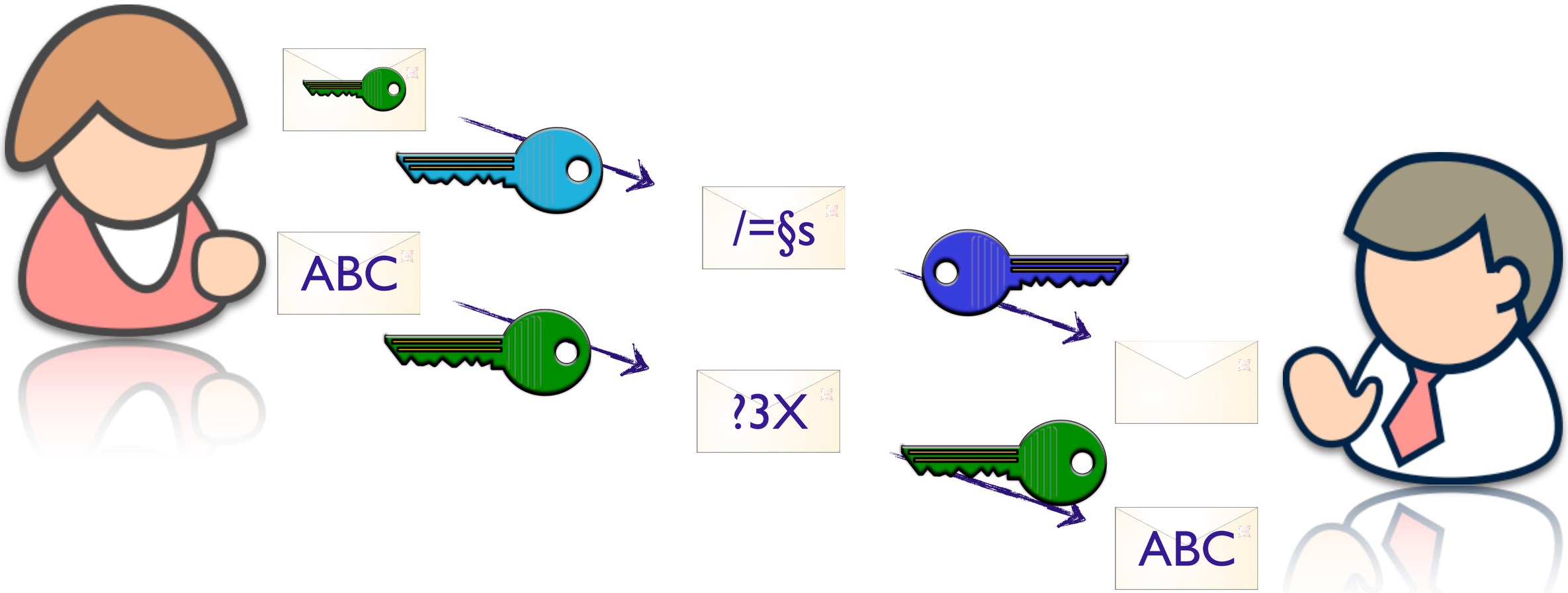
Kombiniertes Verfahren



Kombiniertes Verfahren



Kombiniertes Verfahren



HTTP**S**

- TLS (aka SSL) ermöglicht verschlüsselte Datenübertragung mit Bordmitteln
- Serverzertifikat muss geprüft werden
  - Checksummenvergleich

```
NSURLRequest *theRequest=[NSURLRequest requestWithURL:
                                    [NSURL URLWithString:@"https://…"];

NSURLConnection * theConnection = [[NSURLConnection alloc]
                            initWithRequest:theRequest delegate:self];
…
```

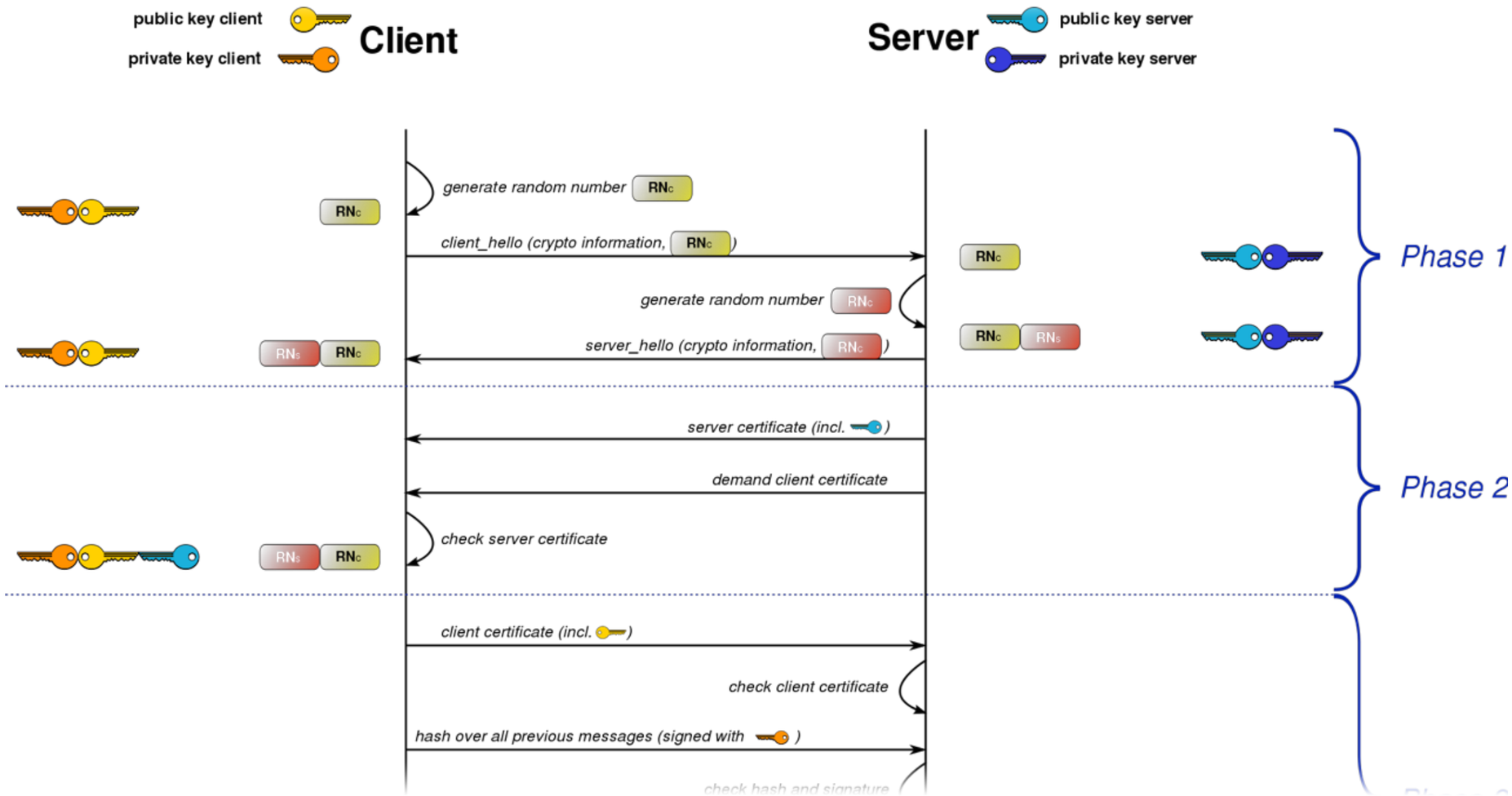
```
#define SERVER_CERT_HASH @"1893bf2d3e527d44f6f1b68632f97e237a48cae8"

- (BOOL)connection:(NSURLConnection *)connection
            canAuthenticateAgainstProtectionSpace:… {
    return ([[protectionSpace authenticationMethod]
         isEqual:NSURLAuthenticationMethodServerTrust]);
}

- (void)connection:(NSURLConnection *)connection
            didReceiveAuthenticationChallenge:… {
             …
  cert = SecTrustGetCertificateAtIndex(trust, certIndex);
  certData = SecCertificateCopyData(cert);
  trusted |= [SERVER_CERT_HASH isEqual:[self sha1HexDigest:(NSData*)
  certData]];
  …
```

```
#define SERVER_CERT_HASH @"1893bf2d3e527d44f6f1b68632f97e237a48cae8"

- (BOOL)connection:(NSURLConnection *)connection
             canAuthenticateAgainstProtectionSpace:… {
    return ([[protectionSpace authenticationMethod]
          isEqual:NSURLAuthenticationMethodServerTrust]);
}

- (void)connection:(NSURLConnection *)connection
             didReceiveAuthenticationChallenge:… {
              …
  cert = SecTrustGetCertificateAtIndex(trust, certIndex);
  certData = SecCertificateCopyData(cert);
  trusted |= [SERVER_CERT_HASH isEqual:[self sha1HexDigest:(NSData*)
  certData]];
  …
```

Insgesamt ca. 60 Zeilen

certcheck.m

HTTPS Handshake



Eigene Verschlüsselung

- Nur zu Beginn der Kommunikation
- Client erstellt zufälligen AES-Schlüssel
- Sendet diesen verschlüsselt mit bekanntem RSA-Public-Key zum Server
- Server dekodiert diesen mittels RSA-Private-Key
- Security.framework

AES (Symmetrisch)

```
#import <CommonCrypto/CommonCryptor.h>

CCCryptorStatus cryptStatus = CCCrypt(kCCDecrypt, kCCAlgorithmAES128,
                                      kCCOptionPKCS7Padding | kCCOptionECBMode,
                                      keyPtr, kCCKeySizeAES128,
                                      NULL,
                                      [self bytes], dataLength, /* input */
                                      buffer, bufferSize, /* output */
                                      &numBytesDecrypted);
```

RSA (Asymmetrisch)

```
#import <CommonCrypto/CommonCryptor.h>


SecKeyRef publicKey = SecTrustCopyPublicKey(trust);

OSStatus  sanityCheck = SecKeyEncrypt(publicKey,
                        kSecPaddingPKCS1,
                        (const uint8_t *)[self bytes],
                        keyBufferSize,
                        cipherBuffer,
                        &cipherBufferSize);
```

```
Cipher c = Cipher.getInstance("AES");
SecretKeySpec k = new SecretKeySpec(key, "AES");

c.init(Cipher.ENCRYPT_MODE, k);
byte[]  encryptedData = c.doFinal(clear.getBytes());

c.init(Cipher.DECRYPT_MODE, k);
byte[]  decryptedData = c.doFinal(encrypted.getBytes());
```


```
KeyStore keyStore = KeyStore.getInstance("PKCS12");
keyStore.load(new FileInputStream(…), password);

Cipher decryptCipher = Cipher.getInstance("RSA");
        decryptCipher.init(Cipher.DECRYPT_MODE,keyStore.getKey
("…",password));
byte[] messageDecrypte = decryptCipher.doFinal(messageCrypte);
```

```
Cipher c = Cipher.getInstance("AES");
SecretKeySpec k = new SecretKeySpec(key, "AES");

c.init(Cipher.ENCRYPT_MODE, k);
byte[]  encryptedData = c.doFinal(clear.getBytes());

c.init(Cipher.DECRYPT_MODE, k);
byte[]  decryptedData = c.doFinal(encrypted.getBytes());
```

```
CertificateFactory cf = CertificateFactory.getInstance("X.509");
trustedCert = (X509Certificate) cf.generateCertificate(Application.
          getInstance().getResources().openRawResource(R.raw.encryption));
encryptCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");

encryptCipher.init(Cipher.ENCRYPT_MODE,trustedCert.getPublicKey());
byte[] messageEncrypte = encryptCipher.doFinal(key);
```

Fragen?

Der finale Code:

http://pascal-bihler.de/macoun2011-final.zip

Vielen Dank