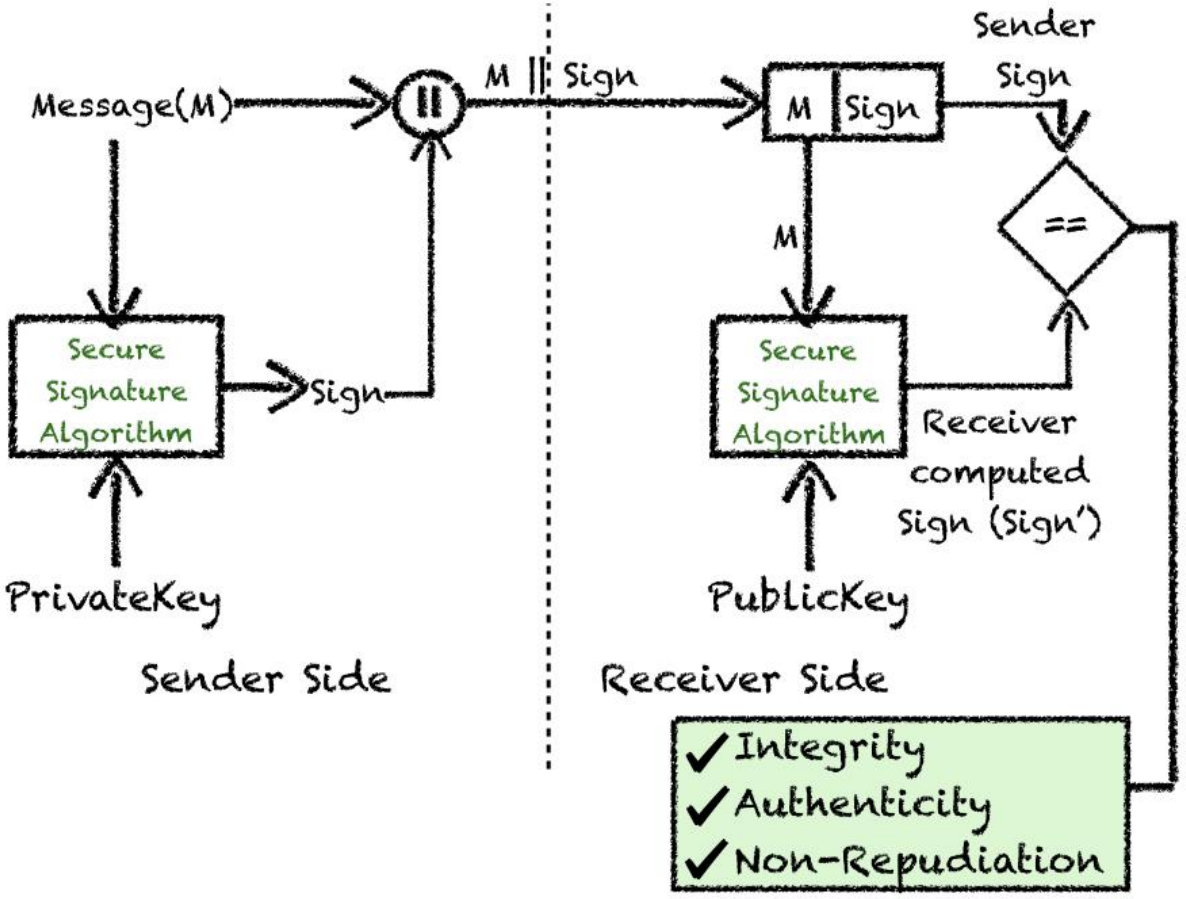
Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Kryptographie

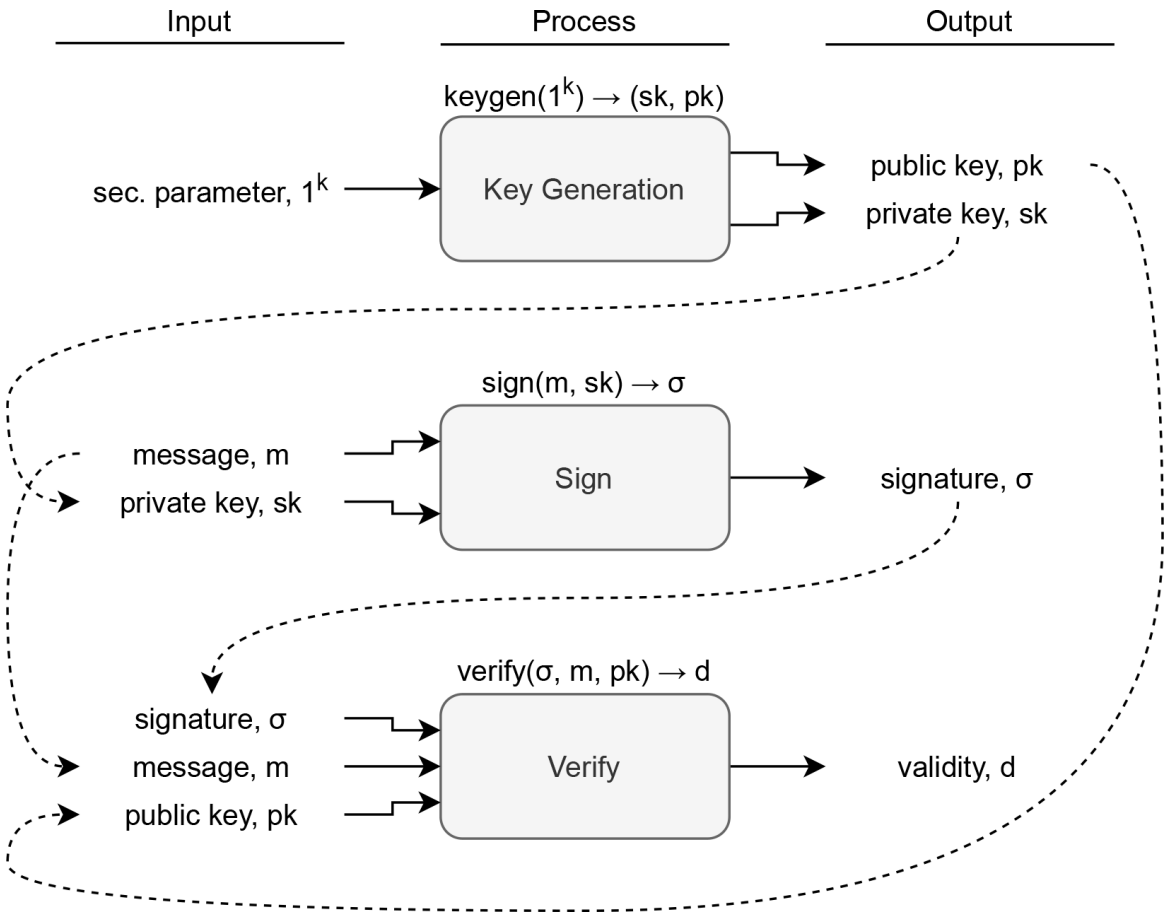
Prof. Dr. Björn Grohmann

SIGNATURES

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law



SIGNATURES


Input
Process
Output
keygen(1^k) → (sk, pk)
sec. parameter, 1^k
Key Generation
public key, pk
private key, sk
sign(m, sk) → σ
message, m
private key, sk
Sign
signature, σ
verify(σ, m, pk) → d
signature, σ
message, m
public key, pk
Verify
validity, d

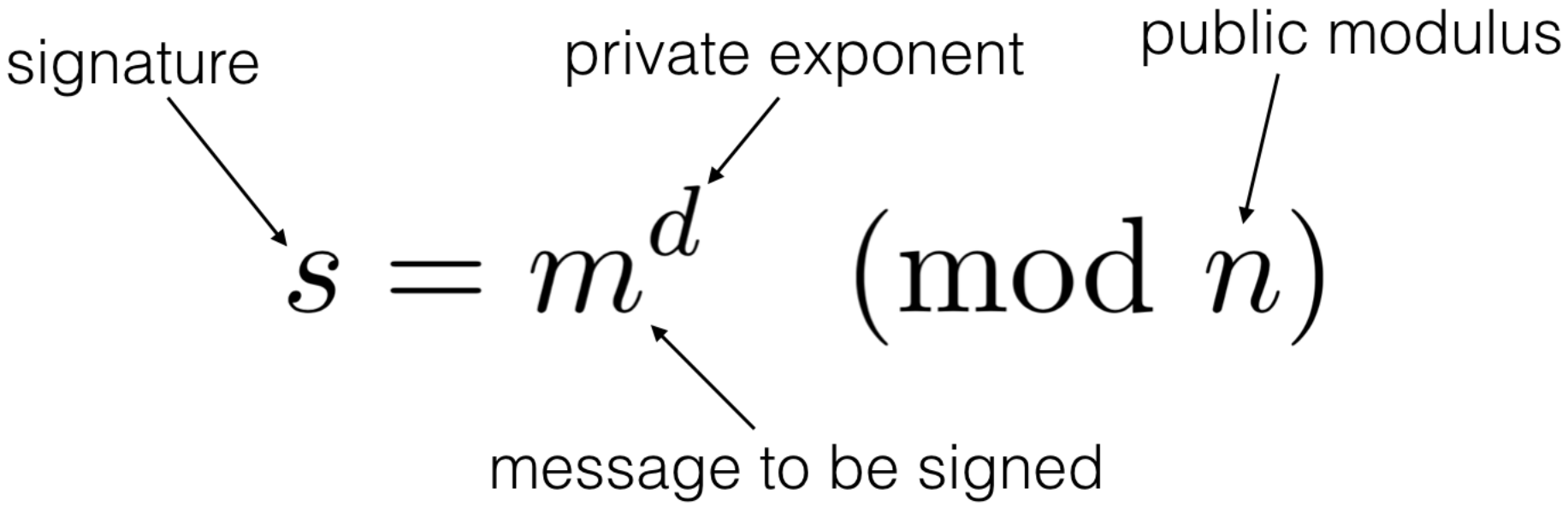
SIGNATURES

Possible Adversarial Goals

- Total break
  - Oscar can derive Alice's private signing algorithm
- Selective forgery
  - Oscar can create a valid signature on a message chosen by someone else, with some non-negligible probability
- Existential forgery
  - Oscar can create a valid signature for at least one message

BEISPIEL: RSA

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law



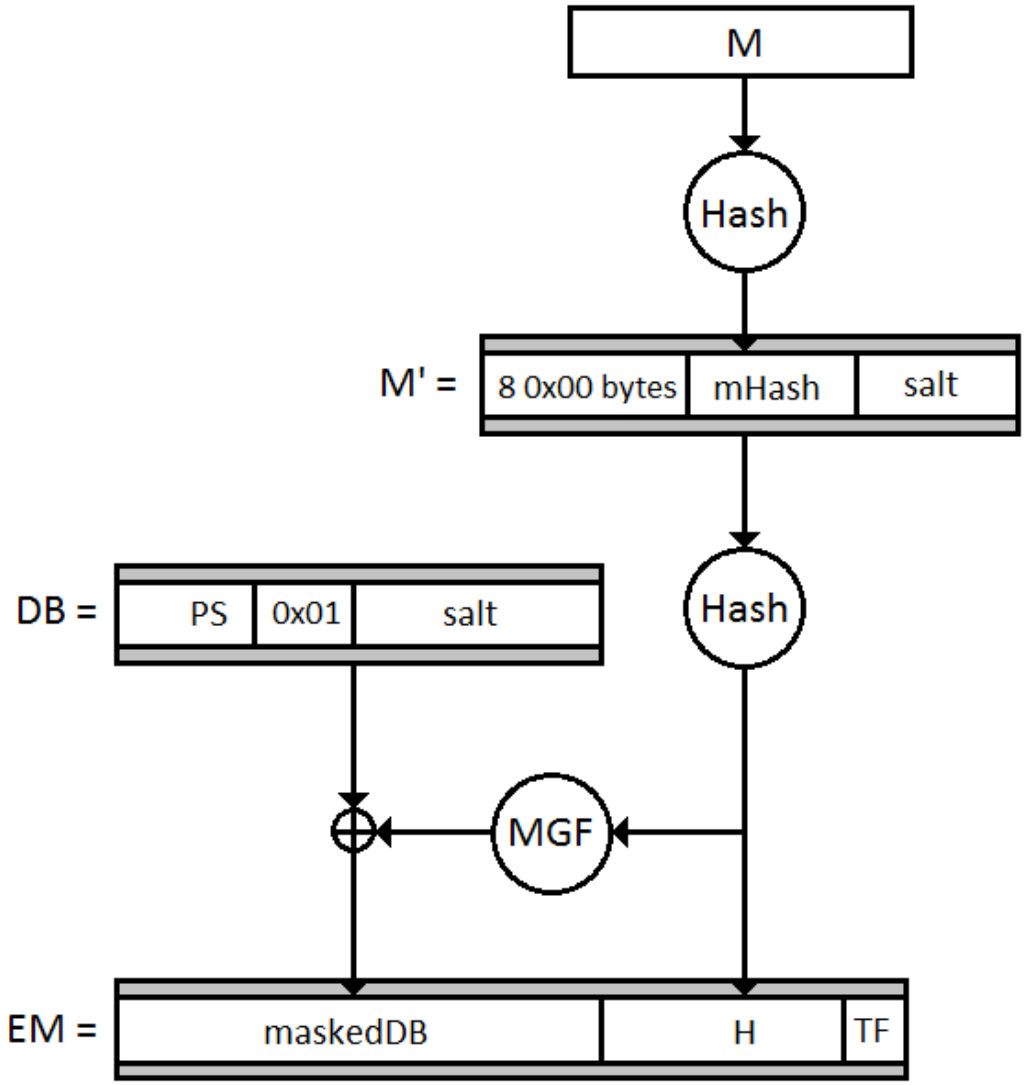
BEISPIEL: RSA

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Existential Forgery Attack Against RSA Digital Signature

Alice	Oscar	Bob
		$k_{pr} = d$ $k_{pub} = (n,e)$
	$\xleftarrow{(n,e)}$	$\xleftarrow{(n,e)}$
	1. choose signature: $s \in \mathbb{Z}_n$ 2. compute message: $x \equiv s^e \bmod n$	
	$\xleftarrow{(x,s)}$	
verification: $s^e \equiv x' \bmod n$ since $x' = x$ $\Longrightarrow$ valid signature!		

RSA-PSS


M
Hash
M' =
8 0x00 bytes
mHash
salt
DB =
PS
0x01
salt
Hash
MGF
EM =
maskedDB
H
TF

ECDSA

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Elliptic Curve Digital Signature Algorithm

For Alice to sign a message m, she follows these steps:

1. Calculate $e = \mathrm{HASH}(m)$. (Here HASH is a cryptographic hash function, such as SHA-2, with the output converted to an integer.)
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order n. (Note that z can be *greater* than n but not *longer*.[2])
3. Select a **cryptographically secure random** integer k from $[1, n-1]$.
4. Calculate the curve point $(x_1, y_1) = k \times G$. ← Generator of the EC-group
5. Calculate $r = x_1 \mod n$. If $r = 0$, go back to step 3.
6. Calculate $s = k^{-1}(z + rd_A) \mod n$. If $s = 0$, go back to step 3. ← Secret key of Alice
7. The signature is the pair (r, s). (And $(r, -s \mod n)$ is also a valid signature.)

ECDSA

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Elliptic Curve Digital Signature Algorithm

Signature verification algorithm [edit]

For Bob to authenticate Alice's signature, he must have a copy of her public-key curve point Q_A. Bob can verify Q_A is a valid curve point as follows:

1. Check that Q_A is not equal to the identity element O, and its coordinates are otherwise valid
2. Check that Q_A lies on the curve
3. Check that $n \times Q_A = O$

After that, Bob follows these steps:

1. Verify that r and s are integers in $[1, n-1]$. If not, the signature is invalid.
2. Calculate $e = \mathrm{HASH}(m)$, where HASH is the same function used in the signature generation.
3. Let z be the L_n leftmost bits of e.
4. Calculate $u_1 = zs^{-1} \mod n$ and $u_2 = rs^{-1} \mod n$.
5. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$. If $(x_1, y_1) = O$ then the signature is invalid.
6. The signature is valid if $r \equiv x_1 \pmod n$, invalid otherwise.

Public Key of Alice

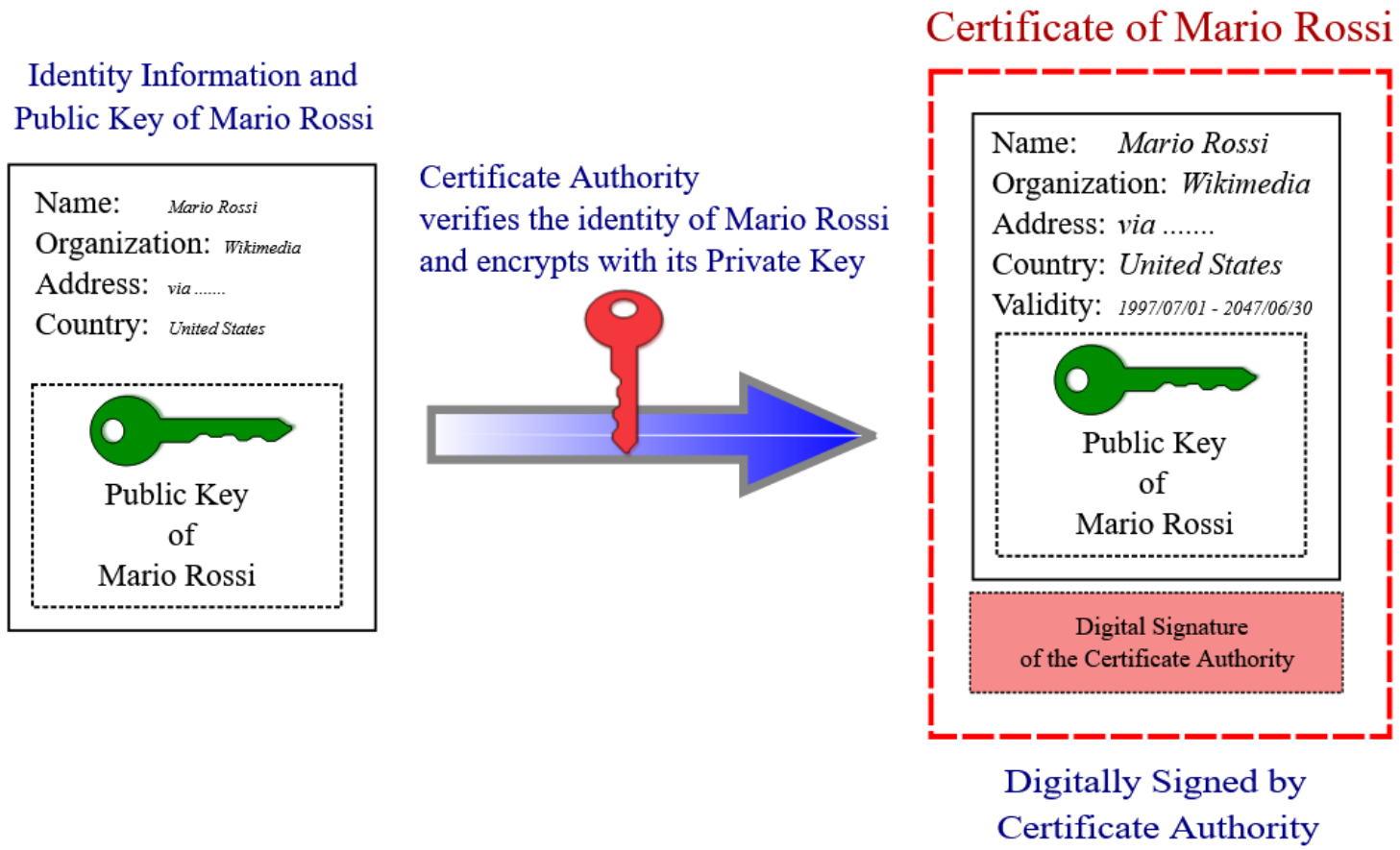
CERTIFICATES (ZERTIFIKATE)

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Wie kann Alice sich eigentlich sicher sein, dass der "Public Key" auch wirklich zu Bob gehört?

CERTIFICATES

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law



CERTIFICATES

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Network Working Group
Request for Comments: 5280
Obsoletes: 3280, 4325, 4630
Category: Standards Track

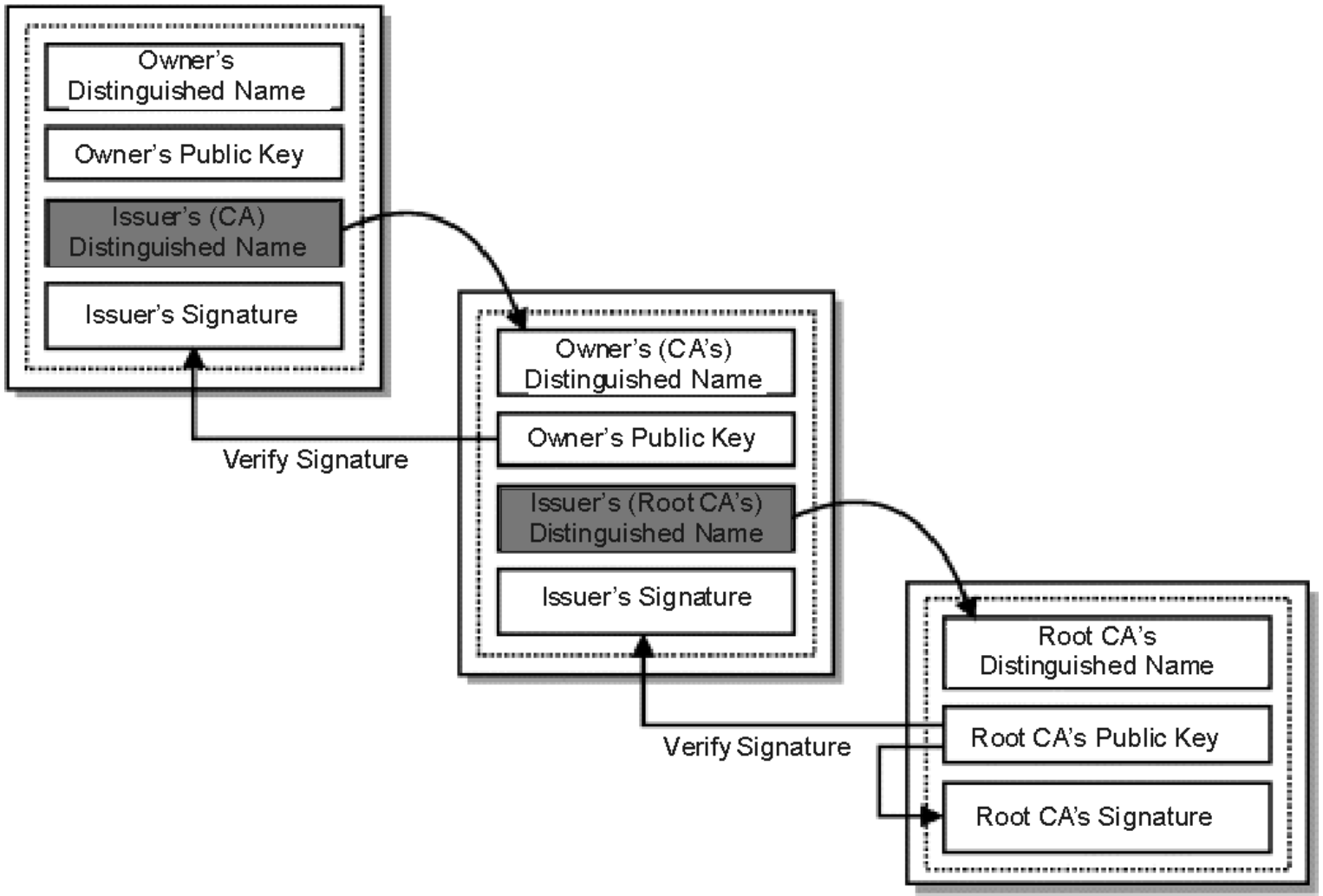
D. Cooper
NIST
S. Santesson
Microsoft
S. Farrell
Trinity College Dublin
S. Boeyen
Entrust
R. Housley
Vigil Security
W. Polk
NIST
May 2008

Internet X.509 Public Key Infrastructure Certificate
and Certificate Revocation List (CRL) Profile

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signature            BIT STRING  }


TBSCertificate  ::=  SEQUENCE  {
     version         [0]  Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  Extensions OPTIONAL
                          -- If present, version MUST be v3 --  }
```

CERTIFICATES



CERTIFICATES

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Reasons for revocation [edit]

Reasons to revoke a certificate according to RFC 5280[6] are:

- `unspecified` (0)
- `keyCompromise` (1)
- `cACompromise` (2)
- `affiliationChanged` (3)
- `superseded` (4)
- `cessationOfOperation` (5)
- `certificateHold` (6)
- `removeFromCRL` (8)
- `privilegeWithdrawn` (9)
- `aACompromise` (10)

Note that value 7 is not used.

BEISPIEL: IDENTITY BASED ENCRYPTION

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Appears in SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615, 2003. An extended abstract of this paper appears in the Proceedings of Crypto 2001, volume 2139 of Lecture Notes in Computer Science, pages 213–229, Springer-Verlag, 2001.

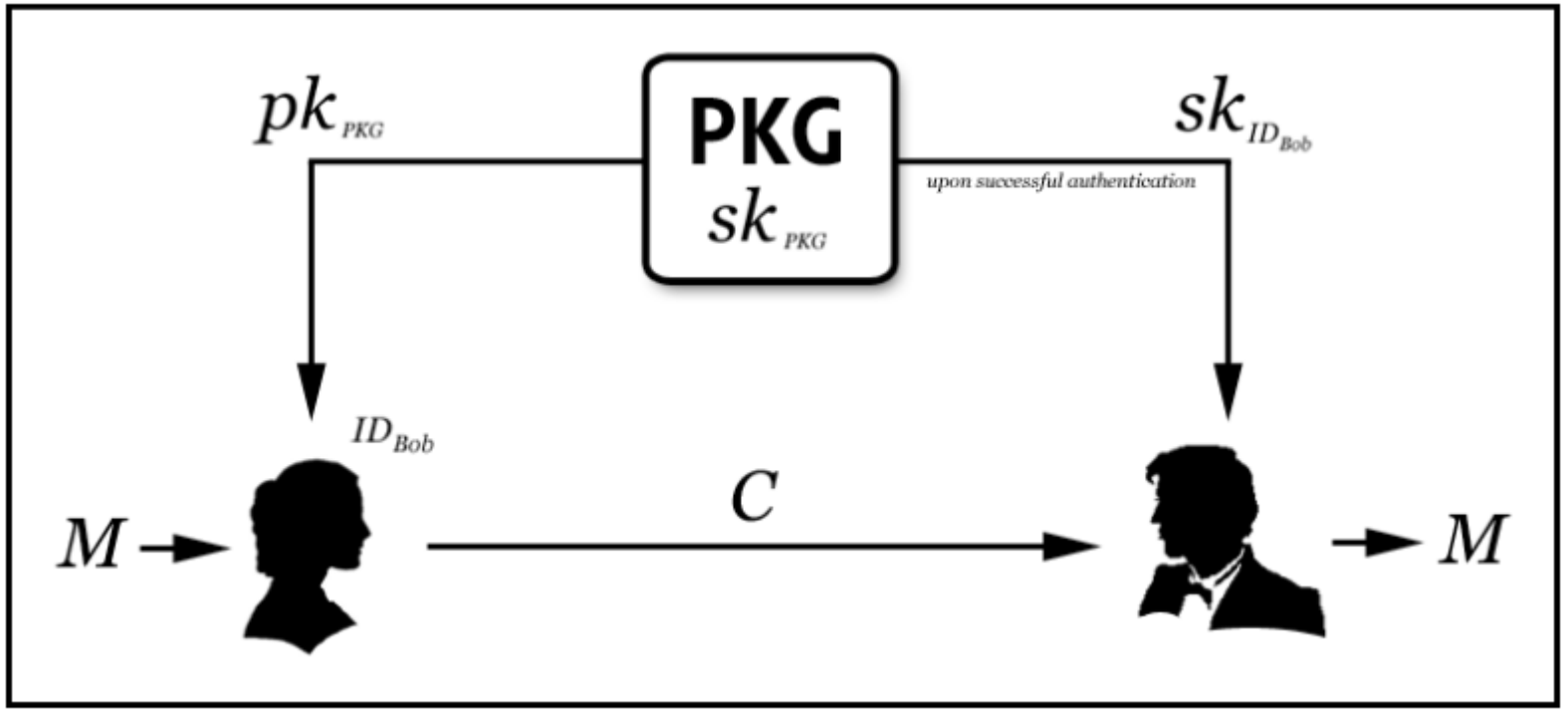
Identity-Based Encryption from the Weil Pairing

Dan Boneh*
dabo@cs.stanford.edu

Matthew Franklin†
franklin@cs.ucdavis.edu

BEISPIEL: IDENTITY BASED ENCRYPTION

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law



Alice encrypting a message for Bob using Identity-Based Encryption

BEISPIEL: IDENTITY BASED ENCRYPTION

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q for some large prime q. Our IBE system makes use of a *bilinear* map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ between these two groups. The map must satisfy the following properties:

1. Bilinear: We say that a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is *bilinear* if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$. Observe that since $\mathbb{G}_1, \mathbb{G}_2$ are groups of prime order this implies that if P is a generator of $\mathbb{G}_1$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

BEISPIEL: IDENTITY BASED ENCRYPTION

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Setup: Given a security parameter $k \in \mathbb{Z}^+$, the algorithm works as follows:

Step 1: Run $\mathcal{G}$ on input k to generate a prime q, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order q, and an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Choose a random generator $P \in \mathbb{G}_1$.

Step 2: Pick a random $s \in \mathbb{Z}_q^*$ and set $P_{pub} = sP$.

Step 3: Choose a cryptographic hash function $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$. Choose a cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n. The security analysis will view H_1, H_2 as random oracles.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n$. The system parameters are $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$. The master-key is $s \in \mathbb{Z}_q^*$.

BEISPIEL: IDENTITY BASED ENCRYPTION

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Extract: For a given string $\mathsf{ID} \in \{0,1\}^*$ the algorithm does: (1) computes $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$, and (2) sets the private key d_{ID} to be $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ where s is the master key.

Encrypt: To encrypt $M \in \mathcal{M}$ under the public key ID do the following: (1) compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$, (2) choose a random $r \in \mathbb{Z}_q^*$, and (3) set the ciphertext to be

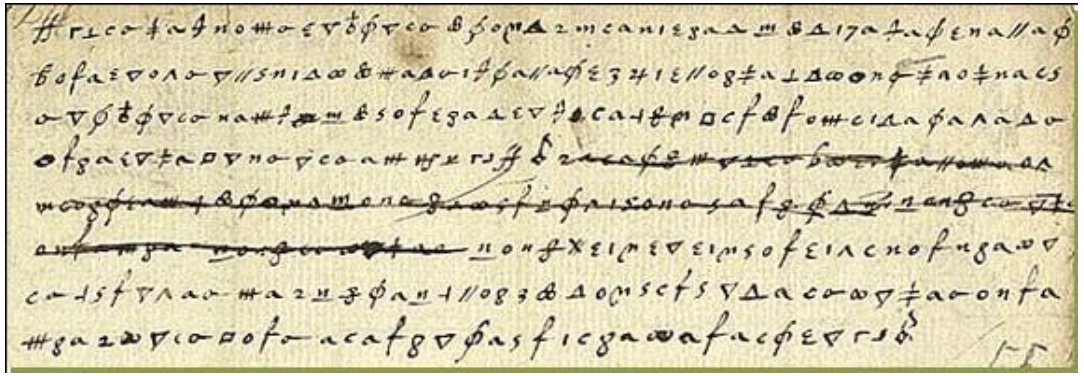
$$C = \langle rP,\ M \oplus H_2(g_{\mathsf{ID}}^r) \rangle \quad \text{where} \quad g_{\mathsf{ID}} = \hat{e}(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*$$

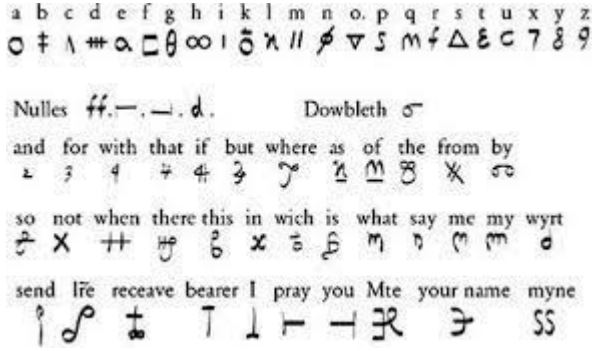
Decrypt: Let $C = \langle U, V \rangle \in \mathcal{C}$ be a ciphertext encrypted using the public key ID. To decrypt C using the private key $d_{\mathsf{ID}} \in \mathbb{G}_1^*$ compute:

$$V \oplus H_2(\hat{e}(d_{\mathsf{ID}}, U)) = M$$

Das funktioniert, da
e(d, U) = e(sQ, rP) = e(Q, sP)^r

THE PLOT (1586)





Anthony Babington

Thomas Phelippes

Maria Stuart

Elizabeth I

Sir Francis Walsingham

Gilbert Gifford

Beer barrel

SCHICHTENMODELL

Conspiracy layer

Carrier layer

Beer barrel

layer

SECURITY BEER BARREL LAYER

Conspiracy layer

Carrier layer

Beer barrel

layer

SECURITY BEER BARREL LAYER

SECURITY CARRIER LAYER

Conspiracy layer

Carrier layer

Beer barrel

layer

SECURITY CONSPIRACY LAYER

Conspiracy layer

Carrier layer

Beer barrel

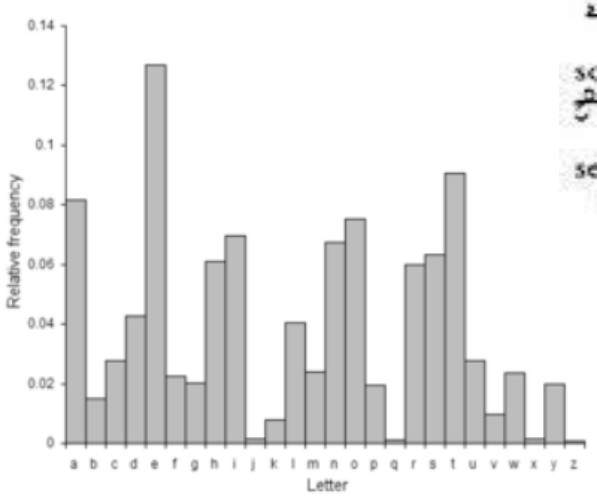
layer

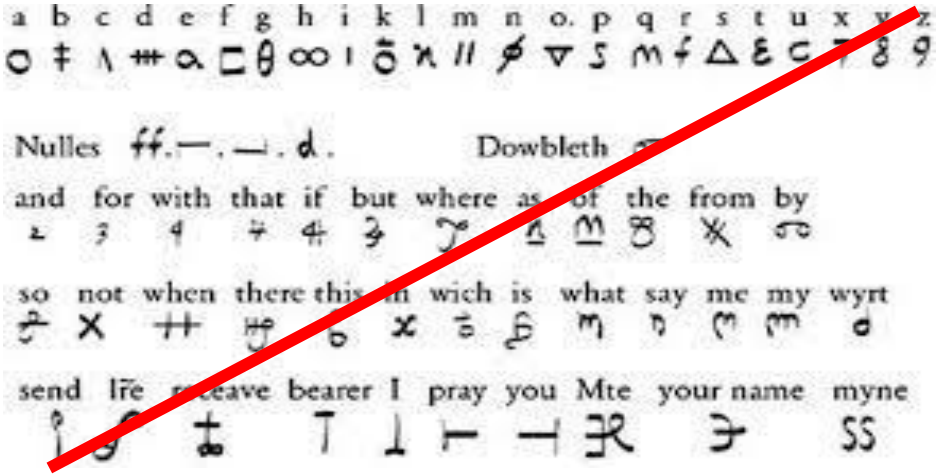
SECURITY CONSPIRACY LAYER

Encryption

Substitution cipher

E	11.1607%	56.88	M	3.0129%	15.36
A	8.4966%	43.31	H	3.0034%	15.31
R	7.5809%	38.64	G	2.4705%	12.59
I	7.5448%	38.45	B	2.0720%	10.56
O	7.1635%	36.51	F	1.8121%	9.24
T	6.9509%	35.43	Y	1.7779%	9.06
N	6.6544%	33.92	W	1.2899%	6.57
S	5.7351%	29.23	K	1.1016%	5.61
L	5.4893%	27.98	V	1.0074%	5.13
C	4.5388%	23.13	X	0.2902%	1.48
U	3.6308%	18.51	Z	0.2722%	1.39
D	3.3844%	17.25	J	0.1965%	1.00
P	3.1671%	16.14	Q	0.1962%	(1)

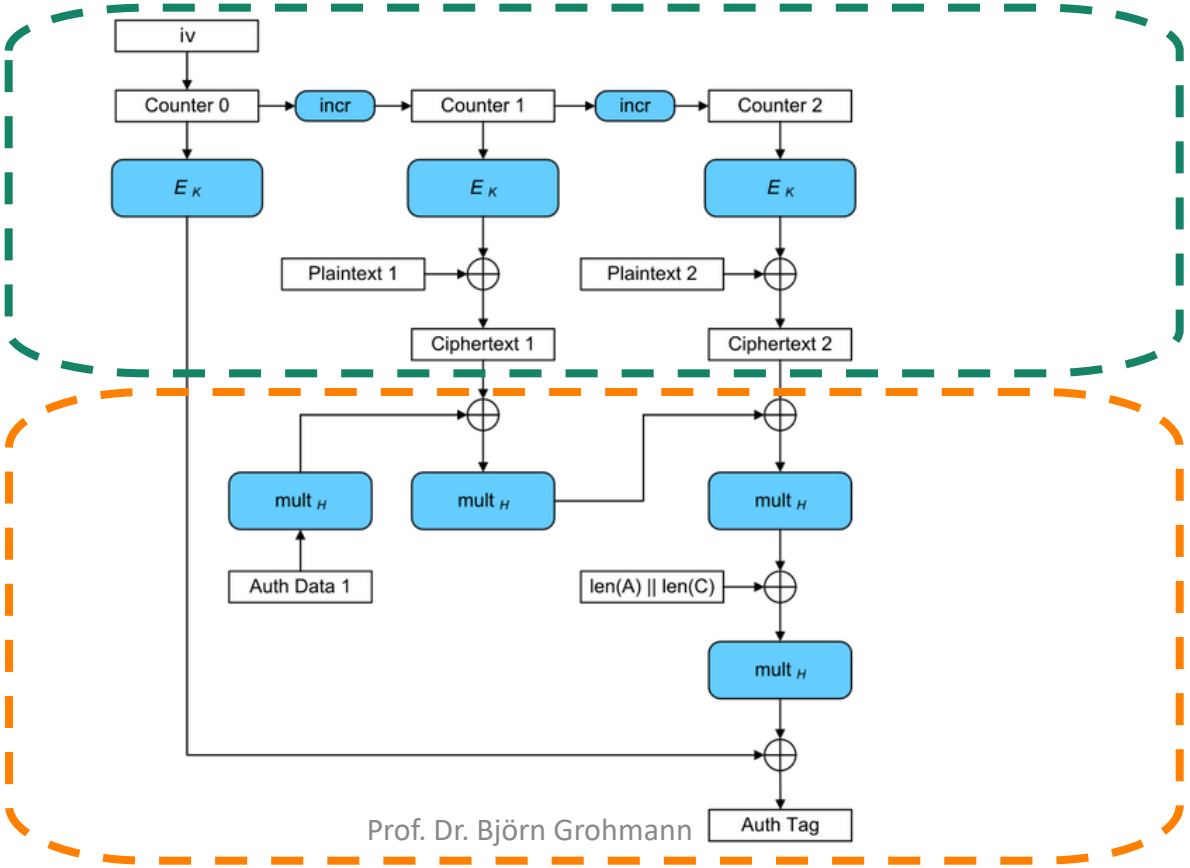




SECURITY CONSPIRACY LAYER

Authenticated Encryption

AES-Ctr-Mode

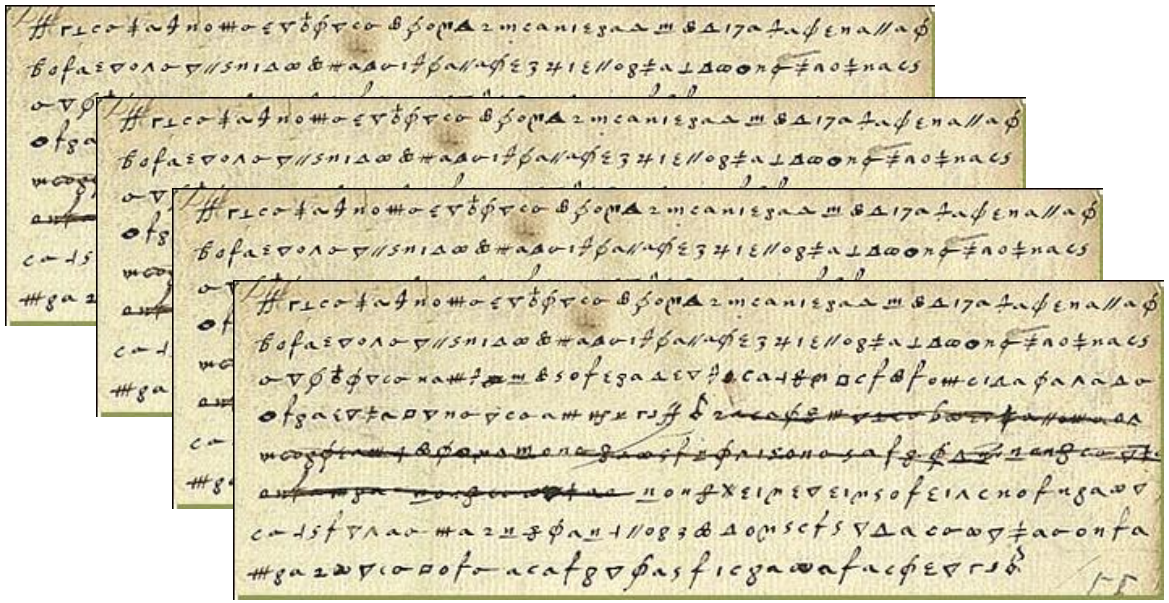


GMAC

SECURITY CONSPIRACY LAYER

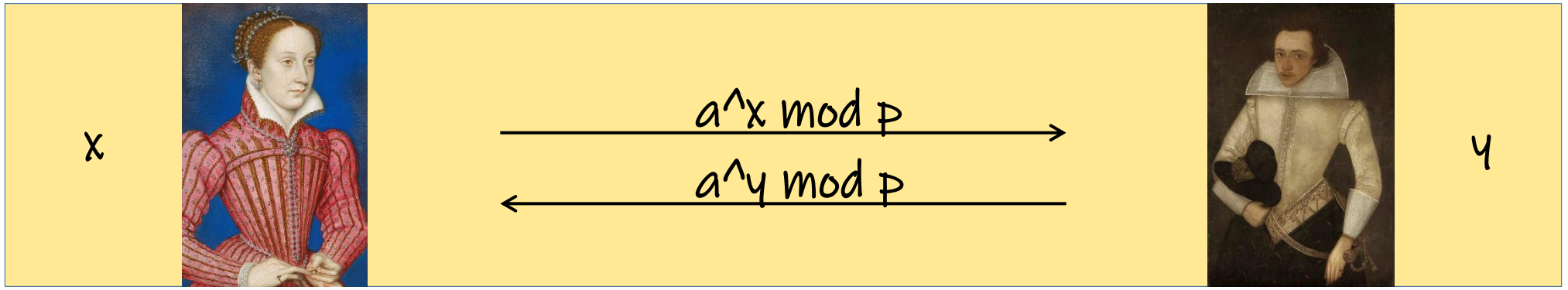
Encryption Keys

a b c d e f g h i k l m n o. p q r s t u x y z



Pre-shared Key

SECURITY CONSPIRACY LAYER

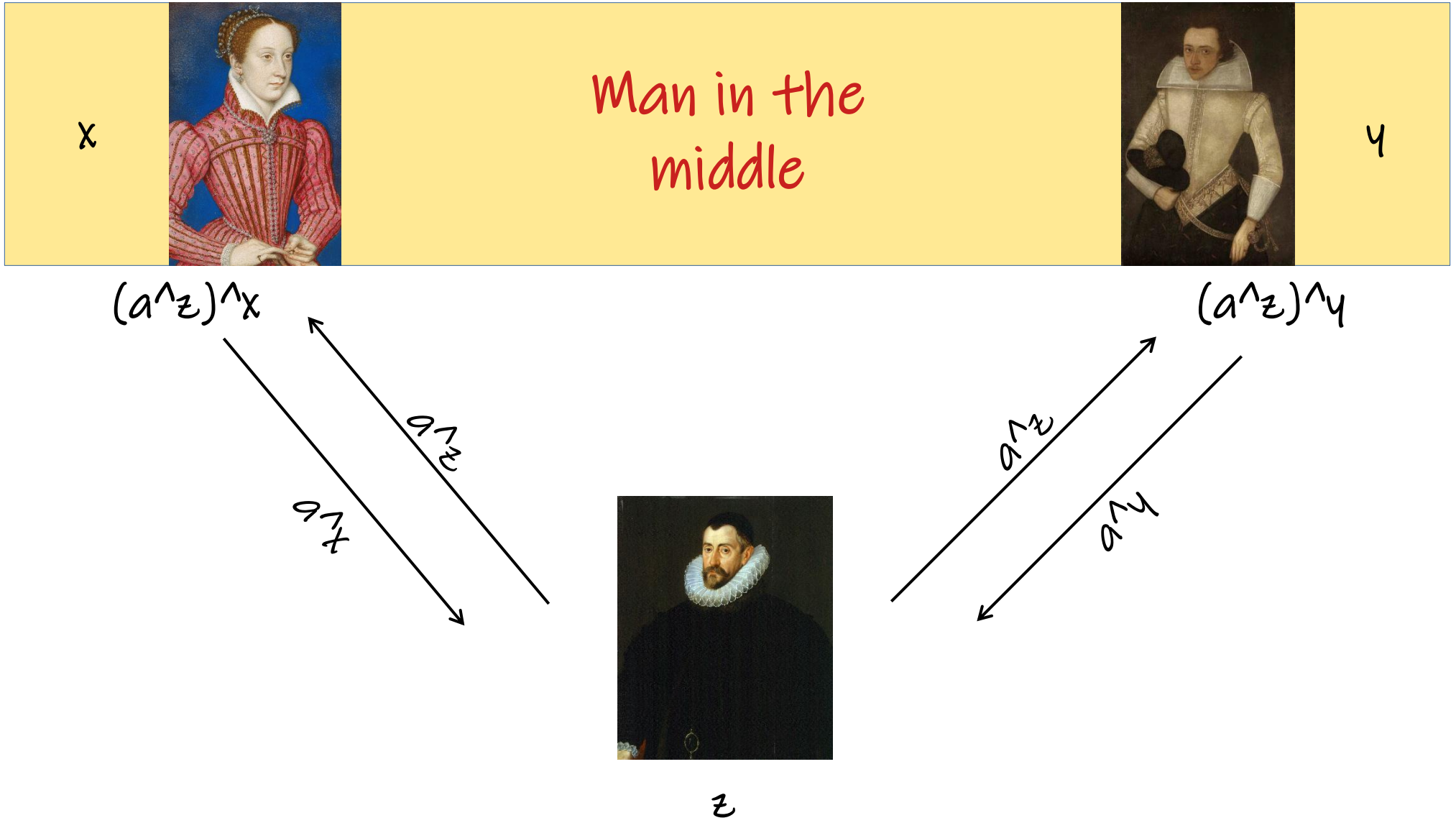


Whitfield Diffie is a graduate student doing research in cryptography in the Department of Electrical Engineering at Stanford University. He previously worked for the Mitre Corporation and at the Artificial Intelligence Laboratories of both MIT and Stanford, doing research in symbolic mathematical manipulation and the theory of program correctness. He holds a bachelor of science degree in mathematics from MIT, from which he graduated in 1965.

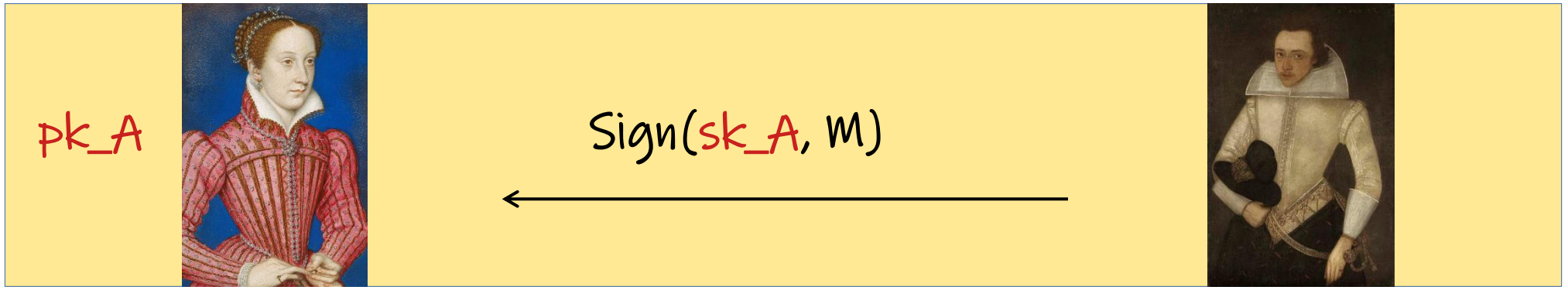
Martin E. Hellman, an associate professor of electrical engineering at Stanford University, is doing research in cryptography and communication theory. He was an assistant professor at MIT from 1969-1971 and on the staff of IBM's Watson Research Center during 1968-1969. He received the BE degree from NYU in 1966 and the MS and PhD degrees from Stanford in 1967 and 1969, all in electrical engineering.

Dr. Hellman is a member of the IEEE Information Theory Group's Board of Governors and was an associate editor of the *Transactions on Communications.* He is the author of over 25 articles.

SECURITY CONSPIRACY LAYER



SECURITY CONSPIRACY LAYER



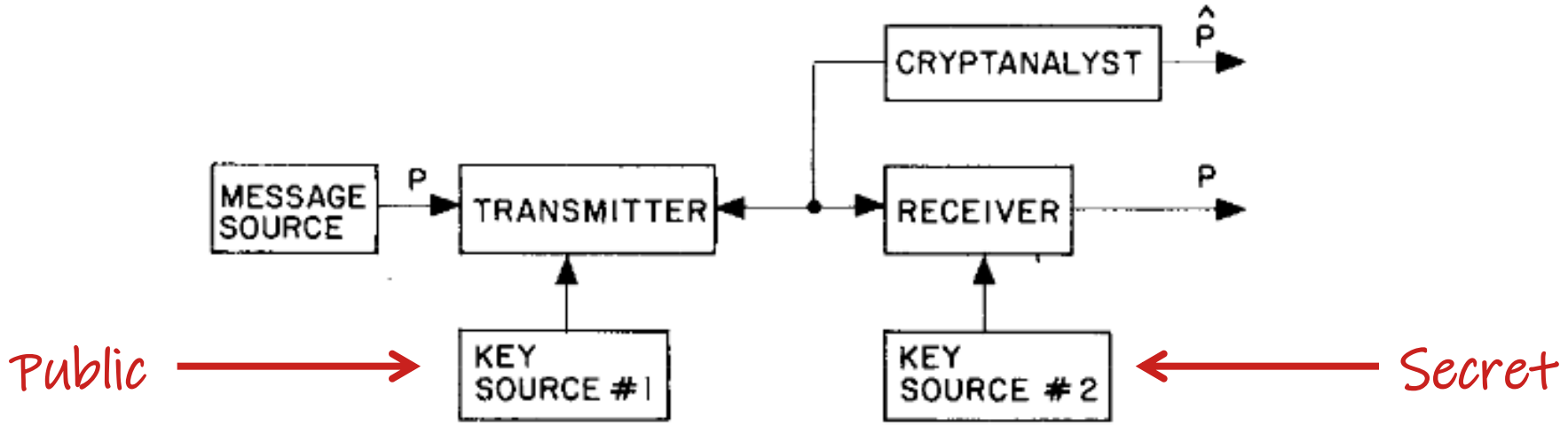


Fig. 2. Flow of information in public key system.

SECURITY CONSPIRACY LAYER

```
TBSCertificate  ::=  SEQUENCE  {
     version         [0]  Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                          -- If present, version MUST be v2 or v3
     extensions      [3]  Extensions OPTIONAL
                          -- If present, version MUST be v3 --  }
```

TRANSPORT LAYER SECURITY

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Internet Engineering Task Force (IETF)
Request for Comments: 8446
Obsoletes: 5077, 5246, 6961
Updates: 5705, 6066
Category: Standards Track
ISSN: 2070-1721

E. Rescorla
Mozilla

August 2018

The Transport Layer Security (TLS) Protocol Version 1.3

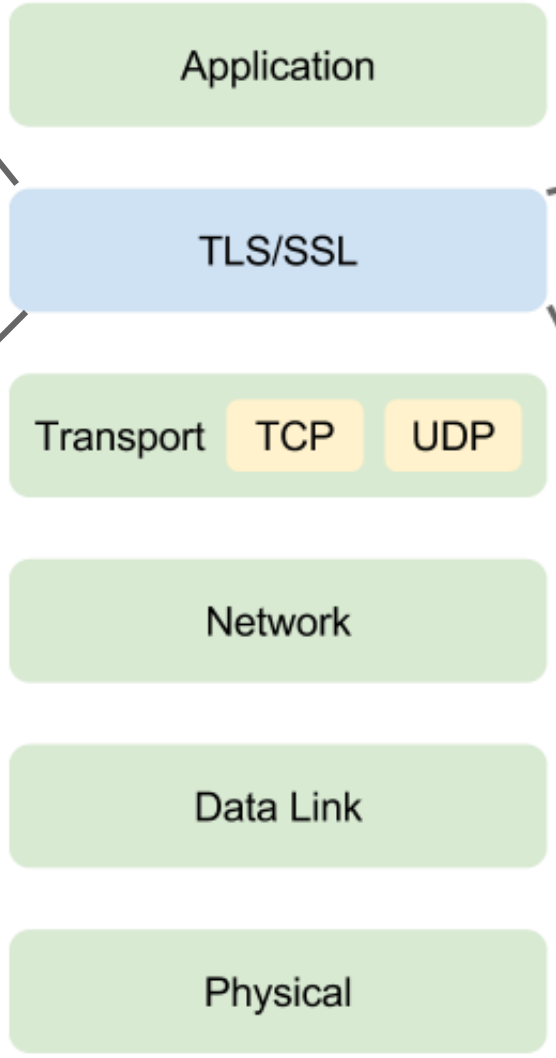
Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

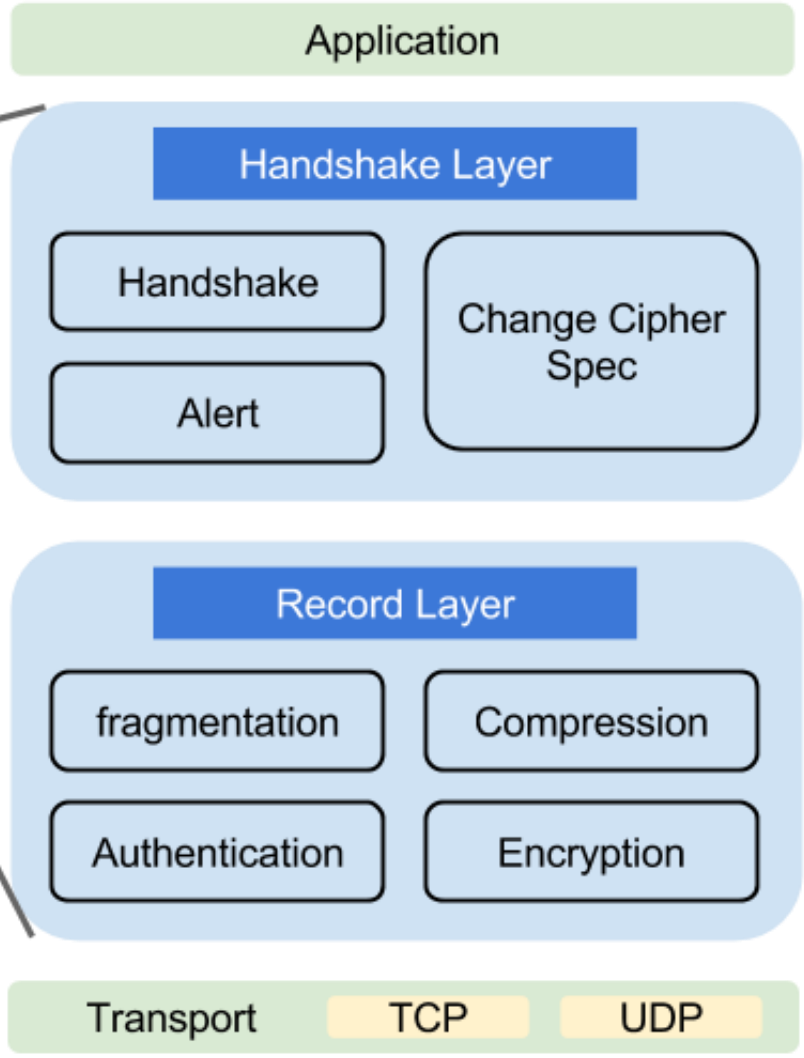
The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:

- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).
- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.

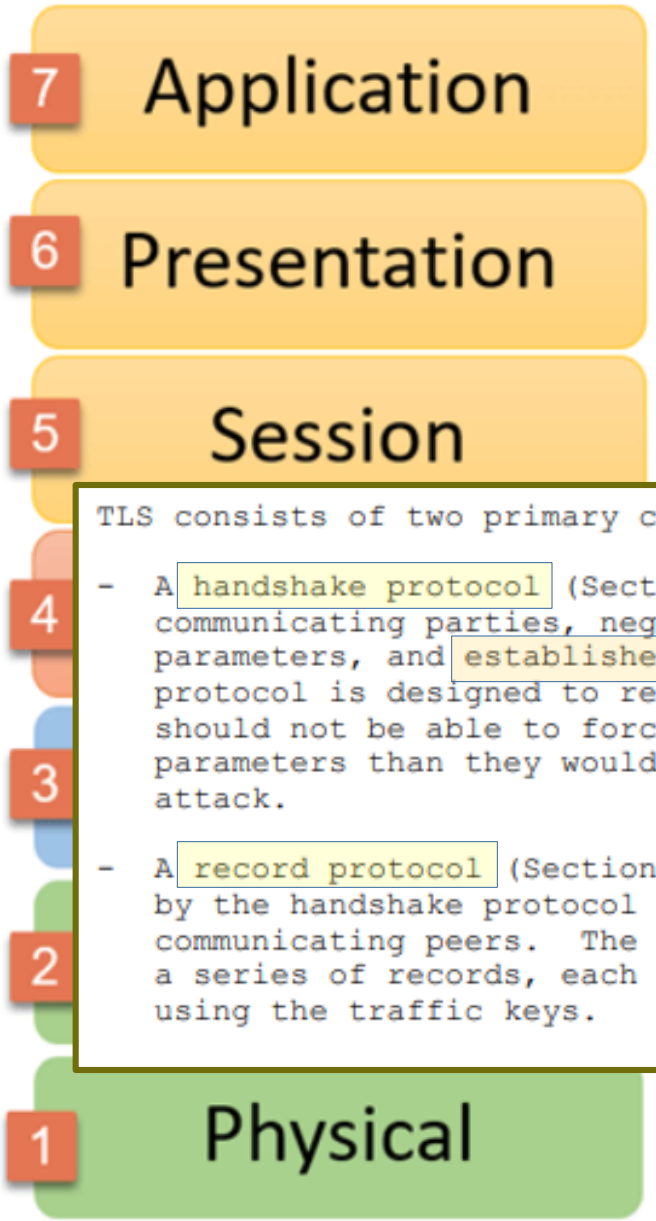
OSI Reference Model



Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law



OSI Reference Model



Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Application

TLS/SSL

Physical

TLS consists of two primary components:

- A handshake protocol (Section 4) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering; an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

- A record protocol (Section 5) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

Application

Handshake Layer

Handshake

Alert

Change Cipher Spec

Only for compatibility reasons in v1.3

Record Layer

fragmentation

Compression

Removed in v1.3

Authentication

Encryption

Transport TCP UDP

Description	Value
TLS_AES_128_GCM_SHA256	{0x13,0x01}
TLS_AES_256_GCM_SHA384	{0x13,0x02}
TLS_CHACHA20_POLY1305_SHA256	{0x13,0x03}
TLS_AES_128_CCM_SHA256	{0x13,0x04}
TLS_AES_128_CCM_8_SHA256	{0x13,0x05}

TLS supports three basic key exchange modes:

- (EC)DHE (Diffie-Hellman over either finite fields or elliptic curves)
- PSK-only
- PSK with (EC)DHE

```
enum {

    /* Elliptic Curve Groups (ECDHE) */
    secp256r1(0x0017), secp384r1(0x0018), secp521r1(0x0019),
    x25519(0x001D), x448(0x001E),

    /* Finite Field Groups (DHE) */
    ffdhe2048(0x0100), ffdhe3072(0x0101), ffdhe4096(0x0102),
    ffdhe6144(0x0103), ffdhe8192(0x0104),

    /* Reserved Code Points */
    ffdhe_private_use(0x01FC..0x01FF),
    ecdhe_private_use(0xFE00..0xFEFF),
    (0xFFFF)
} NamedGroup;
```

```
enum {
    /* RSASSA-PKCS1-v1_5 algorithms */
    rsa_pkcs1_sha256(0x0401),
    rsa_pkcs1_sha384(0x0501),
    rsa_pkcs1_sha512(0x0601),

    /* ECDSA algorithms */
    ecdsa_secp256r1_sha256(0x0403),
    ecdsa_secp384r1_sha384(0x0503),
    ecdsa_secp521r1_sha512(0x0603),

    /* RSASSA-PSS algorithms with public key OID rsaEncryption */
    rsa_pss_rsae_sha256(0x0804),
    rsa_pss_rsae_sha384(0x0805),
    rsa_pss_rsae_sha512(0x0806),

    /* EdDSA algorithms */
    ed25519(0x0807),
    ed448(0x0808),

    /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
    rsa_pss_pss_sha256(0x0809),
    rsa_pss_pss_sha384(0x080a),
    rsa_pss_pss_sha512(0x080b),

    /* Legacy algorithms */
    rsa_pkcs1_sha1(0x0201),
    ecdsa_sha1(0x0203),

    /* Reserved Code Points */
    private_use(0xFE00..0xFFFF),
    (0xFFFF)
} SignatureScheme;
```

Application

TLS/SSL

5 Session

TLS consists of two primary components:

- A handshake protocol (Section 4) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering; an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.

4

3

- A record protocol (Section 5) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

2

Physical

1 Physical

OSI Reference Model

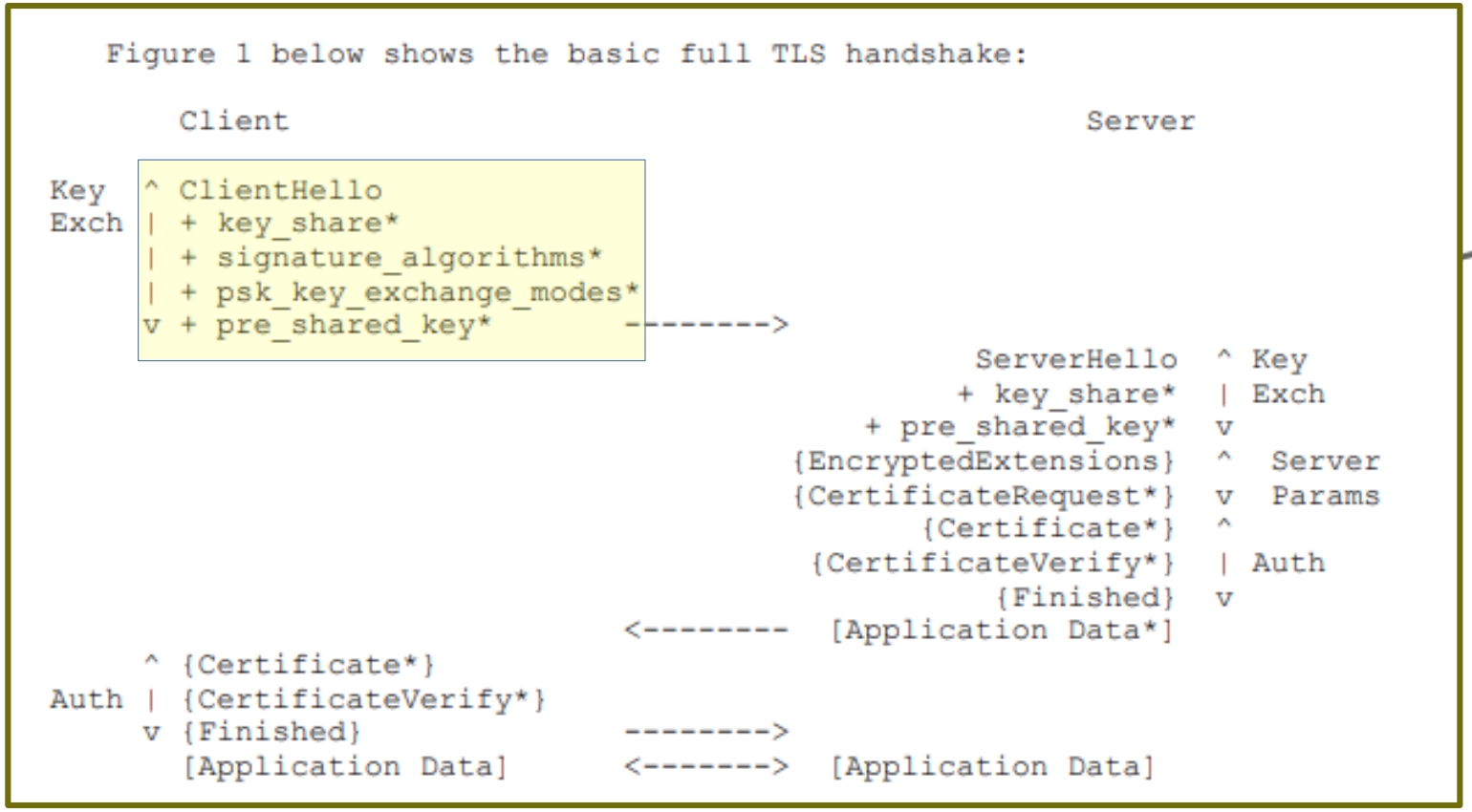


Figure 1 below shows the basic full TLS handshake:

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                ServerHello  ^ Key
                                              + key_share*   | Exch
                                         + pre_shared_key*   v
                                     {EncryptedExtensions}   ^  Server
                                     {CertificateRequest*}   v  Params
                                            {Certificate*}   ^
                                      {CertificateVerify*}   | Auth
                                                {Finished}   v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

2 Data Link

1 Physical

Data Link

Physical

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Application

Handshake Layer

Handshake

Alert

Change Cipher Spec

Record Layer

fragmentation

Compression

Authentication

Encryption

Transport TCP UDP

```
16          TLS handshake protocol
  03 01       SSL version 3.1 (TLS 1.0)
  01 37       Length of payload (311 bytes)
```

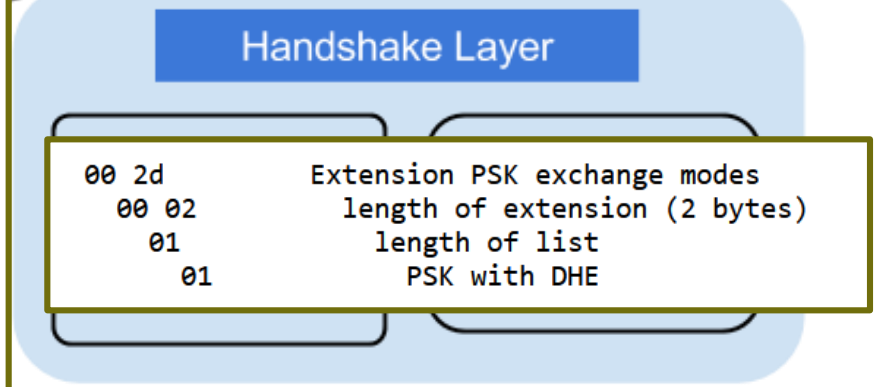
```
01          Handshake type Client Hello
  00 01 33  Length of payload (307 bytes)
  03 03     SSL version 3.3 (TLS 1.2)
```

```
00 33         Extension key share
  00 26         length of extension (38 bytes)
    00 24         length of key share (36 bytes)
      00 1d         Group x25519
        00 20         Length of key exchange (32 bytes)     RFC 7748
          e7 62 a7 5a 88 e1 e9 21 c9 4a 80 0a 0e 65 51 d7 fc c8
          55 df f7 b2 e8 9d a6 b5 60 6b ba c0 0f 2f              key
```

```
Key   ^ ClientHello
Exch  | + key_share*
      | + signature_algorithms*
      | + psk_key_exchange_modes*
      v + pre_shared_key*
```

```
00 0d     Extension signature algorithms
  00 30     length (48 bytes)
    00 2e     length of list
      04 03   ecdsa_secp256r1_sha256
      05 03   ecdsa_secp384r1_sha384
      06 03   ecdsa_secp521r1_sha512
      08 07   ed25519
      08 08   ed448
      08 09   rsa_pss_pss_sha256
      08 0a   rsa_pss_pss_sha384
      08 0b   rsa_pss_pss_sha512
      08 04   rsa_pss_rsae_sha256
      08 05   rsa_pss_rsae_sha384
      08 06   rsa_pss_rsae_sha512
      04 01   rsa_pkcs1_sha256
      05 01   rsa_pkcs1_sha384
      06 01   rsa_pkcs1_sha512
      03 03   sha224_ecdsa
      02 03   ecdsa_sha1
      03 01   sha224_rsa
      02 01   rsa_pkcs1_sha1
      03 02   sha224_dsa
      02 02   sha1_dsa
      04 02   sha256_dsa
      05 02   sha384_dsa
      06 02   sha512_dsa
```

```
0d 19 8c 01 36 7a b0 a8 0a 1b bc 5e 33 e9 a4 e8 6e
2a ea 74 ec 19 0b 78 9c 25 4f 45 12 f3 1b bc   client random
20        Session ID length (32 bytes)
c1 63 ad d1 41 8b b4 75 8d ec ed b5 28 47 13 52 64
60 14 ff ad 7c 2b 6c b4 8c 7d 4c 53 c8 2d 58   session ID
00 3e     length of cipher suites (62 bytes)
  13 02   AES_256_GCM_SHA384
  13 03   CHACHA20_POLY1305_SHA256
  13 01   AES_128_GCM_SHA256
  c0 2c   ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  c0 30   ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  00 9f   DHE_RSA_WITH_AES_256_GCM_SHA384
  cc a9   ECDHE_ECDSA_WITH_CHACHA_20_POLY1305_SHA256
  cc a8   ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  cc aa   DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
  c0 2b   ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  c0 2f   ECDHE_RSA_WITH_AES_128_GCM_SHA256
  00 9e   DHE_RSA_WITH_AES_128_GCM_SHA256
  c0 24   ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
  c0 28   ECDHE_RSA_WITH_AES_256_CBC_SHA384
  00 6b   DHE_RSA_WITH_AES_256_CBC_SHA384
  c0 23   ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
  c0 27   ECDHE_RSA_WITH_AES_128_CBC_SHA256
  00 67   DHE_RSA_WITH_AES_128_CBC_SHA256
  c0 0a   ECDHE_ECDSA_WITH_AES_256_CBC_SHA
  c0 14   ECDHE_RSA_WITH_AES_256_CBC_SHA
  00 39   DHE_RSA_WITH_AES_256_CBC_SHA
  c0 09   ECDHE_ECDSA_WITH_AES_128_CBC_SHA
  c0 13   ECDHE_RSA_WITH_AES_128_CBC_SHA
  00 33   DHE_RSA_WITH_AES_128_CBC_SHA
  00 9d   RSA_WITH_AES_256_GCM_SHA384
  00 9c   RSA_WITH_AES_128_GCM_SHA256
  00 3d   RSA_WITH_AES_256_CBC_SHA256
  00 3c   RSA_WITH_AES_128_CBC_SHA256
  00 35   RSA_WITH_AES_256_CBC_SHA
  00 2f   RSA_WITH_AES_128_CBC_SHA
  00 ff   psuedo-cipher-suite "renegotiation SCSV supported"
01        compression methods length (1 byte)
00        compression method (no compression)
```




```
00 2d         Extension PSK exchange modes
  00 02         length of extension (2 bytes)
    01            length of list
      01            PSK with DHE
```

```
00 0b     Extension EC points format
  00 04     length of formats (4 bytes)
    03        length of list
    00        uncompressed
    01        ANSI X.962 compressed prime
    02        ANSI X.962 compressed char2
00 0a     Extension Supported groups
  00 0c     length of supported groups (12 bytes)
    00 0a     length of list
      00 1d   x25519              RFC 7748
      00 17   secp256r1
      00 1e   x448
      00 19   secp521r1
      00 18   secp348r1
```

OSI Reference Model

Figure 1 below shows the basic full TLS handshake:

```
       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                ServerHello  ^ Key
                                              + key_share*   | Exch
                                         + pre_shared_key*   v
                                     {EncryptedExtensions}   ^  Server
                                     {CertificateRequest*}   v  Params
                                            {Certificate*}   ^
                                      {CertificateVerify*}   | Auth
                                                {Finished}   v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]
```

2 Data Link

1 Physical

Data Link

Physical

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Application

Handshake Layer

Handshake

Alert

Change Cipher Spec

Record Layer

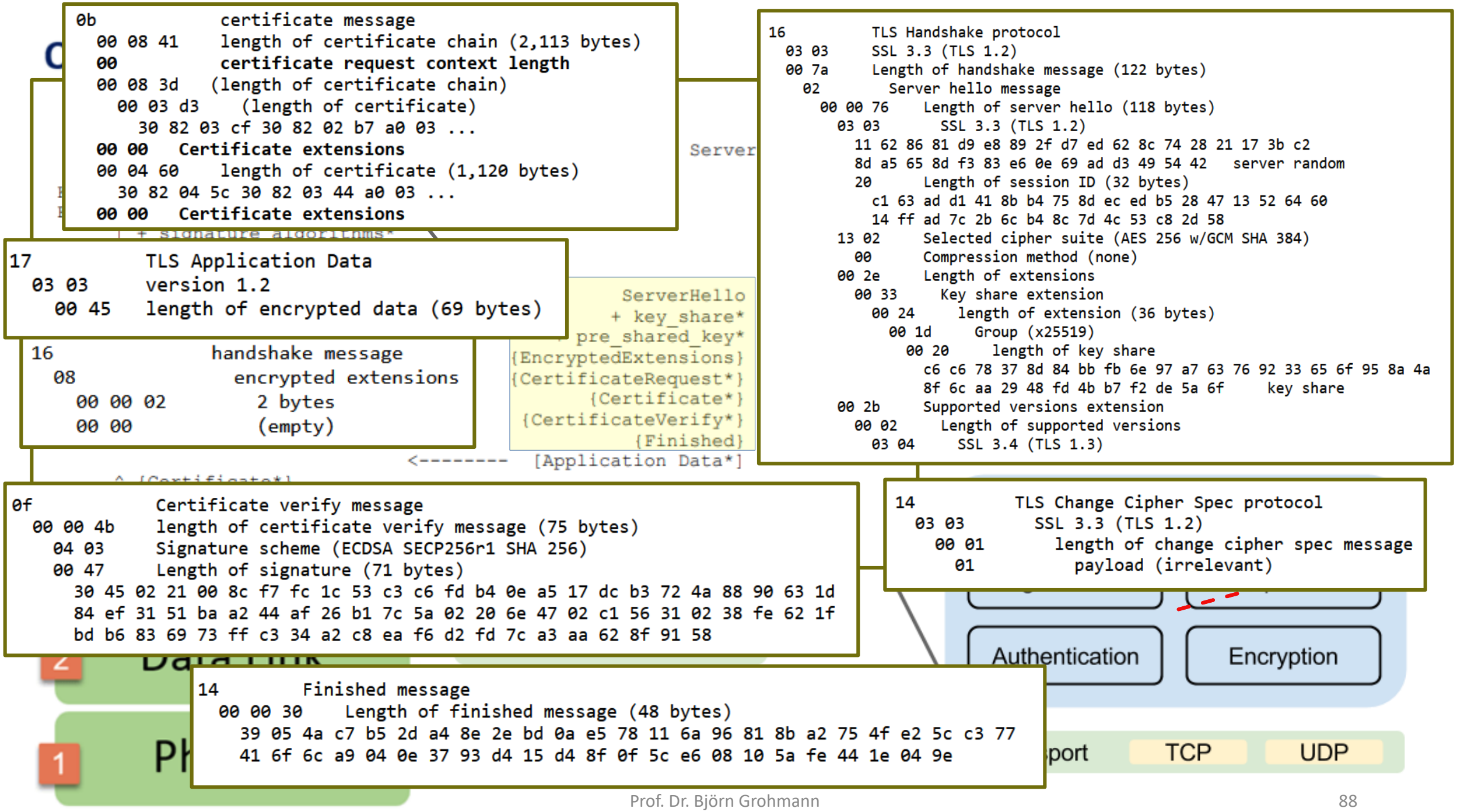
fragmentation

Compression

Authentication

Encryption

Transport TCP UDP


0b certificate message
00 08 41 length of certificate chain (2,113 bytes)
00 certificate request context length
00 08 3d (length of certificate chain)
00 03 d3 (length of certificate)
30 82 03 cf 30 82 02 b7 a0 03 ...
00 00 Certificate extensions
00 04 60 length of certificate (1,120 bytes)
30 82 04 5c 30 82 03 44 a0 03 ...
00 00 Certificate extensions
16 TLS Handshake protocol
03 03 SSL 3.3 (TLS 1.2)
00 7a Length of handshake message (122 bytes)
02 Server hello message
00 00 76 Length of server hello (118 bytes)
03 03 SSL 3.3 (TLS 1.2)
11 62 86 81 d9 e8 89 2f d7 ed 62 8c 74 28 21 17 3b c2
8d a5 65 8d f3 83 e6 0e 69 ad d3 49 54 42 server random
20 Length of session ID (32 bytes)
c1 63 ad d1 41 8b b4 75 8d ec ed b5 28 47 13 52 64 60
14 ff ad 7c 2b 6c b4 8c 7d 4c 53 c8 2d 58
13 02 Selected cipher suite (AES 256 w/GCM SHA 384)
00 Compression method (none)
00 2e Length of extensions
00 33 Key share extension
00 24 length of extension (36 bytes)
00 1d Group (x25519)
00 20 length of key share
c6 c6 78 37 8d 84 bb fb 6e 97 a7 63 76 92 33 65 6f 95 8a 4a
8f 6c aa 29 48 fd 4b b7 f2 de 5a 6f key share
00 2b Supported versions extension
00 02 Length of supported versions
03 04 SSL 3.4 (TLS 1.3)
17 TLS Application Data
03 03 version 1.2
00 45 length of encrypted data (69 bytes)
16 handshake message
08 encrypted extensions
00 00 02 2 bytes
00 00 (empty)
ServerHello
+ key_share*
pre_shared_key*
{EncryptedExtensions}
{CertificateRequest*}
{Certificate*}
{CertificateVerify*}
{Finished}
<-------- [Application Data*]
0f Certificate verify message
00 00 4b length of certificate verify message (75 bytes)
04 03 Signature scheme (ECDSA SECP256r1 SHA 256)
00 47 Length of signature (71 bytes)
30 45 02 21 00 8c f7 fc 1c 53 c3 c6 fd b4 0e a5 17 dc b3 72 4a 88 90 63 1d
84 ef 31 51 ba a2 44 af 26 b1 7c 5a 02 20 6e 47 02 c1 56 31 02 38 fe 62 1f
bd b6 83 69 73 ff c3 34 a2 c8 ea f6 d2 fd 7c a3 aa 62 8f 91 58
14 TLS Change Cipher Spec protocol
03 03 SSL 3.3 (TLS 1.2)
00 01 length of change cipher spec message
01 payload (irrelevant)
Authentication
Encryption
14 Finished message
00 00 30 Length of finished message (48 bytes)
39 05 4a c7 b5 2d a4 8e 2e bd 0a e5 78 11 6a 96 81 8b a2 75 4f e2 5c c3 77
41 6f 6c a9 04 0e 37 93 d4 15 d4 8f 0f 5c e6 08 10 5a fe 44 1e 04 9e
TCP
UDP

OSI Reference Model

```
   Figure 1 below shows the basic full TLS handshake:

         Client                                           Server

Key   ^ ClientHello
Exch  | + key_share*
      | + signature_algorithms*
      | + psk_key_exchange_modes*
      v + pre_shared_key*       -------->
                                                ServerHello  ^ Key
                                              + key_share*   | Exch
                                         + pre_shared_key*   v
                                     {EncryptedExtensions}   ^  Server
                                     {CertificateRequest*}   v  Params
                                            {Certificate*}   ^
                                      {CertificateVerify*}   | Auth
                                                {Finished}   v
                           <--------  [Application Data*]

      ^ {Certificate*}
Auth  | {CertificateVerify*}
      v {Finished}              -------->
        [Application Data]      <------->  [Application Data]
```

2 Data Link

1 Physical

Data Link

Physical

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Application

Handshake Layer

Handshake

Alert

Change Cipher Spec

Record Layer

fragmentation

Compression

Authentication

Encryption

Transport TCP UDP

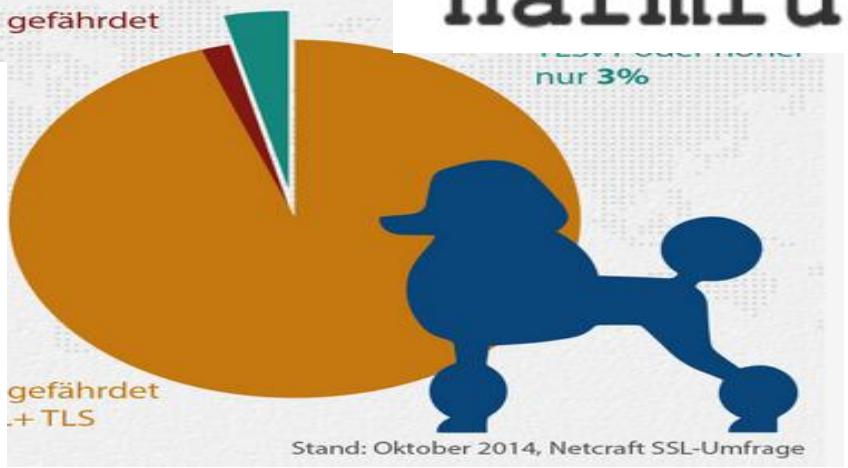
TRANSPORT LAYER SECURITY

SSL and TLS protocols

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecated in 2021 (RFC 8996)[8][9][10]
TLS 1.1	2006	Deprecated in 2021 (RFC 8996)[8][9][10]
TLS 1.2	2008	
TLS 1.3	2018	

„ATTACKS ALWAYS BETTER; THEY NEVER GET WORSE“

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law



TRANSPORT LAYER SECURITY

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Cipher security against publicly known feasible attacks

Cipher			Protocol version						Status
Type	Algorithm	Nominal strength (bits)	SSL 2.0	SSL 3.0 [n 1][n 2][n 3][n 4]	TLS 1.0 [n 1][n 3]	TLS 1.1 [n 1]	TLS 1.2 [n 1]	TLS 1.3	
Block cipher with mode of operation	AES GCM[58][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	AES CCM[59][n 5]		N/A	N/A	N/A	N/A	Secure	Secure	
	AES CBC[n 6]		N/A	Insecure	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	Camellia GCM[60][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	Camellia CBC[61][n 6]		N/A	Insecure	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	ARIA GCM[62][n 5]	256, 128	N/A	N/A	N/A	N/A	Secure	N/A	
	ARIA CBC[62][n 6]		N/A	N/A	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	SEED CBC[63][n 6]	128	N/A	Insecure	Depends on mitigations	Depends on mitigations	Depends on mitigations	N/A	
	3DES EDE CBC[n 6][n 7]	112[n 8]	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	
	GOST 28147-89 CNT[57][n 7]	256	N/A	N/A	Insecure	Insecure	Insecure	N/A	Defined in RFC 4357
	IDEA CBC[n 6][n 7][n 9]	128	Insecure	Insecure	Insecure	Insecure	N/A	N/A	Removed from TLS 1.2
	DES CBC[n 6][n 7][n 9]	56	Insecure	Insecure	Insecure	Insecure	N/A	N/A	
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	Forbidden in TLS 1.1 and later
	RC2 CBC[n 6][n 7]	40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
Stream cipher	ChaCha20-Poly1305[68][n 5]	256	N/A	N/A	N/A	N/A	Secure	Secure	Defined for TLS 1.2 in RFCs
	RC4[n 11]	128	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	Prohibited in all versions of TLS by RFC 7465
		40[n 10]	Insecure	Insecure	Insecure	N/A	N/A	N/A	
None	Null[n 12]	–	Insecure	Insecure	Insecure	Insecure	Insecure	N/A	Defined for TLS 1.2 in RFCs

TRANSPORT LAYER SECURITY

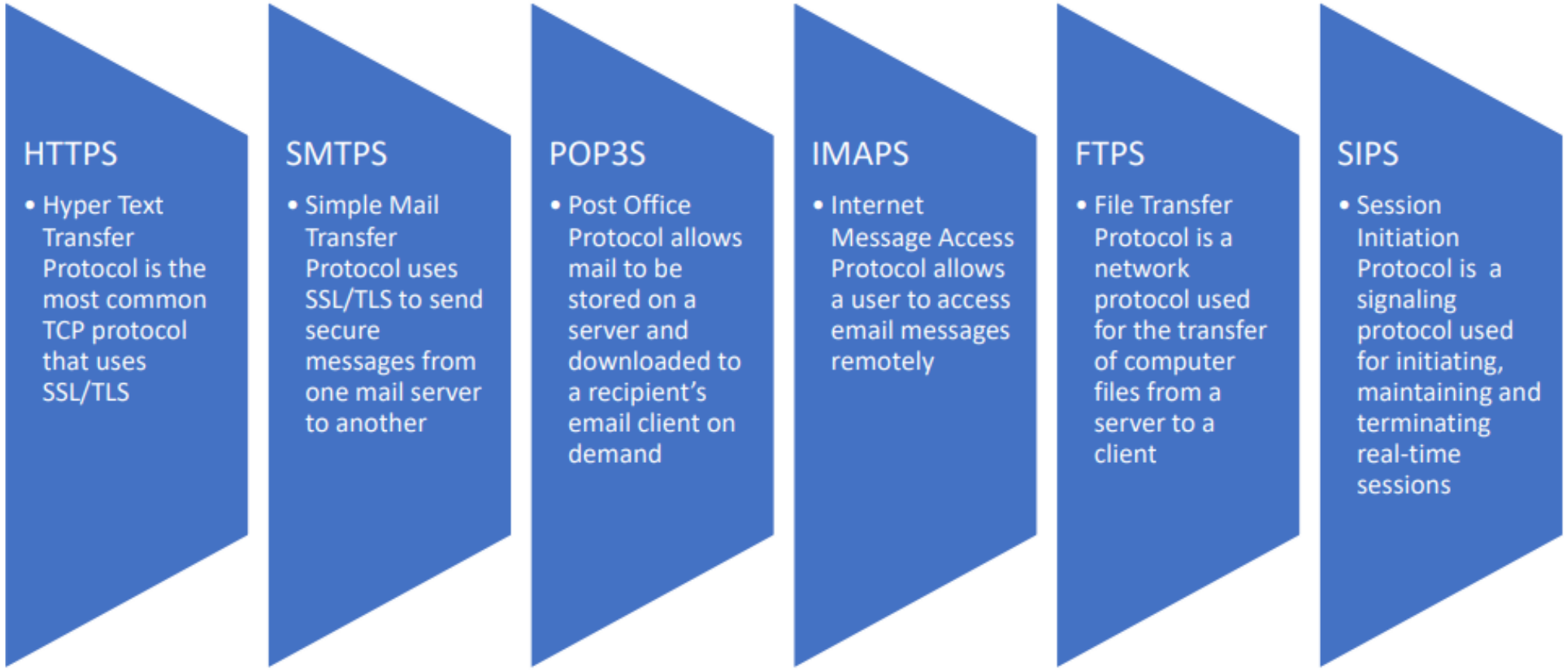
Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Data integrity

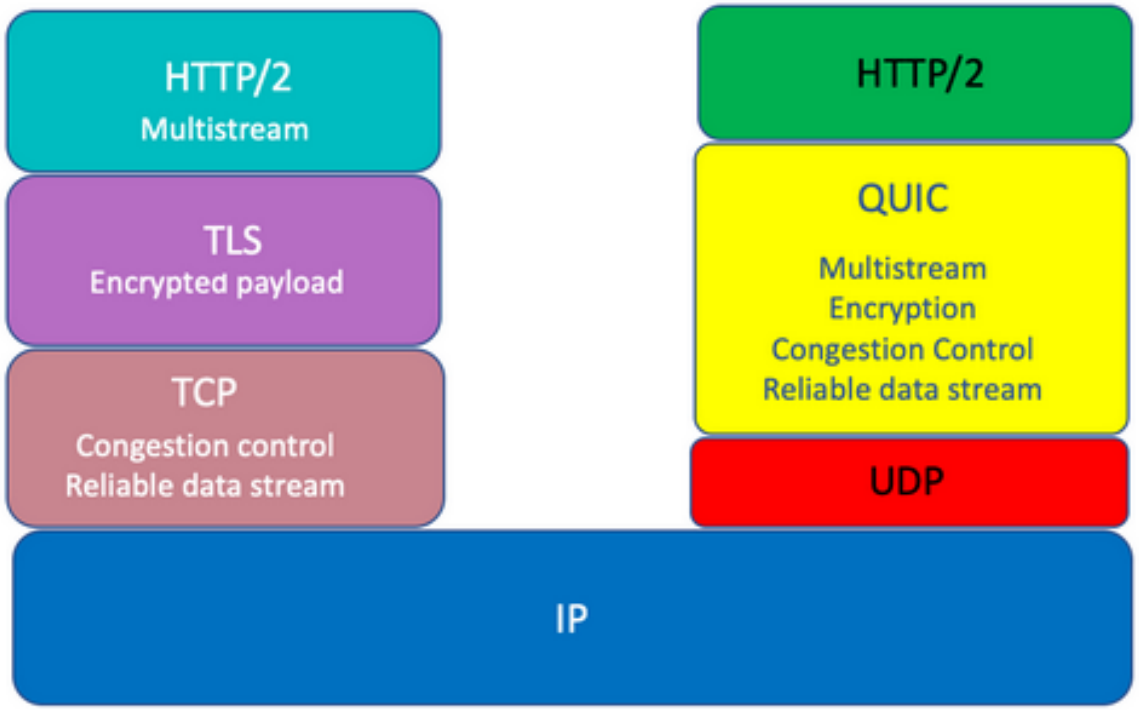
Algorithm	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Status
HMAC-MD5	Yes	Yes	Yes	Yes	Yes	No	Defined for TLS 1.2 in RFCs
HMAC-SHA1	No	Yes	Yes	Yes	Yes	No	
HMAC-SHA256/384	No	No	No	No	Yes	No	
AEAD	No	No	No	No	Yes	Yes	
GOST 28147-89 IMIT[57]	No	No	Yes	Yes	Yes		Proposed in RFC drafts
GOST R 34.11-94[57]	No	No	Yes	Yes	Yes		

TRANSPORT LAYER SECURITY

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

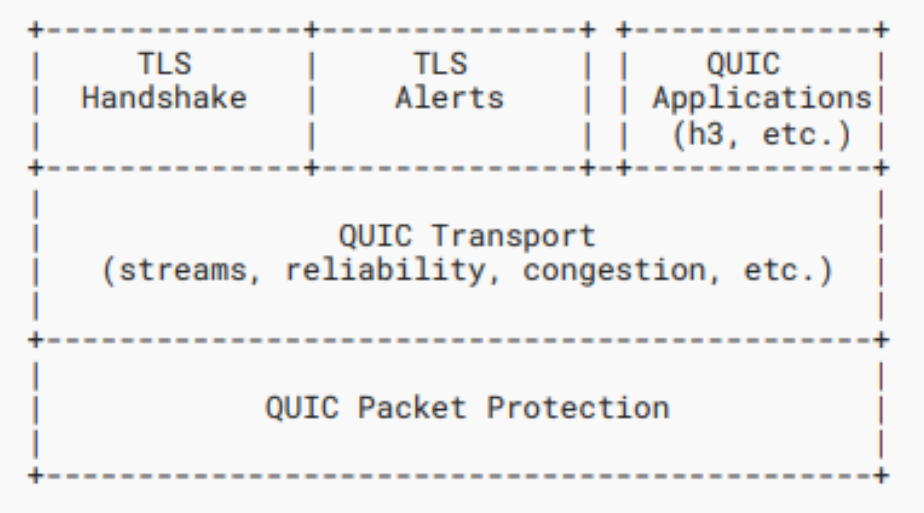


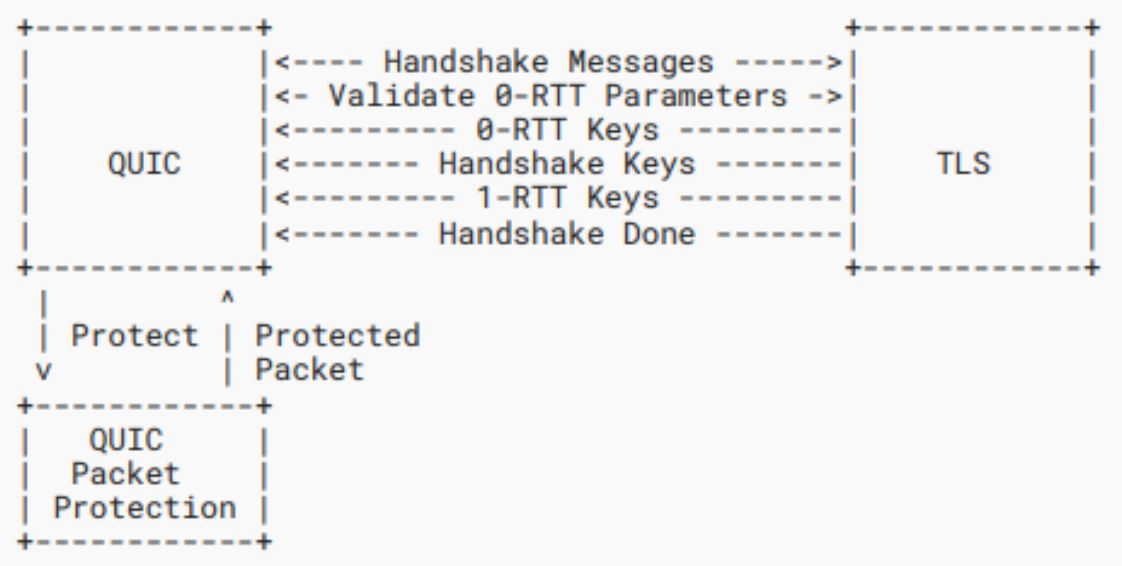
QUIC

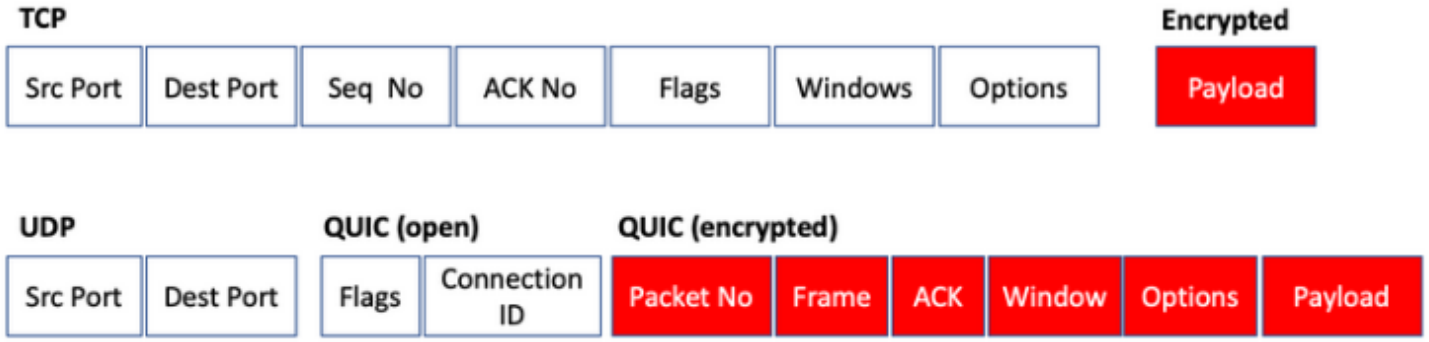

HTTP/2
Multistream
TLS
Encrypted payload
TCP
Congestion control
Reliable data stream
HTTP/2
QUIC
Multistream
Encryption
Congestion Control
Reliable data stream
UDP
IP

QUIC

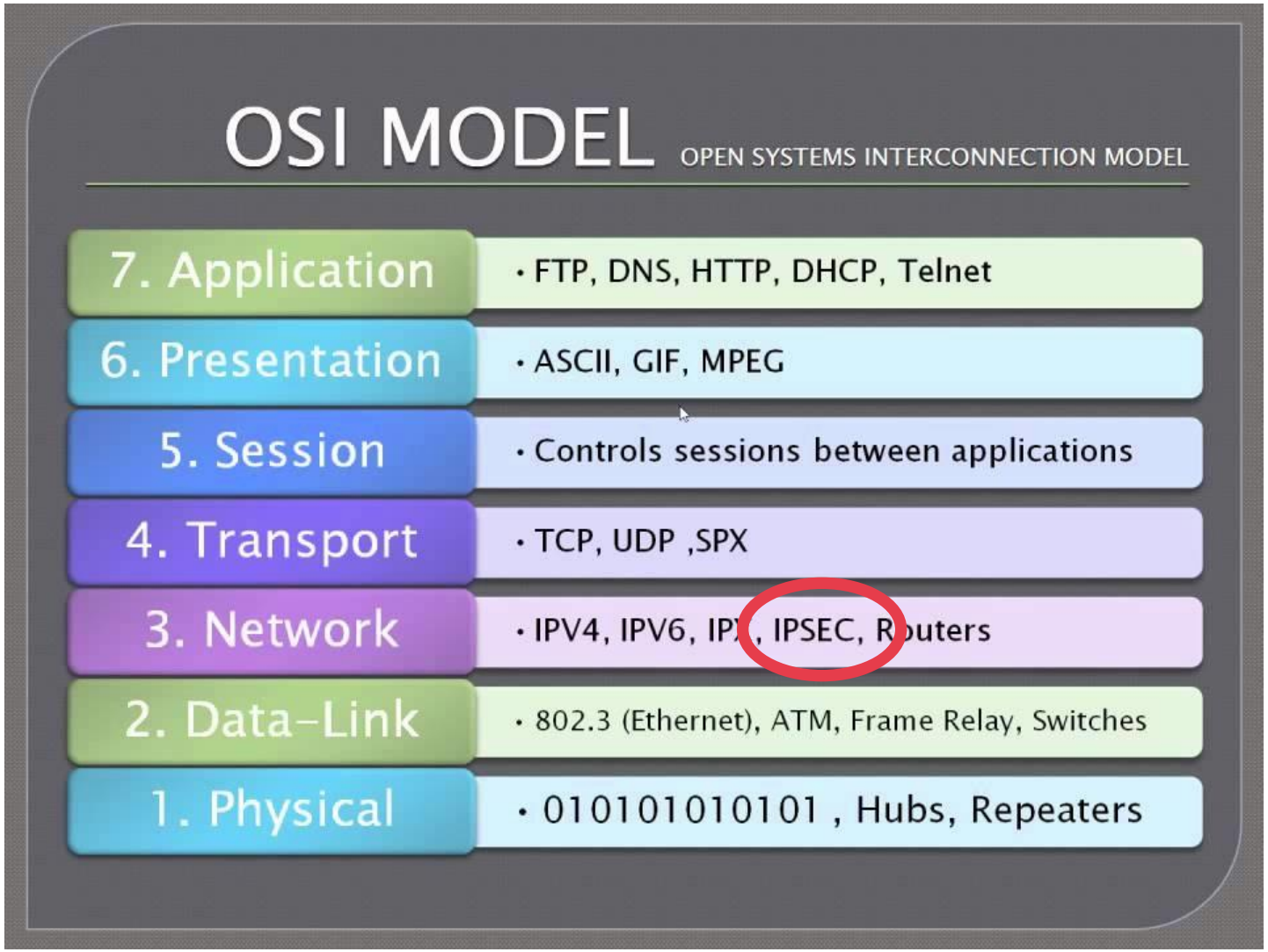
Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law



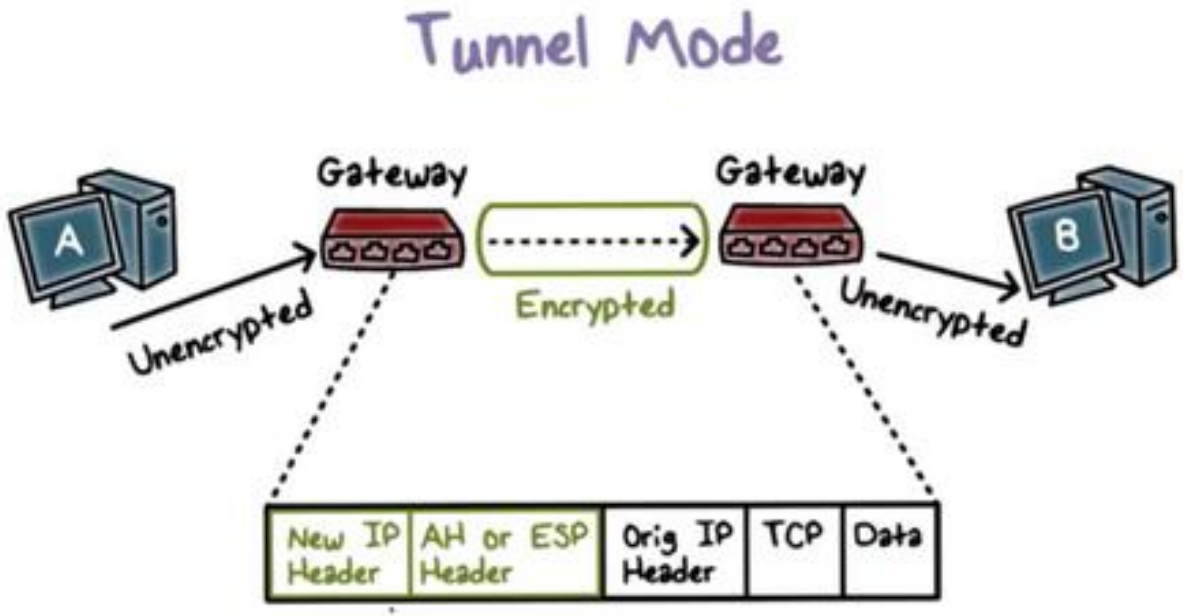




IPSEC

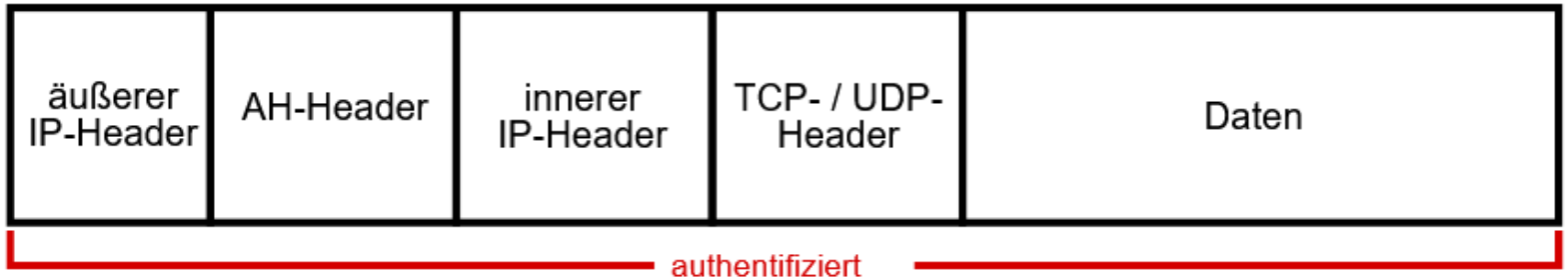

OSI MODEL
OPEN SYSTEMS INTERCONNECTION MODEL
7. Application
• FTP, DNS, HTTP, DHCP, Telnet
6. Presentation
• ASCII, GIF, MPEG
5. Session
• Controls sessions between applications
4. Transport
• TCP, UDP ,SPX
3. Network
• IPV4, IPV6, IP
, IPSEC, R
uters
2. Data-Link
• 802.3 (Ethernet), ATM, Frame Relay, Switches
1. Physical
• 010101010101 , Hubs, Repeaters

IPSEC

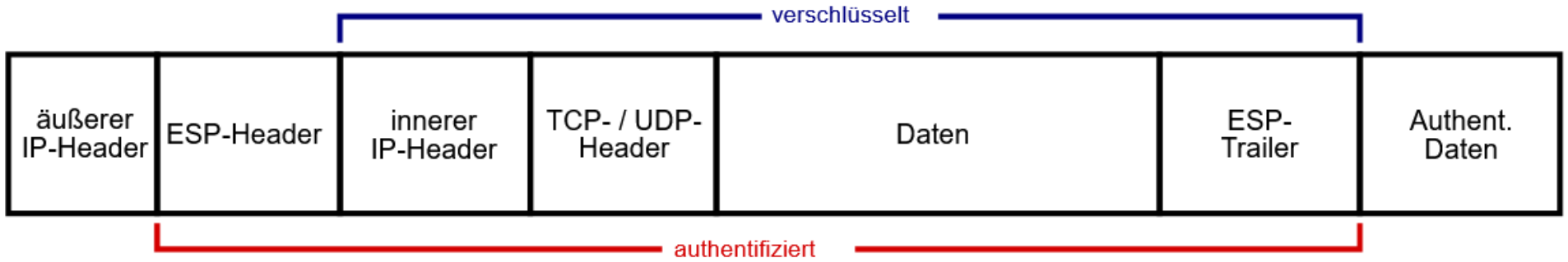

Tunnel Mode
Gateway
Gateway
A
B
Unencrypted
Encrypted
Unencrypted
New IP Header
AH or ESP Header
Orig IP Header
TCP
Data

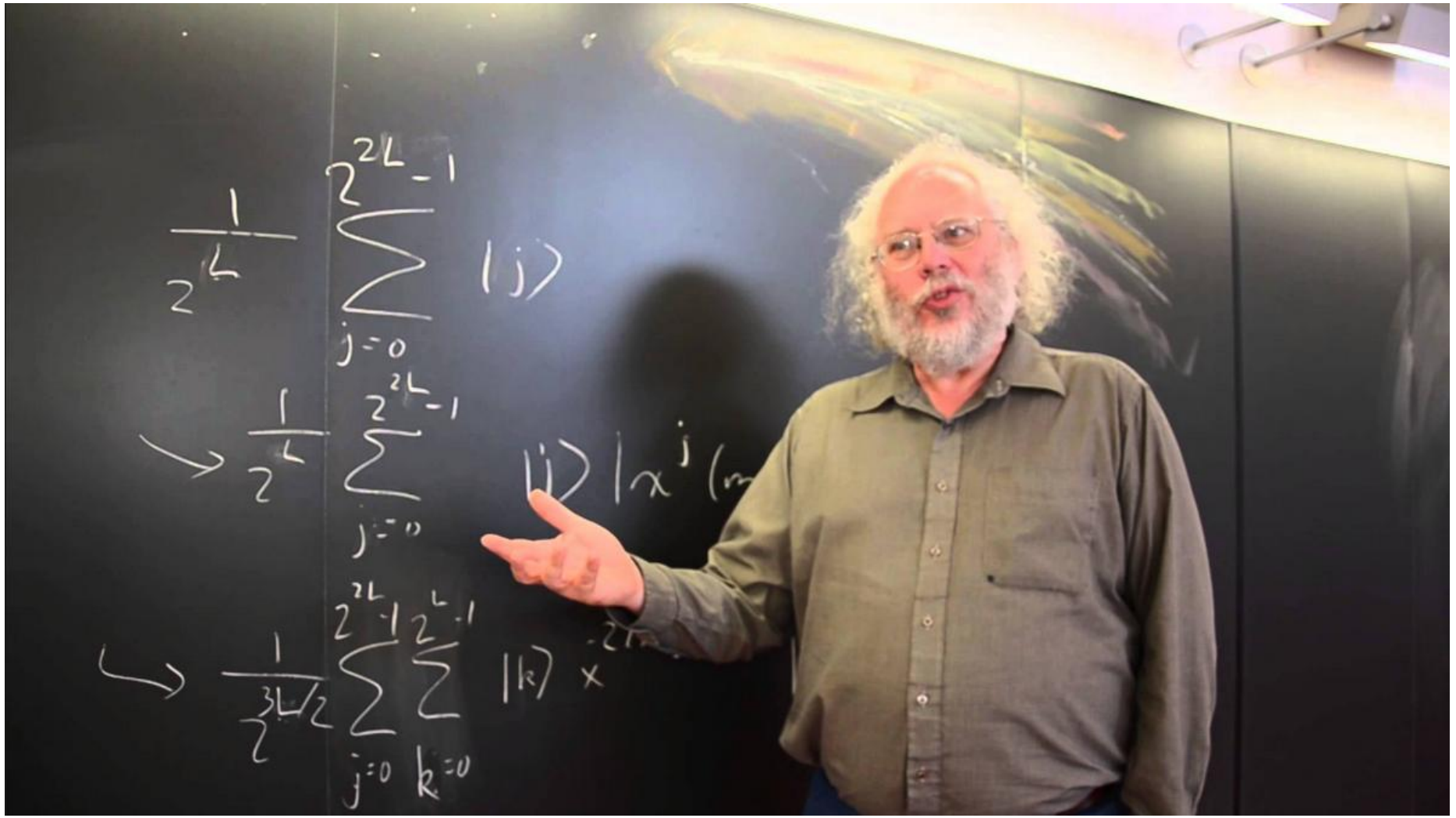
IPSEC

AH im Tunnelmodus



ESP im Tunnelmodus




$\frac{1}{2^L}\sum_{j=0}^{2^{2L}-1}|j\rangle$
$\hookrightarrow \frac{1}{2^L}\sum_{j=0}^{2^{2L}-1}|j\rangle|x^j(m$
$\hookrightarrow \frac{1}{2^{3L/2}}\sum_{j=0}^{2^{2L}-1}\sum_{k=0}^{2^L-1}|k\rangle x$

"Die Quanten sind doch eine hoffnungslose Schweinerei!" *Max Born*

"Denn wenn man nicht zunächst über die Quantentheorie entsetzt ist, kann man sie doch unmöglich verstanden haben." *Niels Bohr*

"Wer glaubt, die Quantentheorie verstanden zu haben, hat sie nicht verstanden." *Richard Feynman*

"Die ganzen 50 Jahre bewusster Grübelei haben mich der Antwort der Frage ‚Was sind Lichtquanten' nicht näher gebracht. Heute glaubt zwar jeder Lump, er wisse es, aber er täuscht sich…" *Albert Einstein*

"Wenn ich von Schrödingers Katze höre, greife ich nach meiner Waffe." *Stephen Hawking*

KEY EXCHANGE

Quantum ~~Cryptography~~
Quantum Computing
Post Quantum Cryptography

QUANTUM MECHANICS

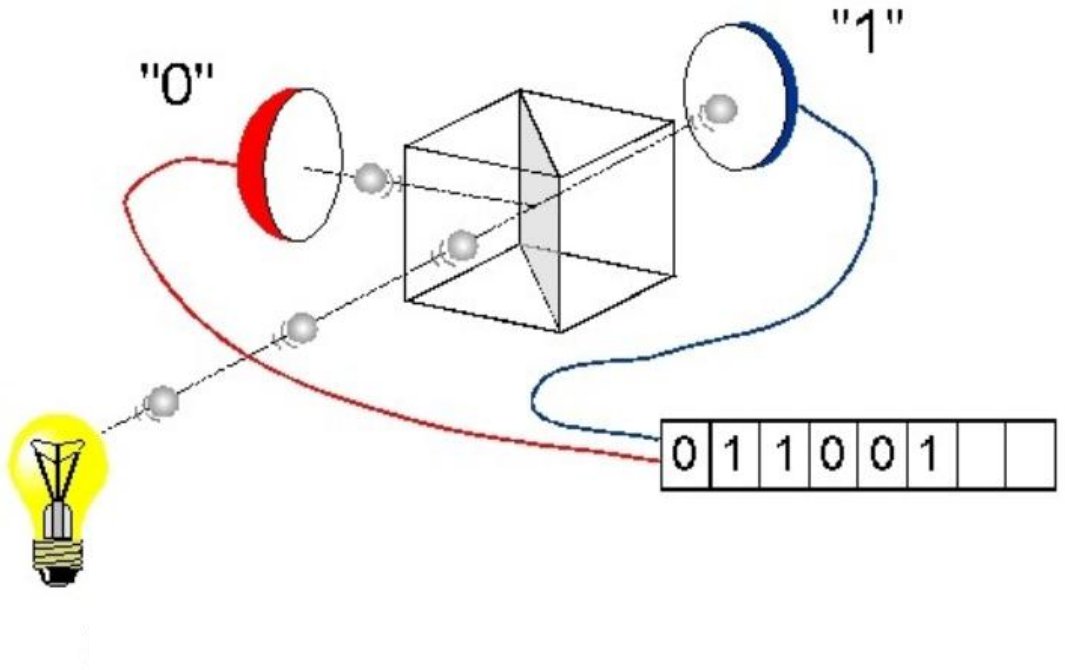
Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

- Nondeterminism
- Superposition
- Entanglement
- Uncertainty

QUANTUM MECHANICS

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

- Nondeterminism



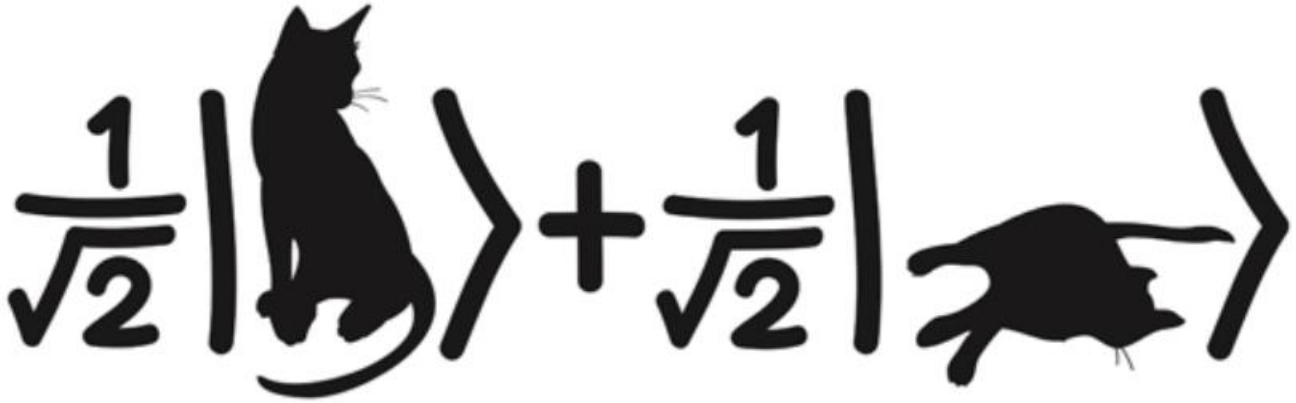
QUANTUM MECHANICS

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

- Nondeterminism

QUANTUM MECHANICS

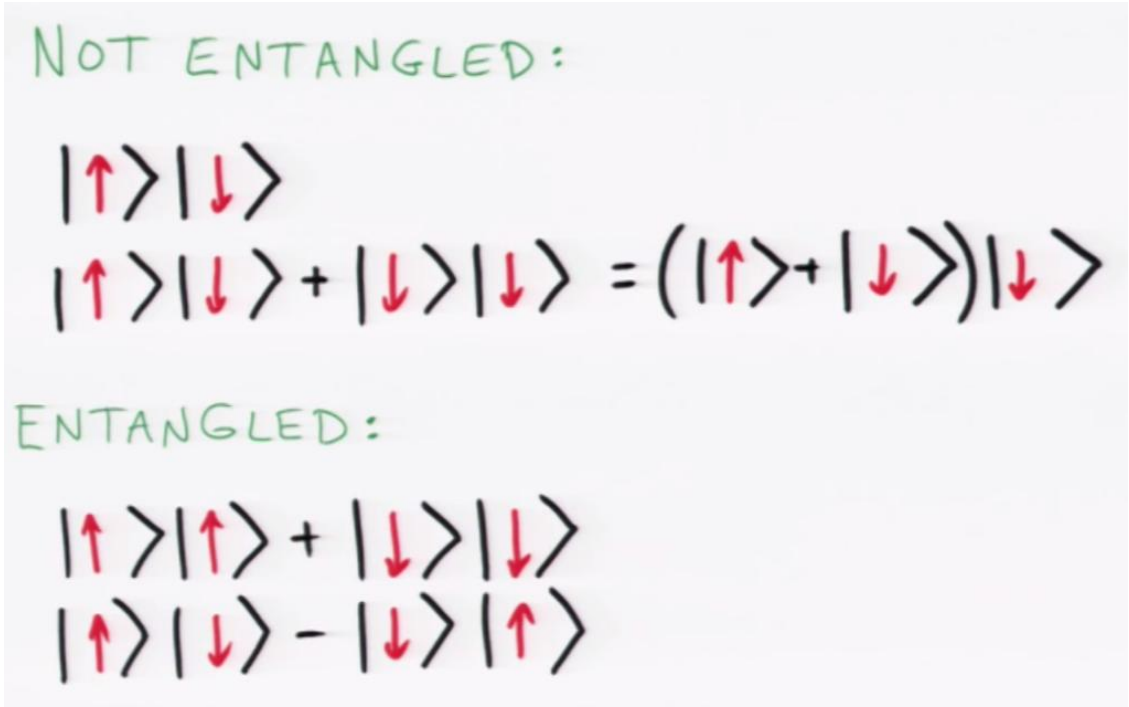
- Superposition



QUANTUM MECHANICS

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

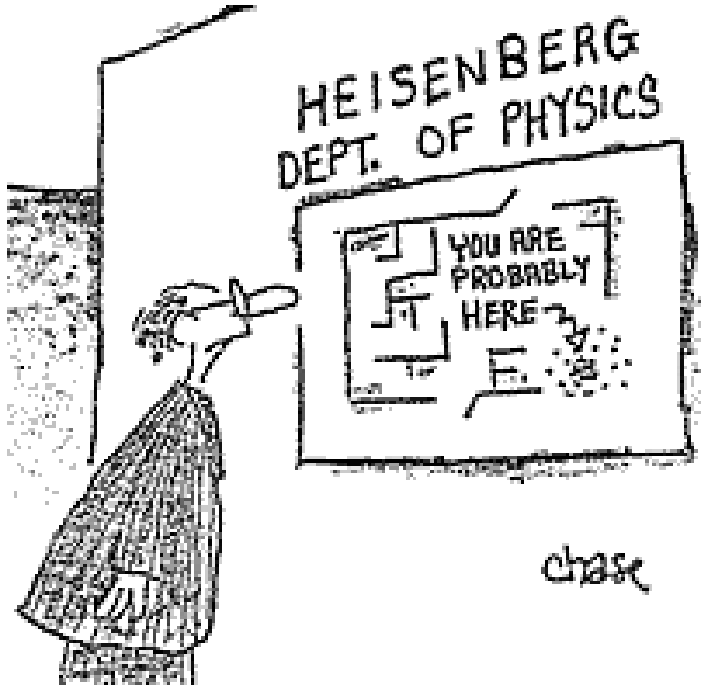
- Entanglement



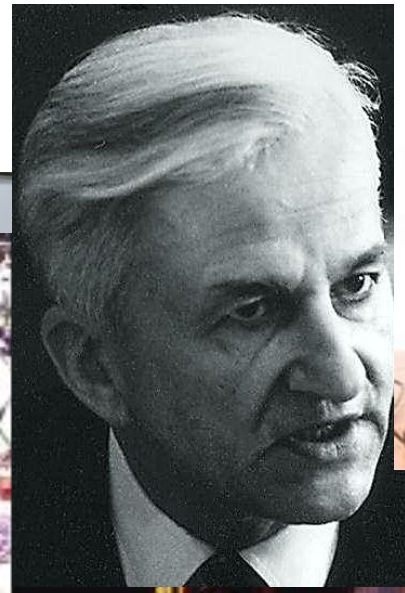
QUANTUM MECHANICS

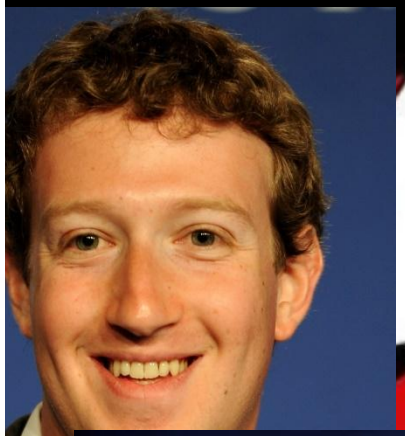
Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

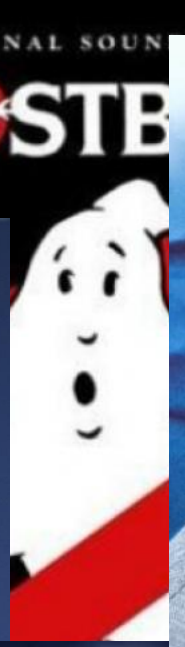
- Uncertainty

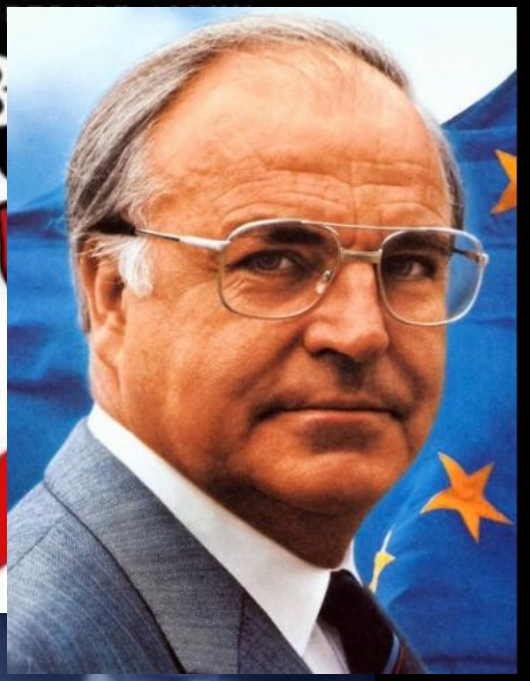


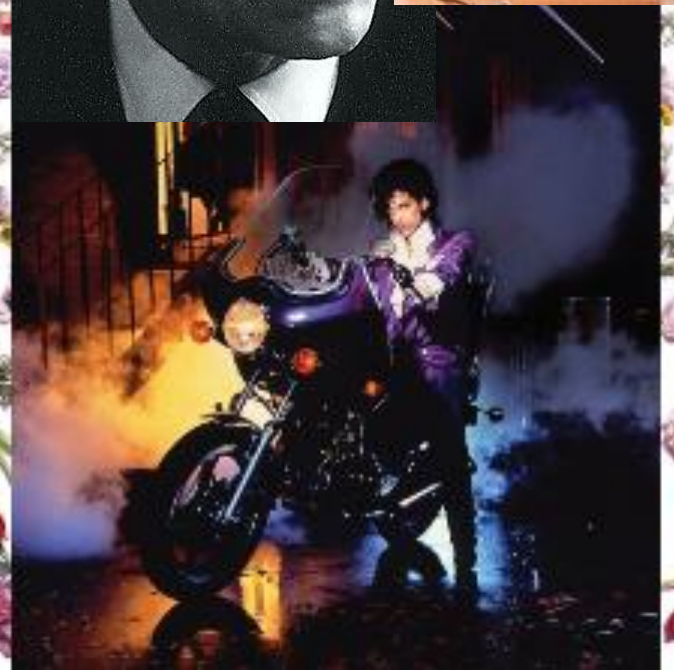
1984











1984

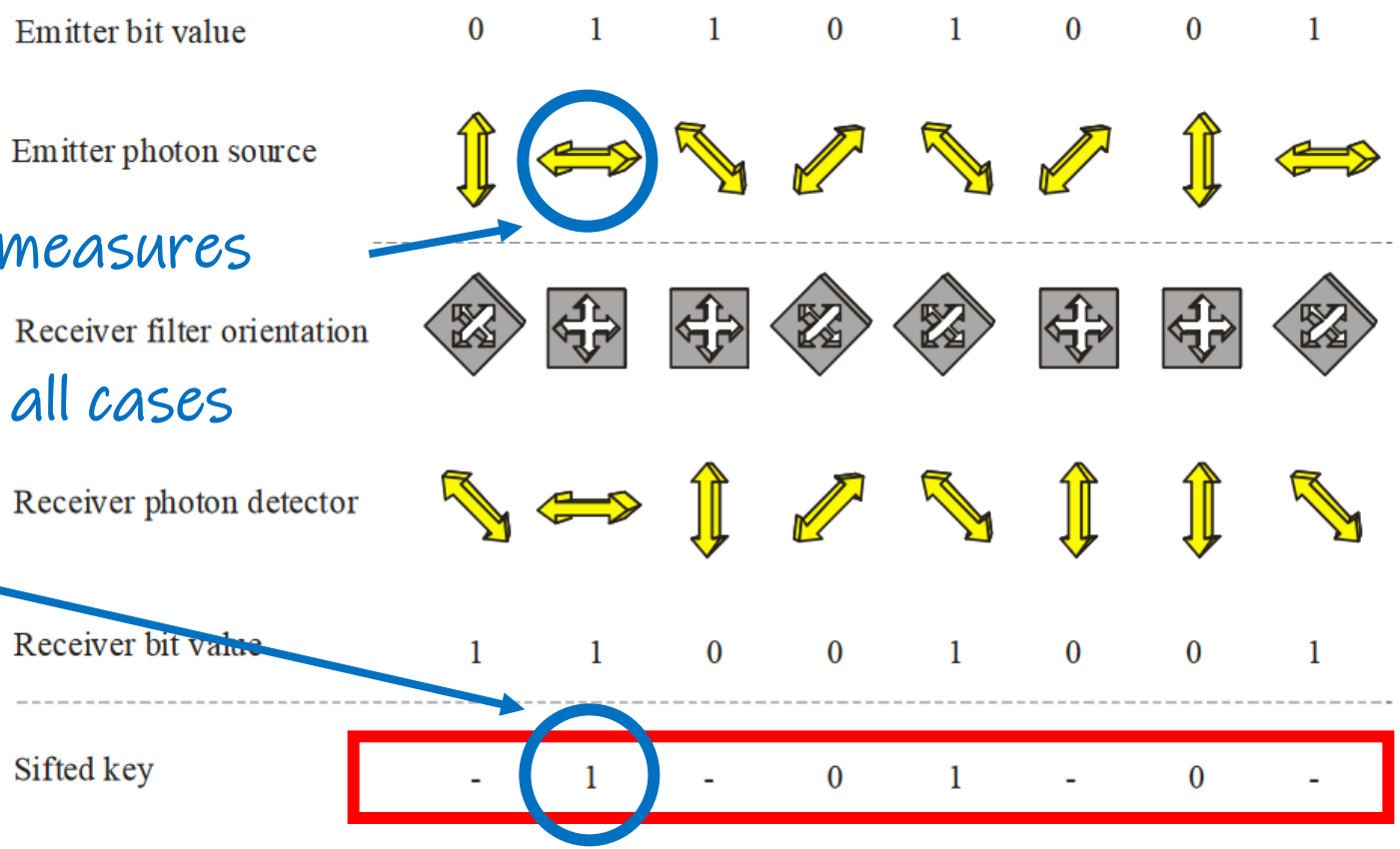
ORIGINAL SOUN
GHOSTB
Pluto
Mars
Saturn
Mercury
Earth
Neptune
Venus
Jupiter
Uranus
WITH THEIR

1984

BB84 PROTOCOL

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Alice chooses random bits and encodes them...



Whenever Eve measures this bit, then in a quarter of all cases this bit flips

Bob chooses a basis and makes a measurement...

...after telling Alice his choice, these are the remaining bits...

BB84 PROTOCOL

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Key Management

Seamless and automated key management

Dual key agreement:	conventional and quantum cryptography
Key refresh rate:	1 key/minute up to 12 encryptors
Quantum Key Distribution:	BB84 and SARG, up to 50 km (100km upon request)
Conventional Key Agreement:	RSA-2048, master key

Cerberis QKD Server

NO CLONING THEOREM

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

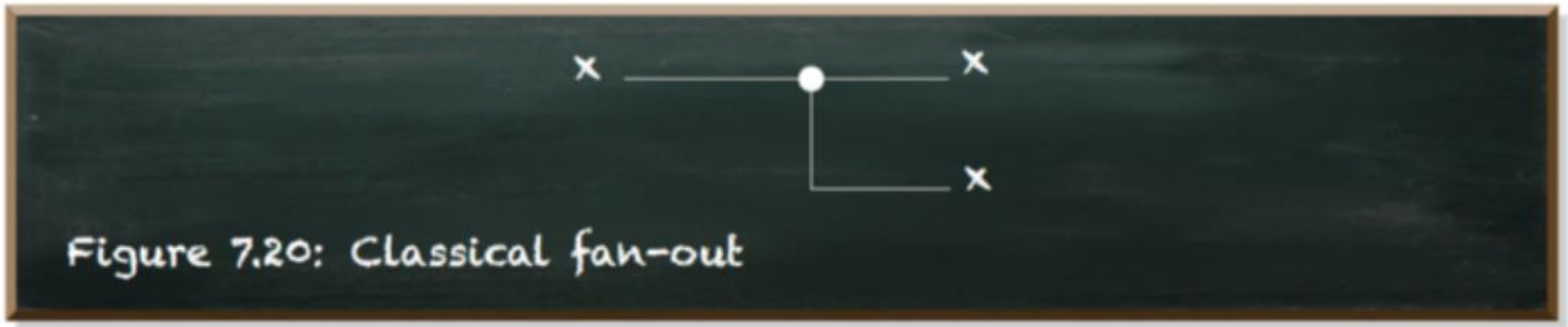


Figure 7.20: Classical fan-out

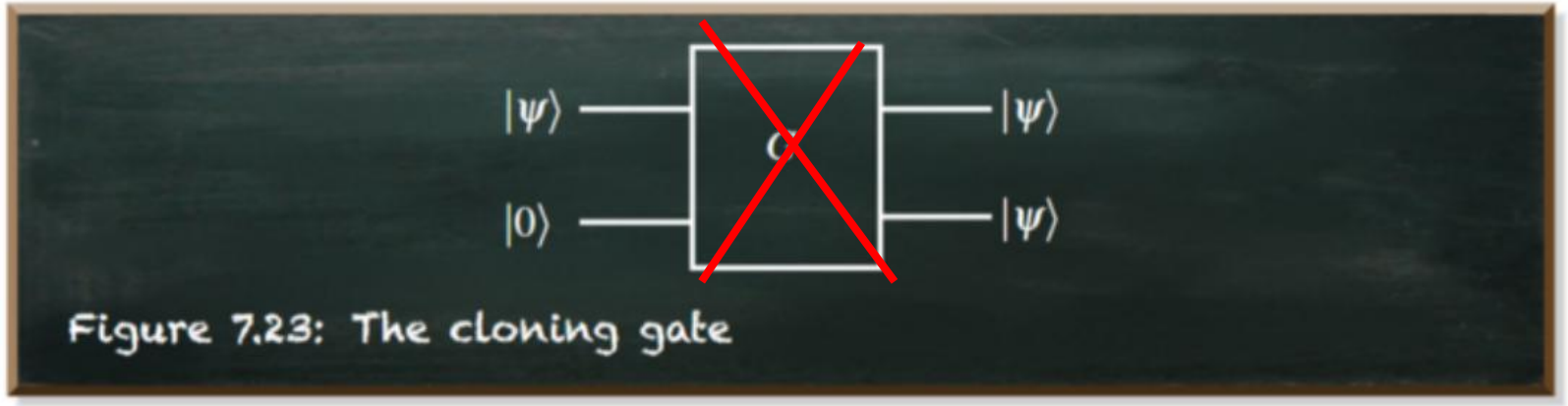
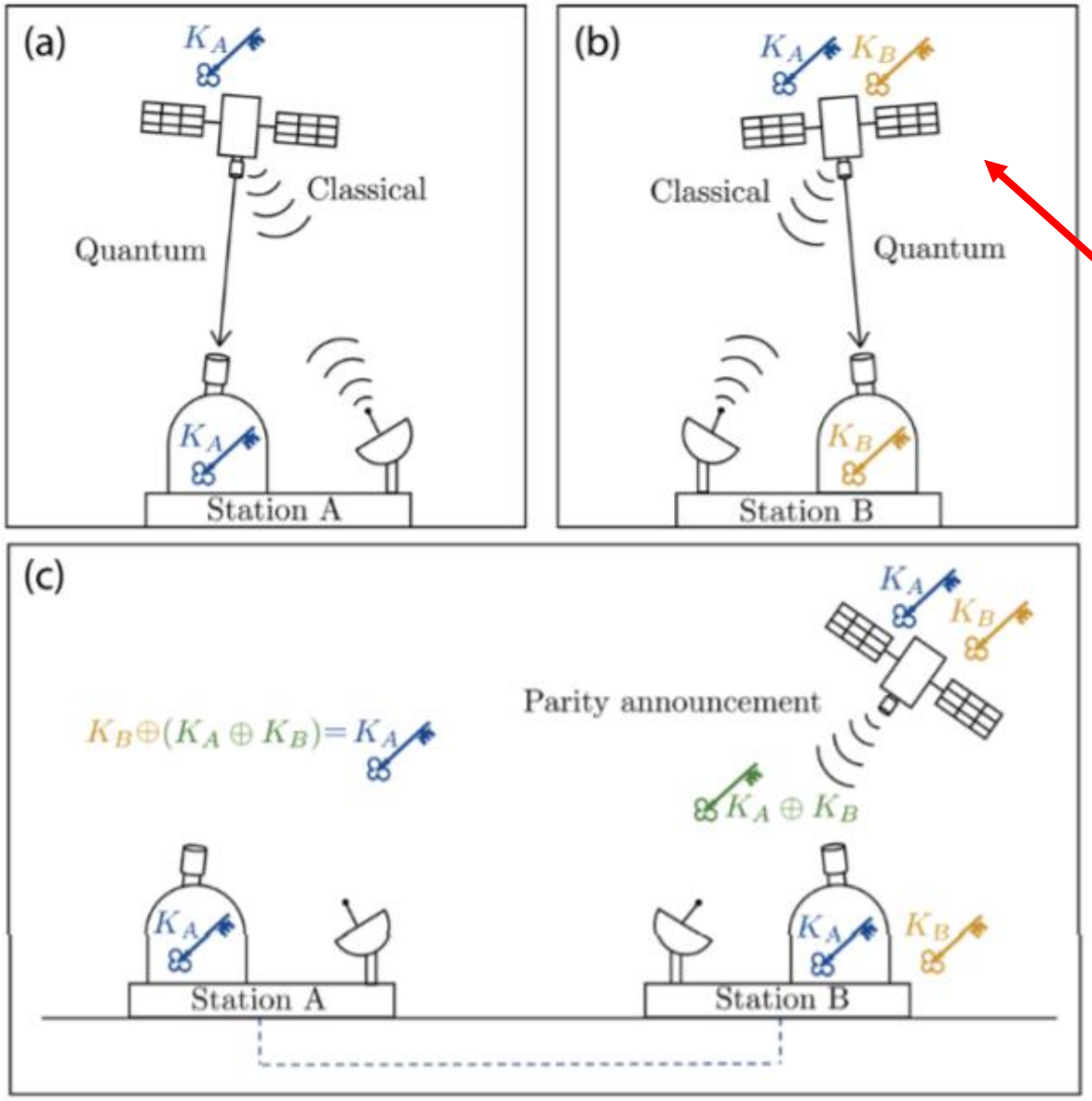


Figure 7.23: The cloning gate

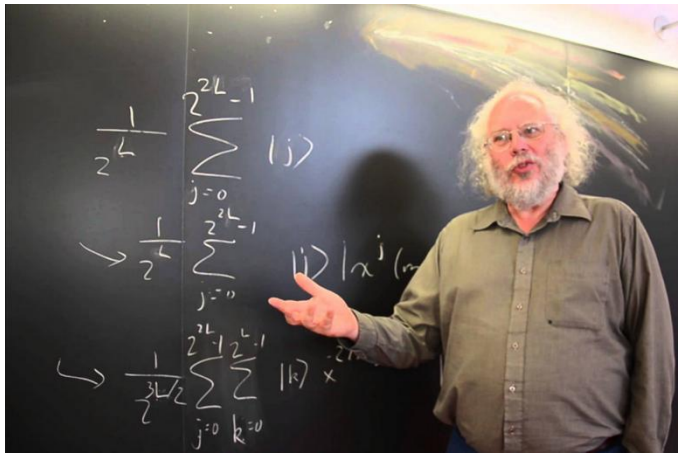
Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law



Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

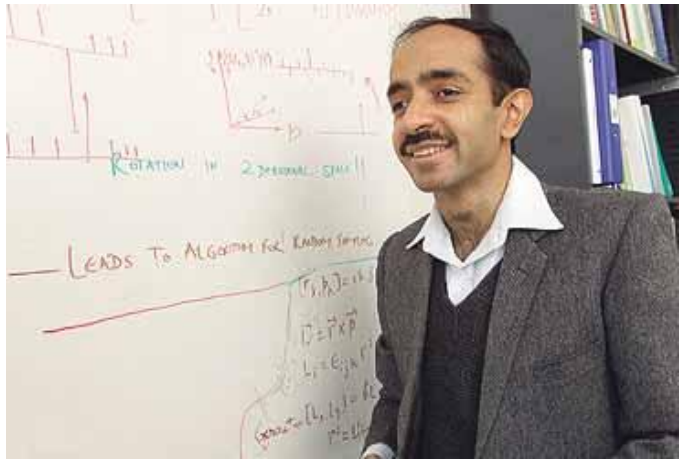
"Give every man thy ear but few thy voice"

BB84 PROTOCOL SATELITE-QKD



Peter Shor can factor integers and compute discrete logarithms in polynomial time

RSA-3072	(encryption)	pre-Shor-sec:	128	post-Shor-sec:	none
RSA-3072	(signature)		128		none
DH-3072	(key exchange)		128		none
DSA-3072	(signature)		128		none
256-bit-ECDH	(key exchange)		128		none
256-bit ECDSA	(signature)		128		none
256-bit ECIES	(encryption)		128		none



Lov Grover can find the needle in a haystack of size N in $O(\sqrt{N})$ steps

AES-128	(block cipher)	pre-Grover-sec:	128	post-Grover-sec:	64
AES-256	(block cipher)		256		128
Salsa20	(stream cipher)		256		128
SHA-256	(hash function)		256*		128*
SHA-3	(hash function)		256*		128*

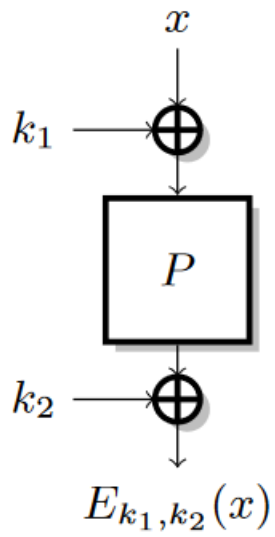
Simon's problem: Given a Boolean function $f : \{0,1\}^n \to \{0,1\}^n$ and the promise that there exists $s \in \{0,1\}^n$ such that for any $(x, y) \in \{0,1\}^n$, $[f(x) = f(y)] \Leftrightarrow [x \oplus y \in \{0^n, s\}]$, the goal is to find s.

Number of queries classical: $\Omega(2^{\frac{n}{2}})$

Number of queries quantum: $O(n)$

EVEN-MANSOUR (BROKEN)

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

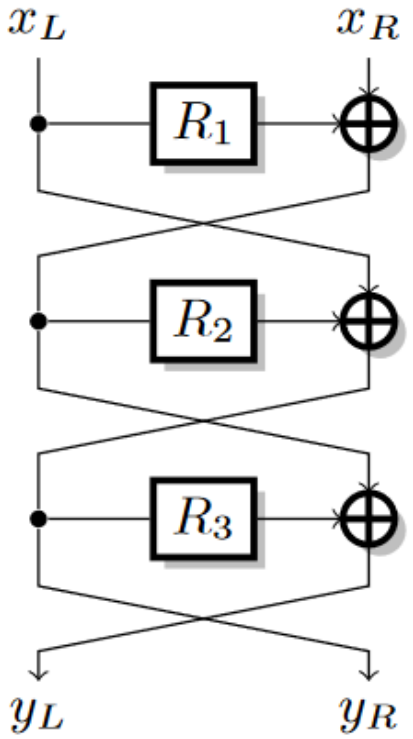


$$E_{k_1,k_2}(x) = P(x \oplus k_1) \oplus k_2$$

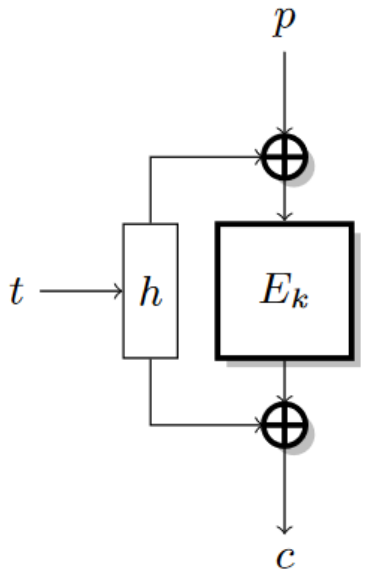
$$f : \{0,1\}^n \to \{0,1\}^n$$
$$x \mapsto E_{k_1,k_2}(x) \oplus P(x) = P(x \oplus k_1) \oplus P(x) \oplus k_2$$

$$\boxed{f(x \oplus k_1) = f(x)}$$

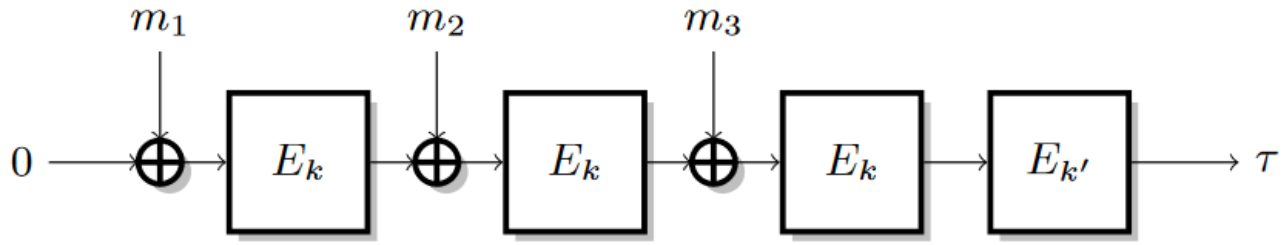
EVEN-MANSOUR (BROKEN)
3-RUNDEN-FEISTEL (BROKEN)



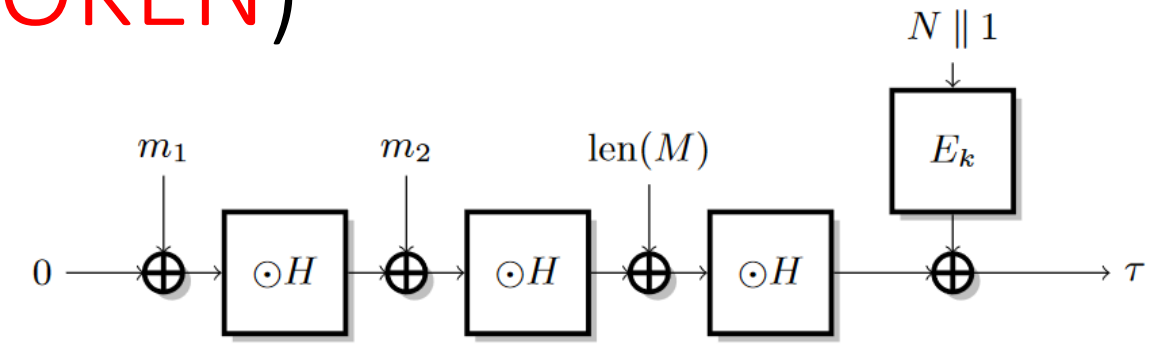
EVEN-MANSOUR (BROKEN)
3-RUNDEN-FEISTEL (BROKEN)
LRW (BROKEN)



EVEN-MANSOUR (BROKEN)
3-RUNDEN-FEISTEL (BROKEN)
LRW (BROKEN)
CBC-MAC (BROKEN)



EVEN-MANSOUR (BROKEN)
3-RUNDEN-FEISTEL (BROKEN)
LRW (BROKEN)
CBC-MAC (BROKEN)
GMAC (BROKEN)



EVEN-MANSOUR (BROKEN)
3-RUNDEN-FEISTEL (BROKEN)
LRW (BROKEN)
CBC-MAC (BROKEN)
GMAC (BROKEN)
PMAC (BROKEN)
GCM (BROKEN)
OCB (BROKEN)
…