Cryptography

Lecture 3, Chapter 7

SSAS F2016
Søren Debois

Cryptography

Motivation

- Preserve confidentiality: only the intended recipient of a message should be able to read it.
- Preserve integrity: An adversary cannot (undetectedly) tamper with a message.

Plan

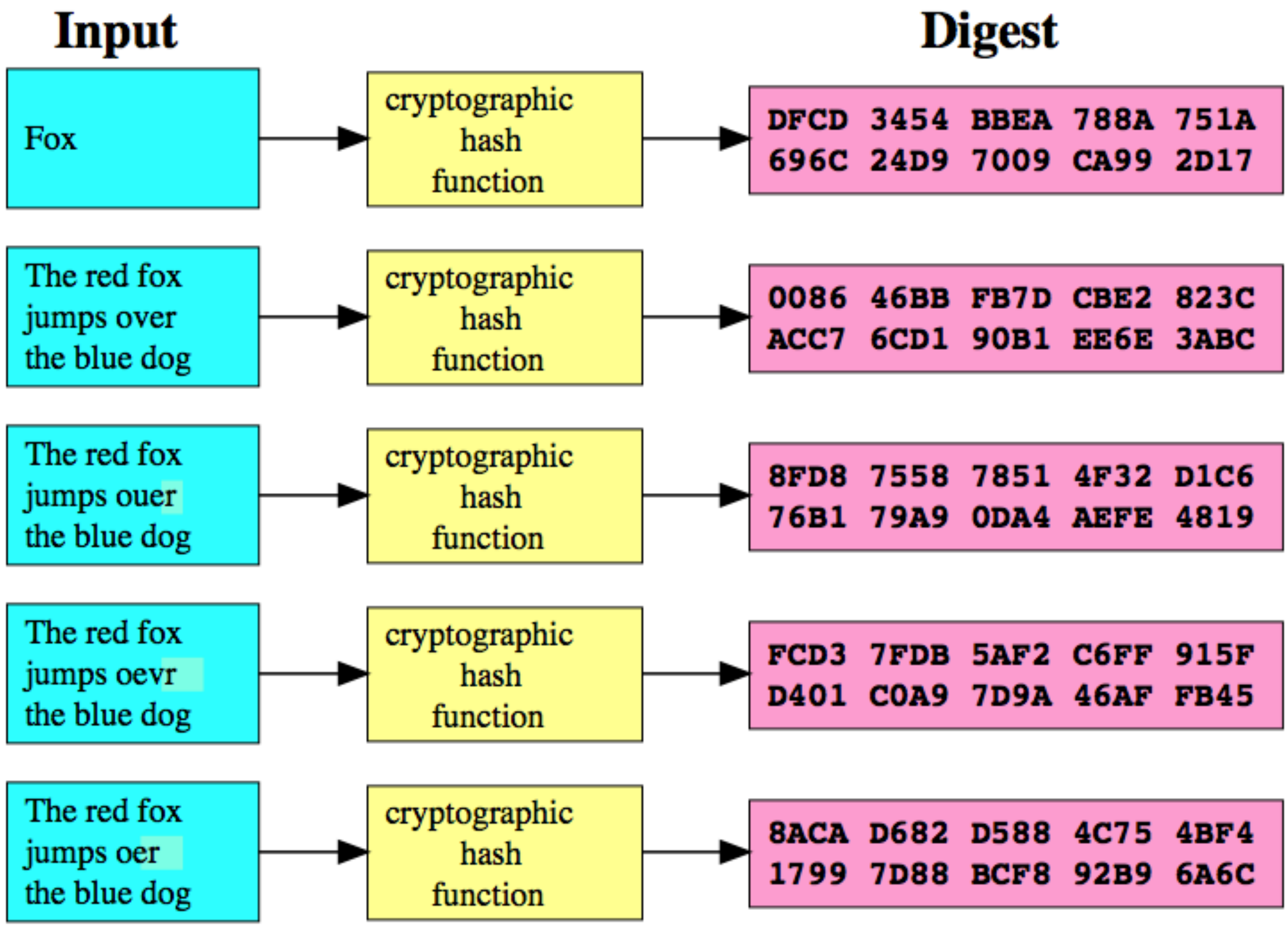
- Hashes
- Symmetric encryption schemes
- Asymmetric encryption schemes
- Signatures
- Certificates
- SSL/TLS

Hashes & Digests

Hashes, digests

- Hash function: Function taking arbitrary length data (“message”) to fixed-length value (“digest”).
- H(M) = h.
- Used in, e.g., hashing, hash table http://en.wikipedia.org/wiki/File:Cryptographic_Hash_Function.svg s (duh).
- Used in, e.g., verifying integrity.
- Used for storing passwords.

Example


Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Hash properties

- Given M, H(M) = h should be easy to compute
- Given h, finding M s.t. H(M) = h should be infeasible to compute
- Given M, finding M’ with H(M) = H(M’) should be infeasible to compute.

Implementations

- MD5. Broken ca. 2005. Collisions are easy to find.
- SHA-1. Discovered likely insecure ca. 2005. Used in SSL.
- SHA-2 aka SHA-256 or SHA-512.
  As yet unbroken.

Salt

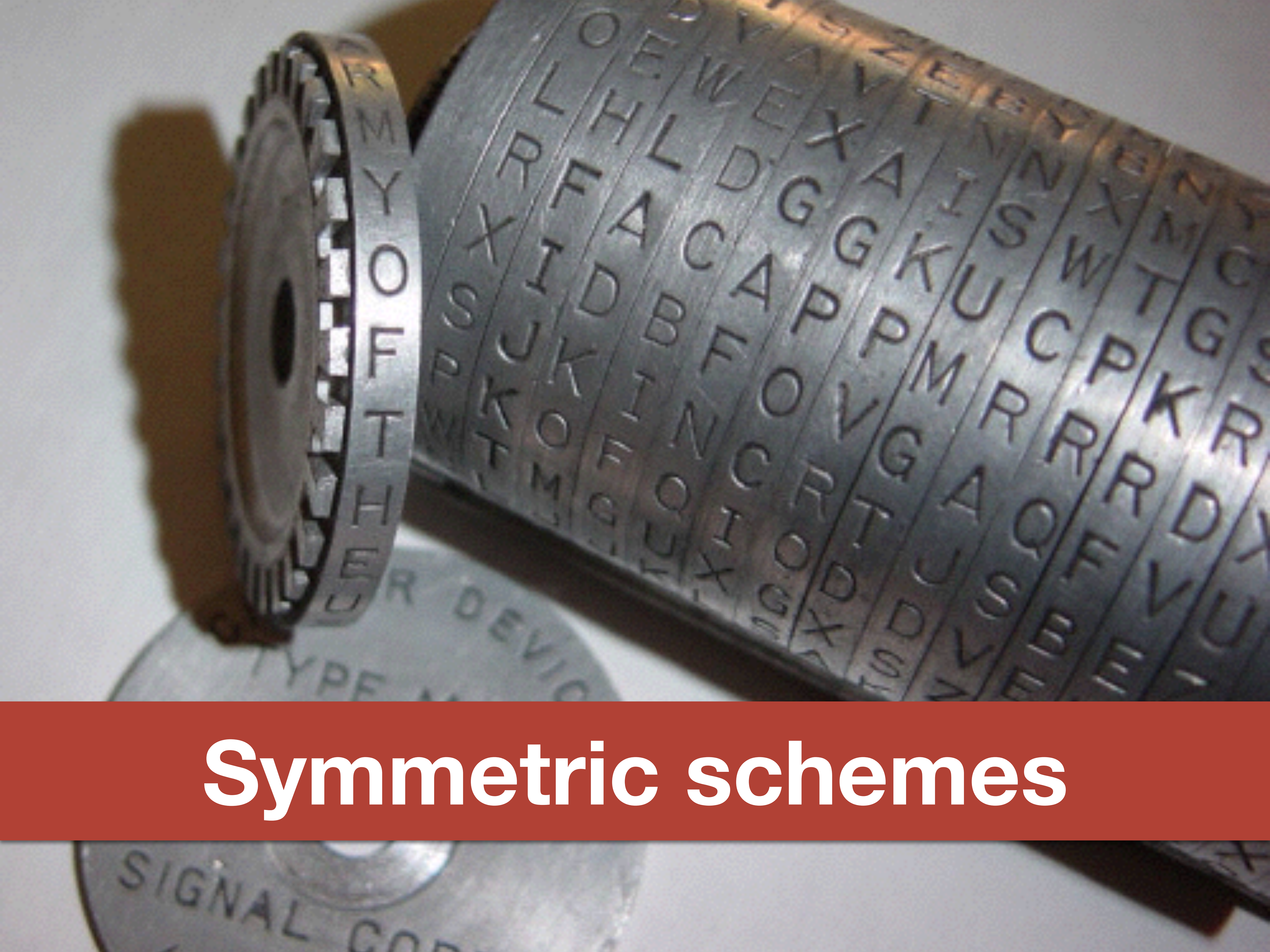
- Recall we store hashes of passwords. Users password input is hashed and compared with the stored hash.

- This works when inverting the hash is computationally infeasible. But:

- An adversary might precompute hashes for a large collection of typical passwords.

- To avoid this, we pick a random value, a *salt*, and add it to password before hashing.

- (Obviously, you need to store the salt with the password.)



Symmetric schemes

Symmetric algorithms

- Encryption: function from *secret key* and *plaintext* to *cipher text*
- Decryption: function from *secret key* and *ciphertext* to *plaintext*.
- $E(K,M) = \{M\}_K$
  $D(K,\{M\}_k) = M$
- Security depends on assumption that $D(\_, \{M\}_K)$ is *infeasible* to compute when you don't know K.
- Best attack: brute-force, chosen plaintext.

Symmetric

Caesar-cipher

Easy to break

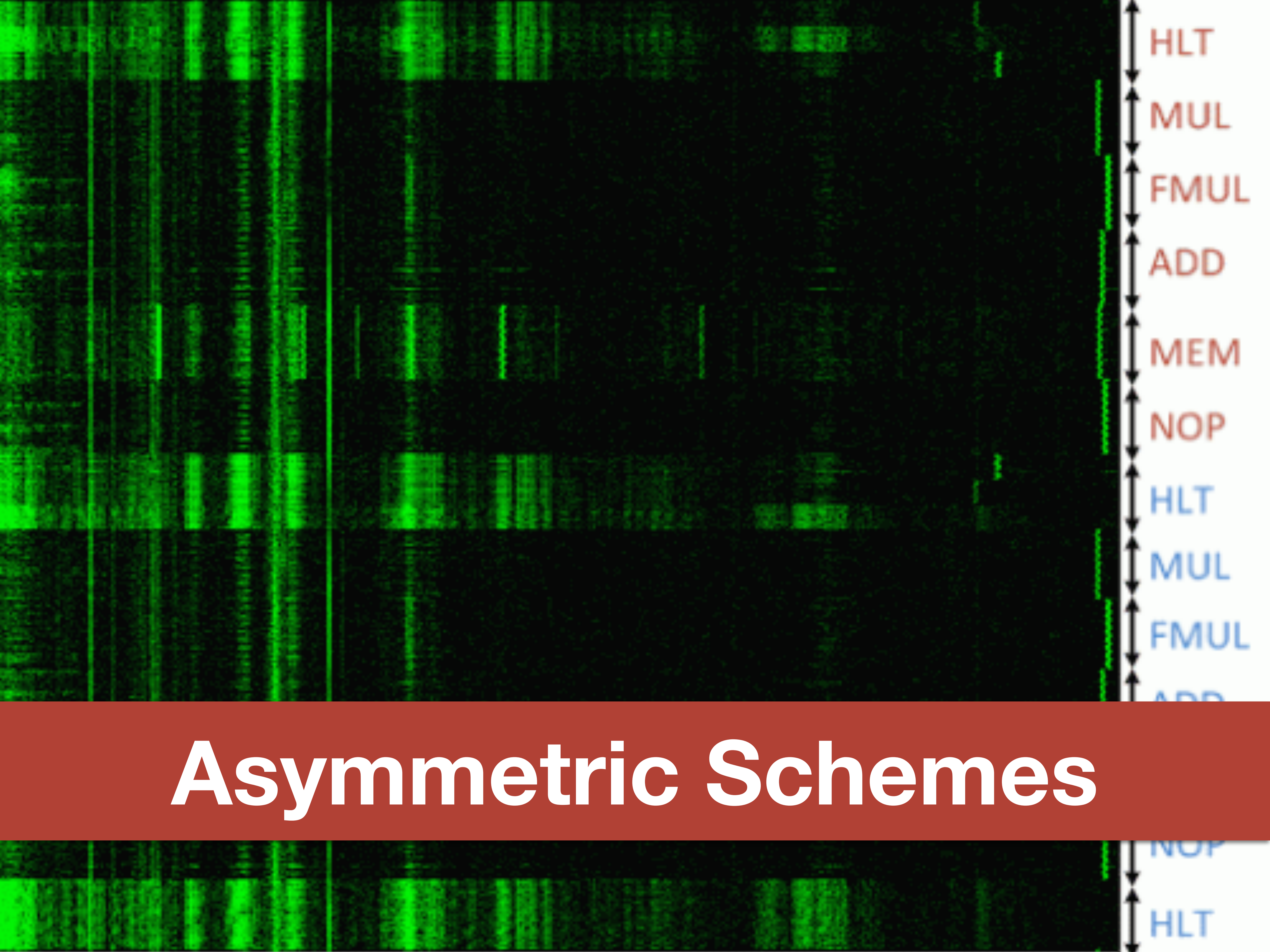
Frequency table for English text:
e: 12.7%, t: 9.1%, a: 8.2%, o: 7.5%

Implementations

- rot13
- DES (broken 1999, use Triple-DES)
- AES (Rijndael). No feasible attacks.
- RC4. Broken.

Symmetric scheme challenges

- Key distribution.
- E.g., how do a bank get key to every customer?
- In general, n parties need n^2 keys.



Asymmetric Schemes

Assymmetric encryption schemes

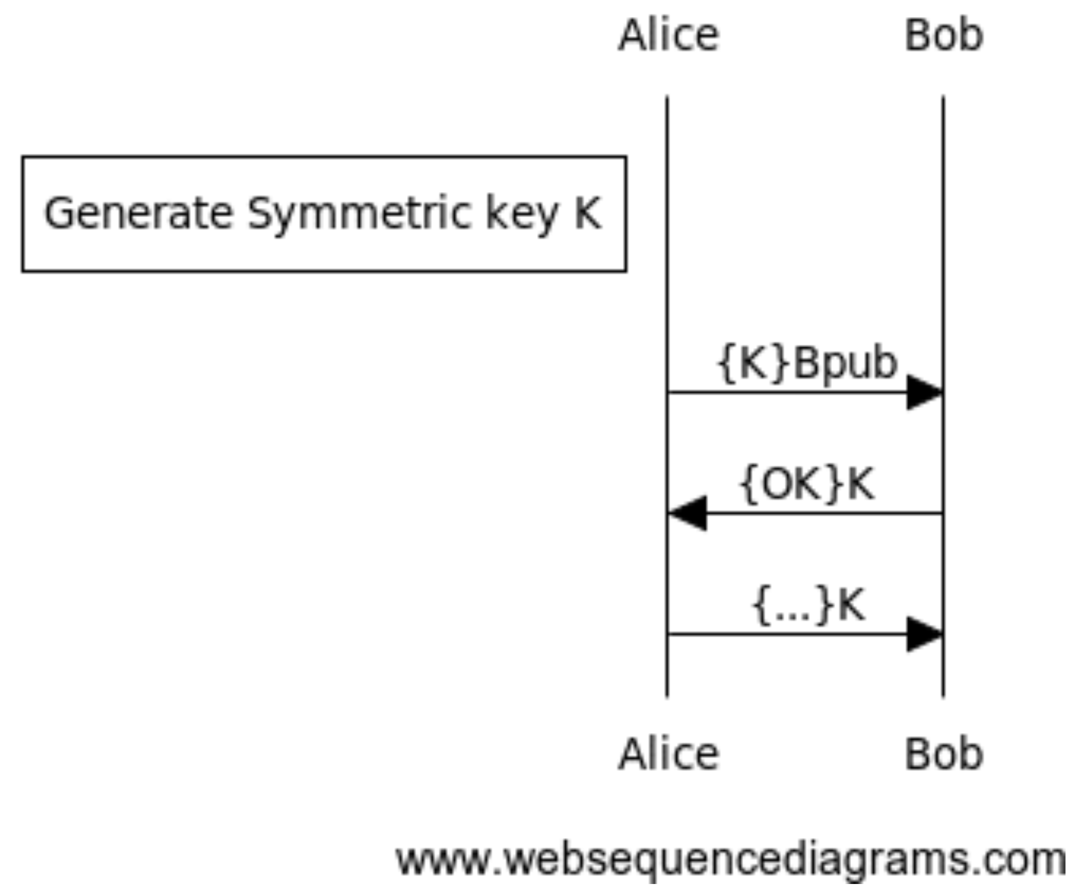
- Pair of keys K_{priv}, K_{pub}
- K_{priv} is secret, I tell it to no-one.
- K_{pub} is public, I tell it to everyone.
- Encryption: $E(K_{pub}, M) = \{M\}_{Kpub}$
- Decryption: $D(K_{priv}, \{M\}_{Kpub}) = M$
  i.e., $D(K_{priv}, E(K_{pub}, M)) = M$

Key distribution?

- Partially solves key distribution; now n parties need only n key-pairs.

Assymmetric Algorithms

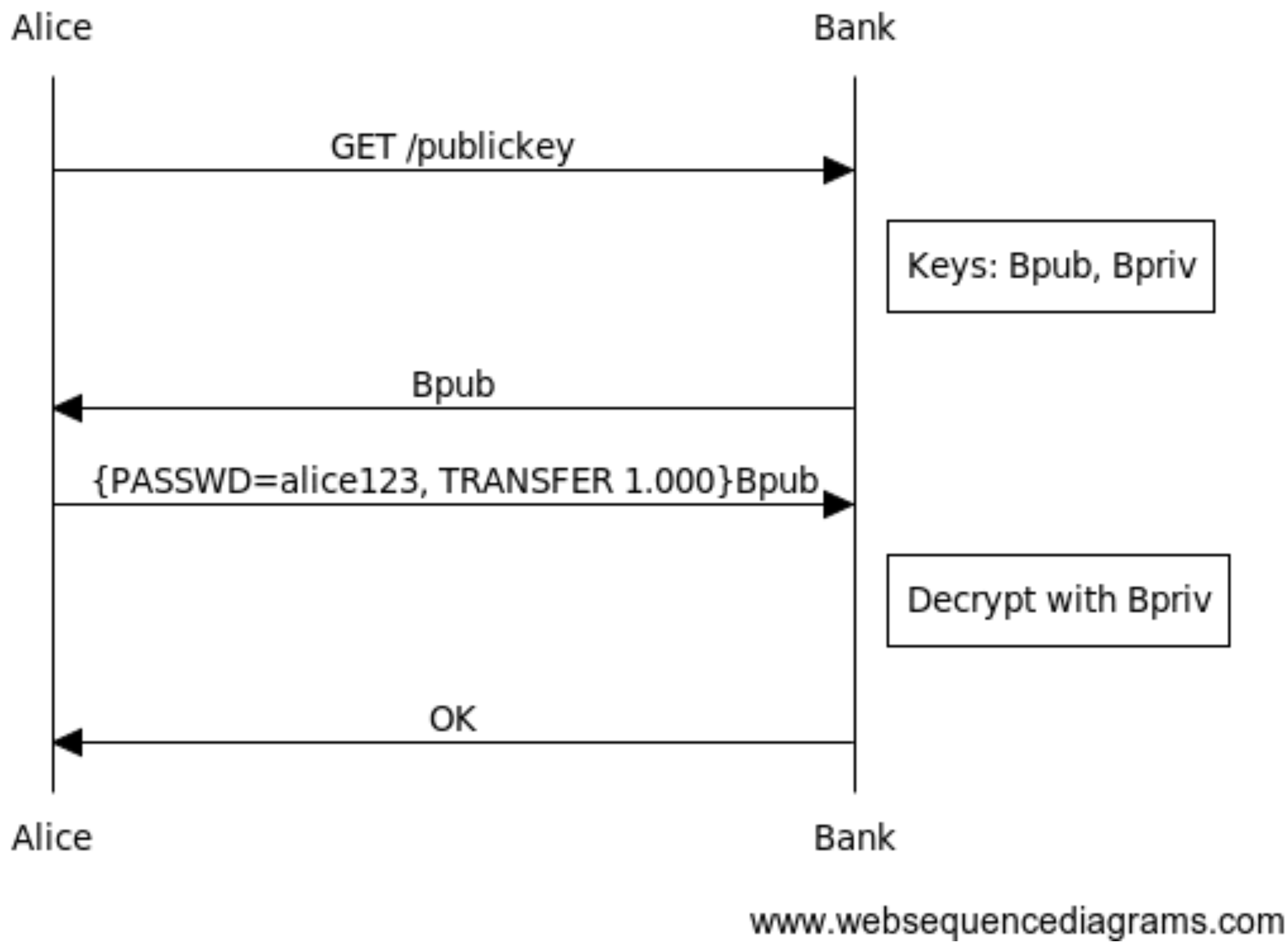
- Slow to compute in practice
- Often used for agreeing on a secret key for a symmetric algorithm.
- RSA. Considered secure for sufficiently large key sizes. (768 bit key broken in 2009 using 2000 years of computing time.)



Question

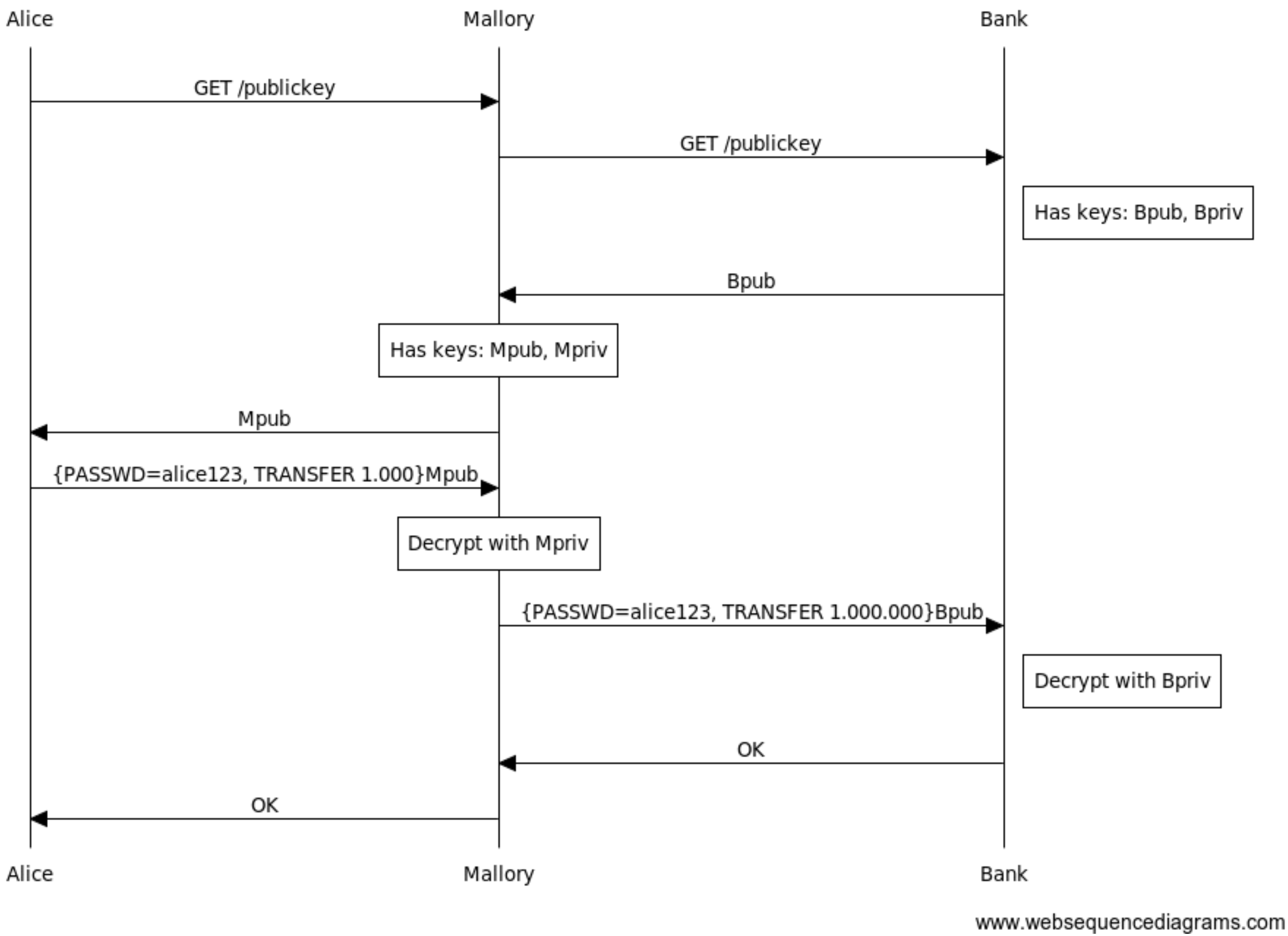
- I’m a bank; my clients net secure net-banking.
- I put my public key K_{pub} on my webpage.
- Clients should:
  - download the public key.
  - encrypt their requests with my public key and send it to me.
  - requests are now communicated securely.
- Yes? No?

That is, this?


Alice
Bank
GET /publickey
Keys: Bpub, Bpriv
Bpub
{PASSWD=alice123, TRANSFER 1.000}Bpub
Decrypt with Bpriv
OK
Alice
Bank
www.websequencediagrams.com

Man-in-the-middle attack

- No!
- If the adversary intercepts my traffice, he can replace the public key of the bank with his own.


Alice
Mallory
Bank
GET /publickey
GET /publickey
Has keys: Bpub, Bpriv
Bpub
Has keys: Mpub, Mpriv
Mpub
{PASSWD=alice123, TRANSFER 1.000}Mpub
Decrypt with Mpriv
{PASSWD=alice123, TRANSFER 1.000.000}Bpub
Decrypt with Bpriv
OK
OK
Alice
Mallory
Bank
www.websequencediagrams.com

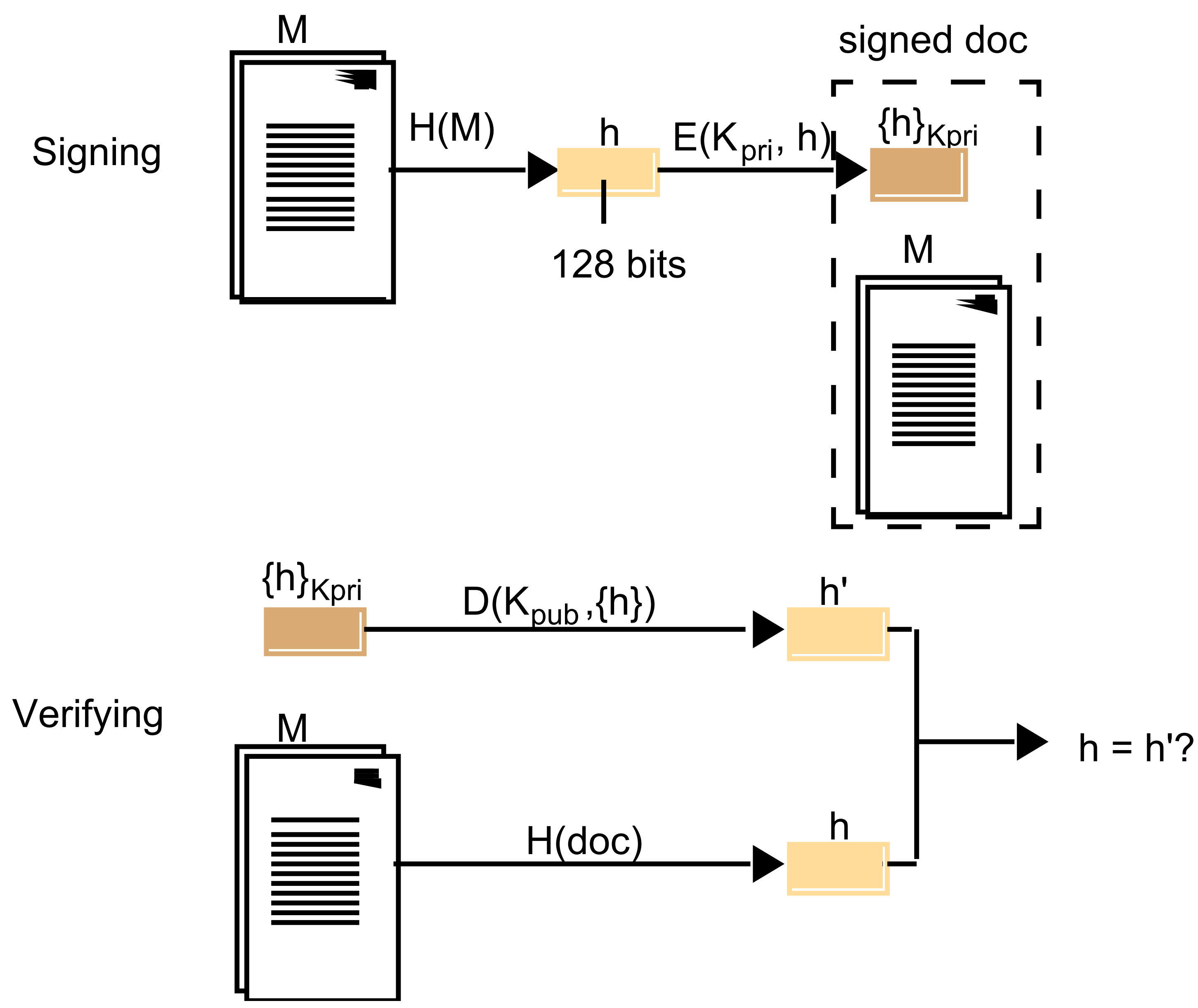
Signatures & Certificates

Signatures

- Authenticity of messages (signee, contents)
- Non-repudiability of messages

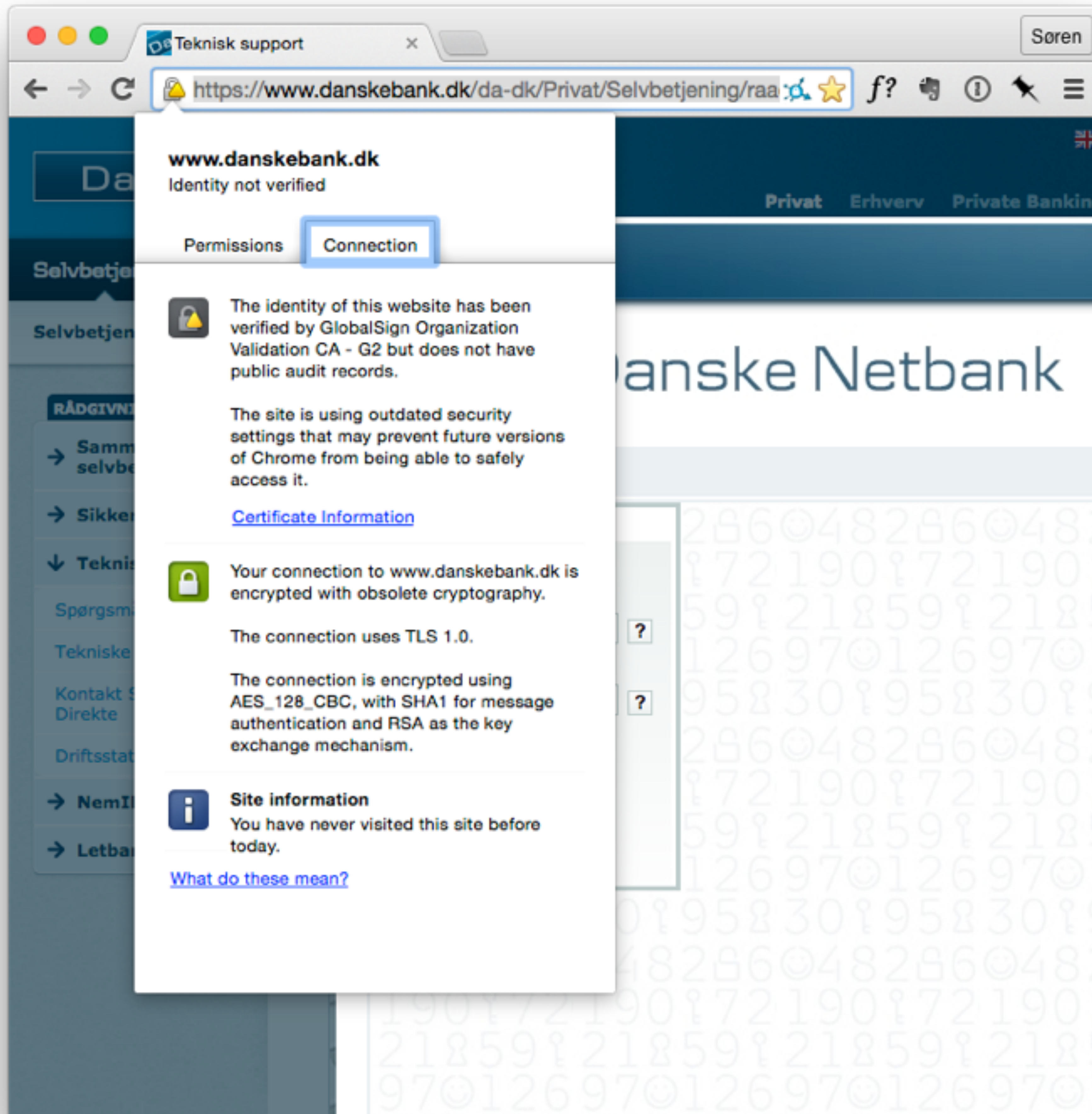
… with asymmetric scheme:

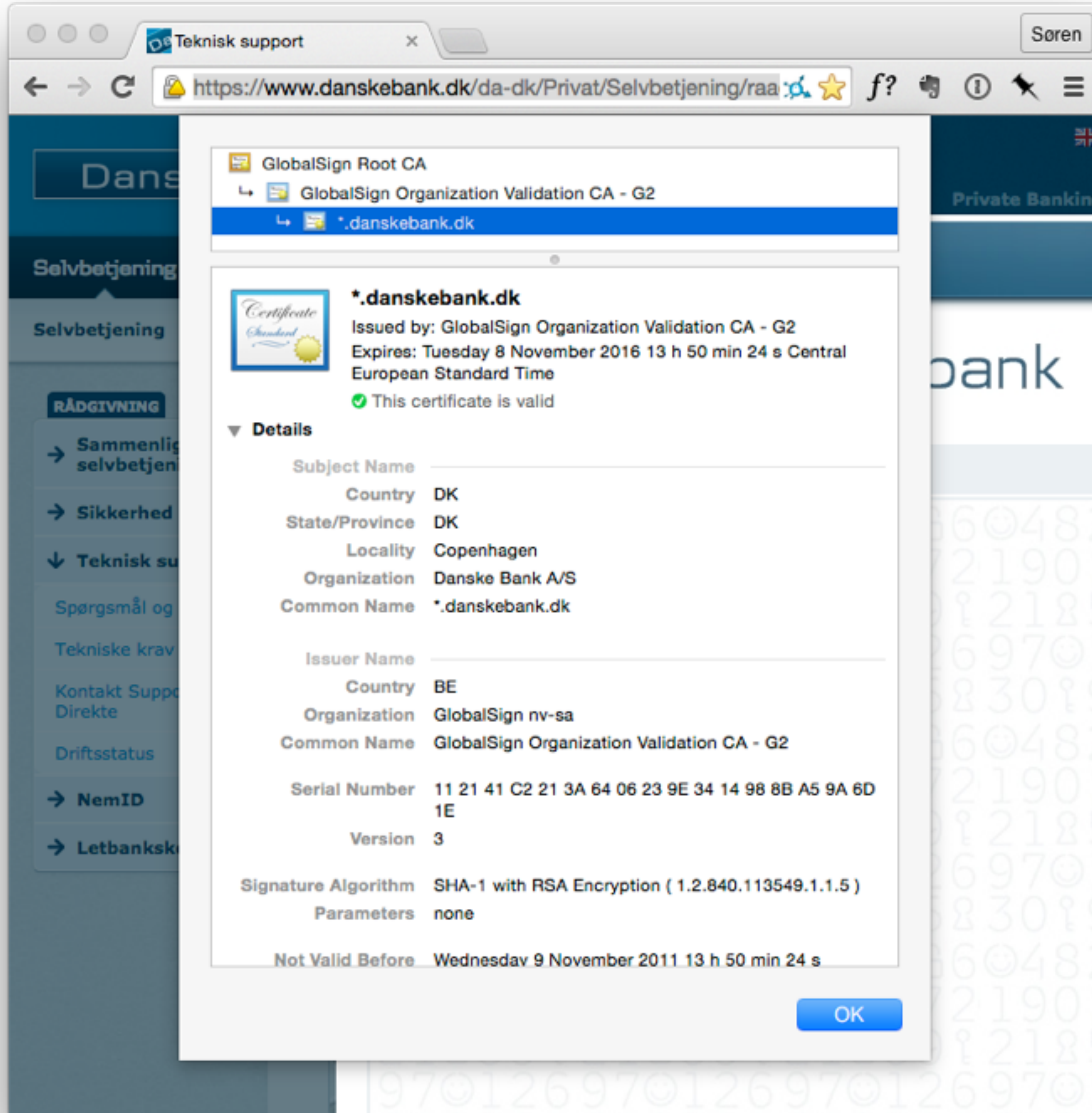
- I have keys K_{pub}, $K_{priv.}$ and a message M.
- I compute a digest (hash) H(M).
- I encrypt the hash with my *private* key S = E(K_{priv}, H(M))
- I send $[M]_K$ = M,S
- Recipient decrypts S with $K_{pub,}$ checks himself if H' = D($K_{pub,}$ S) =? H(M).
- Adversary can't tamper with M, because H' won't match H(M).



Certificates

- Signed public keys.
- I am a Certificate Authority. I have keys K_{pub}, $K_{priv.}$
- The bank “International Bank A/S” has keys B_{pub}, B_{priv}.
- I sign a message M containing B_{pub} and the words “I believe this is the public key of International Bank A/S“, producing $S = E(K_{priv}, H(M))$. This is the certificate.
- Only I have Kpriv, so only I could have made such a certificate.
- International Bank A/S presents the certificate along K_{pub}.
- Anyone who has my public key can verify that I believe K_{pub} belongs to International Bank A/S.

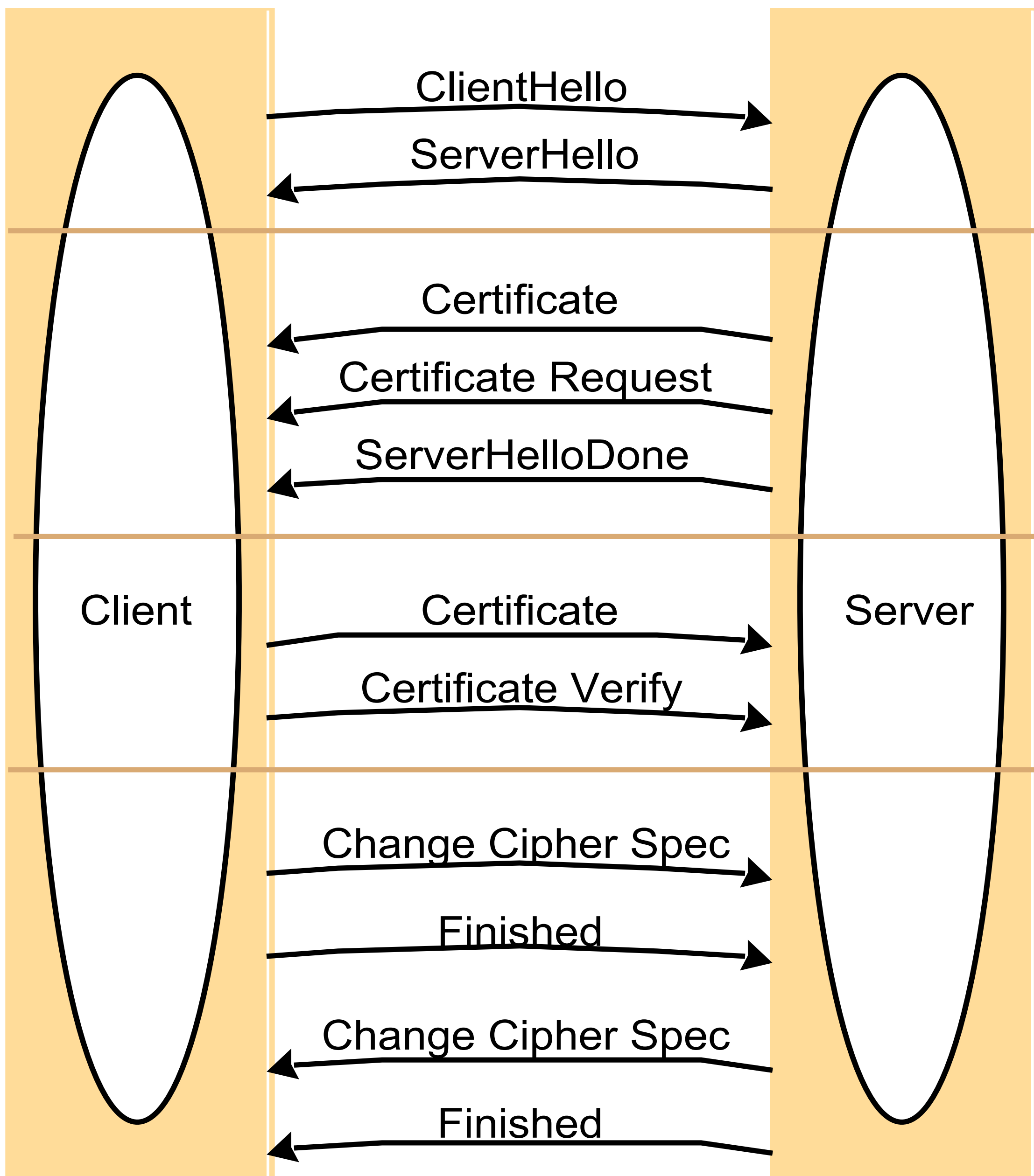

Teknisk support
Søren
https://www.danskebank.dk/da-dk/Privat/Selvbetjening/raa
Privat
Erhverv
Private Banking
www.danskebank.dk
Identity not verified
Permissions
Connection
The identity of this website has been verified by GlobalSign Organization Validation CA - G2 but does not have public audit records.
The site is using outdated security settings that may prevent future versions of Chrome from being able to safely access it.
Certificate Information
Your connection to www.danskebank.dk is encrypted with obsolete cryptography.
The connection uses TLS 1.0.
The connection is encrypted using AES_128_CBC, with SHA1 for message authentication and RSA as the key exchange mechanism.
Site information
You have never visited this site before today.
What do these mean?
anske Netbank


Teknisk support
Søren
https://www.danskebank.dk/da-dk/Privat/Selvbetjening/raa
GlobalSign Root CA
GlobalSign Organization Validation CA - G2
*.danskebank.dk
*.danskebank.dk
Issued by: GlobalSign Organization Validation CA - G2
Expires: Tuesday 8 November 2016 13 h 50 min 24 s Central European Standard Time
This certificate is valid
Details
Subject Name
Country DK
State/Province DK
Locality Copenhagen
Organization Danske Bank A/S
Common Name *.danskebank.dk
Issuer Name
Country BE
Organization GlobalSign nv-sa
Common Name GlobalSign Organization Validation CA - G2
Serial Number 11 21 41 C2 21 3A 64 06 23 9E 34 14 98 8B A5 9A 6D 1E
Version 3
Signature Algorithm SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters none
Not Valid Before Wednesday 9 November 2011 13 h 50 min 24 s
OK
Private Banking
Selvbetjening
Selvbetjening
RÅDGIVNING
Sikkerhed
Spørgsmål og
Tekniske krav
Driftsstatus
NemID

TLS

- Transport Layer Security.
- Replaces earlier SSL. (viz. Danske Bank.)
- Handshake enables
  - exchange of certificates
  - agreement on symmetric key for subsequent encrypted communication.

TLS



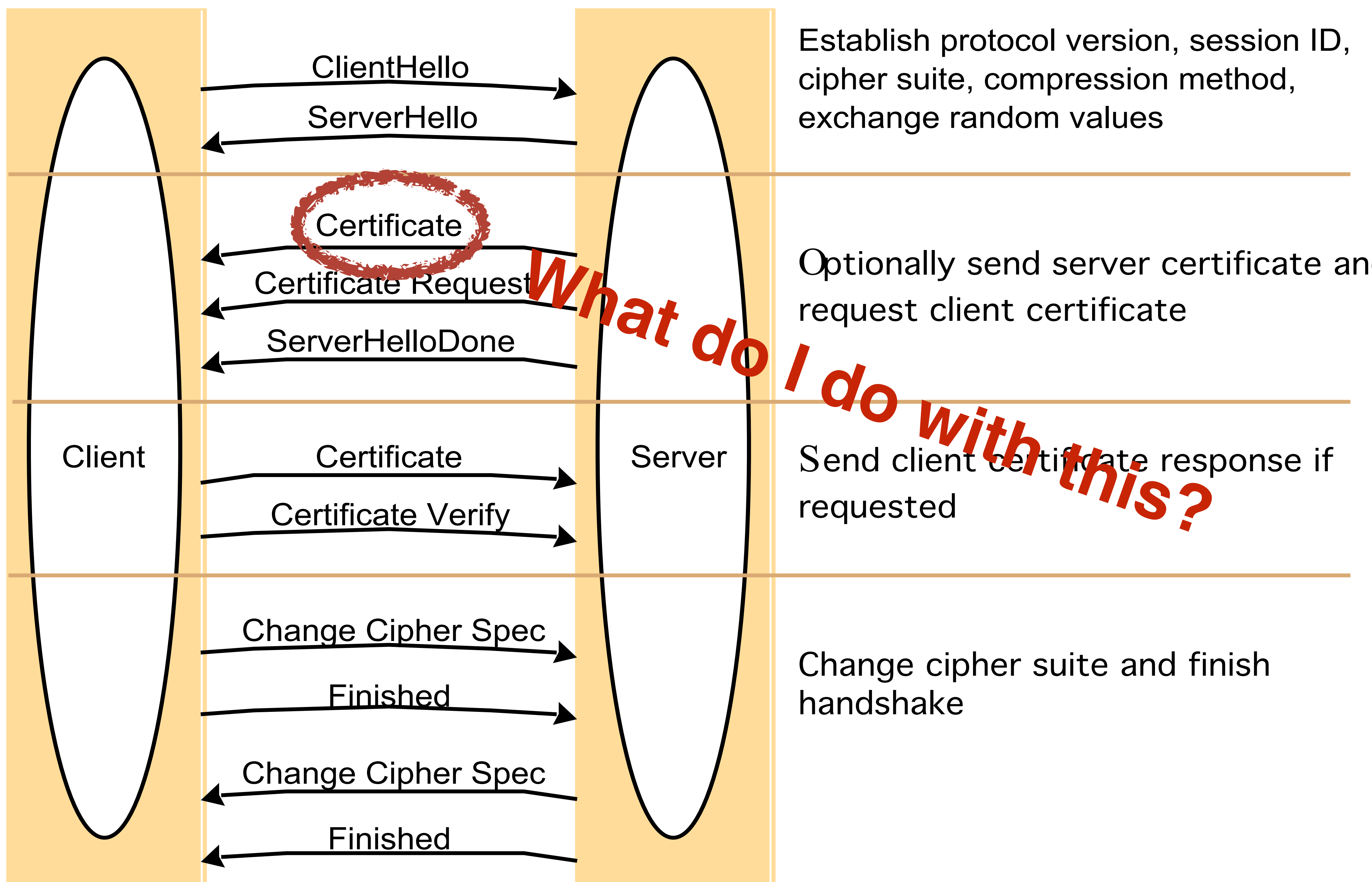
Establish protocol version, session ID, cipher suite, compression method, exchange random values

Optionally send server certificate an request client certificate

Send client certificate response if requested

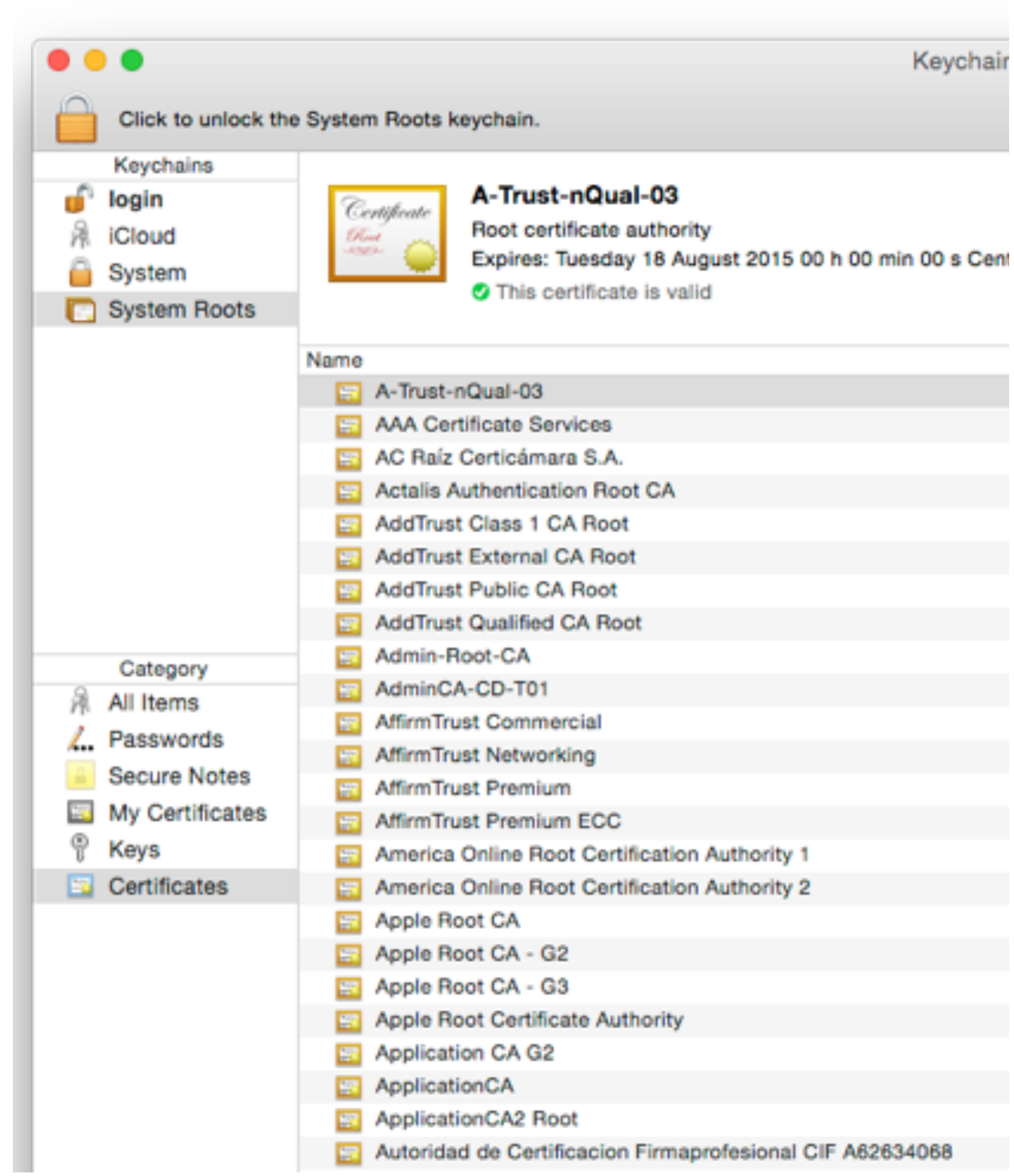
Change cipher suite and finish handshake

TLS


ClientHello
ServerHello
Certificate
Certificate Request
ServerHelloDone
Client
Server
Certificate
Certificate Verify
Change Cipher Spec
Finished
Change Cipher Spec
Finished
Establish protocol version, session ID, cipher suite, compression method, exchange random values
Optionally send server certificate and request client certificate
Send client certificate response if requested
Change cipher suite and finish handshake
What do I do with this?

Certificates and the web

- X.509 certificates
- OSes, browsers come preloaded with “root” certificates from trusted Certificate Authorities.
- Root certificates are signed by themselves and thus implicitly trusted.
- (“Here is the public key of International Bank A/S; you can trust it because I have a certificate made with the corresponding private key” doesn’t give you any connection to International Bank A/S at all.)



Certificates and the web

- X.509 certificates
- OSes, browsers come preloaded with “root” certificates.
- Root certificates are signed by themselves and thus implicitly trusted.
- (“Here is the public key of International Bank A/S; you can trust it because I have a certificate made with the corresponding private key” doesn’t give you any connection to International Bank A/S at all.)
- A certificate you receive is signed by someone.
- Hopefully that someone is someone you trust.
- So you trust the browser.

SuperFish

- Lenovo shipped machines with a self-signed root certificate from a small company called SuperFish.
- SuperFish man-in-the-middled all HTTPS traffic on the local machine in order to insert ads.
- The root-certificate was insufficiently protected; anybody can certify anything for a SuperFish compromised machine.
- Check if your Lenovo machine is affected here (bottom): http://arstechnica.com/security/2015/02/lenovo-pcs-ship-with-man-in-the-middle-adware-that-breaks-https-connections/

Summary

- Hashes
- Symmetric encryption schemes
- Asymmetric encryption schemes
- Signatures
- Certificates
- SSL/TLS

Questions?