

Netzsicherheit 2

Transport Layer Security

Prof. Dr. Jörg Schwenk

www.nds.rub.de

Einordnung und Wiederholung

TCP/IP-Schichtenmodell

7	Anwendungsschicht	Anwendungsschicht	Telnet, <u>FTP</u> , SMTP, <u>HTTP</u> , DNS, <u>IMAP</u>
6	Darstellungsschicht		
5	Sitzungsschicht		
4	Transportschicht	Transportschicht	<u>TCP</u> , UDP
3	Vermittlungsschicht	IP - Schicht	IP
2	Sicherungsschicht	Netzzugangsschicht	Ethernet, Token Ring, PPP, FDDI, IEEE 802.3/802.11
1	Bitübertragungsschicht		

TCP/IP-Schichtenmodell

7	Anwendungsschicht	Anwendungsschicht	Telnet, <u>FTP</u> , SMTP, <u>HTTP</u> , DNS, <u>IMAP</u>
6	Darstellungsschicht		
5	Sitzungsschicht		
TLS			
4	Transportschicht	Transportschicht	<u>TCP</u> , UDP
3	Vermittlungsschicht	IP - Schicht	IP
2	Sicherungsschicht	Netzzugangsschicht	Ethernet, Token Ring, PPP, FDDI, IEEE 802.3/802.11
1	Bitübertragungsschicht		

SSL/TLS: Architektur

Record Layer:

- MAC-then-PAD-then-ENCRYPT,
- transparent für Anwendungsprotokolle wie HTTP

Handshake:

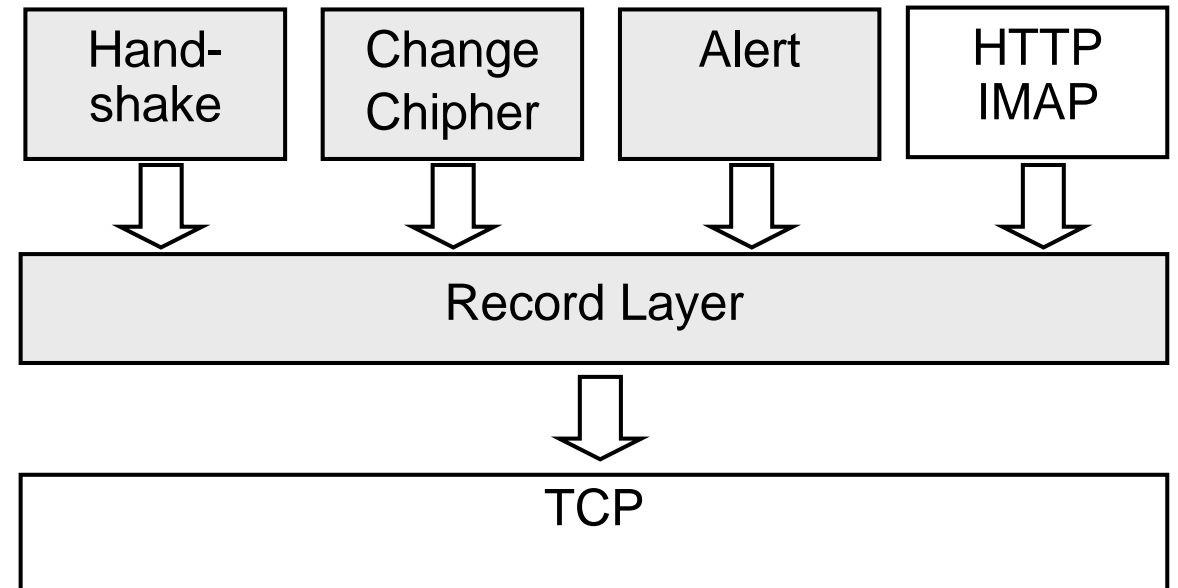
- Authentische Schlüsselvereinbarung

ChangeCipherSpec:

- Umschalten des Record Layer

Alert:

- Fehlermeldungen



Versionen

- SSL 1.0: Nur Netscape-intern
- SSL 2.0: Ab 1995 in Netscape Navigator, unsicher, aber noch 2017 Basis für DROWN
- SSL 3.0: 1996, Basis für alle TLS-Versionen bis 1.2, Historischer RFC 6101
- TLS 1.0 = SSL 3.1: 1999, RFC 2246
- TLS 1.1 = SSL 3.2: 2006, RFC 4346
- TLS 1.2 = SSL 3.3: 2008, RFC 5246
- TLS 1.3: RFC 8446, August 2018

2.1 Das TLS-Ökosystem

TLS 1.0,1.1,1.2

TLS 1.3

MAC-then-PAD-then-Encrypt

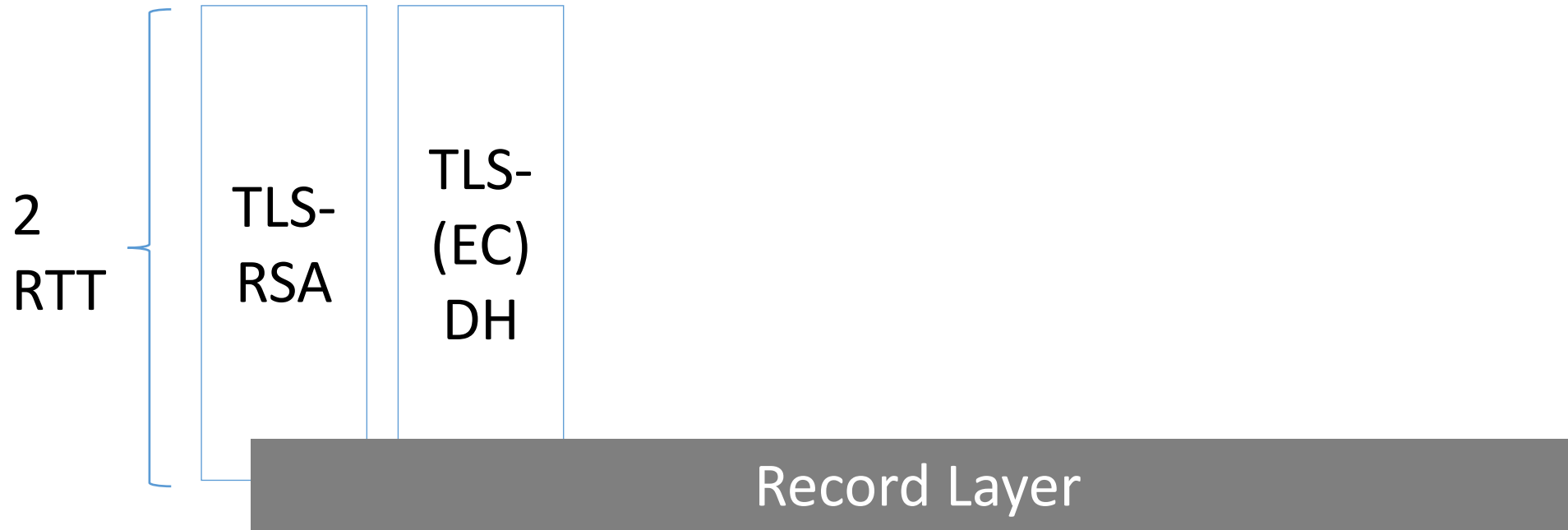
PAD-then-Encrypt-then-MAC

Record Layer

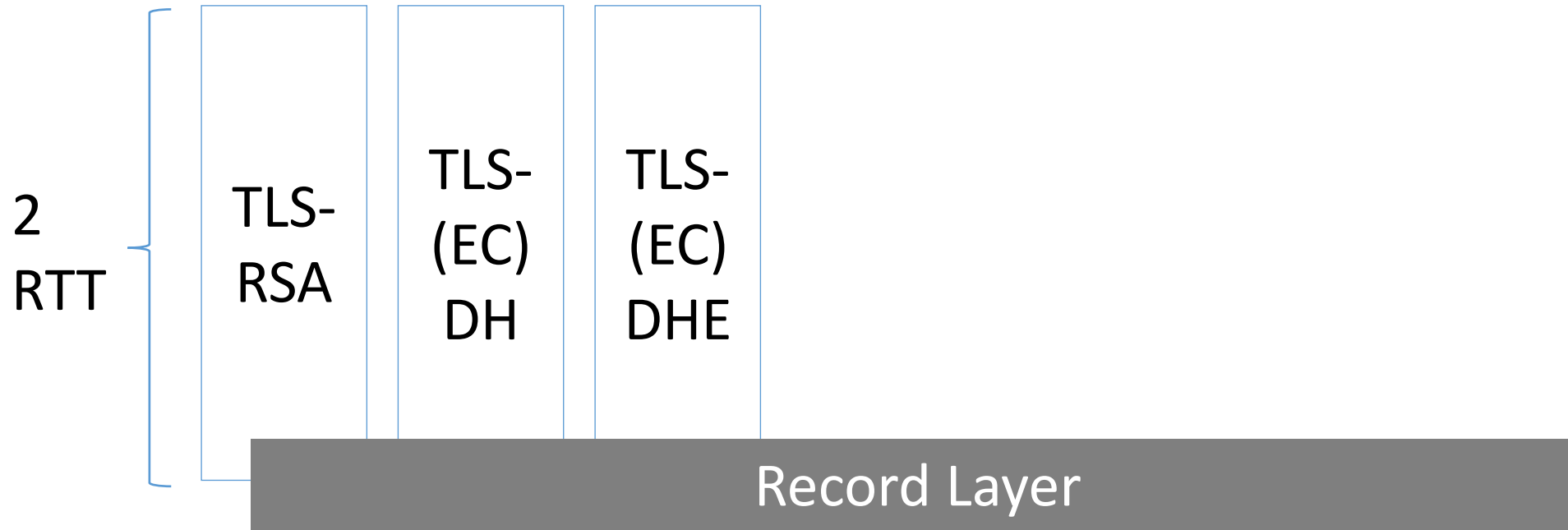
TLS 1.0,1.1,1.2



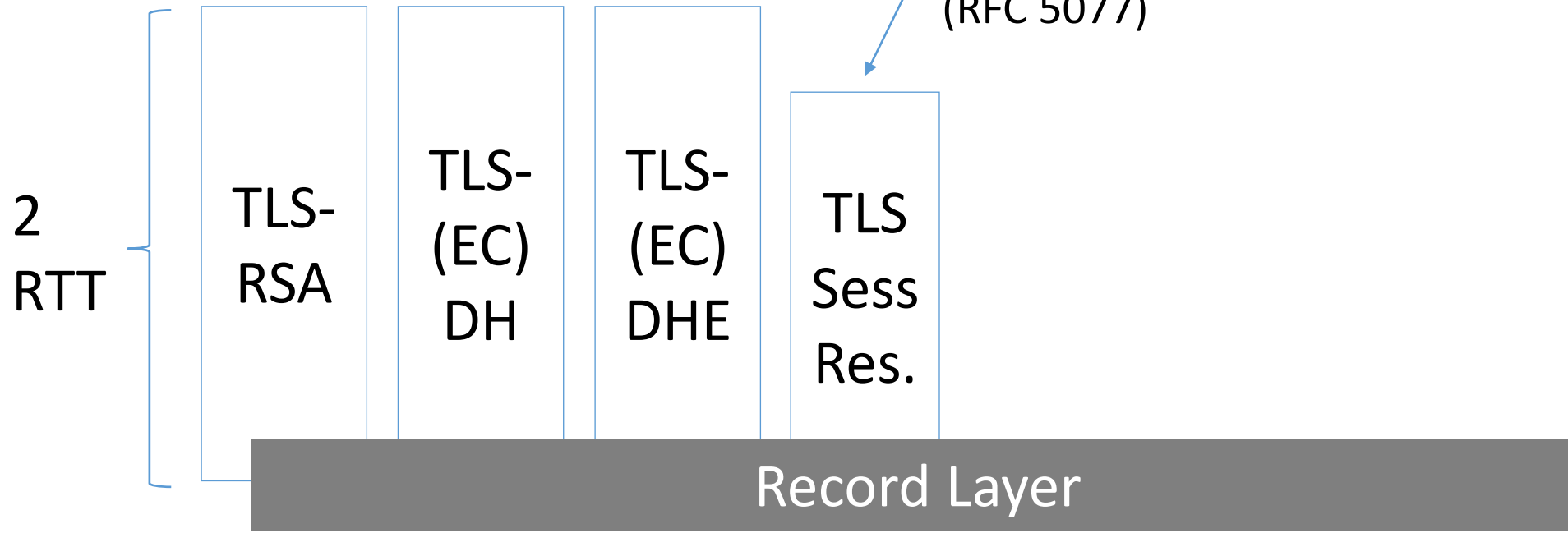
TLS 1.0,1.1,1.2



TLS 1.0,1.1,1.2



TLS 1.0,1.1,1.2



TLS 1.0,1.1,1.2

Session
Tickets
(RFC 5077)

2
RTT

TLS-
RSA

TLS-
(EC)
DH

TLS-
(EC)
DHE

TLS
Sess
Res.

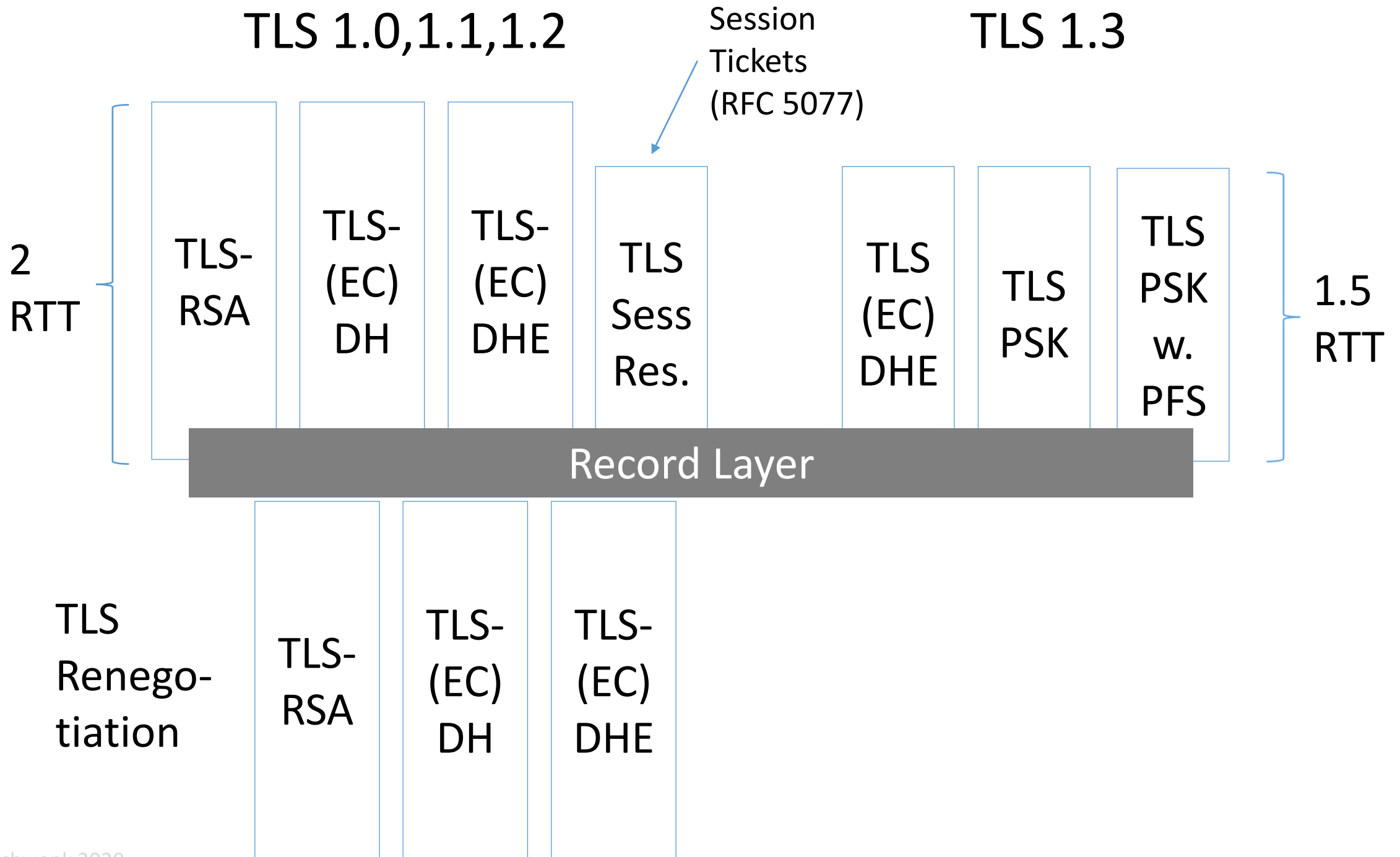
Record Layer

TLS
Renego-
tiation

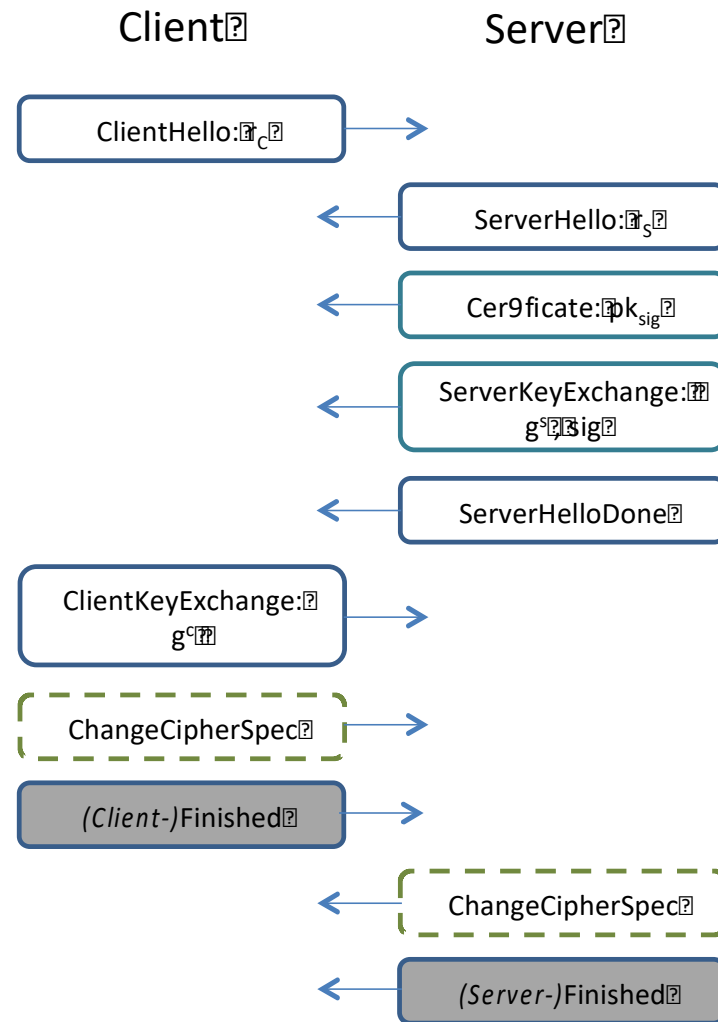
TLS-
RSA

TLS-
(EC)
DH

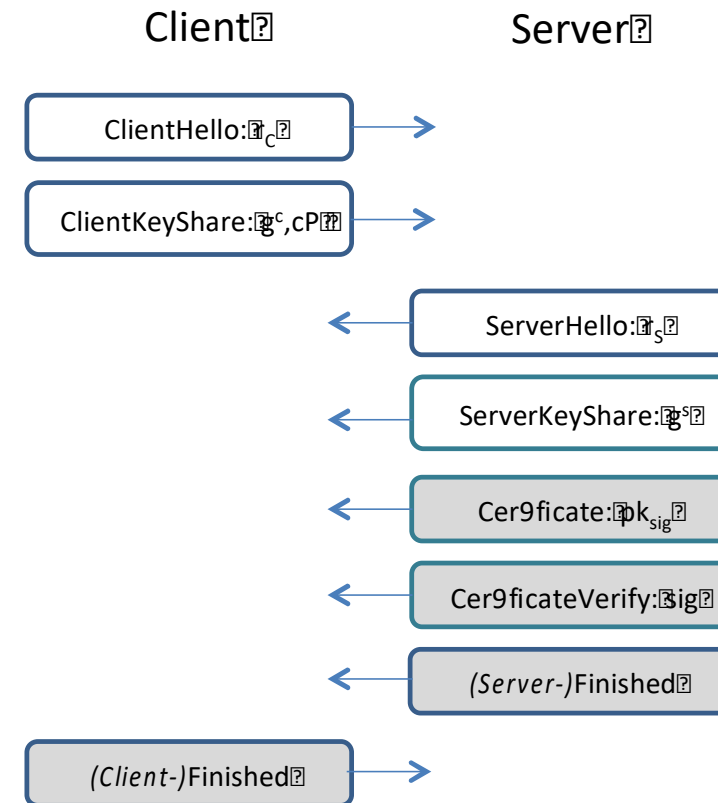
TLS-
(EC)
DHE



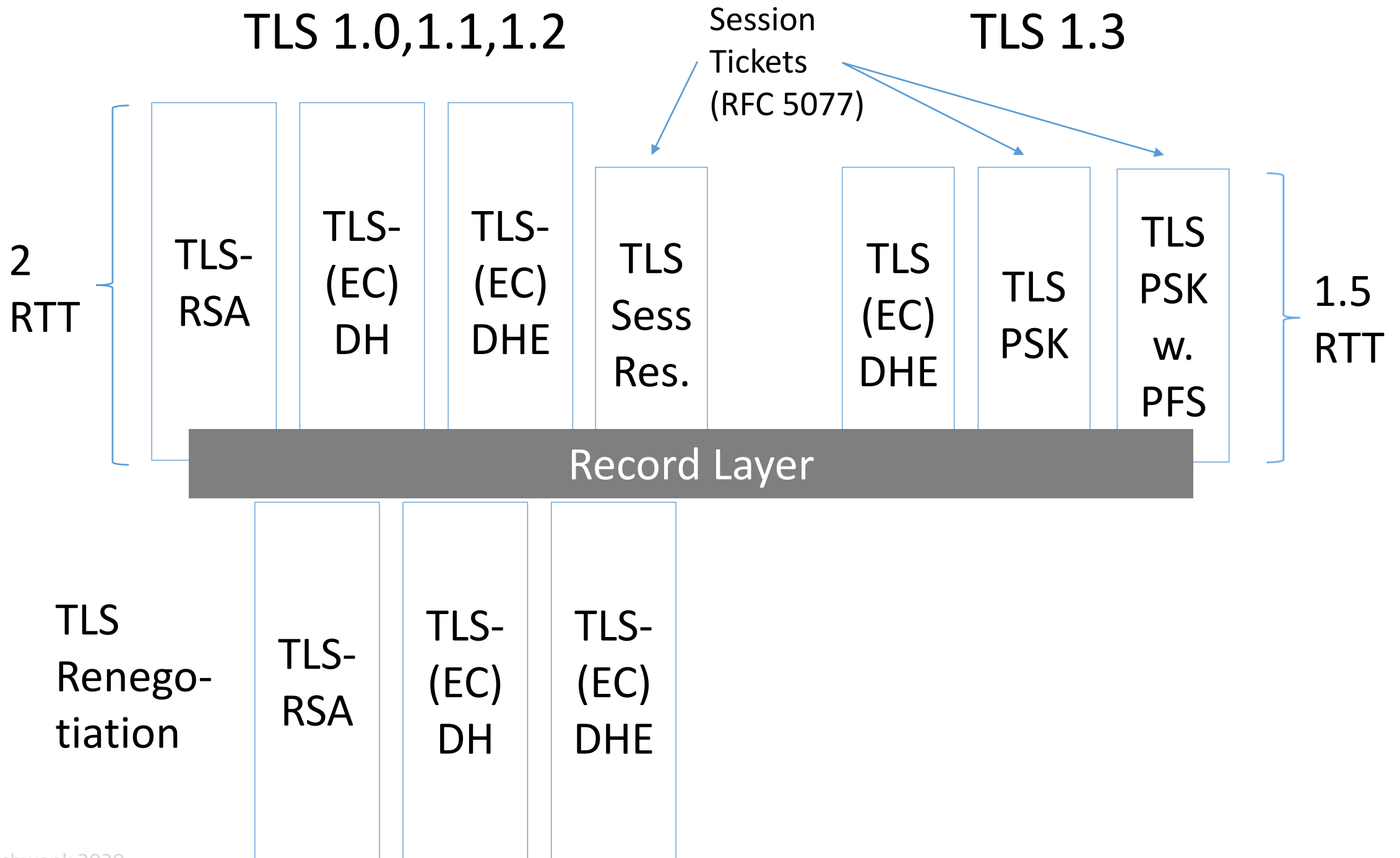
TLS-Ökosystem

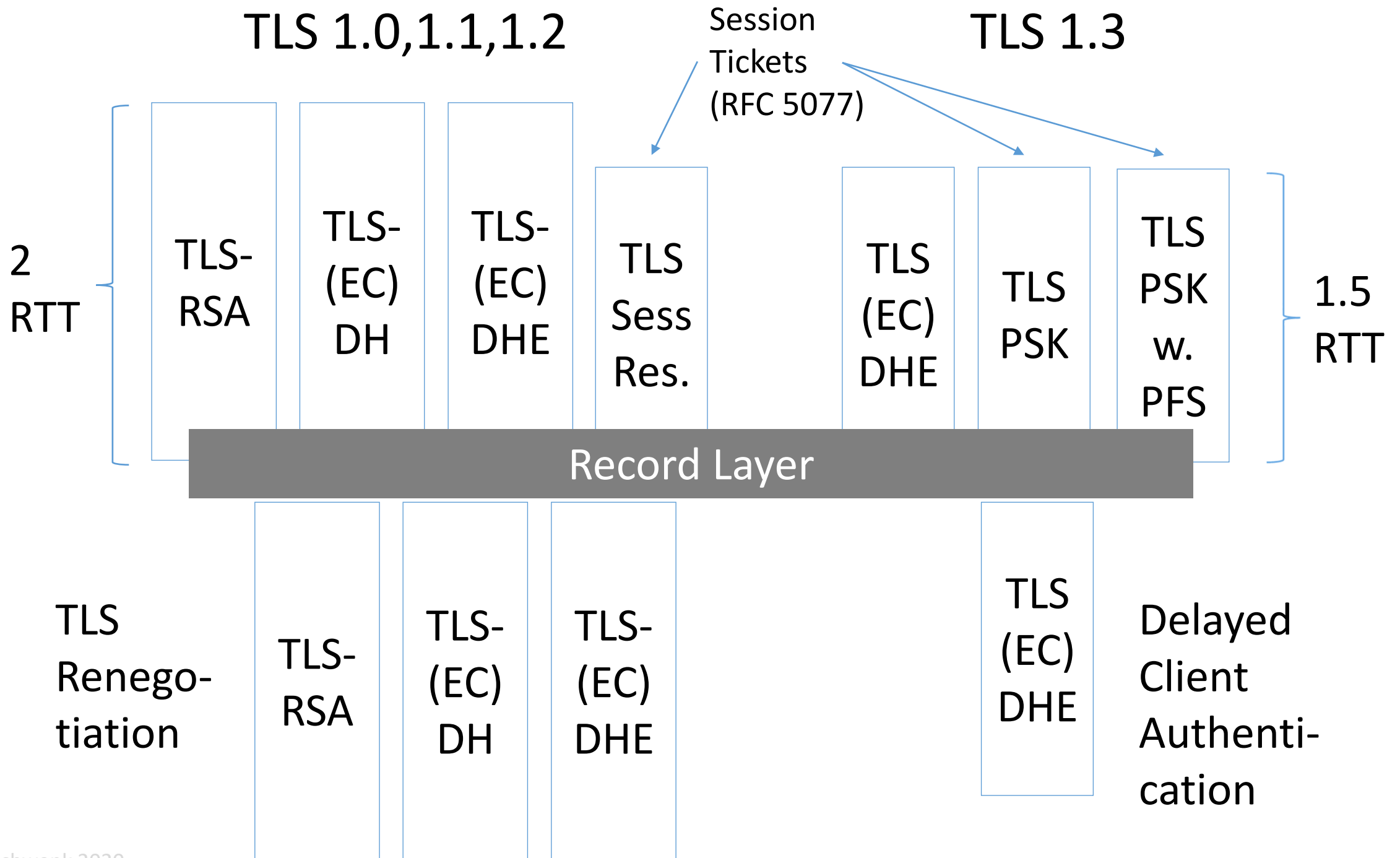


TLS-DHE 1.0, 1.1, 1.2



TLS-DHE 1.3





Aktivierung von TLS

- Neue Protokollnamen in URLs: https, ftps, imaps, ...

Aktivierung von TLS

- Neue Protokollnamen in URLs: https, ftps, imaps, ...
- Well-Known Ports: 443 für https, 993 für imaps, ...

Aktivierung von TLS

- Neue Protokollnamen in URLs: https, ftps, imaps, ...
- Well-Known Ports: 443 für https, 993 für imaps, ...
- Always on: Client wird konfiguriert, immer TLS zu verwenden, z.B. E-Mail-Client; über die ALPN-Extension kann das Anwendungsprotokoll festgelegt werden

Aktivierung von TLS

- Neue Protokollnamen in URLs: https, ftps, imaps, ...
- Well-Known Ports: 443 für https, 993 für imaps, ...
- Always on: Client wird konfiguriert, immer TLS zu verwenden, z.B. E-Mail-Client; über die ALPN-Extension kann das Anwendungsprotokoll festgelegt werden
- HTTP Redirect: Nach Senden einer unverschlüsselten HTTP-Anfrage antwortet der Server mit einem HTTP-Redirect auf eine HTTPS-URL

Aktivierung von TLS

- Neue Protokollnamen in URLs: https, ftps, imaps, ...
- Well-Known Ports: 443 für https, 993 für imaps, ...
- Always on: Client wird konfiguriert, immer TLS zu verwenden, z.B. E-Mail-Client; über die ALPN-Extension kann das Anwendungsprotokoll festgelegt werden
- HTTP Redirect: Nach Senden einer unverschlüsselten HTTP-Anfrage antwortet der Server mit einem HTTP-Redirect auf eine HTTPS-URL
- STARTTLS: Nach Senden einer unverschlüsselten IMAP/POP3/...-Anfrage antwortete der Server mit STARTTLS/AUTH TLS/..., und der Client baut eine TLS-Verbindung zum gleichen Port auf

2.2 TLS Record Layer

HTTP/IMAP/...-Bytestrom		
-------------------------	--	--

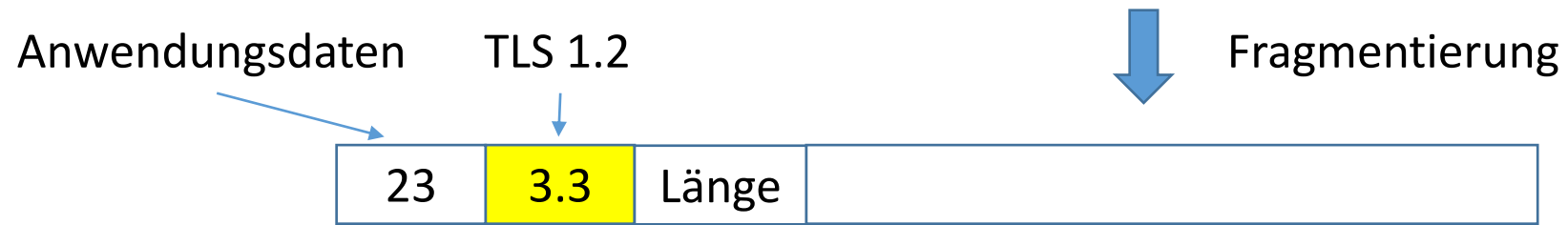


Anwendungsdaten

TLS 1.2

Fragmentierung





```
struct {  
    ContentType type;  
    ProtocolVersion version;  
    uint16 length;  
    opaque fragment[TLSPplaintext.length];  
} TLSPplaintext;
```

HTTP/IMAP/...-Bytestrom

Anwendungsdaten

TLS 1.2

Fragmentierung

23

3.3

Länge

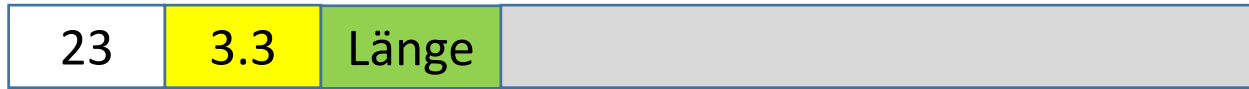
```
struct {  
    ContentType type;  
    ProtocolVersion version;  
    uint16 length;  
    opaque fragment[TLSPplaintext.length];  
} TLSPplaintext;
```

HTTP/IMAP/...-Bytestrom

Anwendungsdaten

TLS 1.2

Fragmentierung



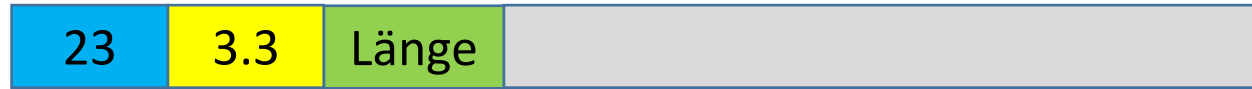
```
struct {  
    ContentType type;  
    ProtocolVersion version;  
    uint16 length;  
    opaque fragment[TLSPplaintext.length];  
} TLSPplaintext;
```

HTTP/IMAP/...-Bytestrom

Anwendungsdaten

TLS 1.2

Fragmentierung



```
struct {  
    ContentType type;  
    ProtocolVersion version;  
    uint16 length;  
    opaque fragment[TLSPplaintext.length];  
} TLSPplaintext;
```

```
enum { change_cipher_spec(20), alert(21), handshake(22),  
        application_data(23), (255)  
} ContentType;
```



Anwendungsdaten

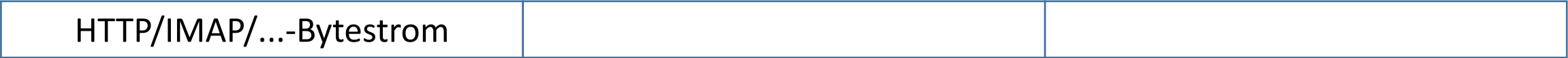
TLS 1.2

Fragmentierung



optional: Kompression





Anwendungsdaten

TLS 1.2

Fragmentierung



optional: Kompression



MAC



Berechnung des MAC

$$\text{MAC} := \text{HMAC}_{\text{MAC_write_key}}(\text{SQN} | \text{Type} | \text{Version} | \text{Length} | \text{Data})$$

Berechnung des MAC

$\text{MAC} := \text{HMAC}_{\text{MAC_write_key}}(\text{SQN} | \text{Type} | \text{Version} | \text{Length} | \text{Data})$



Aus dem im TLS-Handshake ausgehandelten
MasterSecret abgeleitet

Berechnung des MAC

$\text{MAC} := \text{HMAC}_{\text{MAC_write_key}}(\text{SQN} | \text{Type} | \text{Version} | \text{Length} | \text{Data})$

RECORD Header



A blue arrow points from the 'RECORD Header' box to the 'Type | Version | Length' portion of the MAC calculation input string.

Berechnung des MAC

$\text{MAC} := \text{HMAC}_{\text{MAC_write_key}}(\text{SQN} | \text{Type} | \text{Version} | \text{Length} | \text{Data})$

Klartext

A blue arrow points from the box labeled 'Klartext' to the 'Data' field within the parentheses of the HMAC function in the formula above.

Berechnung des MAC

$\text{MAC} := \text{HMAC}_{\text{MAC_write_key}}(\text{SQN} \mid \text{Type} \mid \text{Version} \mid \text{Length} \mid \text{Data})$



Implizite Sequenznummer, beginnt bei 0

HTTP/IMAP/...-Bytestrom

Anwendungsdaten

TLS 1.2

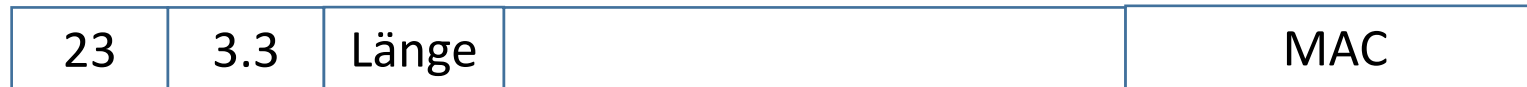
Fragmentierung



optional: Kompression



MAC



PAD

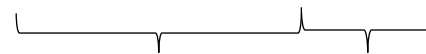


Padding in TLS

Wenn n Byte Padding notwendig sind, wird n mal der Wert $n-1$ angefügt

- Beispiel: AES mit 16 Byte Blocklänge, 61 Byte komprimierter Plaintext, SHA-256 als Hashfunktion

- Plaintext + MAC: $61 + 32 = 93$ Byte
- Vielfaches der Blocklänge: $6 * 16 = 96$ Byte
- Padding: 0x02 0x02 0x02



Paddingbytes

Länge des Padding:

Anzahl der Paddingbytes ohne das Längenbyte

HTTP/IMAP/...-Bytestrom

Anwendungsdaten

TLS 1.2

Fragmentierung



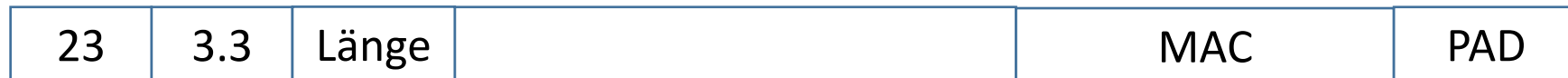
optional: Kompression



MAC



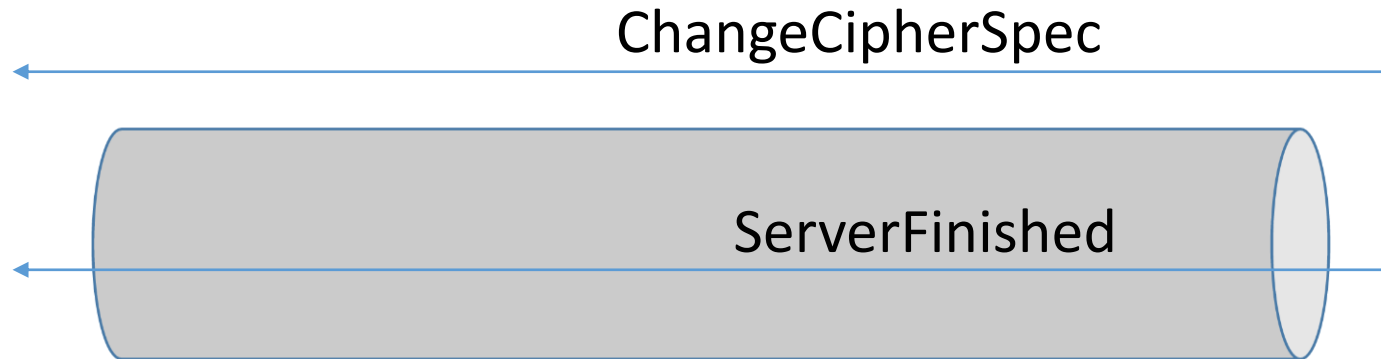
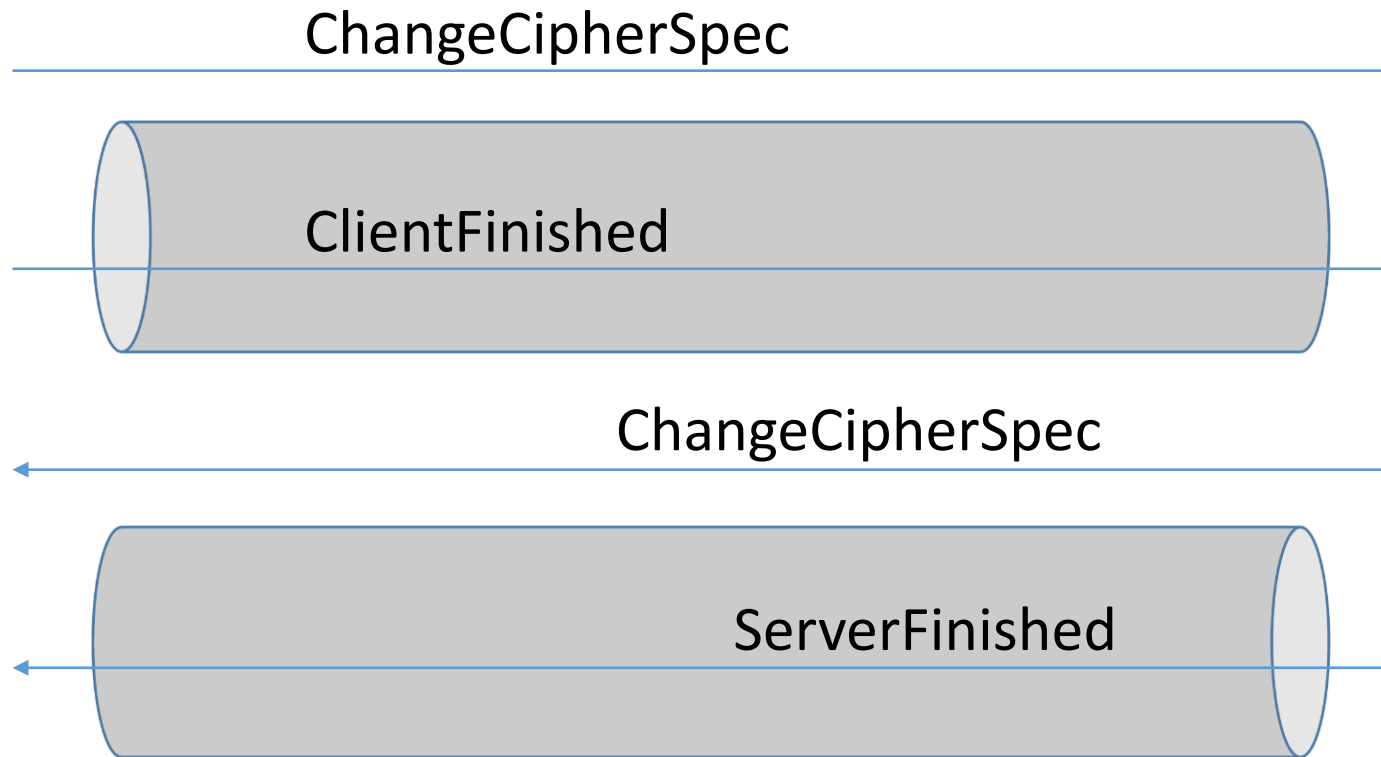
PAD



ENCRYPT



Aktivierung des Record Layer



TLS Record Layer: Exkurs Authenticated Encryption

Exkurs: Authenticated Encryption

[Mihir Bellare](#), [Chanathip Namprempre](#):

Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. [J. Cryptology 21\(4\)](#): 469-491 (2008)

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

https://en.wikipedia.org/wiki/Authenticated_encryption

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC then encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	insecure	secure	insecure

SSH

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt and MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt then MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

TLS 1.2

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

IPsec,
TLS 1.3

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

TLS 1.2

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

Exkurs: Authenticated Encryption

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	insecure	insecure	secure	insecure

Composition Method	Privacy			Integrity	
	IND-CPA	IND-CCA	NM-CPA	INT-PTXT	INT-CTXT
Encrypt-and-MAC	insecure	insecure	insecure	secure	insecure
MAC-then-encrypt	secure	insecure	insecure	secure	insecure
Encrypt-then-MAC	secure	secure	secure	secure	secure

TLS 1.3

Figure 2: Summary of security results for the composite authenticated encryption schemes. The given encryption scheme is assumed to be IND-CPA for both tables while the given MAC is assumed to be weakly unforgeable for the top table and strongly unforgeable for the bottom table.

INT-PTXT

Integrität des Plaintext

- Kein Angreifer darf in der Lage sein, einen 'gültigen' Chiffretext für einen neuen Plaintext zu erstellen

Exkurs: Authenticated Encryption

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{M\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $M \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{C\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $C \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

Exkurs: Authenticated Encryption

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{M\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $M \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{C\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $C \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

Exkurs: Authenticated Encryption

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{M\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $M \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{C\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $C \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

INT-PTXT

Integrität des Chiffretexts

- Kein Angreifer darf in der Lage sein, einen neuen, 'gültigen' Chiffretext zu erstellen

Exkurs: Authenticated Encryption

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{M\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $M \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{C\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $C \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

Exkurs: Authenticated Encryption

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{M\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $M \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{C\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $C \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

Exkurs: Authenticated Encryption

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{M\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $M \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

proc Initialize

$K \xleftarrow{\$} K ; S \leftarrow \emptyset$

proc Enc(M)

$C \xleftarrow{\$} E(K, M) ; S \leftarrow S \cup \{C\} ; \text{Return } C$

proc VF(C)

$M \leftarrow D(K, C)$

If $M \neq \perp$ and $C \notin S$ then win \leftarrow true

Return $(M \neq \perp)$

proc Finalize

Return win

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

Exkurs: Authenticated Encryption

```
proc Initialize  
 $K \xleftarrow{\$} K ; S \leftarrow \emptyset$   
  
proc Enc(M)  
 $C \xleftarrow{\$} E(K, M)$   $S \leftarrow S \cup \{M\}$  Return C  
  
proc VF(C)  
 $M \leftarrow D(K, C)$   
If  $M \neq \perp$  and  $M \notin S$  then win  $\leftarrow$  true  
Return ( $M \neq \perp$ )  
  
proc Finalize  
Return win
```

```
proc Initialize  
 $K \xleftarrow{\$} K ; S \leftarrow \emptyset$   
  
proc Enc(M)  
 $C \xleftarrow{\$} E(K, M)$   $S \leftarrow S \cup \{C\}$  ; Return C  
  
proc VF(C)  
 $M \leftarrow D(K, C)$   
If  $M \neq \perp$  and  $C \notin S$  then win  $\leftarrow$  true  
Return ( $M \neq \perp$ )  
  
proc Finalize  
Return win
```

Figure 5: Game INT-PTXT_{SE} (left) and Game INT-CTXT_{SE} (right) where SE = (K, E, D).

IND-CPA

Standardanforderung an Public-Key-Verschlüsselung

IND-CPA

Standardanforderung an Public-Key-Verschlüsselung

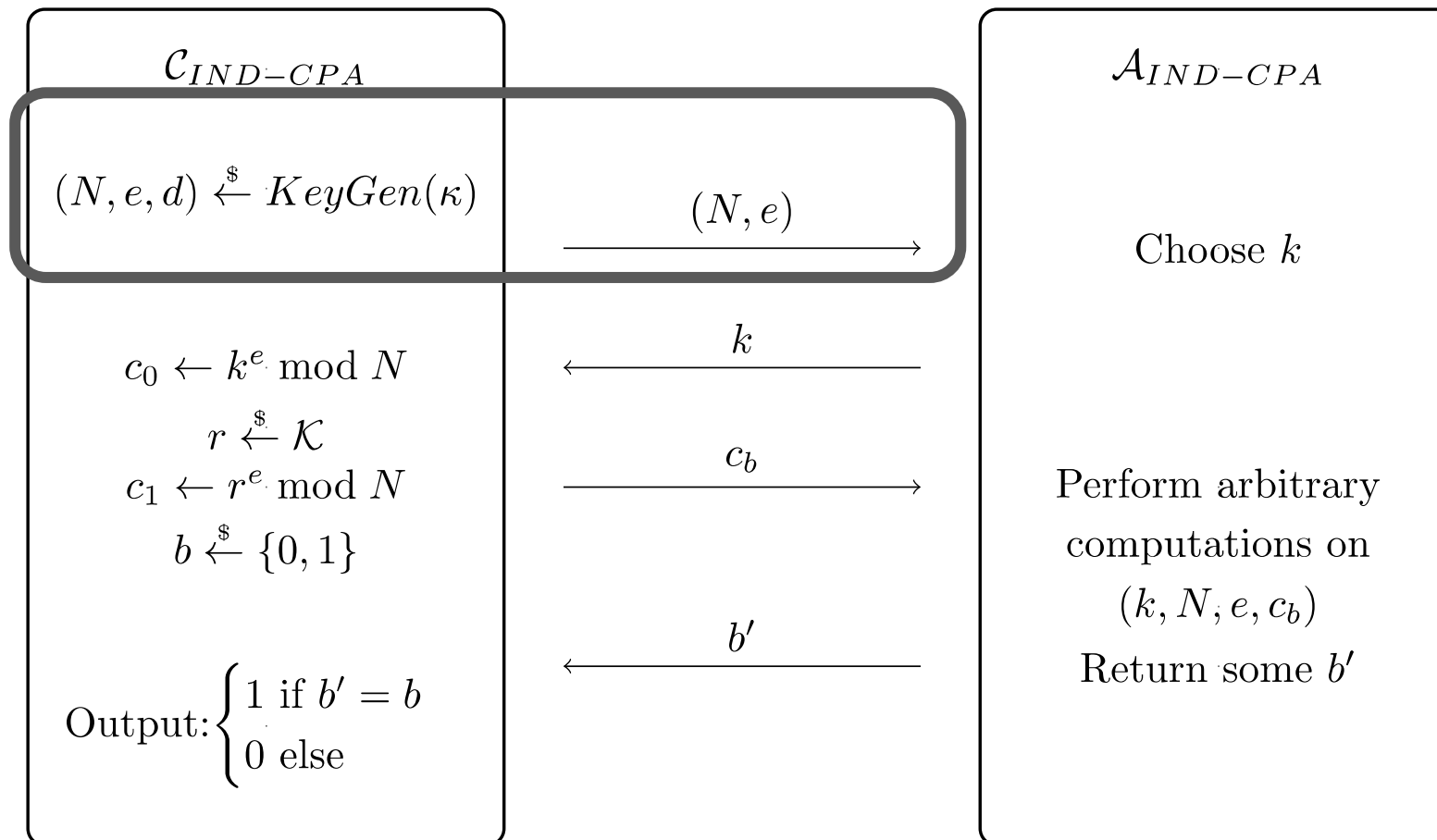
- Angreifer kann selbst (Public Key) oder über ein Orakel (symmetrisch) Klartexte verschlüsseln

IND-CPA

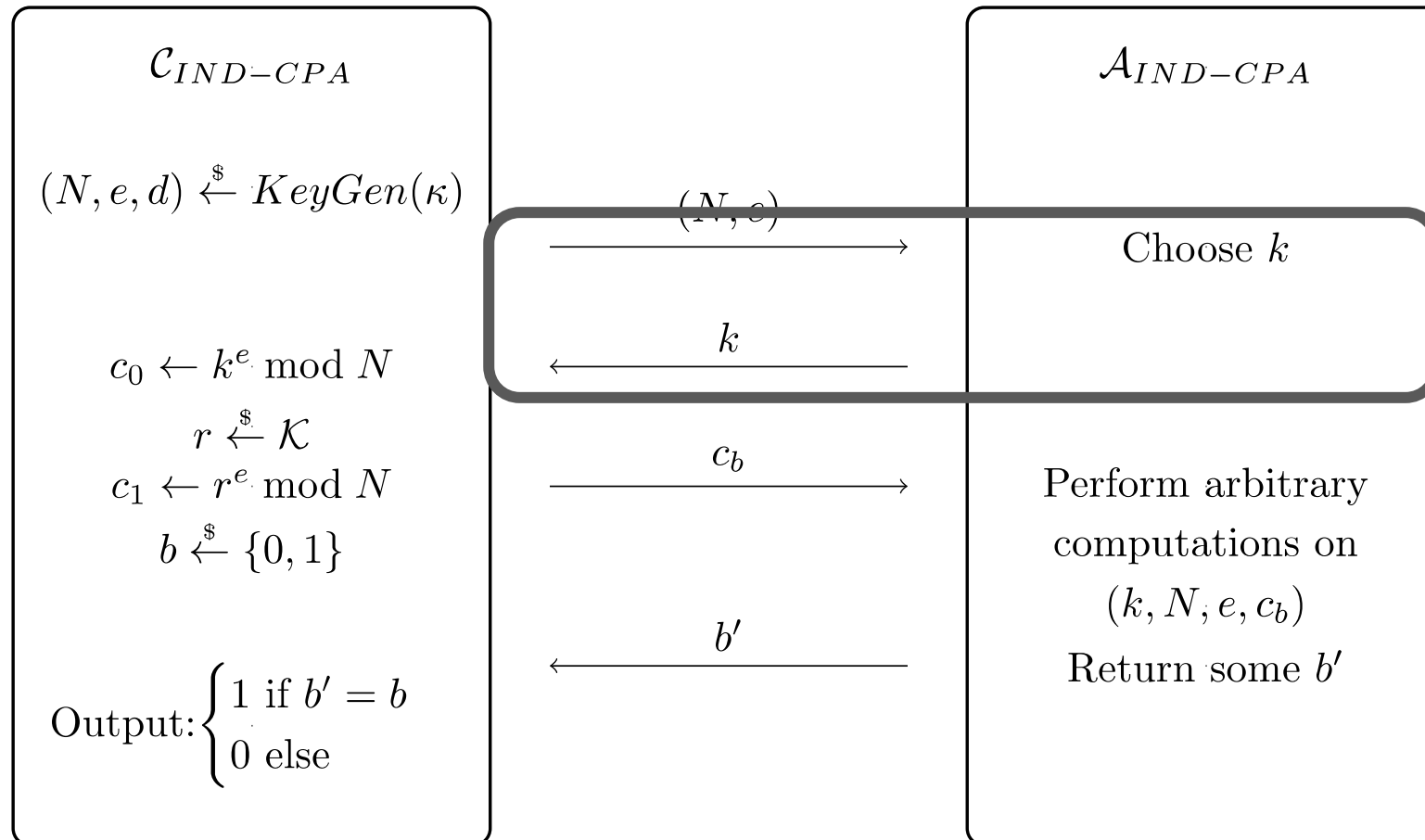
Standardanforderung an Public-Key-Verschlüsselung

- Angreifer kann selbst (Public Key) oder über ein Orakel (symmetrisch) Klartexte verschlüsseln
- Trotzdem darf er nicht in der Lage sein zu entscheiden, ob seine vorgegebene Challenge-Nachricht oder ein Zufallswert verschlüsselt wurden

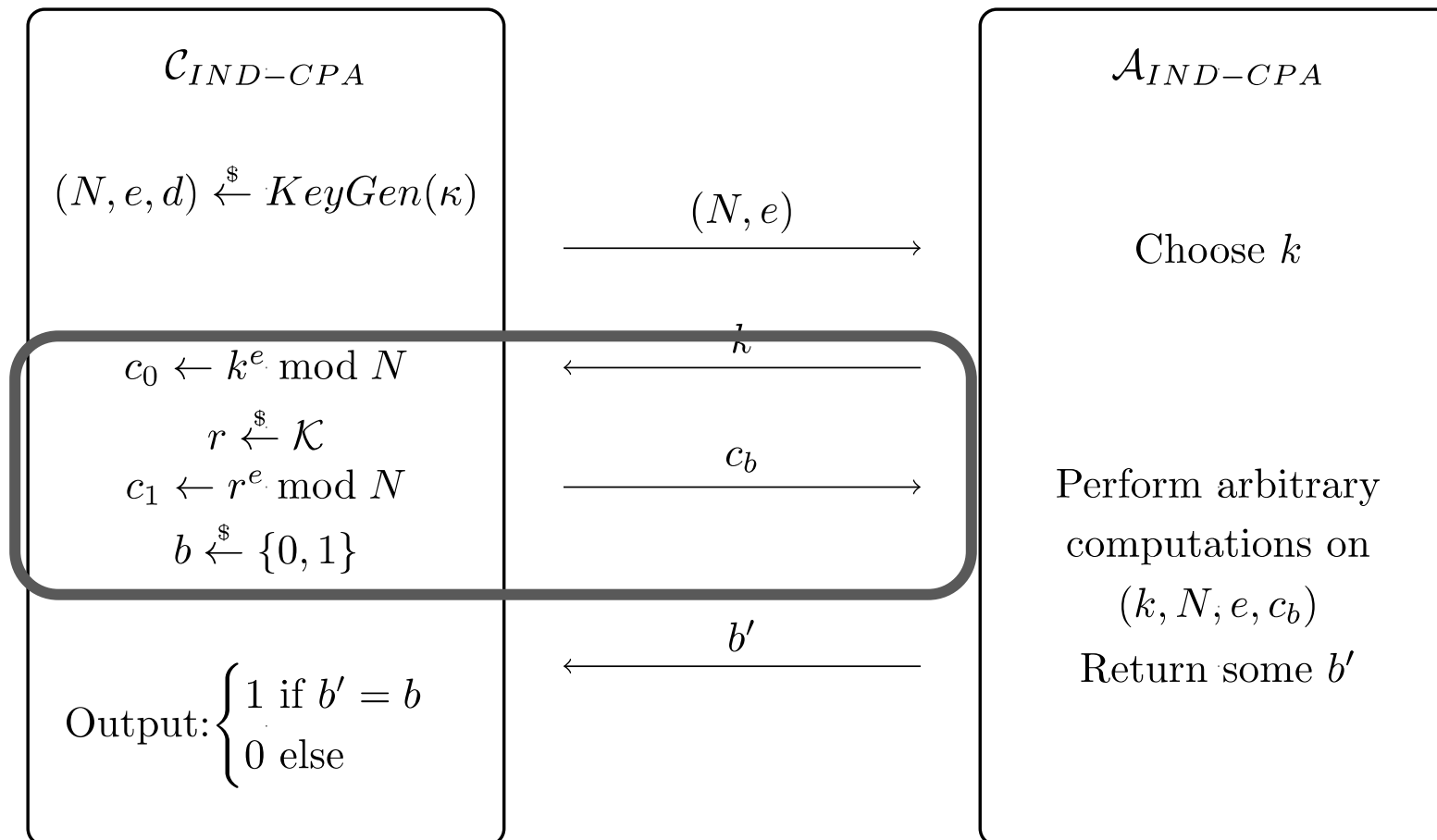
Exkurs: Authenticated Encryption



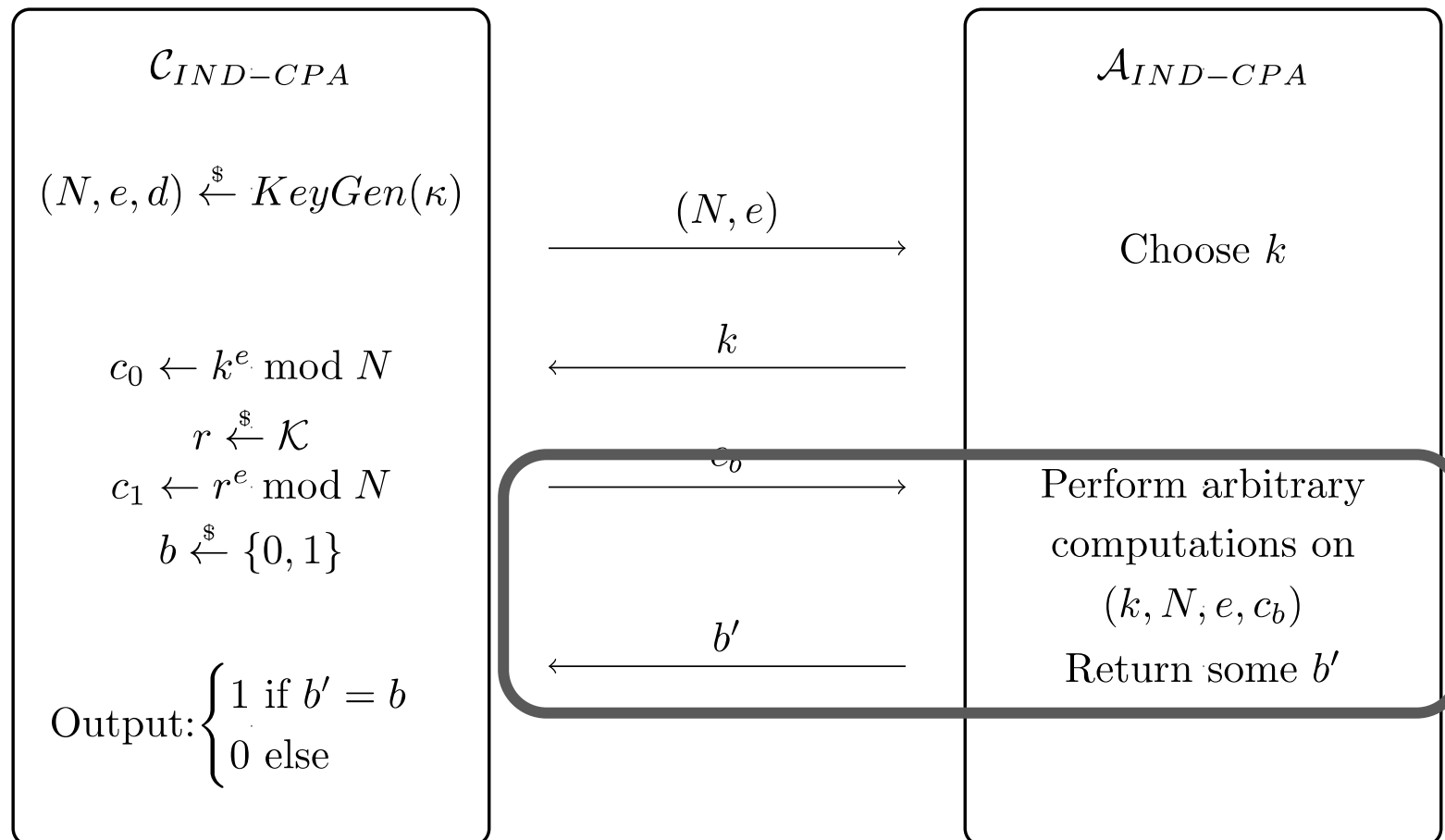
Exkurs: Authenticated Encryption



Exkurs: Authenticated Encryption



Exkurs: Authenticated Encryption



IND-CCA

IND-CPA plus *Entschlüsselungsortakel*

IND-CCA

IND-CPA plus *Entschlüsselungsortakel*

- Angreifer kann selbst (Public Key) oder über ein Orakel (symmetrisch) Klartexte verschlüsseln

IND-CCA

IND-CPA plus *Entschlüsselungsortakel*

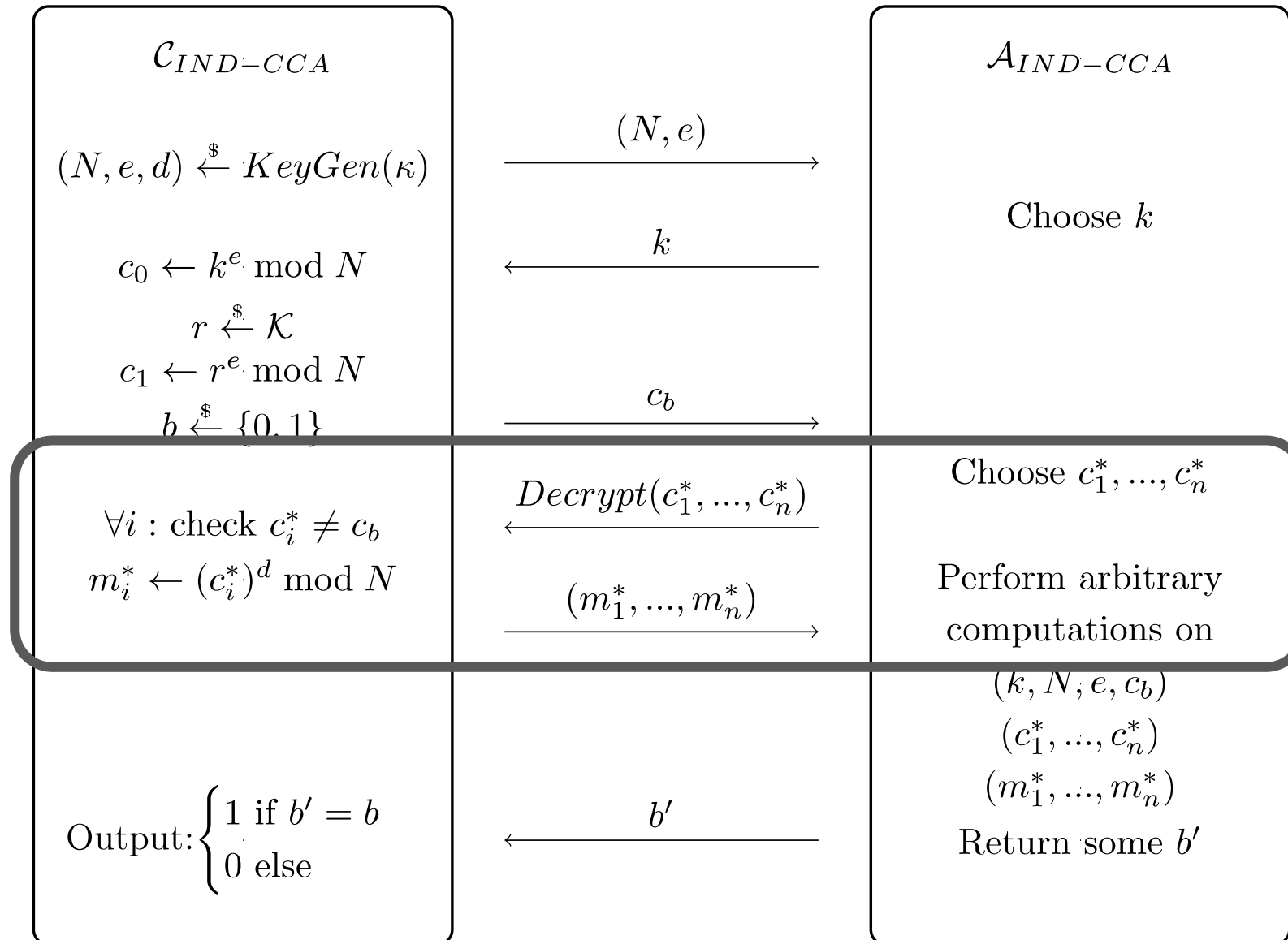
- Angreifer kann selbst (Public Key) oder über ein Orakel (symmetrisch) Klartexte verschlüsseln
- *Angreifer kann sich über ein Orakel beliebige Chiffretexte entschlüsseln lassen*

IND-CCA

IND-CPA plus *Entschlüsselungsortakel*

- Angreifer kann selbst (Public Key) oder über ein Orakel (symmetrisch) Klartexte verschlüsseln
- *Angreifer kann sich über ein Orakel beliebige Chiffretexte entschlüsseln lassen*
- Trotzdem darf er nicht in der Lage sein zu entscheiden, ob seine vorgegebene Challenge-Nachricht oder ein Zufallswert verschlüsselt wurden

Exkurs: Authenticated Encryption



2.3 TLS Handshake

2.3.1 Übersicht

Ciphersuite-Familien

TLS-RSA

mandatory in TLS 1.0 and 1.2

TLS-(EC)DHE

mandatory in TLS 1.1

TLS-(EC)DH

Ciphersuite-Familien

TLS-RSA

TLS-(EC)DHE

TLS-(EC)DH

Negotiation

ClientHello →
← ServerHello
← Certificate

Ciphersuite-Familien

TLS-RSA

TLS-(EC)DHE

TLS-(EC)DH

Negotiation

ClientHello →
← ServerHello
← Certificate

Key Exchange

$pms^e \bmod N \rightarrow$

$pms = pms$

Ciphersuite-Familien

TLS-RSA

TLS-(EC)DHE

TLS-(EC)DH

Negotiation

ClientHello →
← ServerHello
← Certificate

Key Exchange

$\text{pms}^e \bmod N \rightarrow$

Key Exchange

← g^y
 $g^x \rightarrow$

$\text{pms} = \text{pms}$

$\text{pms} = g^{xy}$

Ciphersuite-Familien

TLS-RSA

TLS-(EC)DHE

TLS-(EC)DH

Negotiation

ClientHello →
← ServerHello
← Certificate

Key Exchange

$\text{pms}^e \bmod N \rightarrow$

Key Exchange

← g^y
 $g^x \rightarrow$

Key Exchange

$g^x \rightarrow$

$\text{pms} = \text{pms}$

$\text{pms} = g^{xy}$

$\text{pms} = \text{Certificate}^x$

Ciphersuite-Familien

TLS-RSA

TLS-(EC)DHE

TLS-(EC)DH

Negotiation

ClientHello →
← ServerHello
← Certificate

Key Exchange

$\text{pms}^e \bmod N \rightarrow$

Key Exchange

$\leftarrow g^y$
 $g^x \rightarrow$

Key Exchange

$g^x \rightarrow$

Key Confirmation/Authentication

FINISHED(MAC) →
← FINISHED(MAC)

$\text{pms} = \text{pms}$

$\text{pms} = g^{xy}$

$\text{pms} = \text{Certificate}^x$

TLS Handshake: TLS-RSA

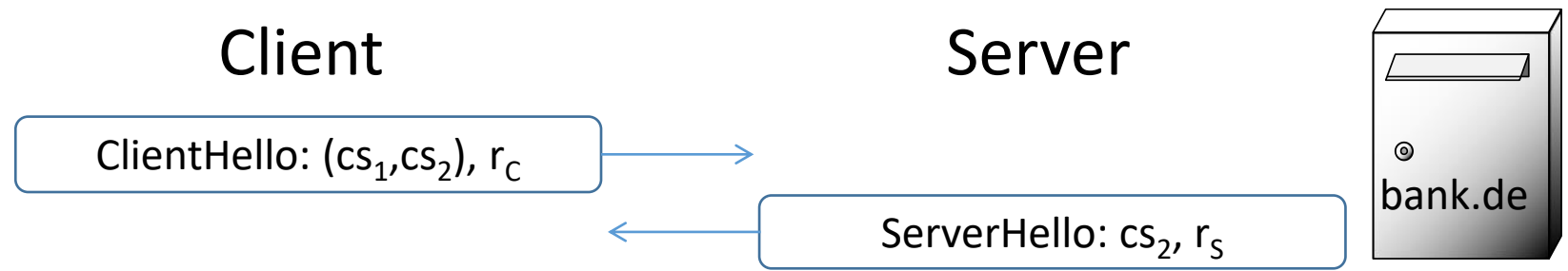
Client

ClientHello: $(cs_1, cs_2), r_c$

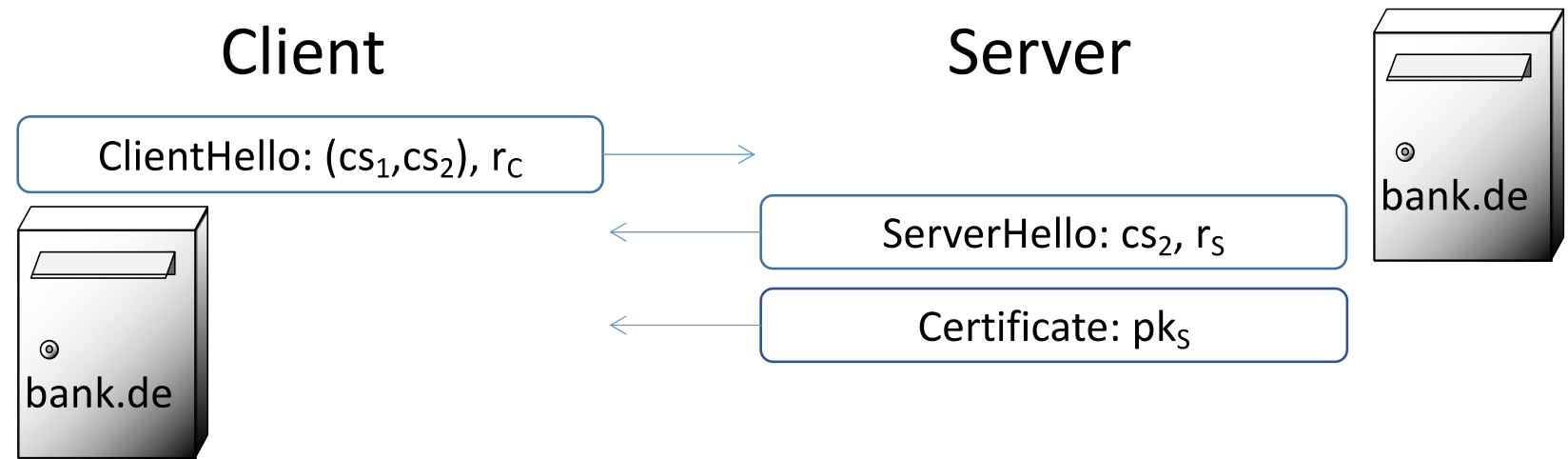
Server



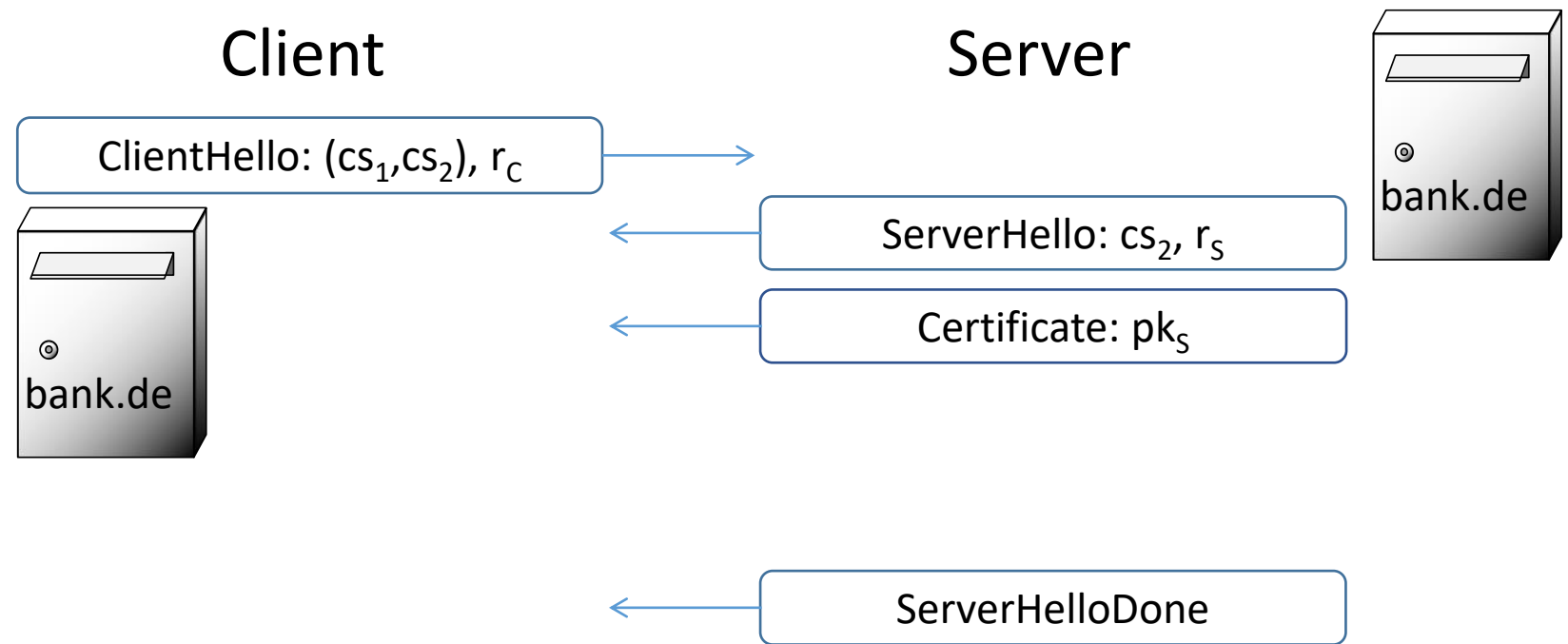
TLS Handshake: TLS-RSA



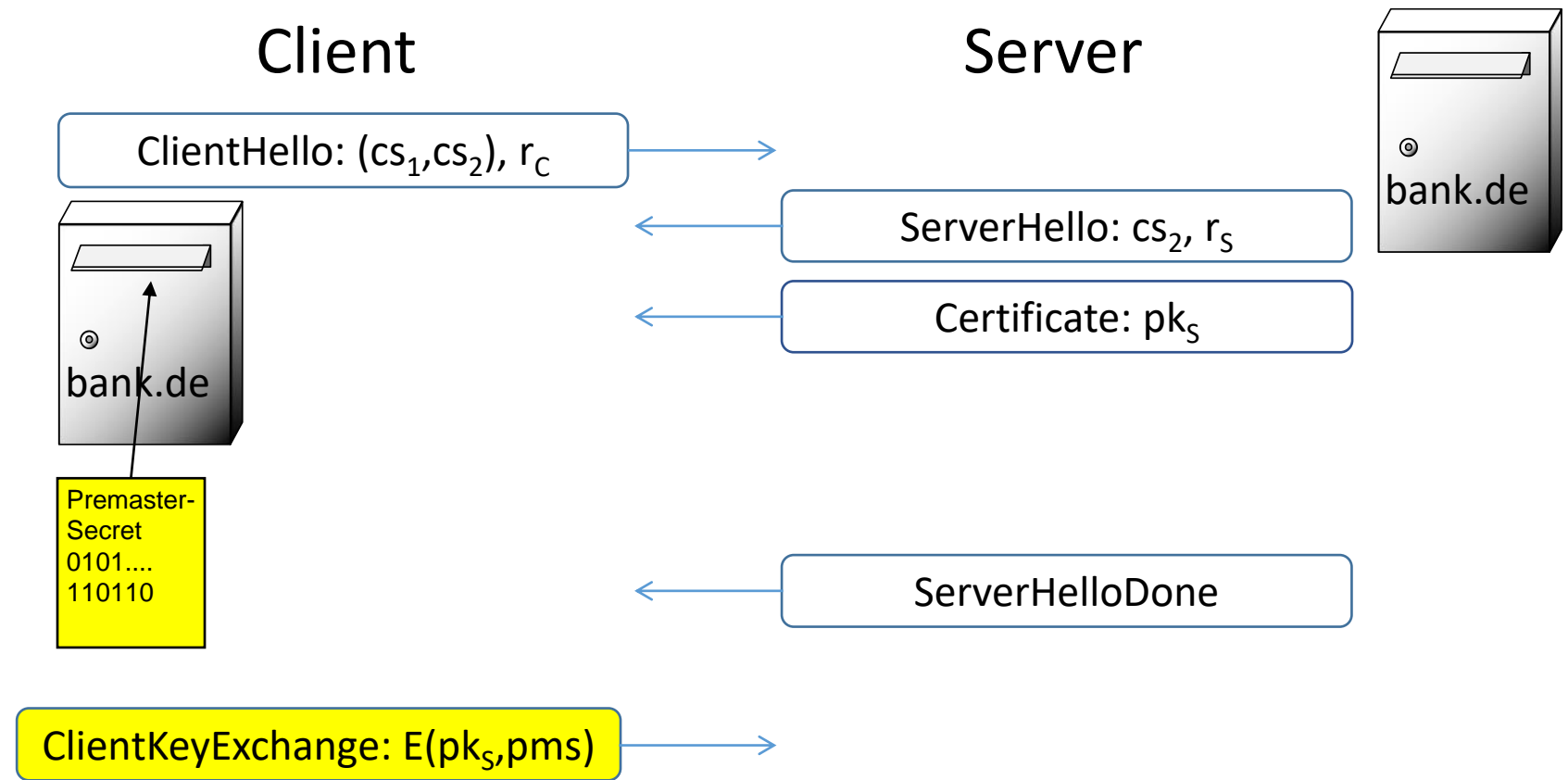
TLS Handshake: TLS-RSA



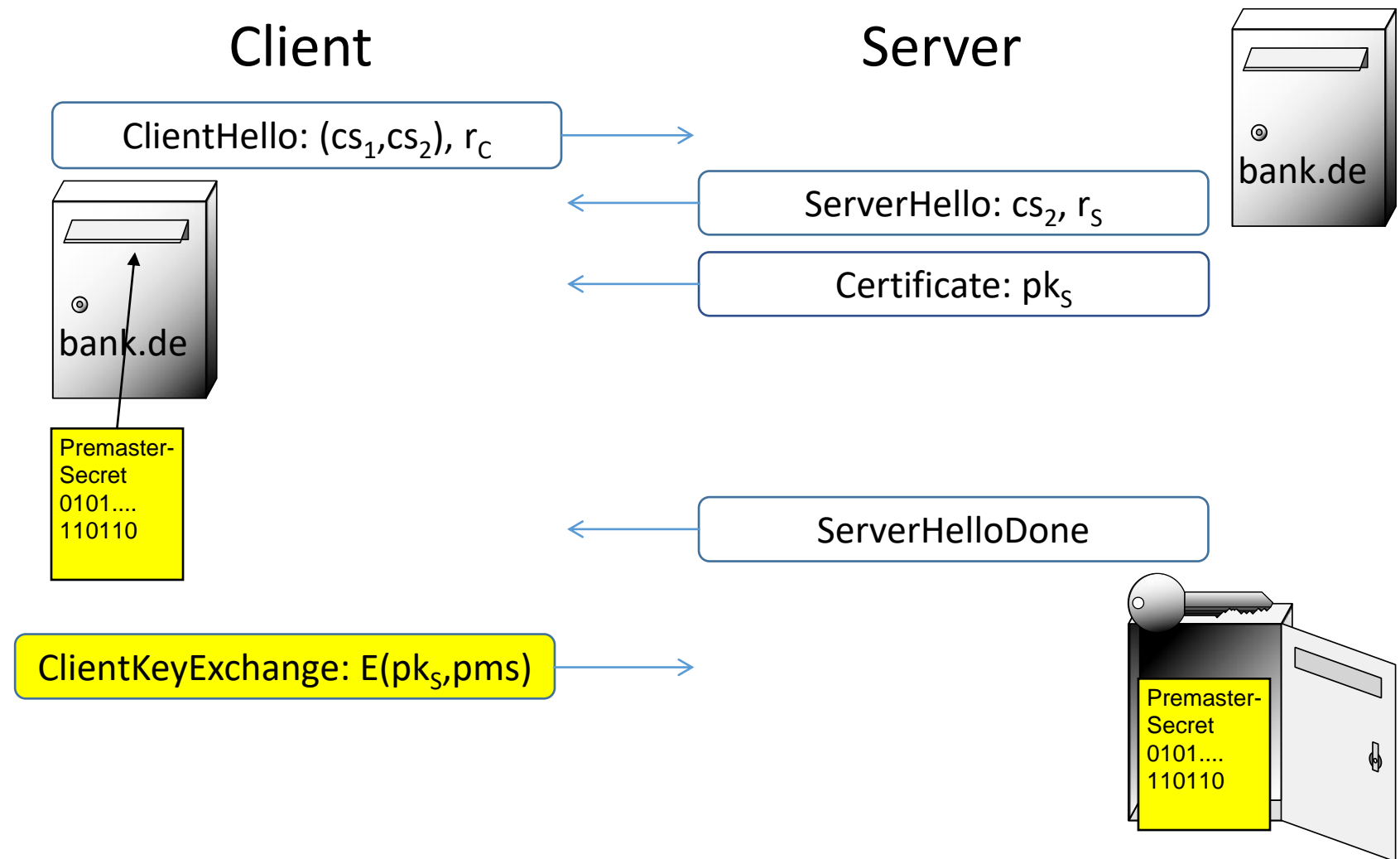
TLS Handshake: TLS-RSA



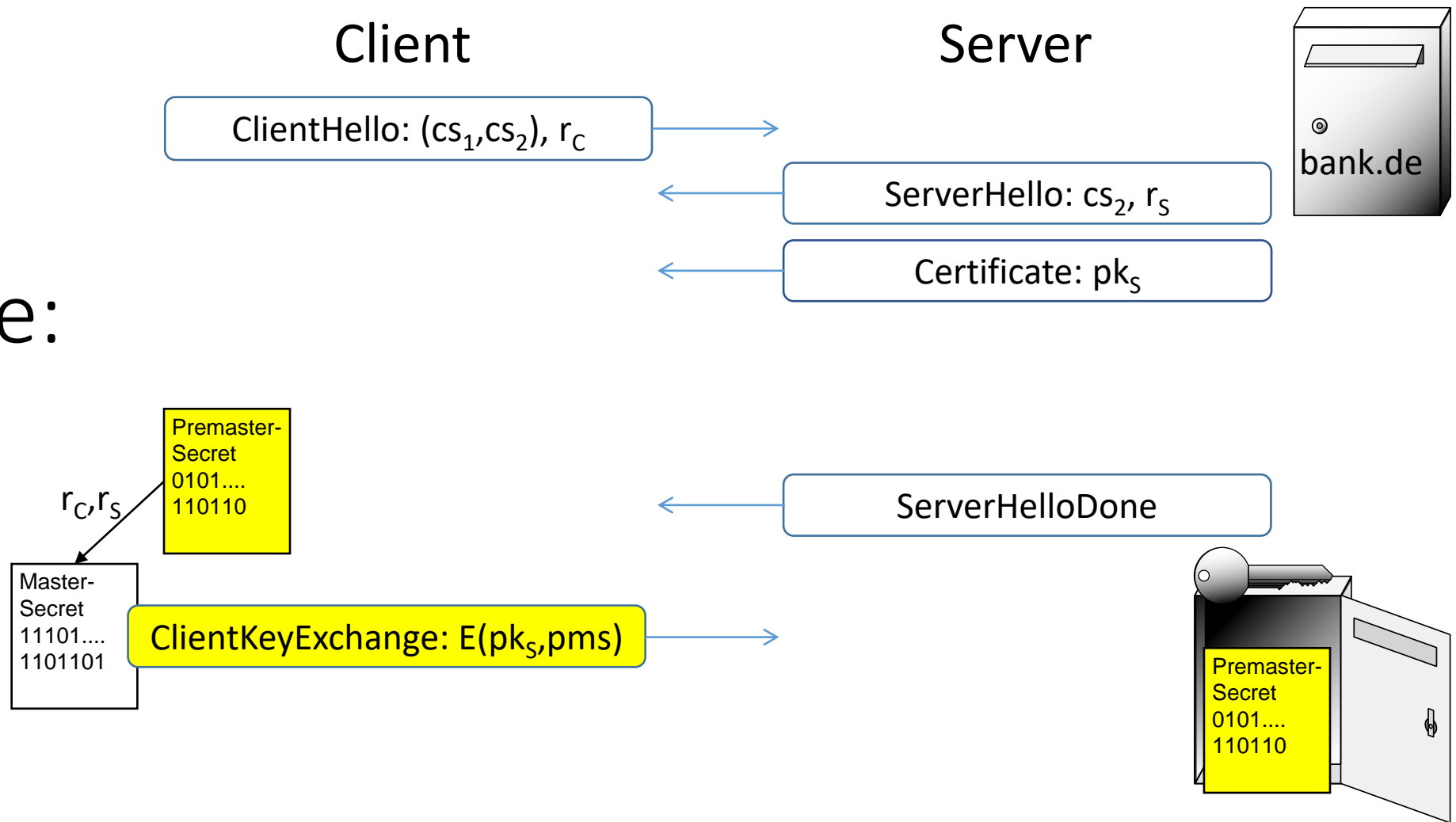
TLS Handshake: TLS-RSA



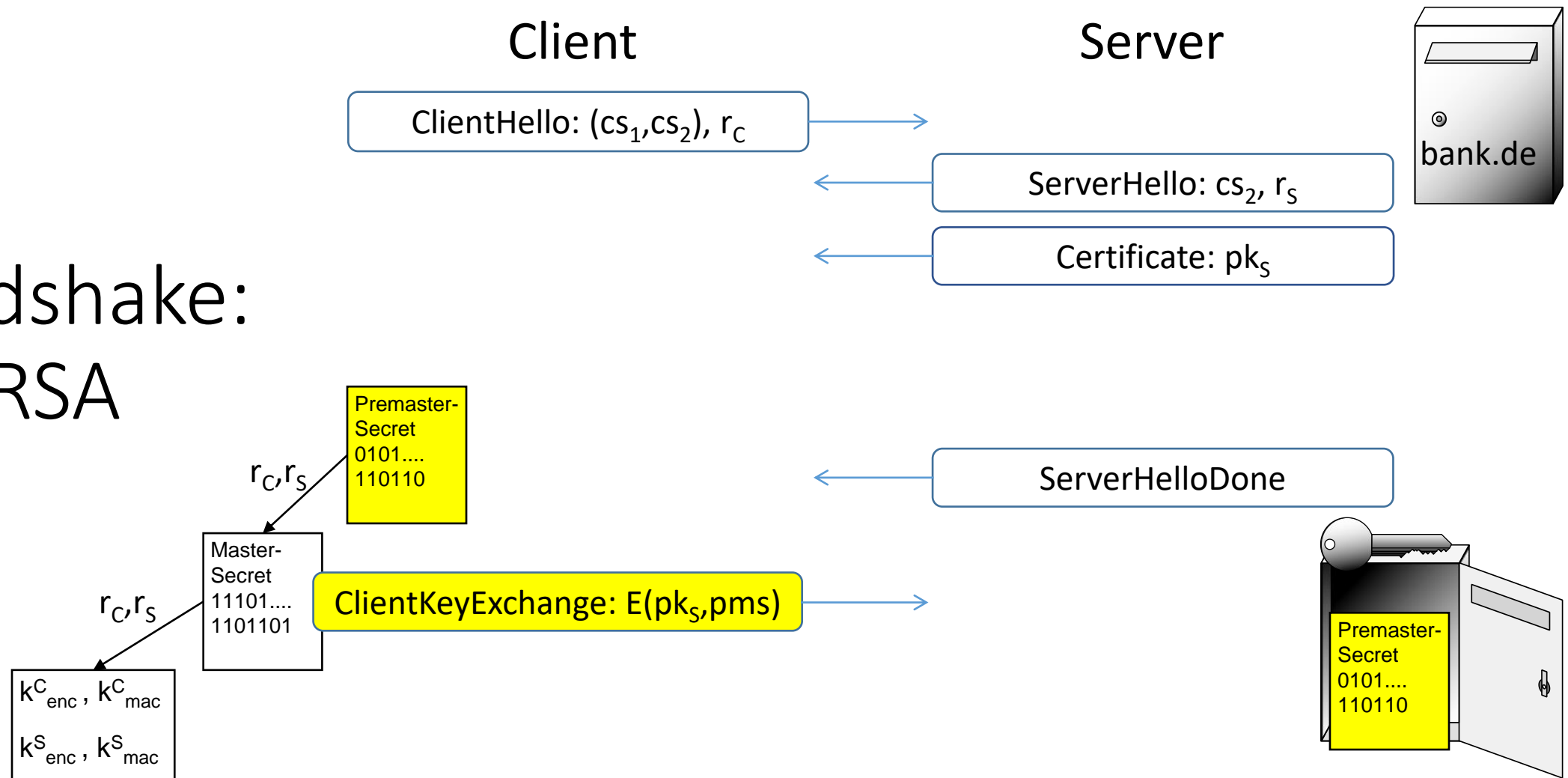
TLS Handshake: TLS-RSA



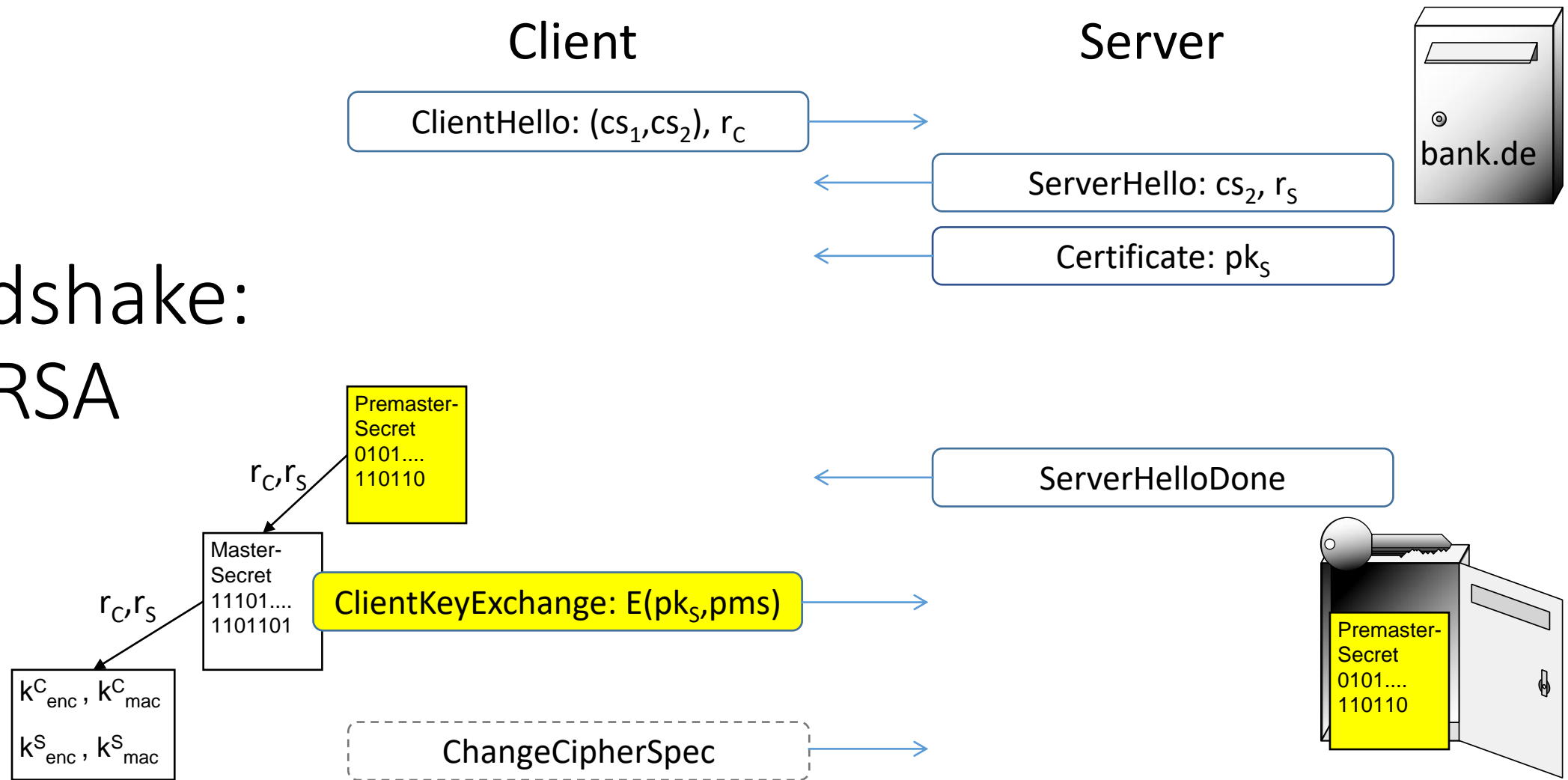
TLS Handshake: TLS-RSA



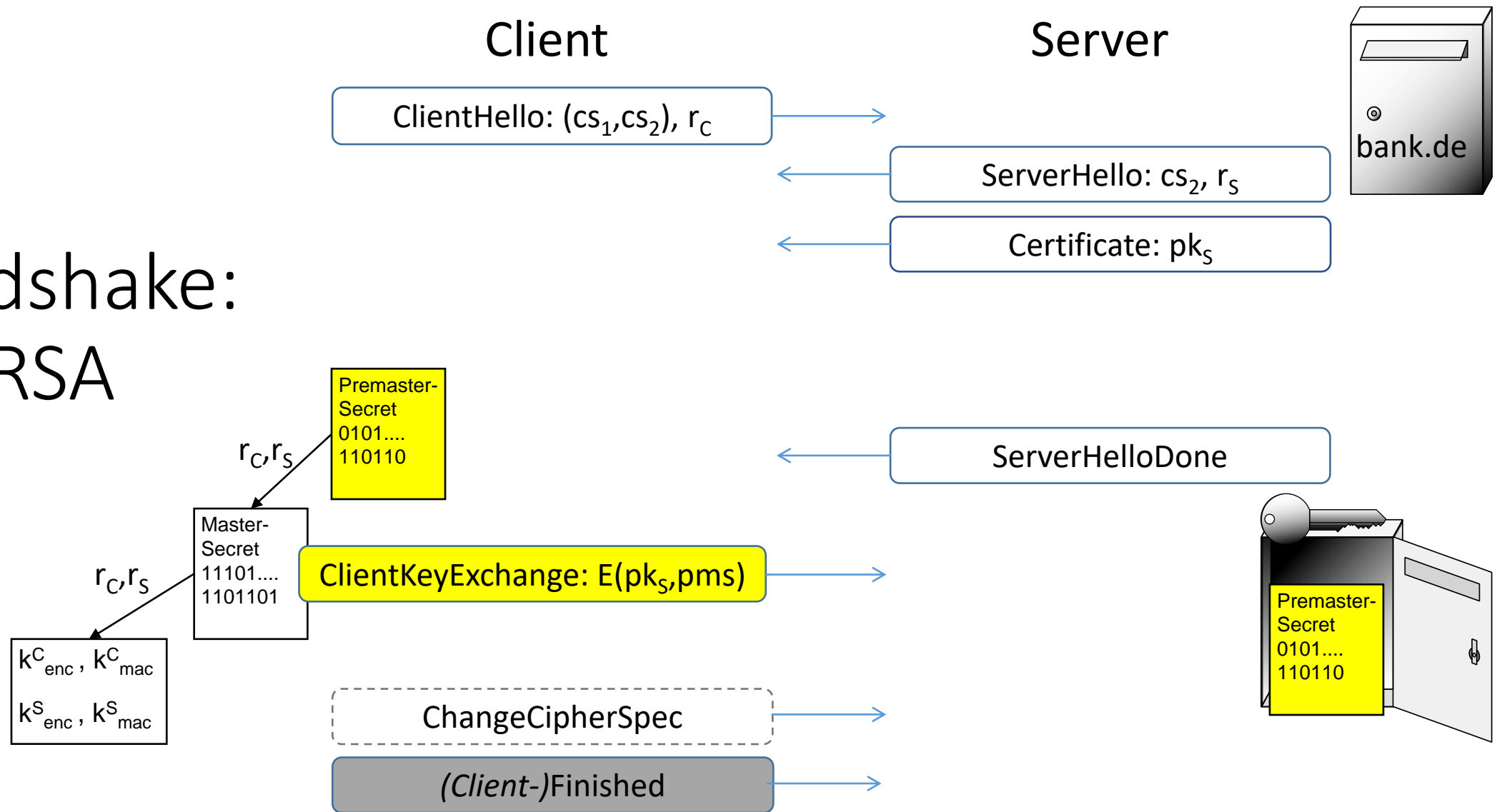
TLS Handshake: TLS-RSA



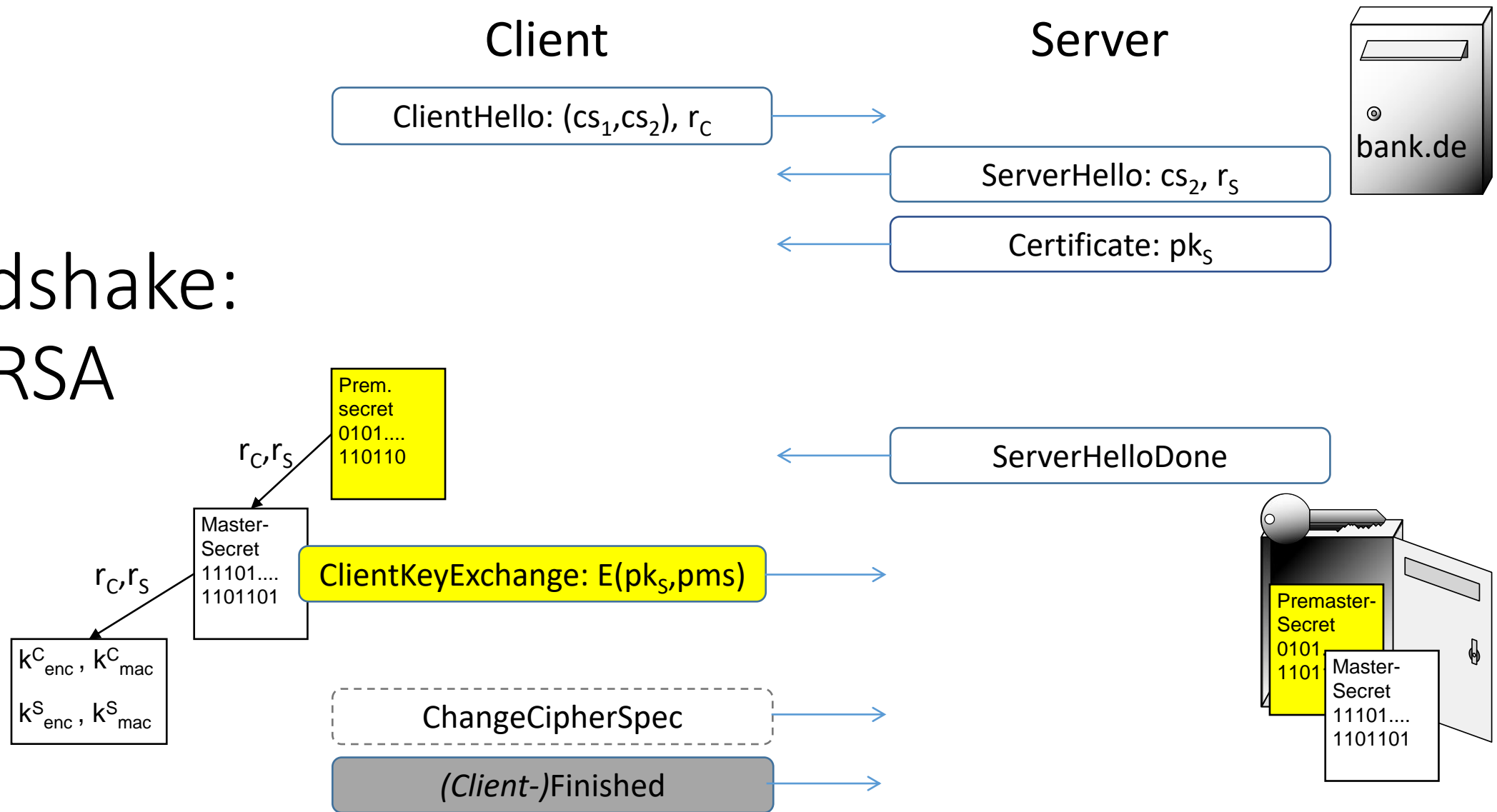
TLS Handshake: TLS-RSA



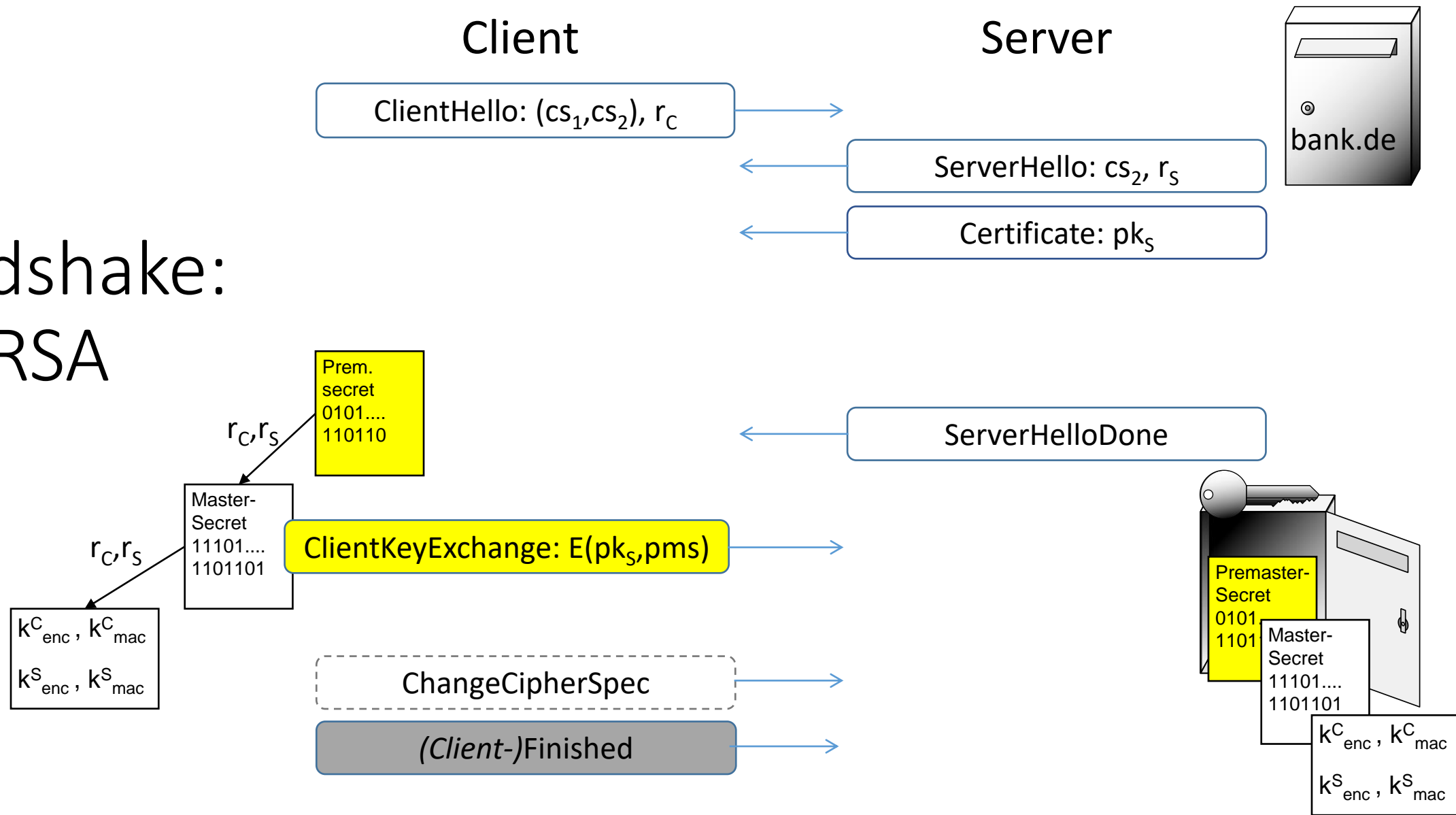
TLS Handshake: TLS-RSA



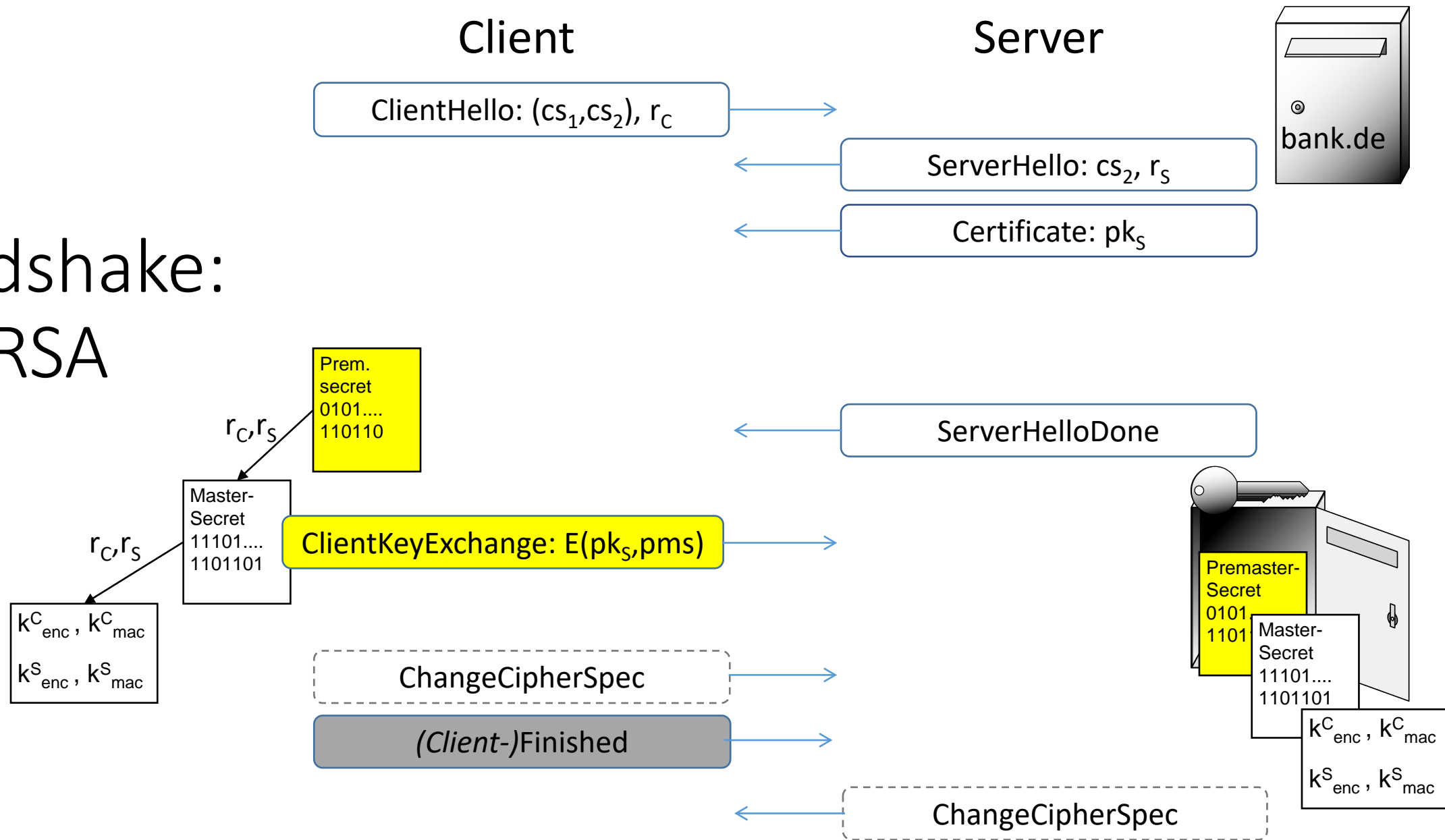
TLS Handshake: TLS-RSA



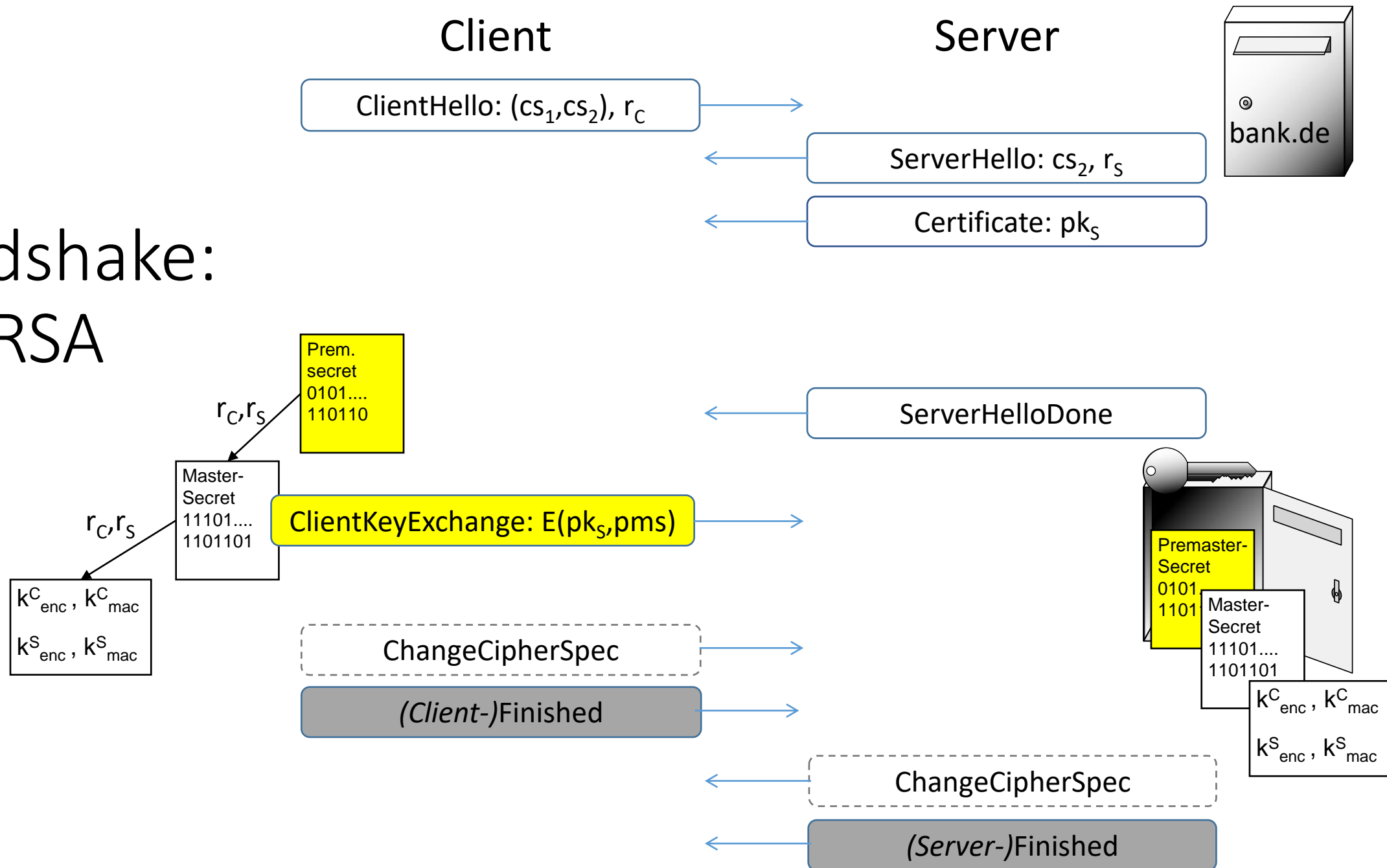
TLS Handshake: TLS-RSA



TLS Handshake: TLS-RSA

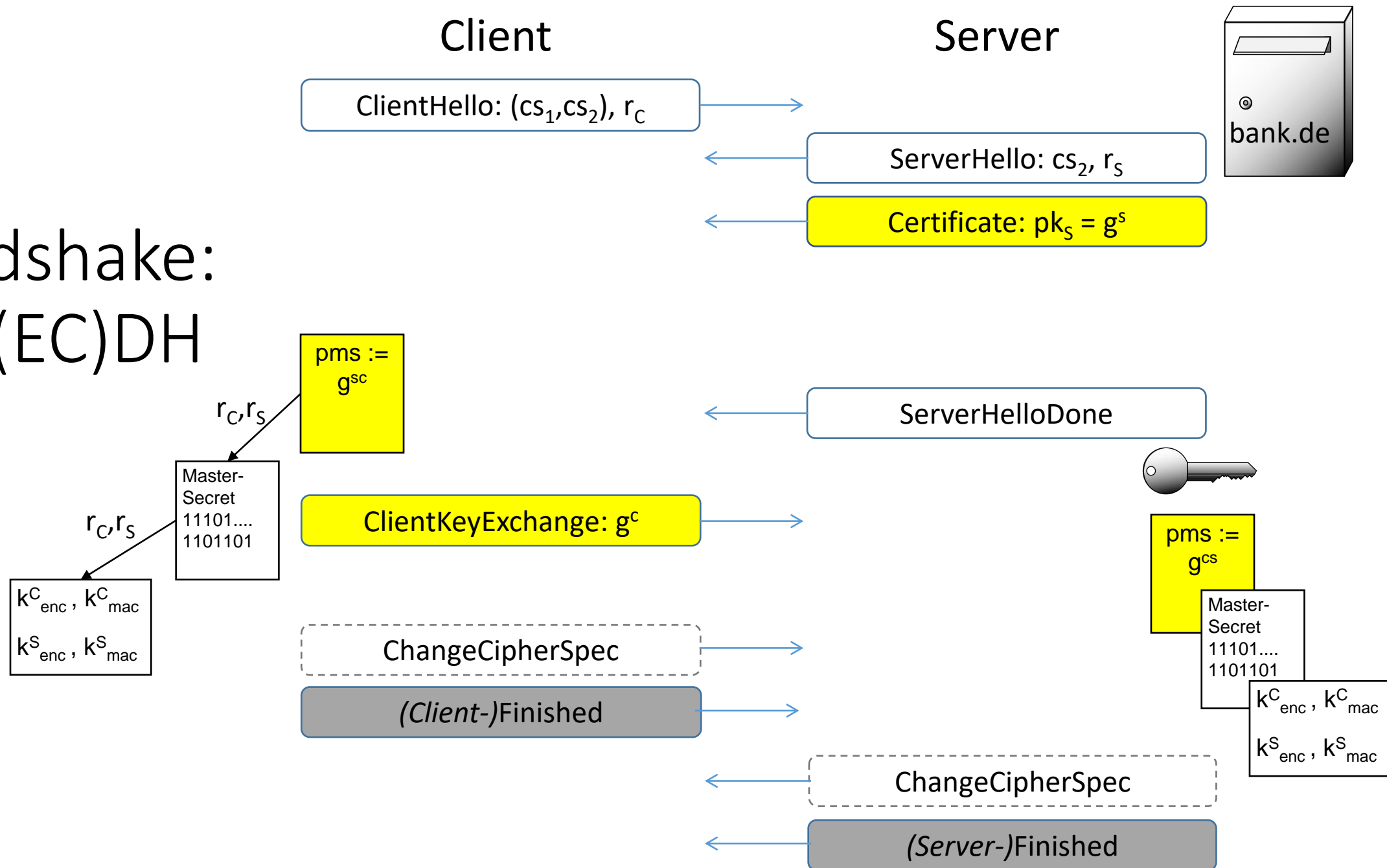


TLS Handshake: TLS-RSA

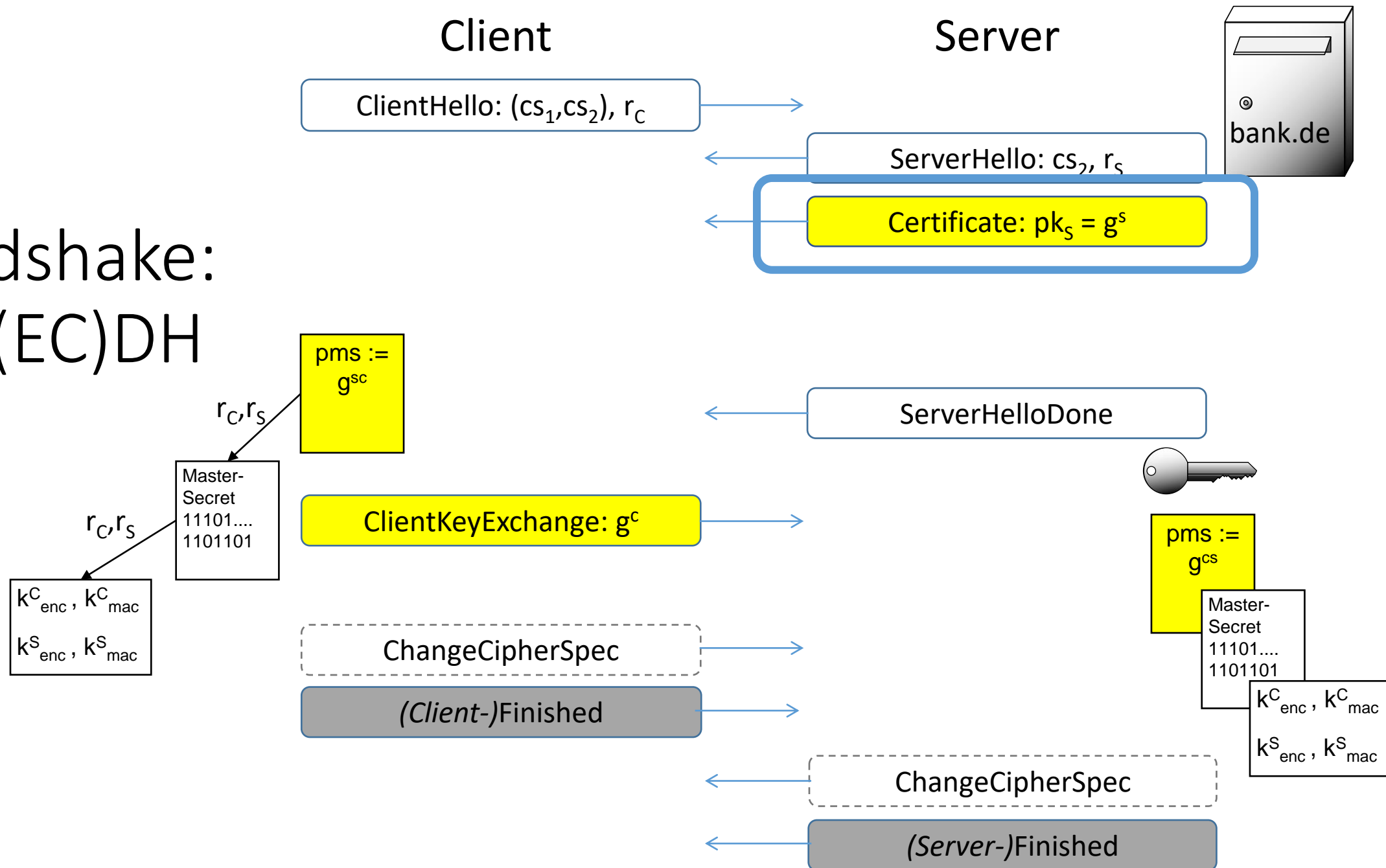


TLS Handshake: TLS-(EC)DH

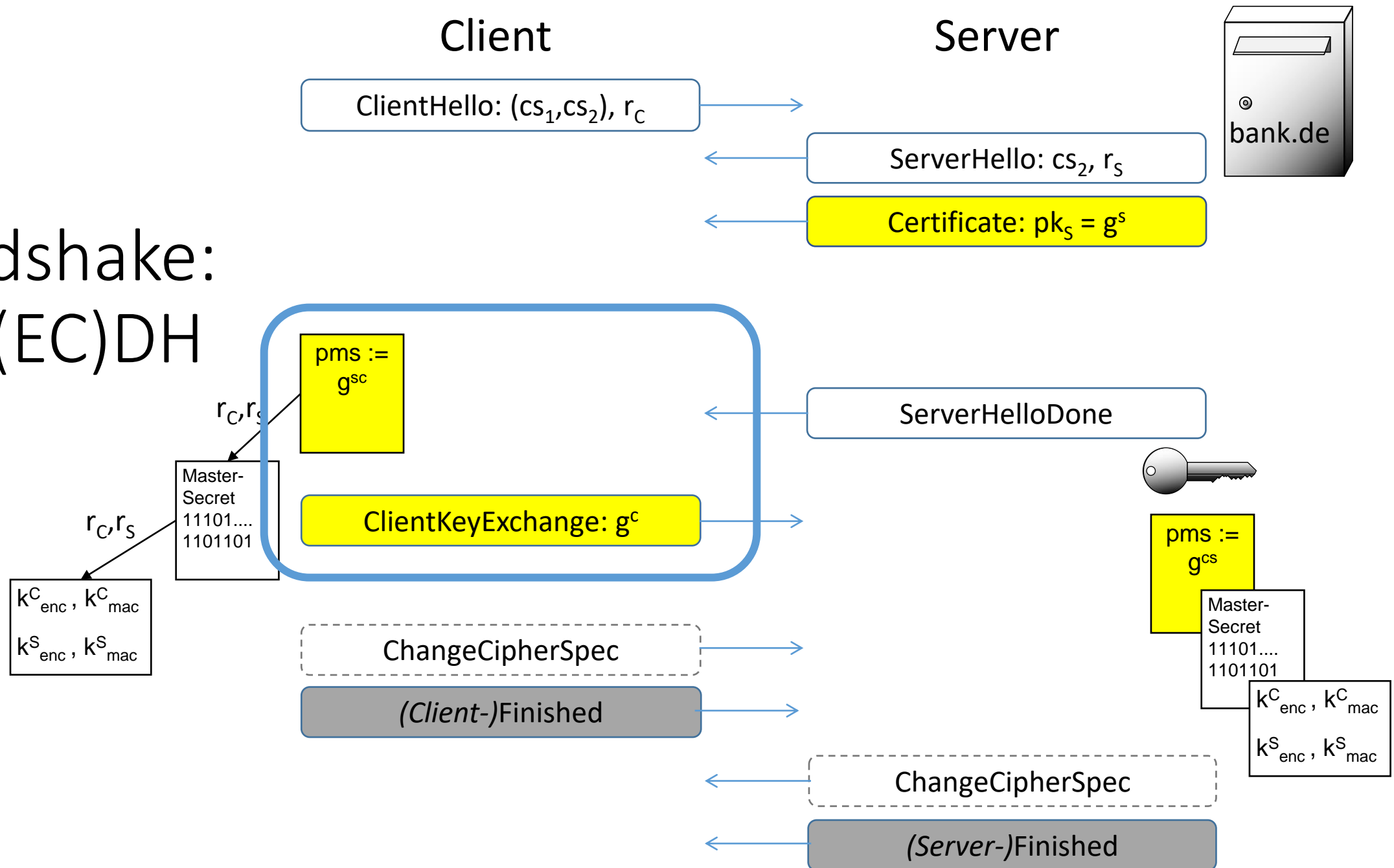
TLS Handshake: TLS-(EC)DH



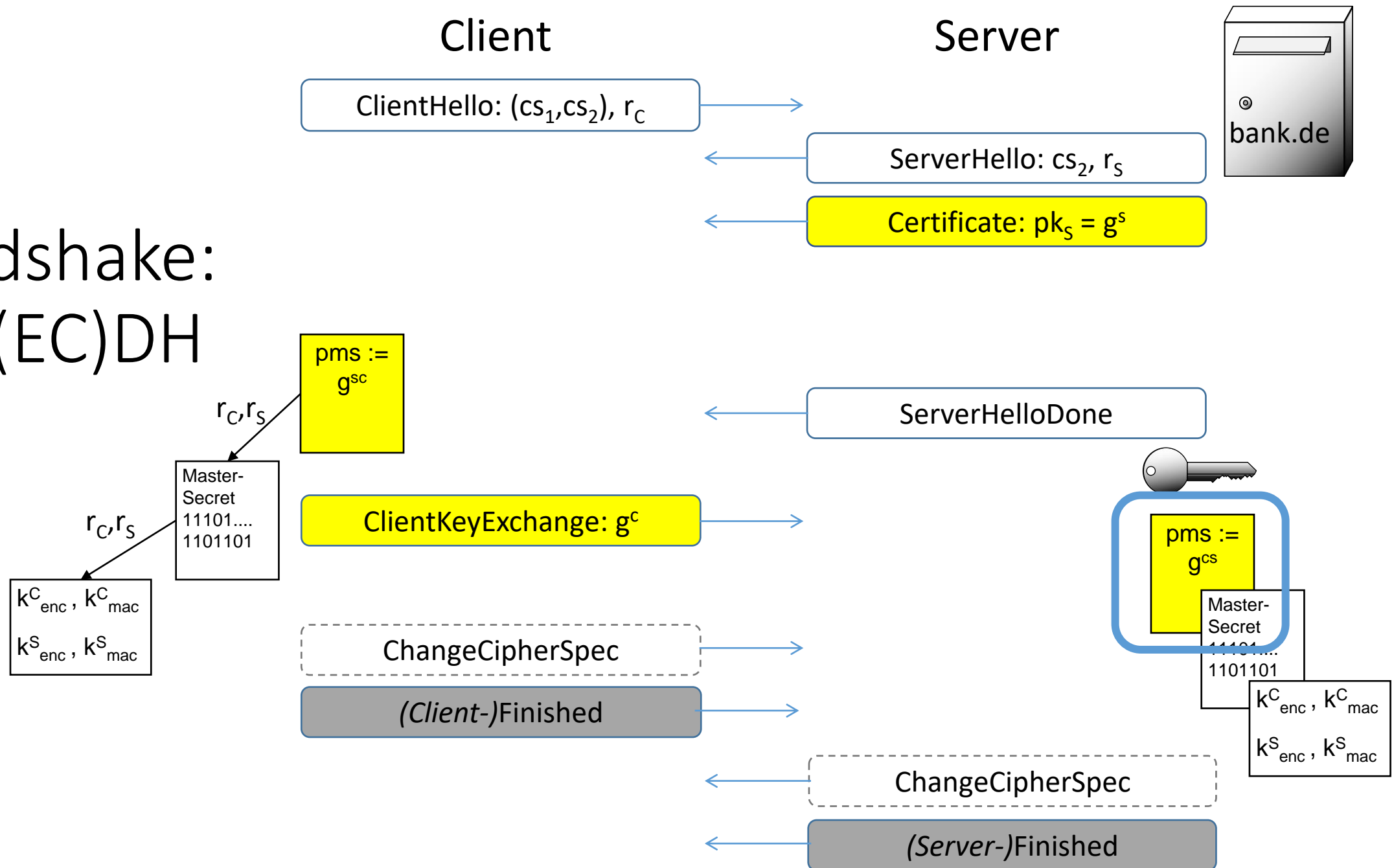
TLS Handshake: TLS-(EC)DH



TLS Handshake: TLS-(EC)DH

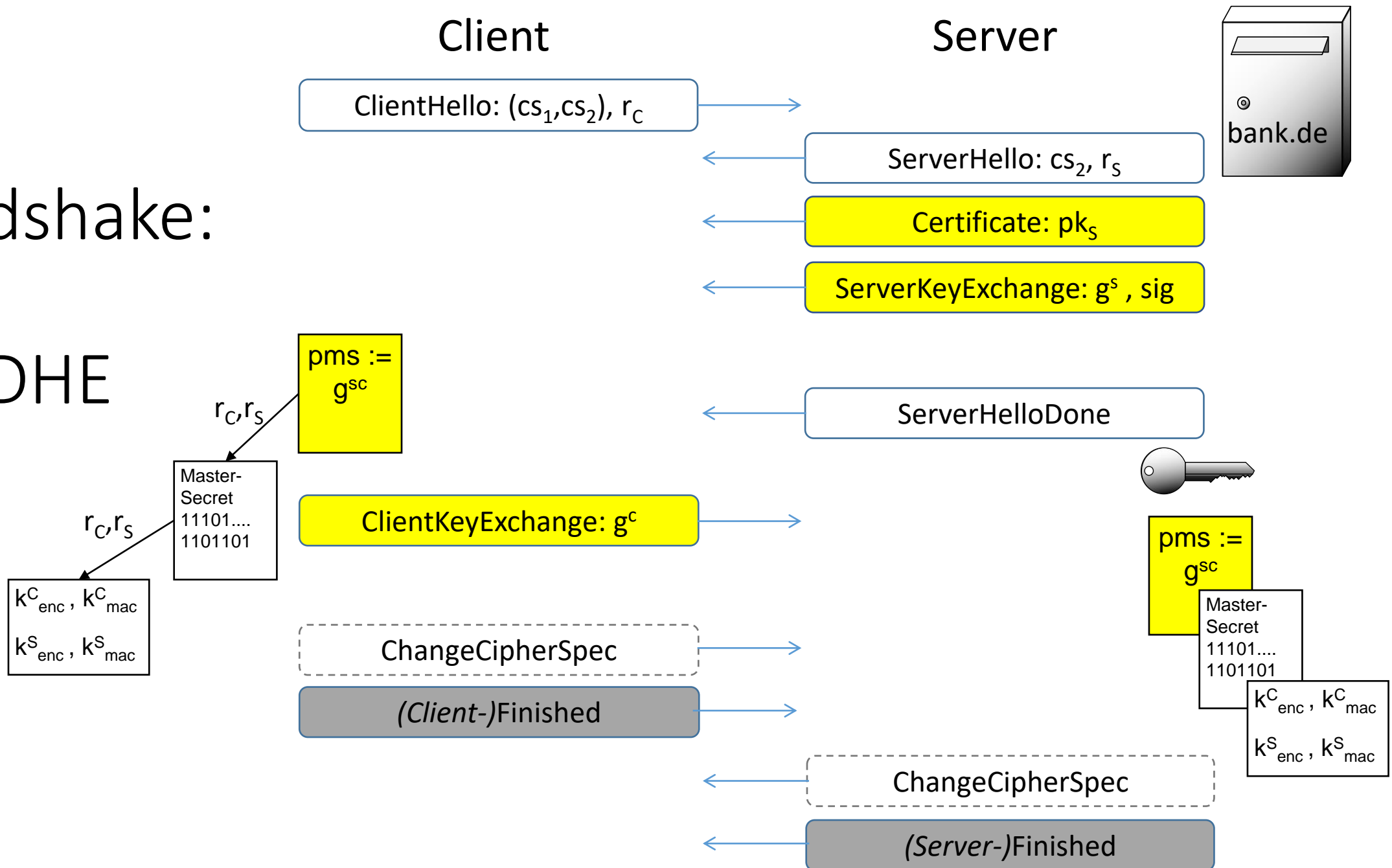


TLS Handshake: TLS-(EC)DH

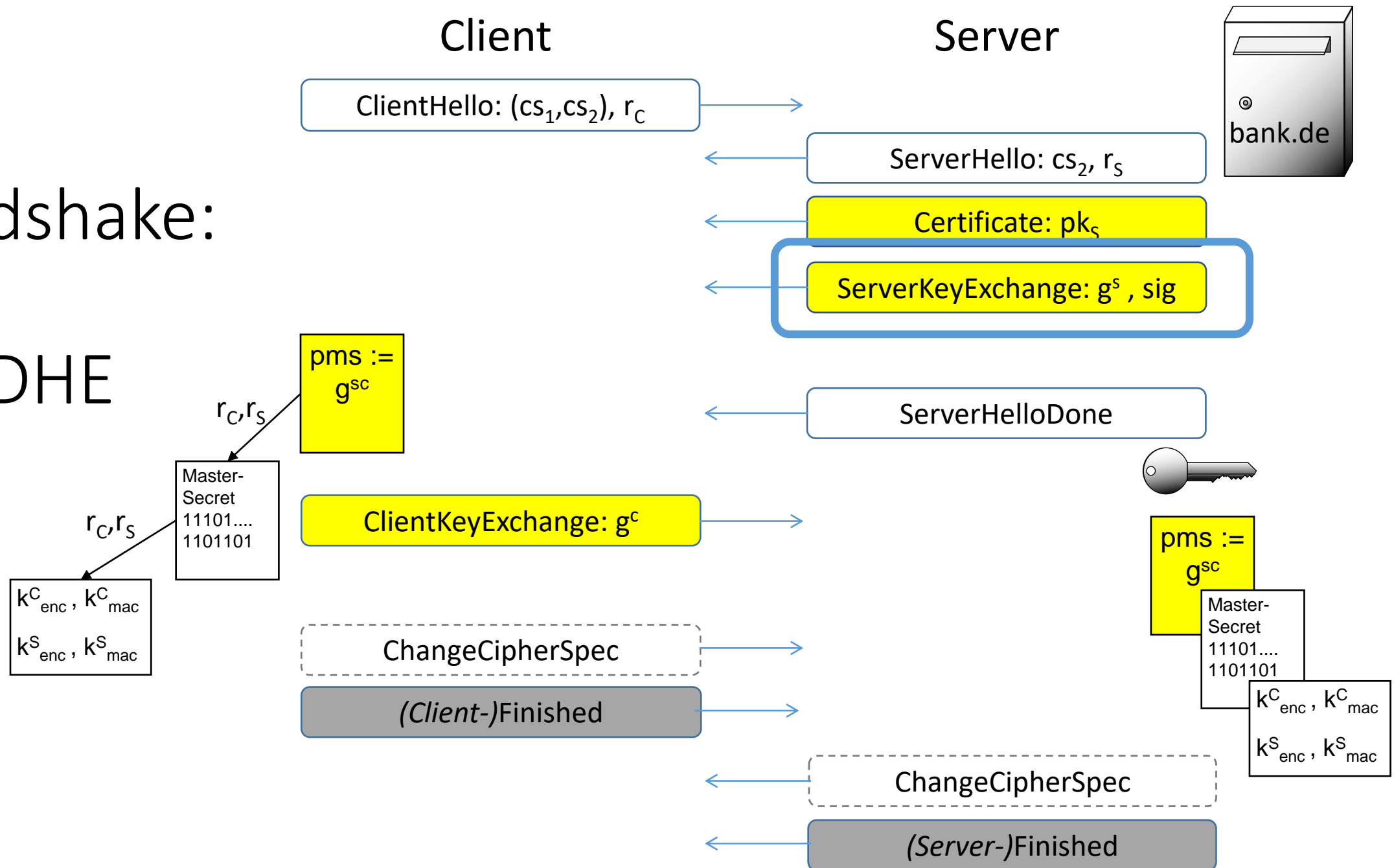


TLS Handshake: TLS-(EC)DHE

TLS Handshake: TLS- (EC)DHE

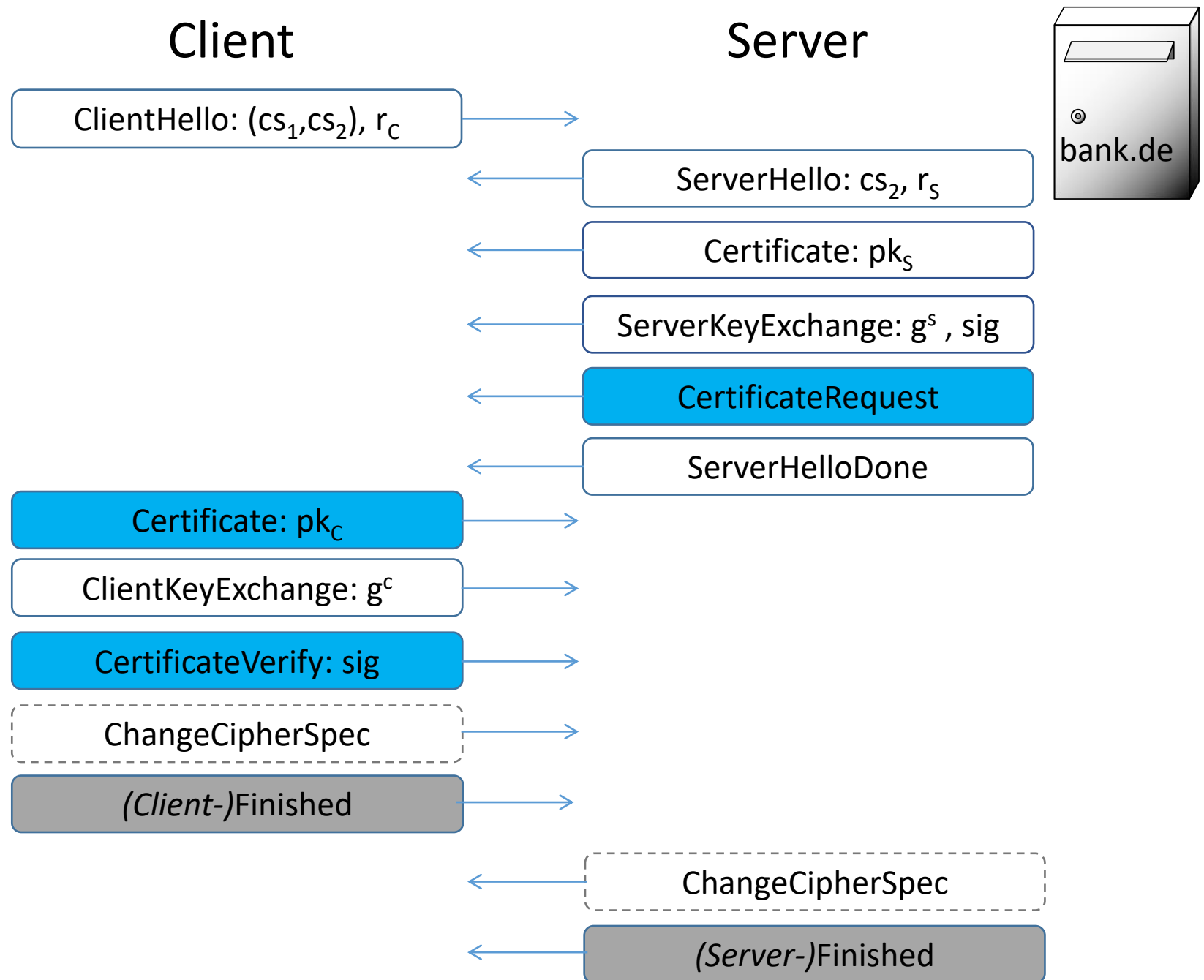


TLS Handshake: TLS- (EC)DHE

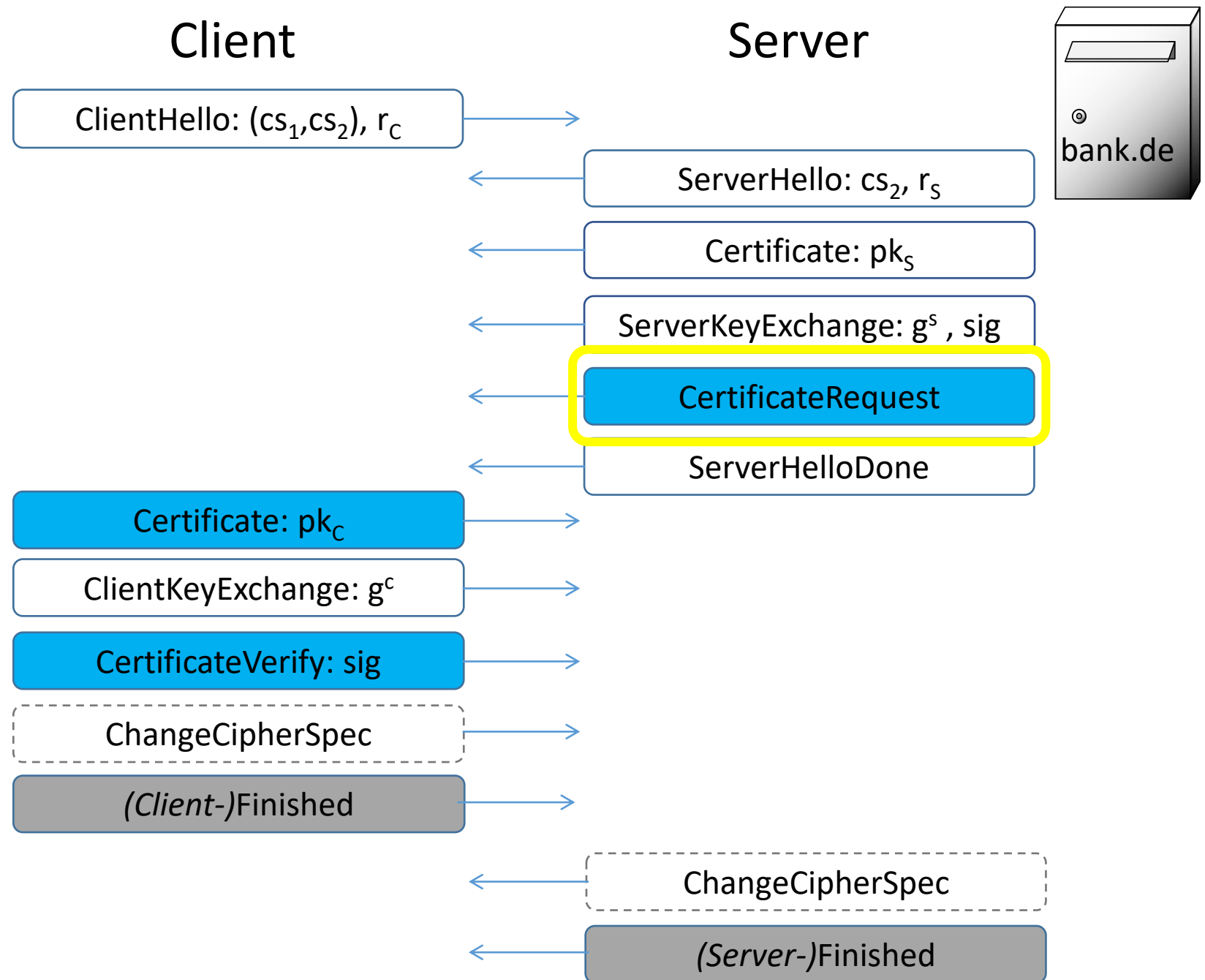


TLS Handshake: TLS-(EC)DHE mit Client-Authentifizierung

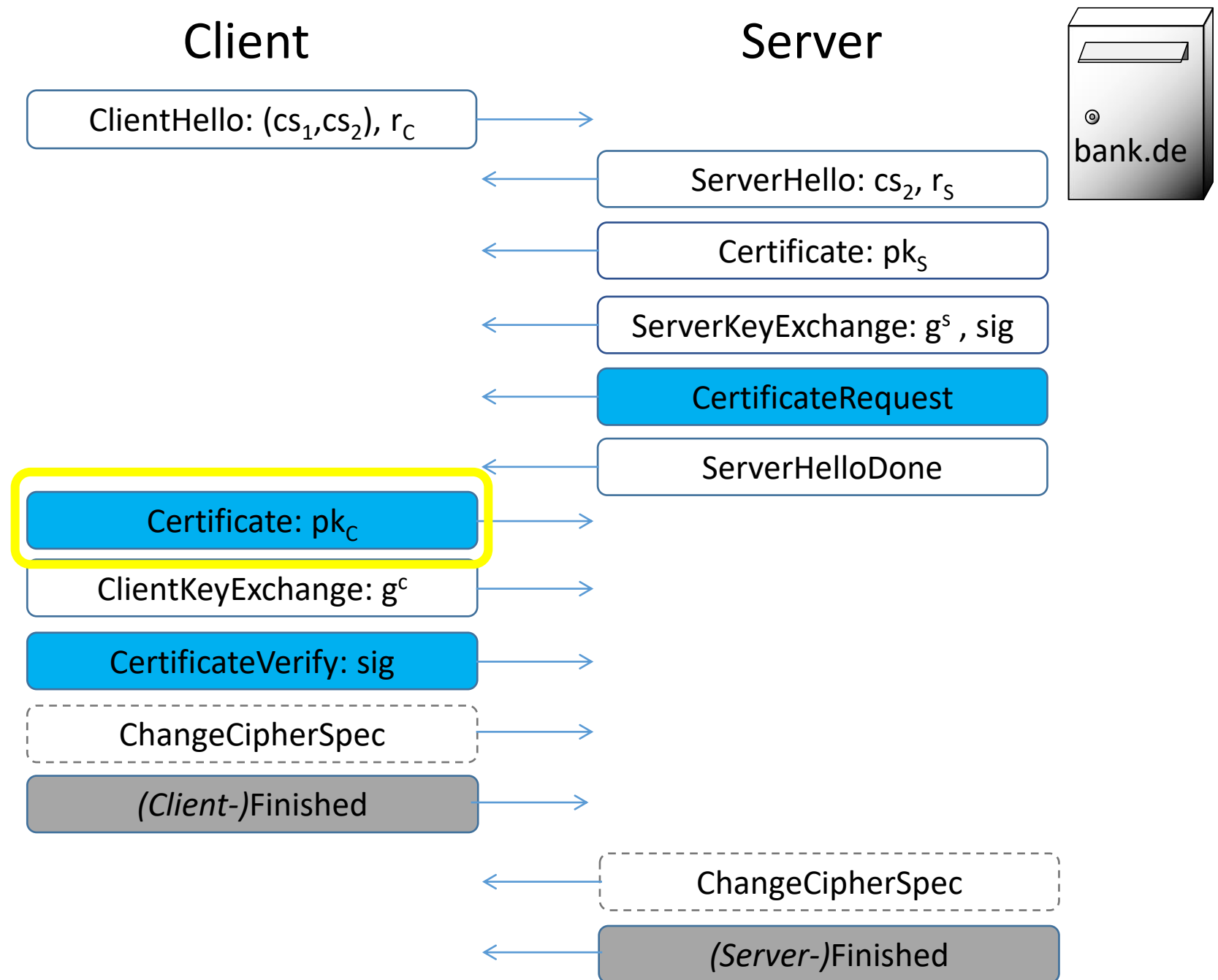
TLS Handshake: TLS- (EC)DHE mutual



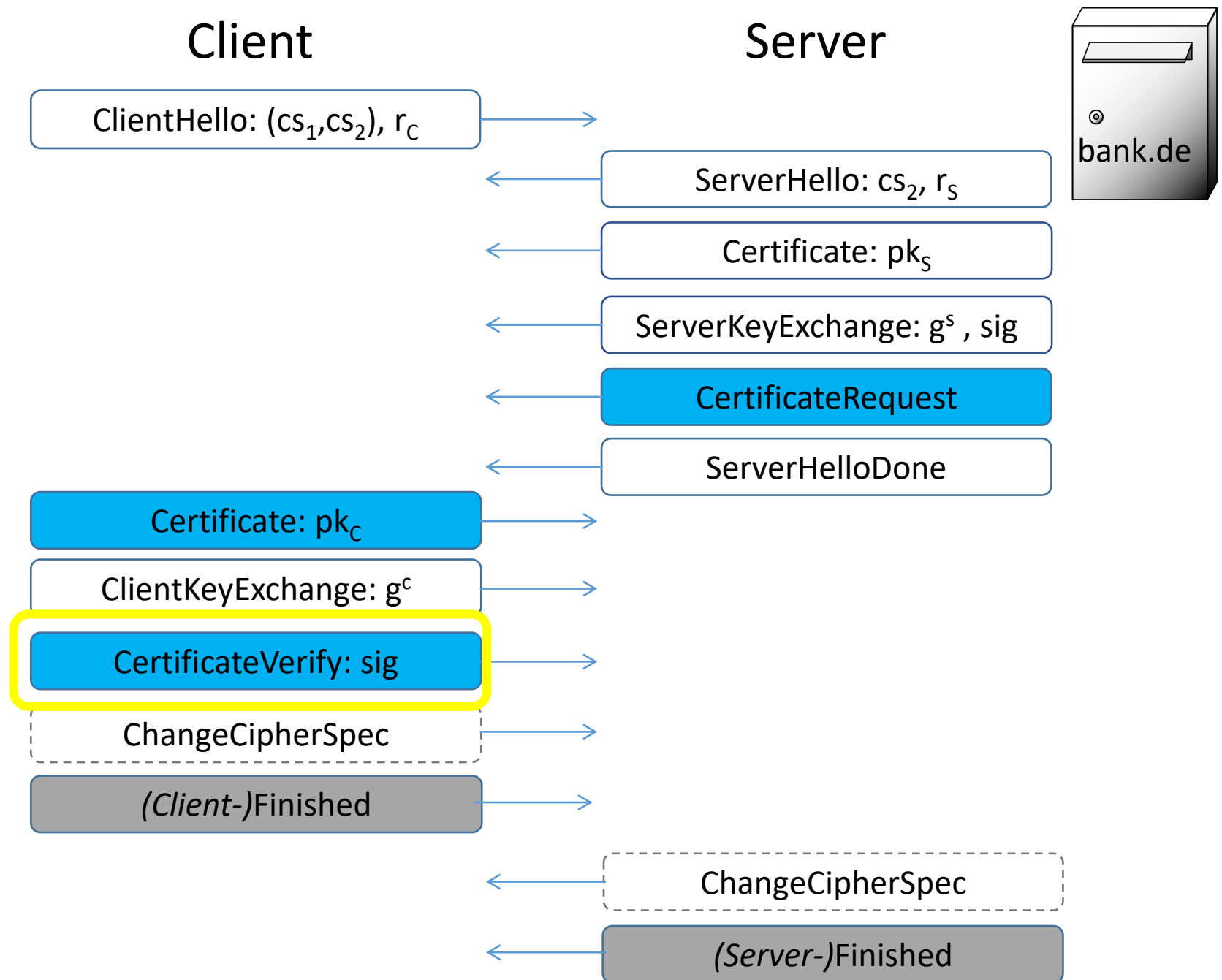
TLS Handshake: TLS- (EC)DHE mutual



TLS Handshake: TLS- (EC)DHE mutual



TLS Handshake: TLS- (EC)DHE mutual



2.3 TLS Handshake

2.3.2 Ciphersuites

Ciphersuites in OpenSSL: TLS 1.2

TLS_RSA_WITH_NULL_SHA256

TLS_RSA_WITH_AES_128_CBC_SHA256

TLS_RSA_WITH_AES_256_CBC_SHA256

TLS_RSA_WITH_AES_128_GCM_SHA256

TLS_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_RSA_WITH_AES_128_CBC_SHA256

TLS_DH_RSA_WITH_AES_256_CBC_SHA256

TLS_DH_RSA_WITH_AES_128_GCM_SHA256

TLS_DH_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_DSS_WITH_AES_128_CBC_SHA256

TLS_DH_DSS_WITH_AES_256_CBC_SHA256

TLS_DH_DSS_WITH_AES_128_GCM_SHA256

TLS_DH_DSS_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256

TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_DSS_WITH_AES_128_CBC_SHA256

TLS_DHE_DSS_WITH_AES_256_CBC_SHA256

TLS_DHE_DSS_WITH_AES_128_GCM_SHA256

TLS_DHE_DSS_WITH_AES_256_GCM_SHA384

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384

TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS_DH_anon_WITH_AES_128_CBC_SHA256

TLS_DH_anon_WITH_AES_256_CBC_SHA256

TLS_DH_anon_WITH_AES_128_GCM_SHA256

TLS_DH_anon_WITH_AES_256_GCM_SHA384

Jede Ciphersuite wird als 2-Byte-Wert codiert übertragen

Ciphersuites in OpenSSL: TLS 1.2

TLS_RSA_WITH_NULL_SHA256

TLS_RSA_WITH_AES_128_CBC_SHA256

TLS_RSA_WITH_AES_256_CBC_SHA256

TLS_RSA_WITH_AES_128_GCM_SHA256

TLS_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_RSA_WITH_AES_128_CBC_SHA256

TLS_DH_RSA_WITH_AES_256_CBC_SHA256

TLS_DH_RSA_WITH_AES_128_GCM_SHA256

TLS_DH_RSA_WITH_AES_256_GCM_SHA384

TLS_DH_DSS_WITH_AES_128_CBC_SHA256

TLS_DH_DSS_WITH_AES_256_CBC_SHA256

TLS_DH_DSS_WITH_AES_128_GCM_SHA256

TLS_DH_DSS_WITH_AES_256_GCM_SHA384

TLS_DHE_RSA_WITH_AES_128_CBC_SHA256

TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

TLS_DHE_DSS_WITH_AES_128_CBC_SHA256

TLS_DHE_DSS_WITH_AES_256_CBC_SHA256

TLS_DHE_DSS_WITH_AES_128_GCM_SHA256

TLS_DHE_DSS_WITH_AES_256_GCM_SHA384

TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384

TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS_DH_anon_WITH_AES_128_CBC_SHA256

TLS_DH_anon_WITH_AES_256_CBC_SHA256

TLS_DH_anon_WITH_AES_128_GCM_SHA256

TLS_DH_anon_WITH_AES_256_GCM_SHA384

keine Certificate-Nachricht,
nur ServerKeyExchange

Viele weitere Ciphersuites unter <https://www.openssl.org/docs/man1.0.2/apps/ciphers.html>

Was muss alles ausgehandelt werden?

- Schlüsselvereinbarung
- Client-Authentifikation
- Record Layer-Verschlüsselung
- Record Layer-Authentifikation
- TLS-PRF (ab TLS 1.2)

Was muss alles ausgehandelt werden?

- Schlüsselvereinbarung
 - RSA-PKE/DHKE/ElGamal KEM, mathematische Gruppe/Ring, Auth. Server
- Client-Authentifikation
- Record Layer-Verschlüsselung
- Record Layer-Authentifikation
- TLS-PRF (ab TLS 1.2)

Was muss alles ausgehandelt werden?

- Schlüsselvereinbarung
 - RSA-PKE/DHKE/ElGamal KEM, mathematische Gruppe/Ring, Auth. Server
- Client-Authentifikation
 - welche PKI, welches Signaturverfahren?
- Record Layer-Verschlüsselung
- Record Layer-Authentifikation
- TLS-PRF (ab TLS 1.2)

Was muss alles ausgehandelt werden?

- Schlüsselvereinbarung
 - RSA-PKE/DHKE/ElGamal KEM, mathematische Gruppe/Ring, Auth. Server
- Client-Authentifikation
 - welche PKI, welches Signaturverfahren?
- Record Layer-Verschlüsselung
 - Block-/Stromchiffre, Algorithmus, Modus, Schlüssellänge
- Record Layer-Authentifikation
- TLS-PRF (ab TLS 1.2)

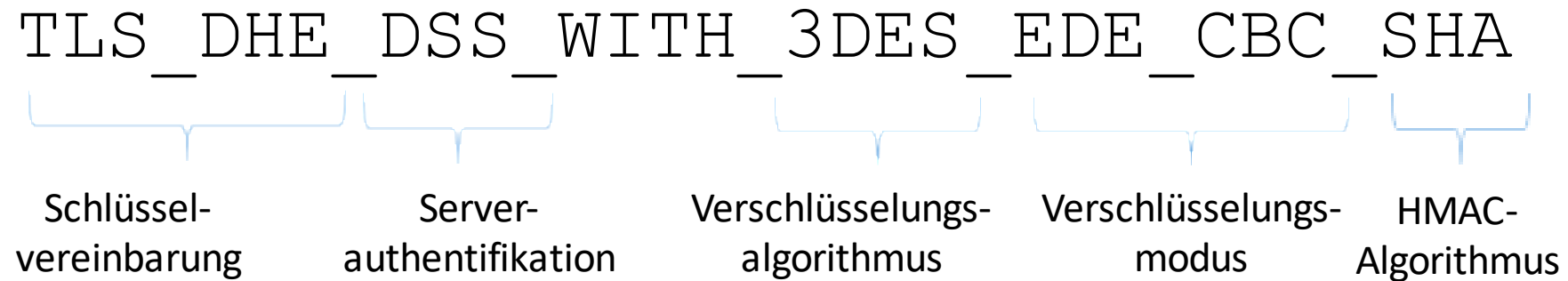
Was muss alles ausgehandelt werden?

- Schlüsselvereinbarung
 - RSA-PKE/DHKE/ElGamal KEM, mathematische Gruppe/Ring, Auth. Server
- Client-Authentifikation
 - welche PKI, welches Signaturverfahren?
- Record Layer-Verschlüsselung
 - Block-/Stromchiffre, Algorithmus, Modus, Schlüssellänge
- Record Layer-Authentifikation
 - HMAC mit welcher Hashfunktion? Sonderfall GCM!
- TLS-PRF (ab TLS 1.2)

Was muss alles ausgehandelt werden?

- Schlüsselvereinbarung
 - RSA-PKE/DHKE/ElGamal KEM, mathematische Gruppe/Ring, Auth. Server
- Client-Authentifikation
 - welche PKI, welches Signaturverfahren?
- Record Layer-Verschlüsselung
 - Block-/Stromchiffre, Algorithmus, Modus, Schlüssellänge
- Record Layer-Authentifikation
 - HMAC mit welcher Hashfunktion? Sonderfall GCM!
- TLS-PRF (ab TLS 1.2)
 - Hashfunktion als Basis für HMAC-basierte TLS-PRF

Bestandteile



Bestandteile

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

Schlüssel-
vereinbarung

Server-
authentifikation

Verschlüsselungs-
algorithmus

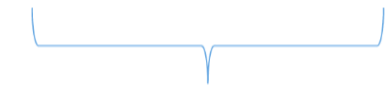
Verschlüsselungs-
modus

HMAC-
Algorithmus

Ciphersuite spezifiziert nicht alle Parameter

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



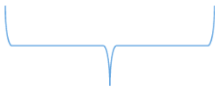
Schlüssel-
vereinbarung

Gibt die Art der Schlüsselvereinbarung an. Mögliche Werte

- TLS_RSA
- TLS_DH
- TLS_ECDH
- TLS_DHE
- TLS_ECDHE

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

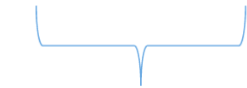


Server-
authentifikation

Gibt an, wie der Server sich authentifiziert.

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



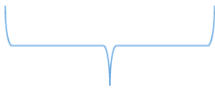
Server-
authentifikation

Gibt an, wie der Server sich authentifiziert.

- TLS-RSA: Dieser Eintrag fehlt hier; Implizite Authentifikation des Servers über die Fähigkeit des Servers, ClientKeyExchange zu entschlüsseln

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



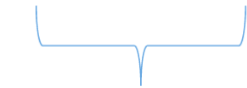
Server-
authentifikation

Gibt an, wie der Server sich authentifiziert.

- TLS-RSA: Dieser Eintrag fehlt hier; Implizite Authentifikation des Servers über die Fähigkeit des Servers, ClientKeyExchange zu entschlüsseln
- TLS-(EC)DH: Das Serverzertifikat ist mit dem angegebenen Algorithmus signiert.
 - mögliche Werte: RSA, DSS, ECDSA, anon

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



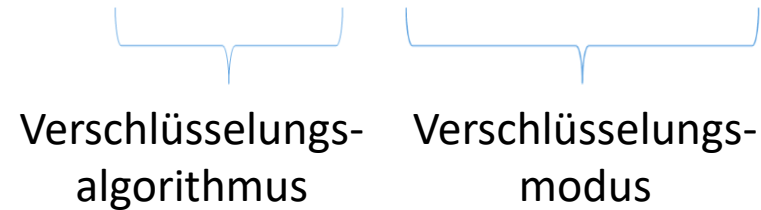
Server-
authentifikation

Gibt an, wie der Server sich authentifiziert.

- TLS-RSA: Dieser Eintrag fehlt hier; Implizite Authentifikation des Servers über die Fähigkeit des Servers, ClientKeyExchange zu entschlüsseln
- TLS-(EC)DH: Das Serverzertifikat ist mit dem angegebenen Algorithmus signiert.
 - mögliche Werte: RSA, DSS, ECDSA, anon
- TLS-(EC)DHE: Die ServerKeyExchange-Nachricht ist, zusammen mit den Zufallszahlen r_C und r_S , mit dem angegebenen Algorithmus signiert.
 - mögliche Werte: RSA, DSS, ECDSA

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



The diagram shows the cipher suite name 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA' with two blue brackets underneath. The first bracket is under '3DES' and points to the label 'Verschlüsselungs-
algorithmus'. The second bracket is under 'EDE_CBC' and points to the label 'Verschlüsselungs-
modus'.

Verschlüsselungs-
algorithmus Verschlüsselungs-
modus

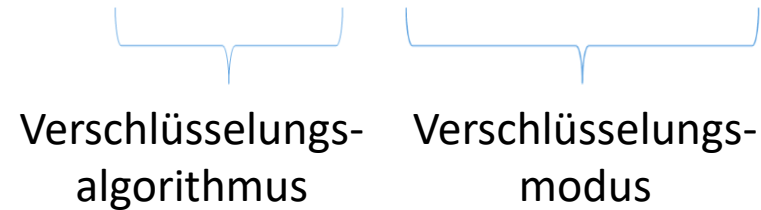
Verschlüsselungsalgorithmus und - modus auf dem Record Layer.

Mögliche Werte:

- NULL

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



The diagram shows the cipher suite name 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA' with two blue curly braces underneath. The first brace is under '3DES' and points to the label 'Verschlüsselungs-
algorithmus'. The second brace is under 'EDE_CBC' and points to the label 'Verschlüsselungs-
modus'.

Verschlüsselungs-
algorithmus Verschlüsselungs-
modus

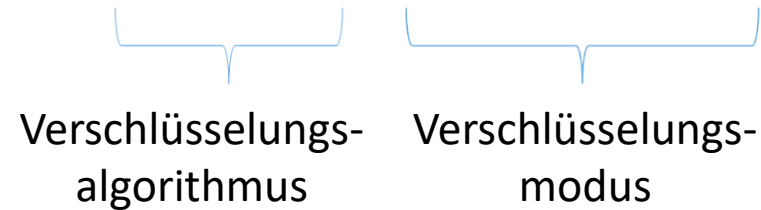
Verschlüsselungsalgorithmus und - modus auf dem Record Layer.

Mögliche Werte:

- NULL
- DES40, DES, 3DES

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



The diagram shows the cipher suite name 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA' with two blue curly braces underneath. The first brace is under '3DES' and points to the label 'Verschlüsselungs-
algorithmus'. The second brace is under 'EDE_CBC' and points to the label 'Verschlüsselungs-
modus'.

Verschlüsselungs-
algorithmus Verschlüsselungs-
modus

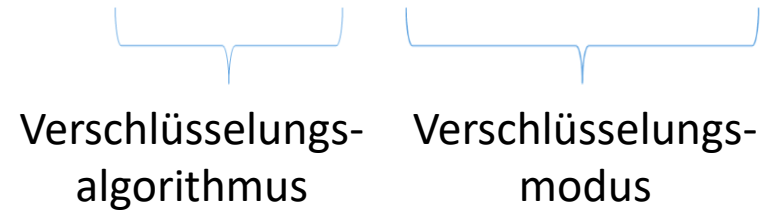
Verschlüsselungsalgorithmus und -modus auf dem Record Layer.

Mögliche Werte:

- NULL
- DES40, DES, 3DES
- RC2, RC4

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



The diagram shows the cipher suite name 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA' with two blue brackets underneath. The first bracket is under '3DES' and points to the label 'Verschlüsselungs-
algorithmus'. The second bracket is under 'EDE_CBC' and points to the label 'Verschlüsselungs-
modus'.

Verschlüsselungs-
algorithmus Verschlüsselungs-
modus

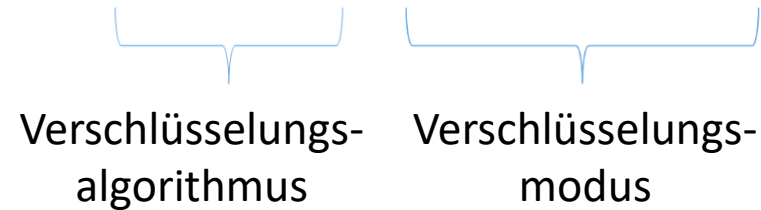
Verschlüsselungsalgorithmus und - modus auf dem Record Layer.

Mögliche Werte:

- NULL
- DES40, DES, 3DES
- RC2, RC4
- IDEA
- AES

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



The diagram shows the cipher suite name 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA' with two blue brackets underneath. The first bracket is under '3DES' and points to the label 'Verschlüsselungs-
algorithmus'. The second bracket is under 'EDE_CBC' and points to the label 'Verschlüsselungs-
modus'.

Verschlüsselungs-
algorithmus Verschlüsselungs-
modus

Verschlüsselungsalgorithmus und - modus auf dem Record Layer.
Mögliche Werte:

- NULL
- DES40, DES, 3DES
- RC2, RC4
- IDEA
- AES
- CAMELLIA, SEED, ARIA
- ChaCha20-Poly1305

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



HMAC-
Algorithmus

Hashfunktion für HMAC.
Mögliche Werte: MD5, SHA,
SHA-256, SHA-386

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



HMAC-
Algorithmus

Hashfunktion für HMAC.

Mögliche Werte: MD5, SHA,
SHA-256, SHA-386

- TLS 1.0, 1.1: Nur für HMAC
im Record Layer

Ciphersuites

TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA



HMAC-
Algorithmus

Hashfunktion für HMAC.

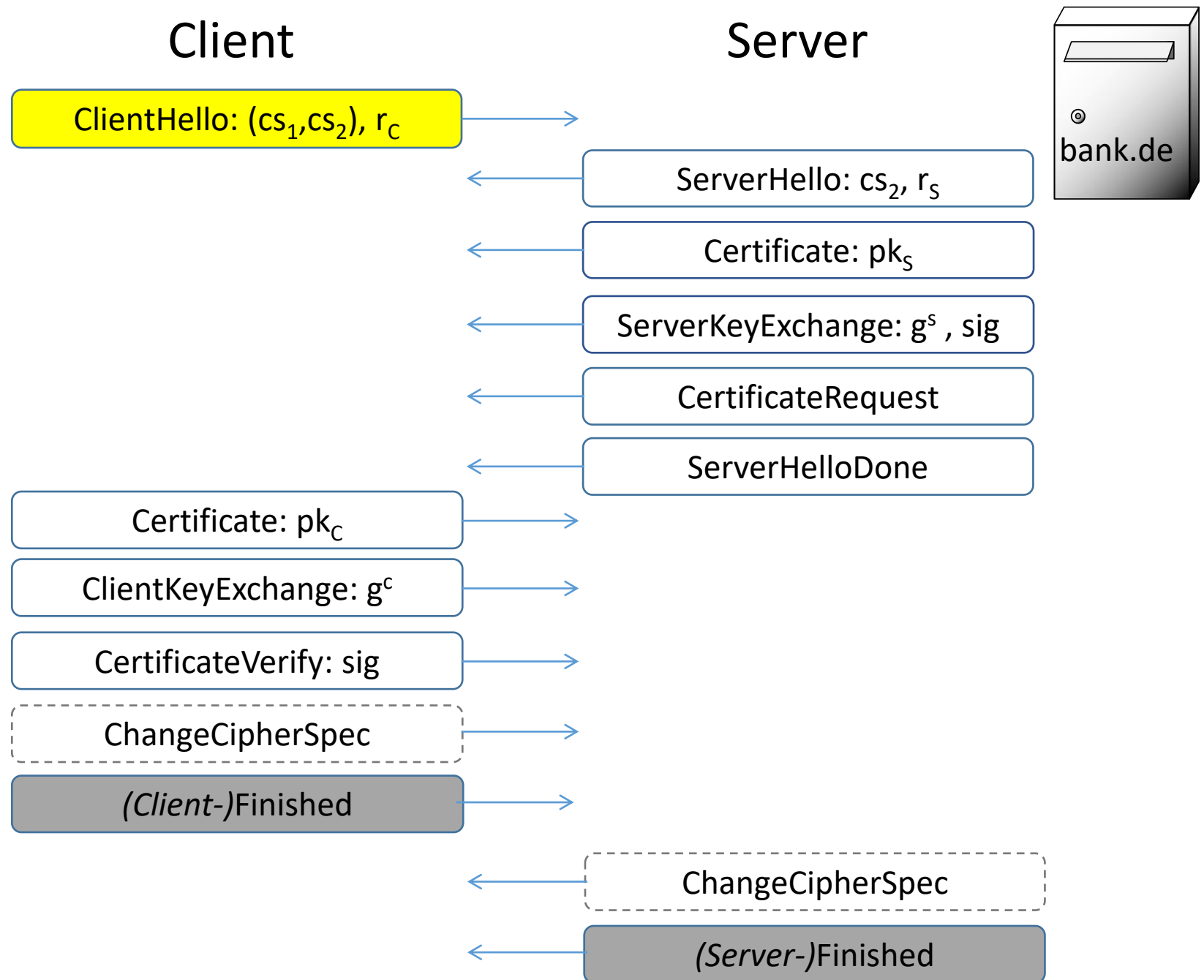
Mögliche Werte: MD5, SHA,
SHA-256, SHA-386

- TLS 1.0, 1.1: Nur für HMAC im Record Layer
- TLS 1.2: Für alle HMAC-Konstruktionen

2.3 TLS Handshake

2.3.3 Abgleich der Fähigkeiten: ClientHello und ServerHello

TLS Handshake



ClientHello !

Record Layer Header:
Auch Handshake-Nachrichten
werden über den Record Layer
übertragen (unverschlüsselt)

Typ: 22	Version: 3.0		Länge...
...Länge	Nachr: 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge CipherSuites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

ClientHello !

Typ der Handshake-Nachricht:
ClientHello

Typ: 22	Version: 3.0	Länge...
...Länge	Nachr: 1	Länge der ...
...Nachricht	version: 3.0	
ClientRandom (32 Byte)		
		Länge ID
SessionID (≤32 Byte)		
Länge CipherSuites	CipherSuite 1	
CipherSuite 2		
...		
	CipherSuite n	
Länge	Komp. 1	... Komp m

ClientHello !

Größte vom Client unterstützte
TLS-Versionsnummer
Hier: SSL 3.0

Typ: 22	Version: 3.0		Länge...
...Länge	Nachr. 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge CipherSuites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

ClientHello !

4 Byte : The current time and date in standard UNIX 32-bit format (seconds since the midnight starting Jan 1, 1970, UTC, ignoring leap seconds) according to the sender's internal clock.

28 Byte : random

Typ: 22	Version: 3.0		Länge...
...Länge	Nachr: 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge CipherSuites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

ClientHello !

Wird nur gesendet wenn
TLS Session Resumption
vom Client gewünscht wird

Typ: 22	Version: 3.0		Länge...
...Länge	Nachr: 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge Cipher Suites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

ClientHello !

Liste der Ciphersuites

Typ: 22	Version: 3.0		Länge...
...Länge	Nachr: 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge CipherSuites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

ClientHello !

Optional: Liste der
Kompressionsalgorithmen

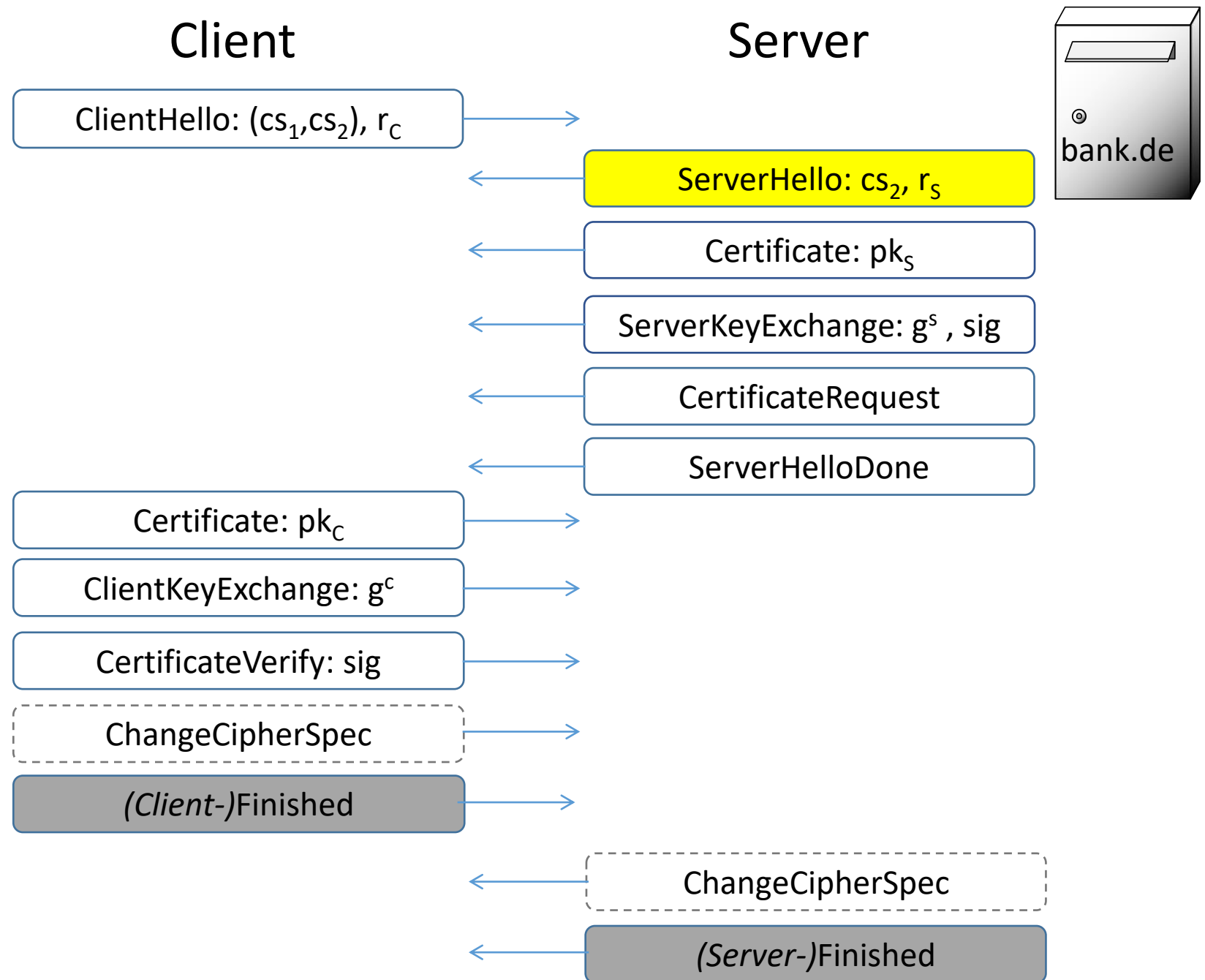
Typ: 22	Version: 3.0		Länge...
...Länge	Nachr: 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge CipherSuites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

ClientHello !

Optional: Liste von Extensions

Typ: 22	Version: 3.0		Länge...
...Länge	Nachr: 1	Länge der ...	
...Nachricht	Version: 3.0		
ClientRandom (32 Byte)			
			Länge ID
SessionID (≤32 Byte)			
Länge CiperSuites		CipherSuite 1	
CipherSuite 2			
		...	
		CipherSuite n	
Länge	Komp. 1	...	Komp m

TLS Handshake



ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob
 - In seiner Datenbank ein MasterSecret unter dieser SessionID abgelegt ist, oder

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob
 - In seiner Datenbank ein MasterSecret unter dieser SessionID abgelegt ist, oder
 - Ob die ClientHello-Nachricht ein TLS Session Ticket enthält

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob
 - In seiner Datenbank ein MasterSecret unter dieser SessionID abgelegt ist, oder
 - Ob die ClientHello-Nachricht ein TLS Session Ticket enthält
 - Fällt diese Prüfungen positiv aus, so kann der Server mit der gleichen SessionID antworten. Dadurch bestätigt er, dass TLS Session Resumption durchgeführt wird.

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob
 - In seiner Datenbank ein MasterSecret unter dieser SessionID abgelegt ist, oder
 - Ob die ClientHello-Nachricht ein TLS Session Ticket enthält
 - Fällt diese Prüfungen positiv aus, so kann der Server mit der gleichen SessionID antworten. Dadurch bestätigt er, dass TLS Session Resumption durchgeführt wird.
 - Ansonsten gibt er in diesem Feld eine neue SessionID vor. In diesem Fall wird der 'normale' Handshake fortgesetzt.

ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob
 - In seiner Datenbank ein MasterSecret unter dieser SessionID abgelegt ist, oder
 - Ob die ClientHello-Nachricht ein TLS Session Ticket enthält
 - Fällt diese Prüfungen positiv aus, so kann der Server mit der gleichen SessionID antworten. Dadurch bestätigt er, dass TLS Session Resumption durchgeführt wird.
 - Ansonsten gibt er in diesem Feld eine neue SessionID vor. In diesem Fall wird der 'normale' Handshake fortgesetzt.
- CipherSuite (2 Byte): Die vom Server ausgewählte CipherSuite.

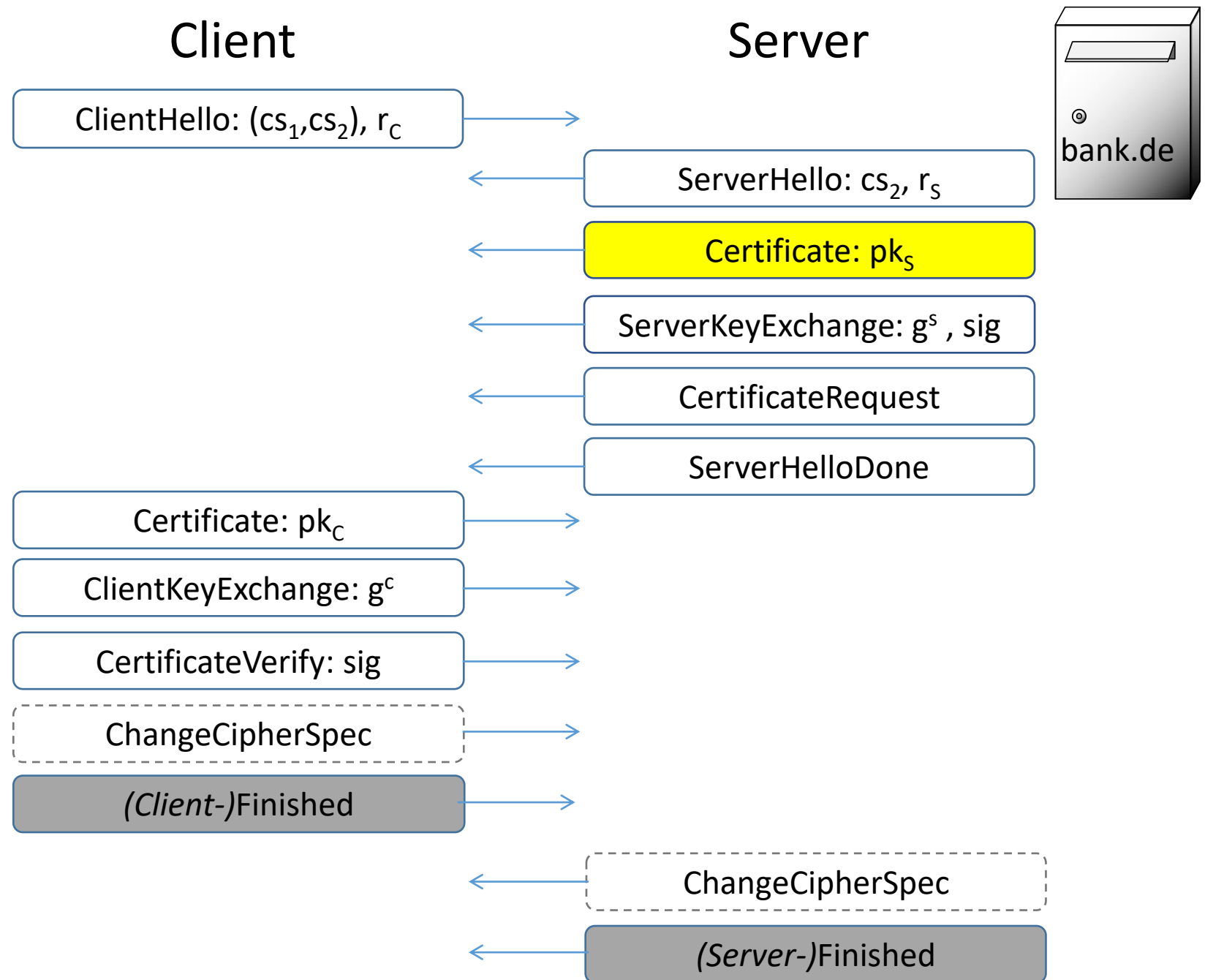
ServerHello

- ProtocolVersion (2 Byte): Hier muss der Server eine TLS-Protokollversion wählen, die gleich oder kleiner der vom Client vorgeschlagenen ist.
- ServerRandom (32 Byte): Ein Zufallswert, der analog zum Client-Random gebildet wird.
- SessionID (bis zu 32 Byte): Mandatory.
 - War in ClientHello eine SessionID enthalten, so prüft der Server, ob
 - In seiner Datenbank ein MasterSecret unter dieser SessionID abgelegt ist, oder
 - Ob die ClientHello-Nachricht ein TLS Session Ticket enthält
 - Fällt diese Prüfungen positiv aus, so kann der Server mit der gleichen SessionID antworten. Dadurch bestätigt er, dass TLS Session Resumption durchgeführt wird.
 - Ansonsten gibt er in diesem Feld eine neue SessionID vor. In diesem Fall wird der 'normale' Handshake fortgesetzt.
- CipherSuite (2 Byte): Die vom Server ausgewählte CipherSuite.
- CompressionMethod (1 Byte): Die vom Server ausgewählte Kompressionsmethode.

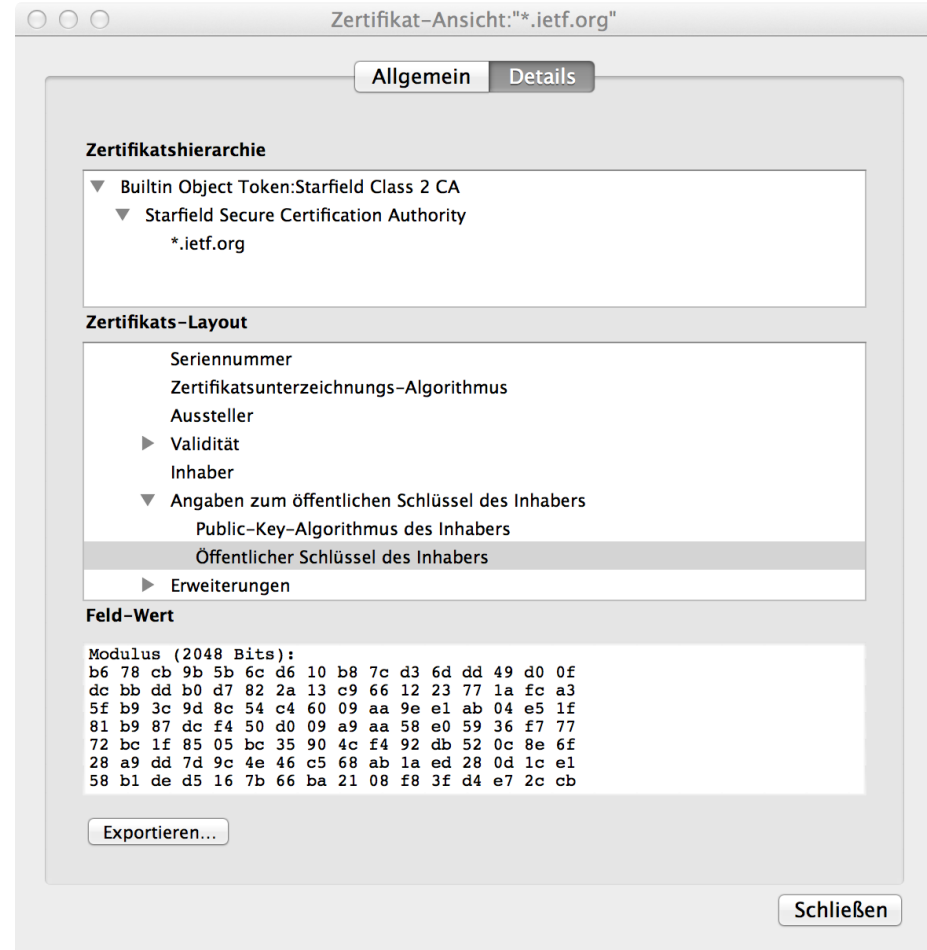
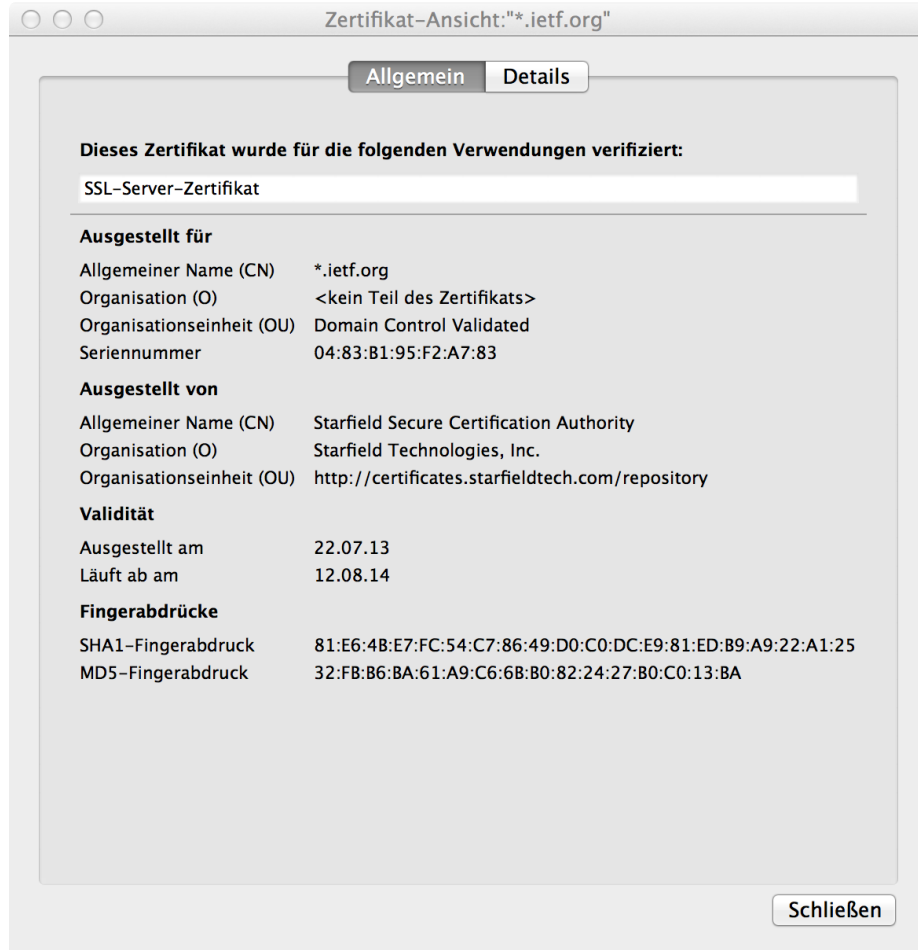
2.3 TLS Handshake

2.3.4 Schlüsselaustausch: Certificate, ServerKeyExchange und ClientKeyExchange

TLS Handshake



Certificate



Certificate

- TLS-RSA
- TLS-(EC)DH
- TLS-(EC)DHE

Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
- TLS-(EC)DH
- TLS-(EC)DHE

Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
 - Key Usage: Verschlüsselung(, Signatur)
- TLS-(EC)DH
- TLS-(EC)DHE

Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
 - Key Usage: Verschlüsselung(, Signatur)
- TLS-(EC)DH
 - Zertifikat mit öffentlichem (EC)DH-Schlüssel
- TLS-(EC)DHE

Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
 - Key Usage: Verschlüsselung(, Signatur)
- TLS-(EC)DH
 - Zertifikat mit öffentlichem (EC)DH-Schlüssel
 - Key Usage: Key Exchange(, Signatur, Verschlüsselung)
- TLS-(EC)DHE

Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
 - Key Usage: Verschlüsselung(, Signatur)
- TLS-(EC)DH
 - Zertifikat mit öffentlichem (EC)DH-Schlüssel
 - Key Usage: Key Exchange(, Signatur, Verschlüsselung)
- TLS-(EC)DHE
 - Zertifikat mit RSA oder (EC)DH-Schlüssel

Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
 - Key Usage: Verschlüsselung(, Signatur)
- TLS-(EC)DH
 - Zertifikat mit öffentlichem (EC)DH-Schlüssel
 - Key Usage: Key Exchange(, Signatur, Verschlüsselung)
- TLS-(EC)DHE
 - Zertifikat mit RSA oder (EC)DH-Schlüssel
 - Key Usage: Signatur(, Verschlüsselung, Key Exchange)

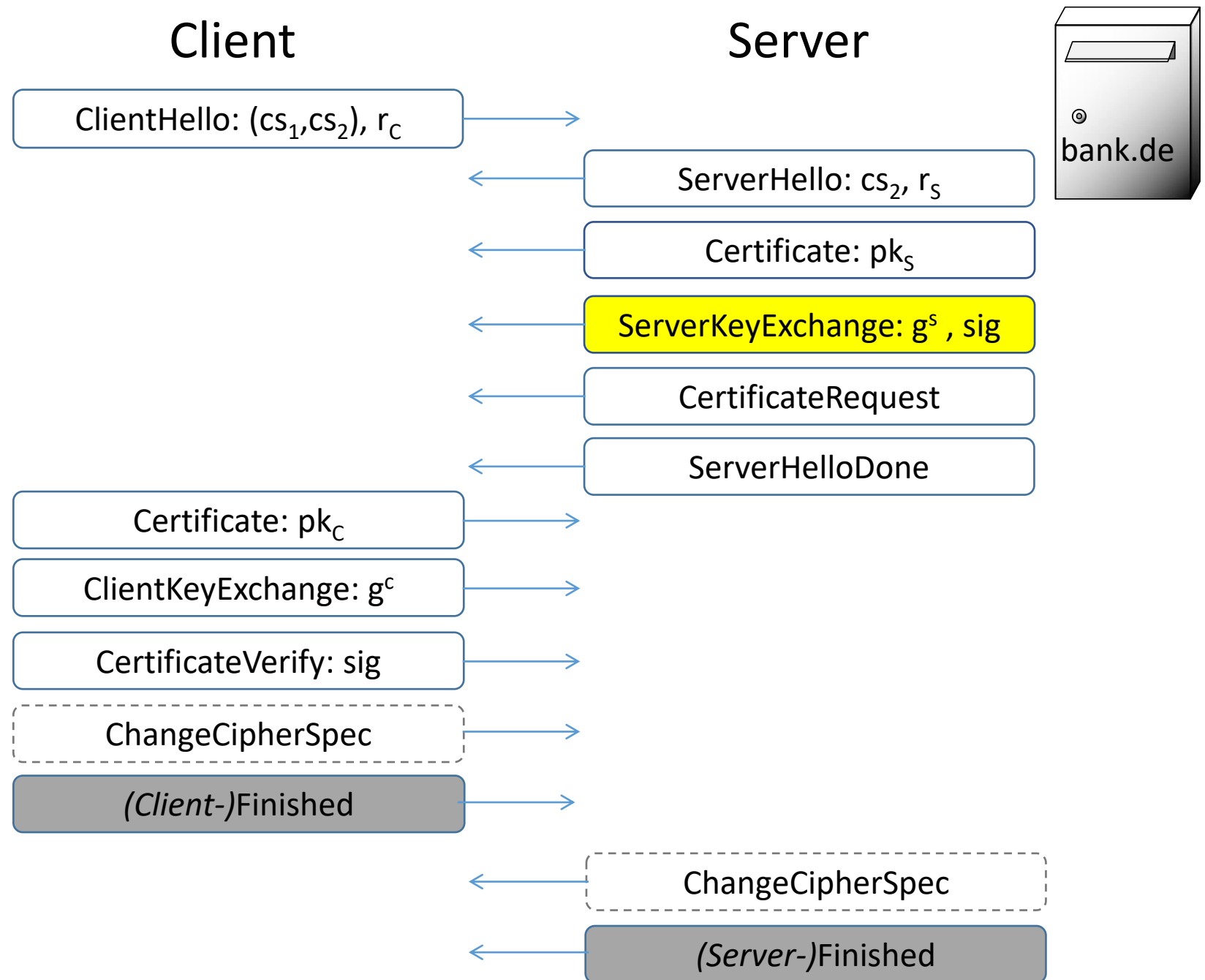
Certificate

- TLS-RSA
 - Zertifikate mit öffentlichem RSA-Schlüssel
 - Key Usage: Verschlüsselung(, Signatur)
- TLS-(EC)DH
 - Zertifikat mit öffentlichem (EC)DH-Schlüssel
 - Key Usage: Key Exchange(, Signatur, Verschlüsselung)
- TLS-(EC)DHE
 - Zertifikat mit RSA oder (EC)DH-Schlüssel
 - Key Usage: Signatur(, Verschlüsselung, Key Exchange)



Kein einzelnes Zertifikat kann alle drei Ciphersuitefamilien abdecken

TLS Handshake



ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = $(g, G, g^s, \text{Signatur})$

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s, Signatur)



Festlegung der mathematischen Gruppe, in der DHKE ausgeführt wird

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s, Signatur)



Festlegung der mathematischen Gruppe, in der DHKE ausgeführt wird

- Primzahlgruppen: Primzahl p , Gruppenelement g

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s, Signatur)

Festlegung der mathematischen Gruppe, in der DHKE ausgeführt wird

- Primzahlgruppen: Primzahl p , Gruppenelement g
- EC-Gruppen:
 - Name der EC-Gruppe oder

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s , Signatur)

Festlegung der mathematischen Gruppe, in der DHKE ausgeführt wird

- Primzahlgruppen: Primzahl p , Gruppenelement g
- EC-Gruppen:
 - Name der EC-Gruppe oder
 - Endlicher Körper $GF(q)$, Parameter (a,b) , Punkt $P=(x,y)$

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s , Signatur)



Diffie-Hellman-Share des Servers

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s, **Signatur**)



Digitale Signatur

ServerKeyExchange – TLS-(EC)DHE

$\text{ServerKeyExchange} = (g, G, g^s, \text{Signatur})$

$\text{Signatur} = \text{SIG.Sign}_{sk}(r_C, r_S, g, G, g^s)$

Die Signatur wird gebildet über

ServerKeyExchange – TLS-(EC)DHE

ServerKeyExchange = (g, G, g^s, Signatur)

Signatur = SIG.Sign_{sk}(r_C, r_S, g, G, g^s)

Die Signatur wird gebildet über

- die Parameter aus ServerKeyExchange

ServerKeyExchange – TLS-(EC)DHE

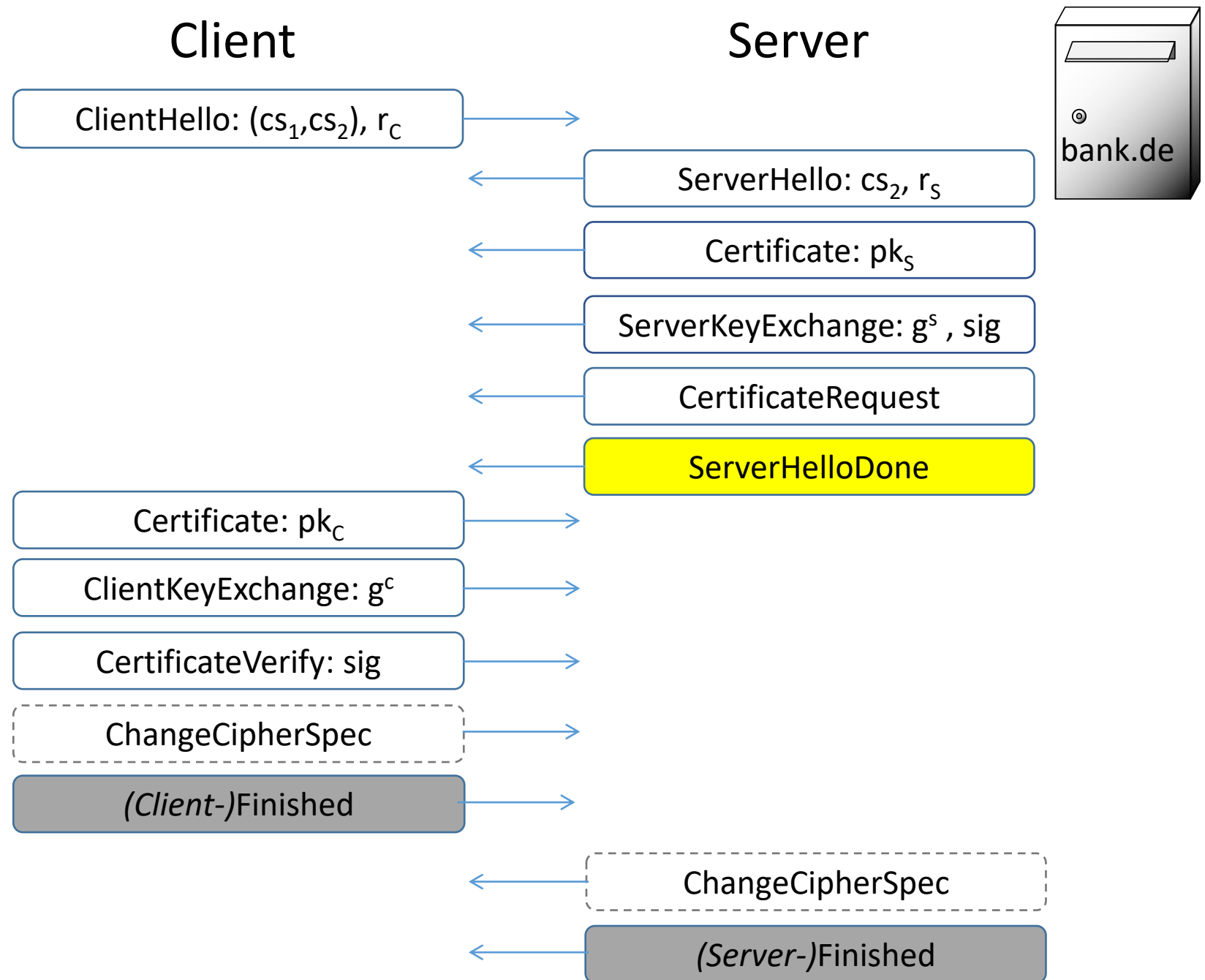
ServerKeyExchange = (g, G, g^s, Signatur)

Signatur = SIG.Sign_{sk}(r_C, r_S, g, G, g^s)

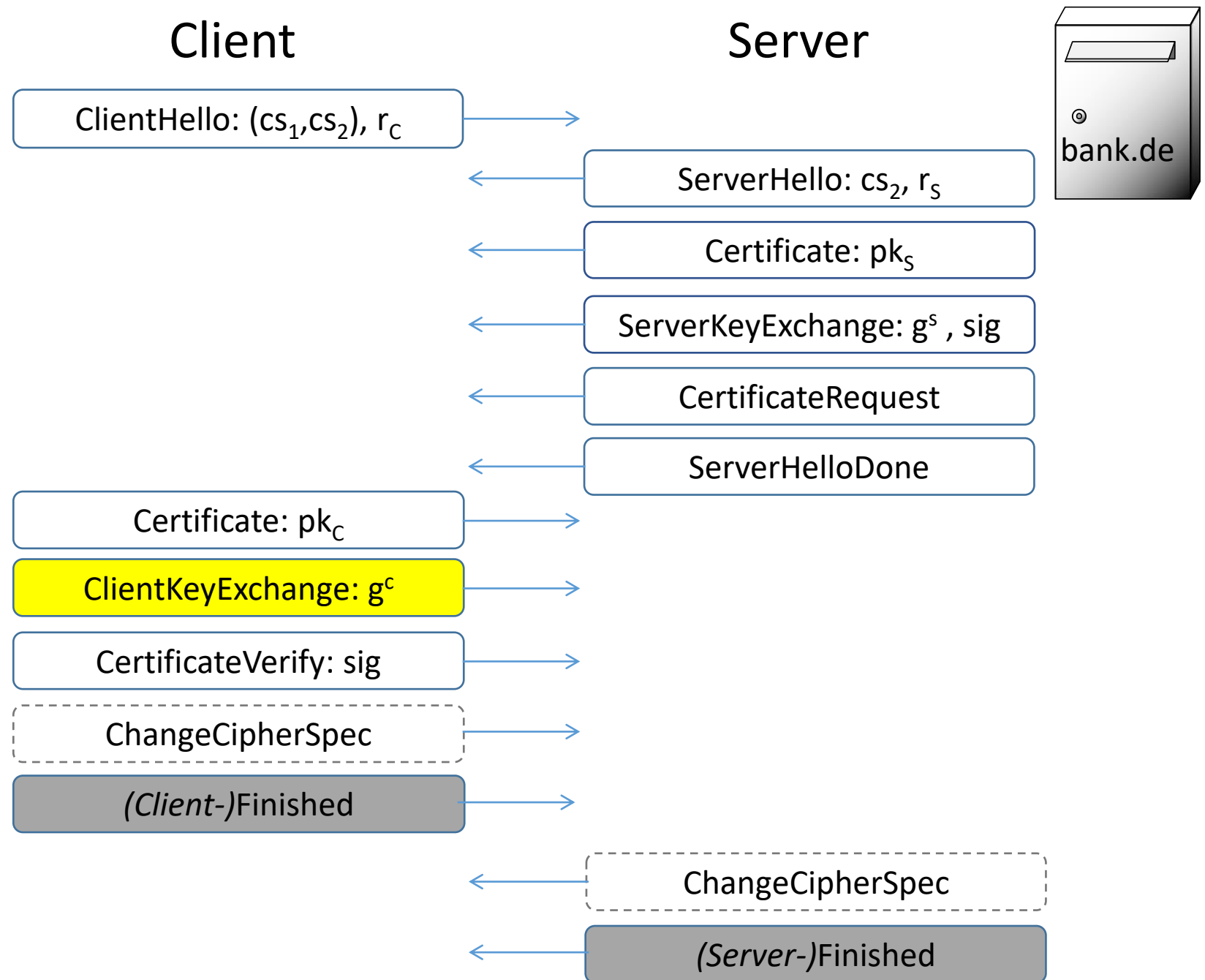
Die Signatur wird gebildet über

- die Parameter aus ServerKeyExchange
- die beiden Zufallswerte aus ClientHello und ServerHello

TLS Handshake



TLS Handshake



ClientKeyExchange

TLS-RSA

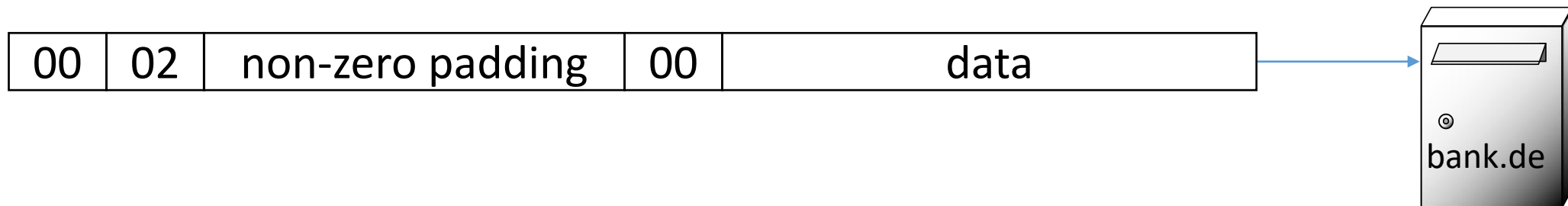
- RSA-Verschlüsselung mit dem Public Key des Servers von



ClientKeyExchange

TLS-RSA

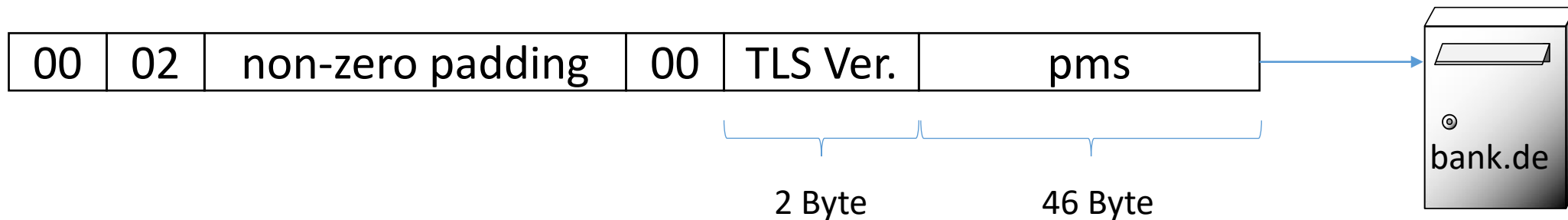
- RSA-Verschlüsselung mit dem Public Key des Servers von
- PKCS#1-Codierung von



ClientKeyExchange

TLS-RSA

- RSA-Verschlüsselung mit dem Public Key des Servers von
- PKCS#1-Codierung von
 - 2 Byte TLS Versionsnummer
 - 46 zufällig gewählten Bytes des PremasterSecret pms



ClientKeyExchange

TLS-RSA

- RSA-Verschlüsselung mit dem Public Key des Servers von
- PKCS#1-Codierung von
 - 2 Byte TLS Versionsnummer
 - 46 zufällig gewählten Bytes des PremasterSecret pms

TLS-DHE und TLS-DH

- g^c

ClientKeyExchange

TLS-RSA

- RSA-Verschlüsselung mit dem Public Key des Servers von
- PKCS#1-Codierung von
 - 2 Byte TLS Versionsnummer
 - 46 zufällig gewählten Bytes des PremasterSecret pms

TLS-DHE und TLS-DH

- g^c

TLS-ECDHE und TLS-ECDH

- cP

Übersicht

Schlüssel-austausch-algorithm.	Benötigter Zertifikats-typ	ServerKey-Exchange benötigt?	Inhalt ClientKey-Exchange	Beschreibung
RSA	RSA Encryption	Nein	Verschlüsseltes Premaster-Secret	Client verschlüsselt Premaster Secret mit öffentlichem Schlüssel des Servers.
RSA Export	RSA Signing	Ja (temporärer RSA-Schlüssel ≤ 512 bit)	Mit temp. RSA-Schlüssel ver-schlüsseltes Premaster Secret	Client verschlüsselt Premaster Secret mit temporärem RSA-Schlüssel des Servers (nur noch relevant wegen Rückwärts-kompatibilität).
DHE - DSS	DSS Signing	Ja ($g^s \bmod p$)	$g^c \bmod p$	Diffie - Hellman - Schlüsselvereinbarung, Server signiert $g^s \bmod p$ mit dem DSS - Schlüssel.
DHE - RSA	RSA Signing	Ja ($g^s \bmod p$)	$g^c \bmod p$	Diffie - Hellman - Schlüsselvereinbarung, Server signiert $g^s \bmod p$ mit dem RSA - Schlüssel.
DH - DSS	DH, signiert mit DSS	Nein ($g^c \bmod p$ im Zertifikat enthalten)	$g^c \bmod p$	Diffie-Hellman-Schlüsselvereinbarung mit festem Serveranteil, Authentisierung über DSS-Zertifikat.
DH - RSA	DH, signiert mit RSA	Nein ($g^c \bmod p$ im Zertifikat enthalten)	$g^c \bmod p$	Diffie-Hellman-Schlüsselvereinbarung mit festem Serveranteil, Authentisierung über RSA-Zertifikat.

2.3 TLS Handshake

2.3.5 Ableitung der Schlüssel

Start: Premaster Secret

- TLS-RSA
- TLS-DH, TLS-DHE
- TLS-ECDH, TLS-ECDHE

Start: Premaster Secret

- TLS-RSA
 - 2 Byte TLS-Versionsnummer || 46 zufällige Byte
- TLS-DH, TLS-DHE
- TLS-ECDH, TLS-ECDHE

48 Byte

Start: Premaster Secret

- TLS-RSA
 - 2 Byte TLS-Versionsnummer || 46 zufällige Byte
- TLS-DH, TLS-DHE
 - g^{xy} ohne führende Nullbytes
- TLS-ECDH, TLS-ECDHE

48 Byte

Länge des Modulus p

Start: Premaster Secret

- TLS-RSA
 - 2 Byte TLS-Versionsnummer || 46 zufällige Byte
- TLS-DH, TLS-DHE
 - g^{xy} ohne führende Nullbytes
- TLS-ECDH, TLS-ECDHE
 - X-Koordinate von xyP

48 Byte

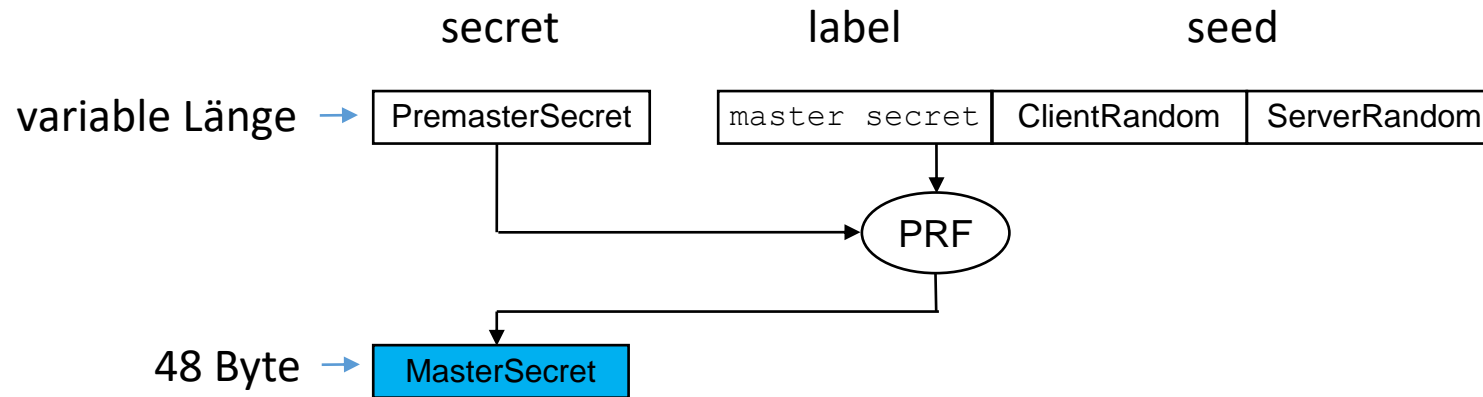
Länge des Modulus p

Element aus $GF(q)$

MasterSecret

```
RFC 5246:      master_secret = PRF(pre_master_secret, "master secret",
                                   ClientHello.random + ServerHello.random)
                                   [0..47];
```

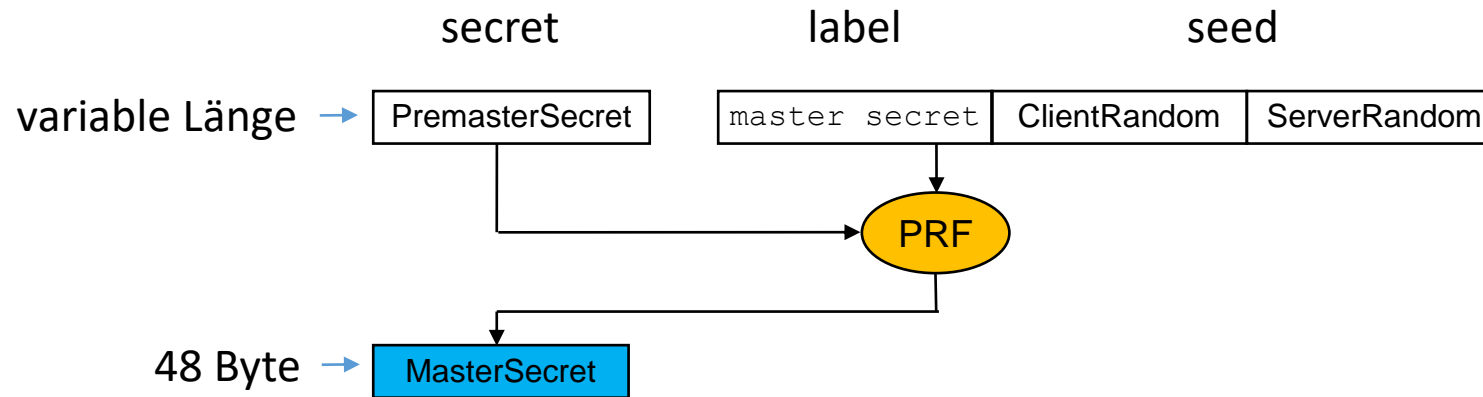
MasterSecret



RFC 5246:

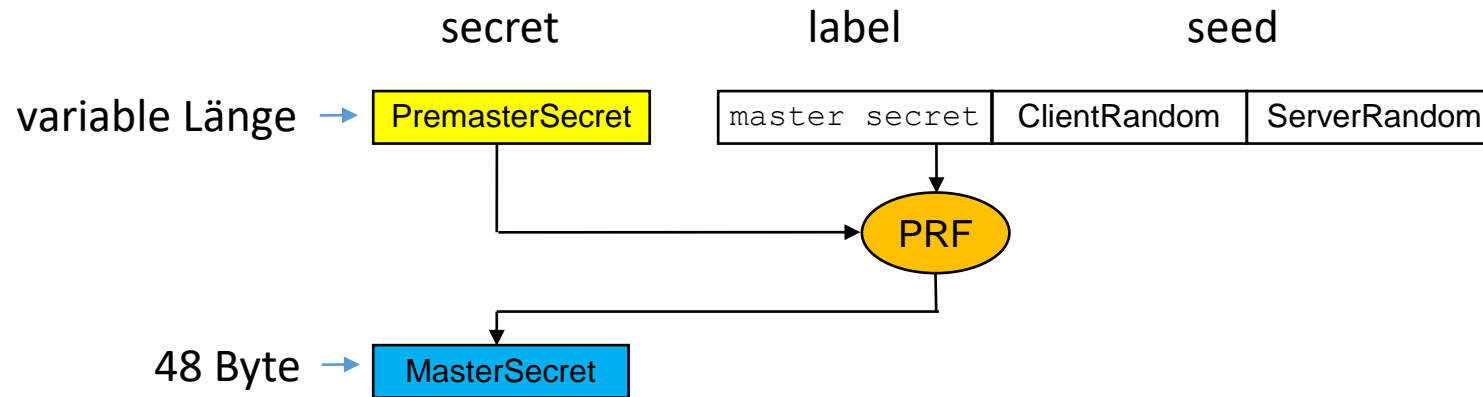
```
master_secret = PRF(pre_master_secret, "master secret",  
                    ClientHello.random + ServerHello.random)  
                    [0..47];
```

MasterSecret



RFC 5246: `master_secret` = `PRF`(`pre_master_secret`, "master secret",
ClientHello.random + ServerHello.random)
[0..47];

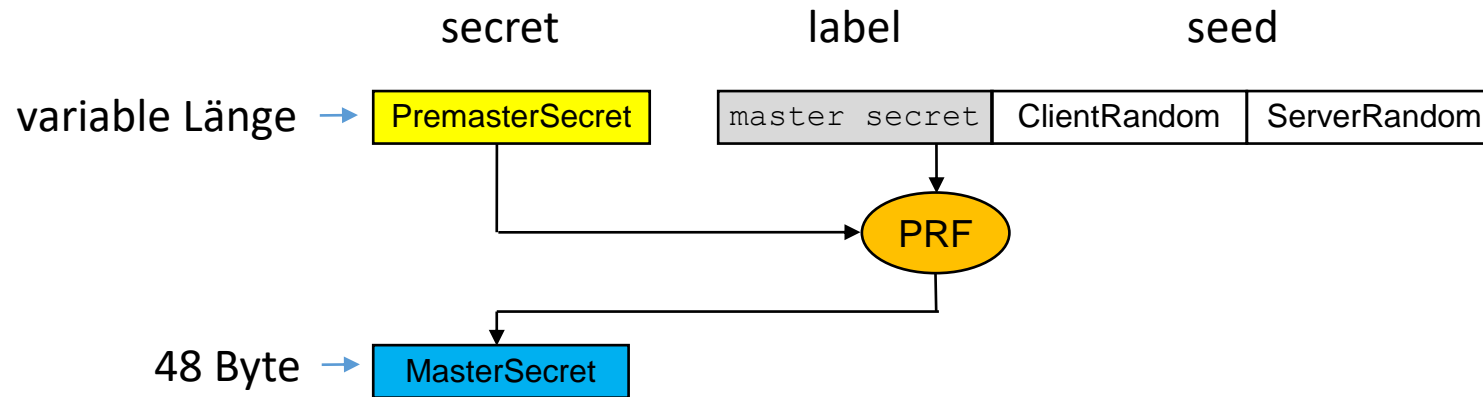
MasterSecret



RFC 5246:

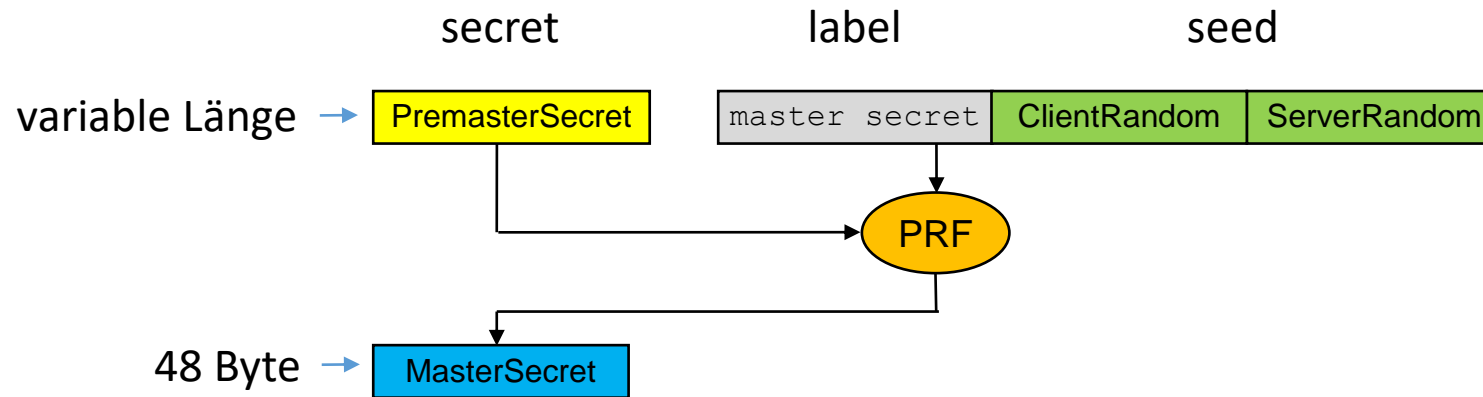
```
master_secret = PRF(pre_master_secret, "master secret",  
                    ClientHello.random + ServerHello.random)  
                    [0..47];
```

MasterSecret



RFC 5246: `master_secret` = `PRF`(`pre_master_secret`, `"master secret"`,
`ClientHello.random + ServerHello.random`
`[0..47]`);

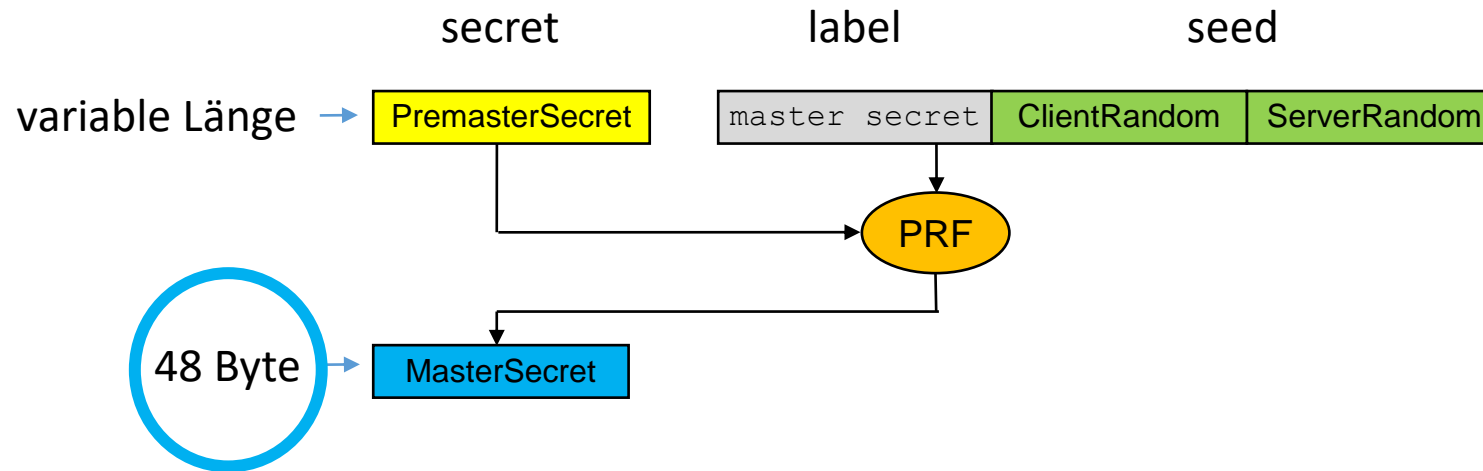
MasterSecret



RFC 5246:

```
master_secret = PRF(pre_master_secret, "master secret",  
ClientHello.random + ServerHello.random)  
[0..47];
```

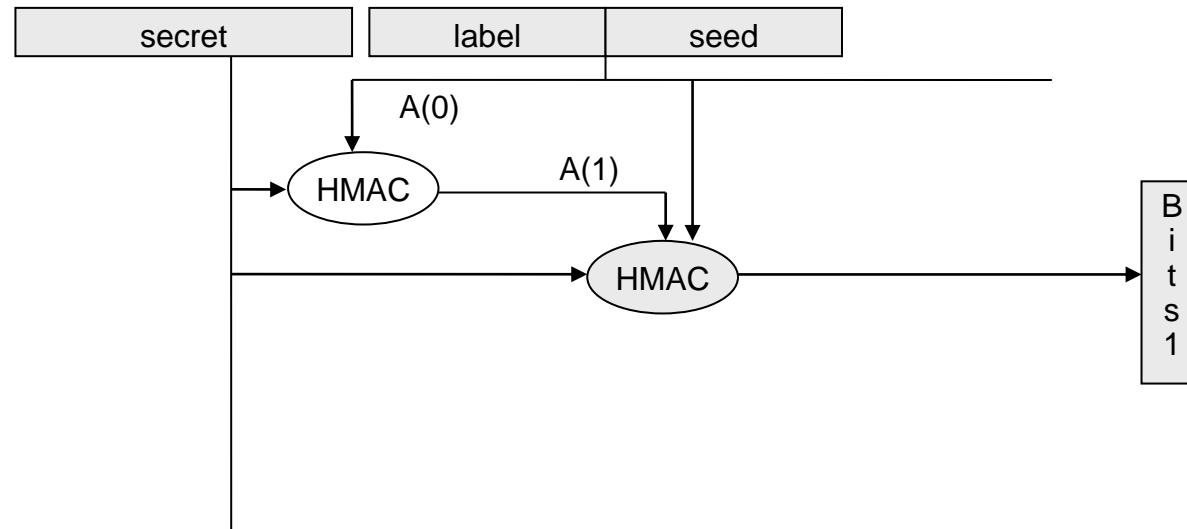
MasterSecret



RFC 5246:

```
master_secret = PRF(pre_master_secret, "master secret",
                    ClientHello.random + ServerHello.random)
                    [0..47];
```

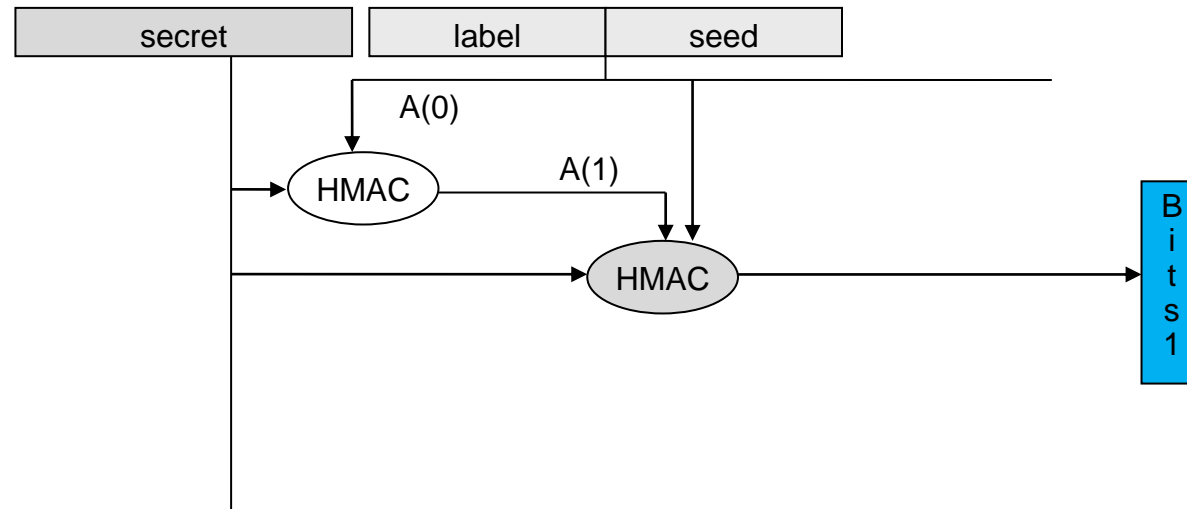
Die TLS-PRF



RFC 5246:

In this section, we define one PRF, based on HMAC. This PRF with the SHA-256 hash function is used for all cipher suites defined in this document and in TLS documents published prior to this document when TLS 1.2 is negotiated. New cipher suites **MUST** explicitly specify a PRF and, in general, **SHOULD** use the TLS PRF with SHA-256 or a stronger standard hash function.

Die TLS-PRF



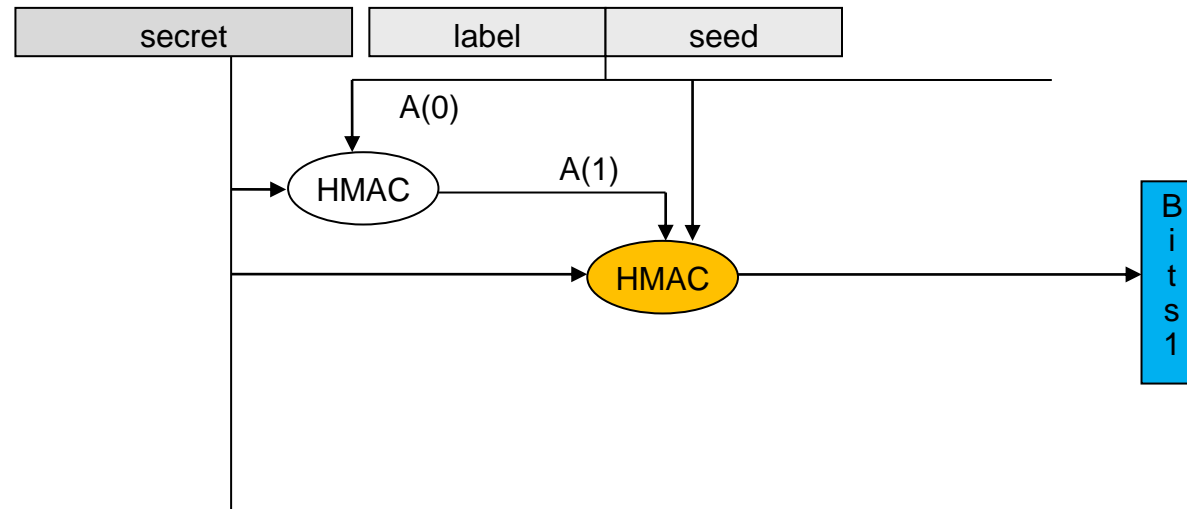
RFC 5246: `P_hash(secret, seed)` = HMAC_hash(secret, A(1) + seed) +
...

A() is defined as:

$A(0) = \text{seed}$

$A(i) = \text{HMAC_hash}(\text{secret}, A(i-1))$

Die TLS-PRF



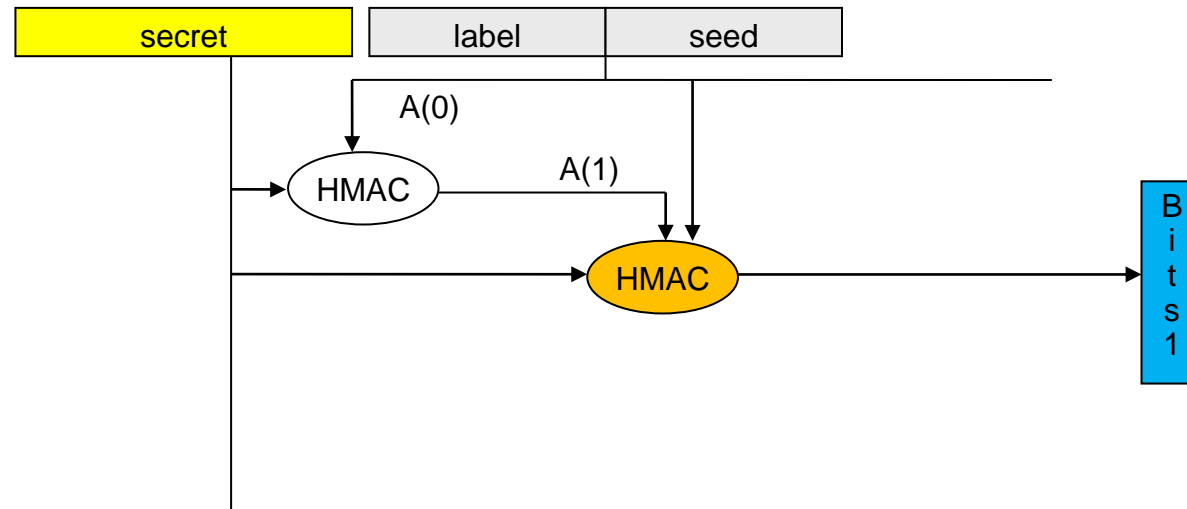
RFC 5246: $P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) + \dots$

$A()$ is defined as:

$A(0) = seed$

$A(i) = HMAC_hash(secret, A(i-1))$

Die TLS-PRF



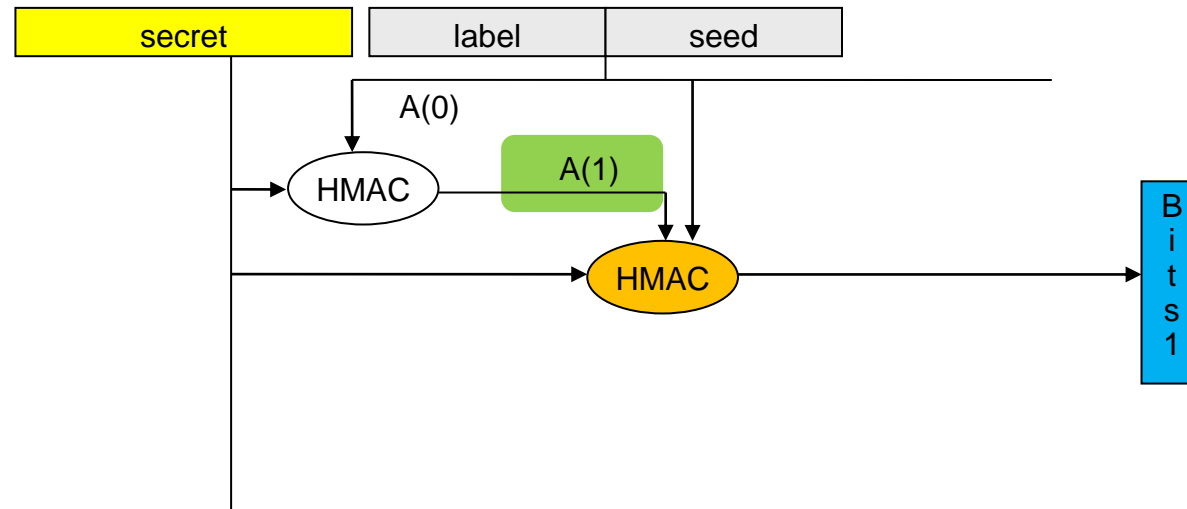
RFC 5246: $P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) + \dots$

$A()$ is defined as:

$A(0) = seed$

$A(i) = HMAC_hash(secret, A(i-1))$

Die TLS-PRF



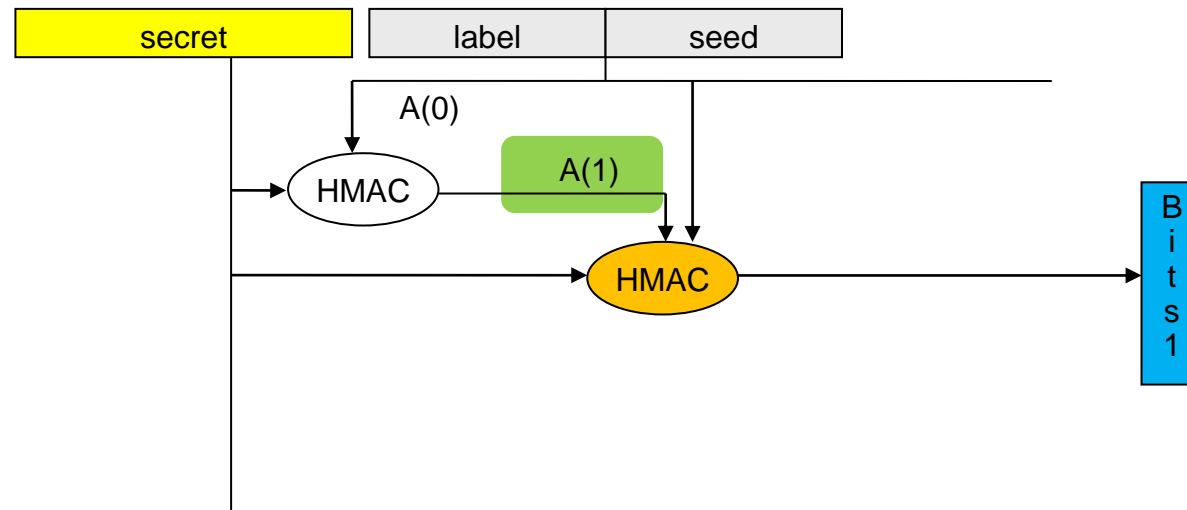
RFC 5246: $P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) + \dots$

$A()$ is defined as:

$A(0) = seed$

$A(i) = HMAC_hash(secret, A(i-1))$

Die TLS-PRF



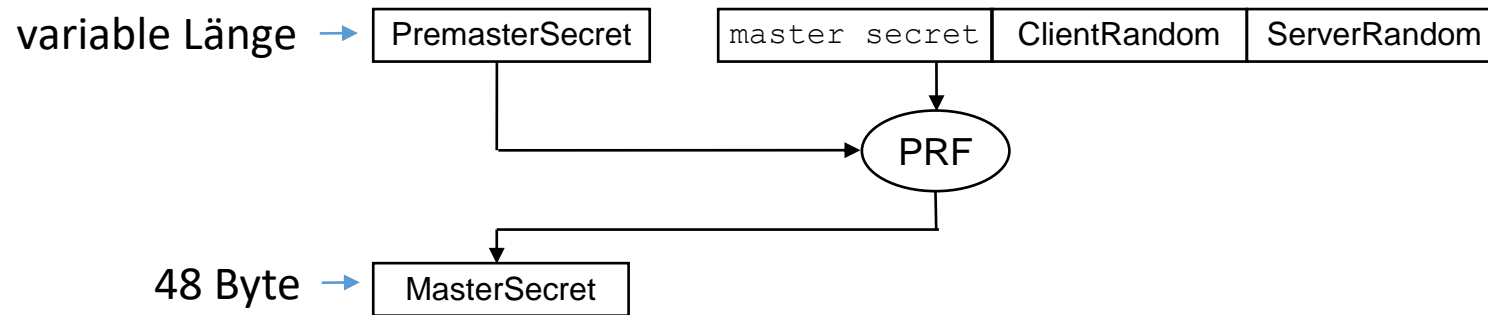
RFC 5246: $P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) + \dots$

$A()$ is defined as:

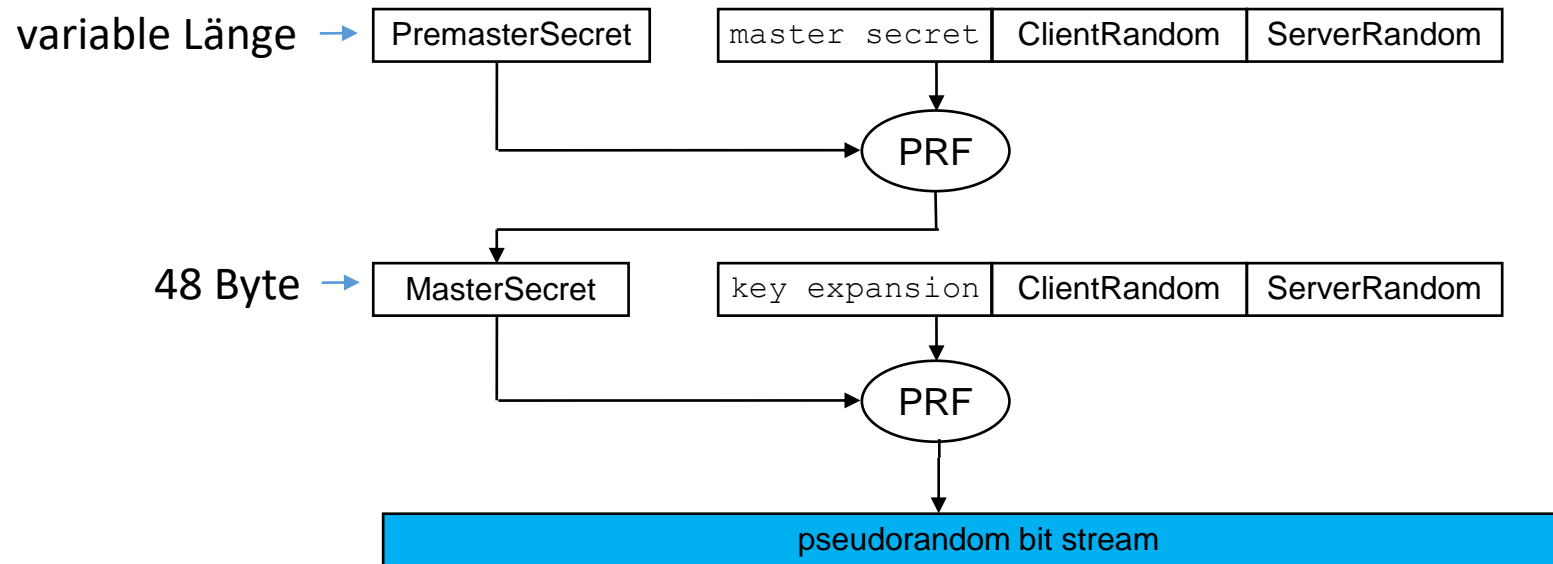
$A(0) = seed$

$A(i) = HMAC_hash(secret, A(i-1))$

Schlüsselmaterial für Record Layer

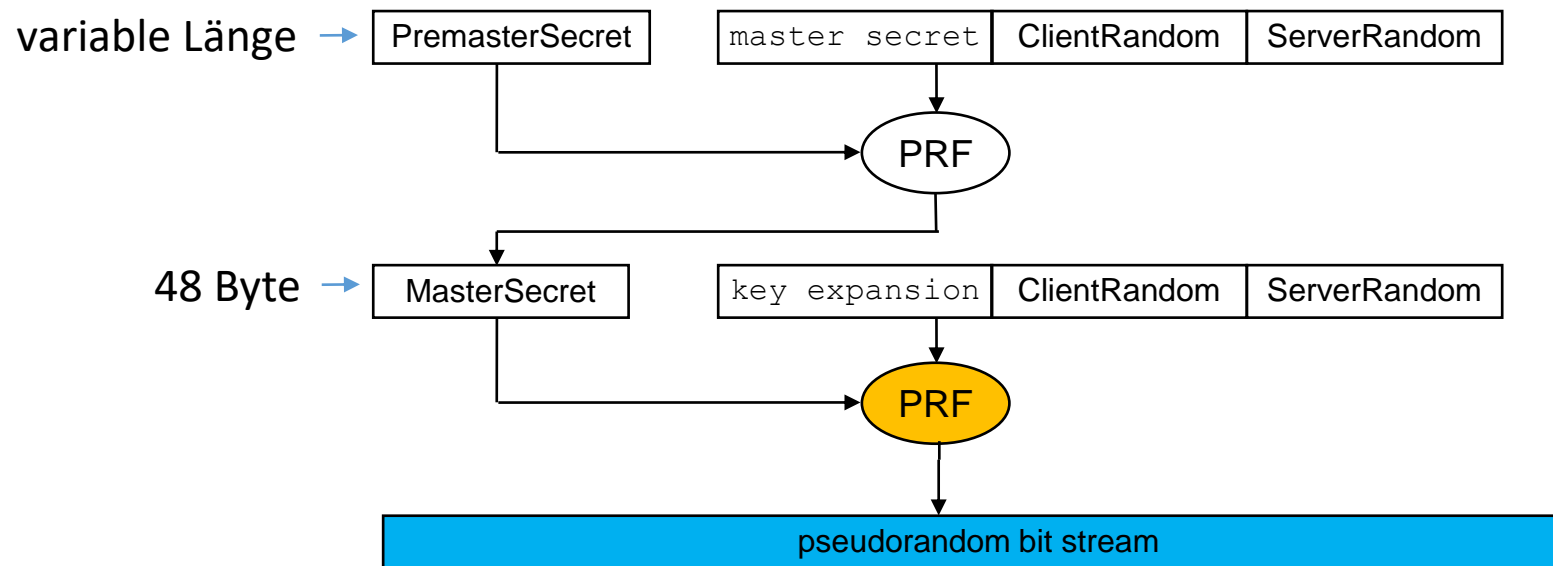


Schlüsselmaterial für Record Layer



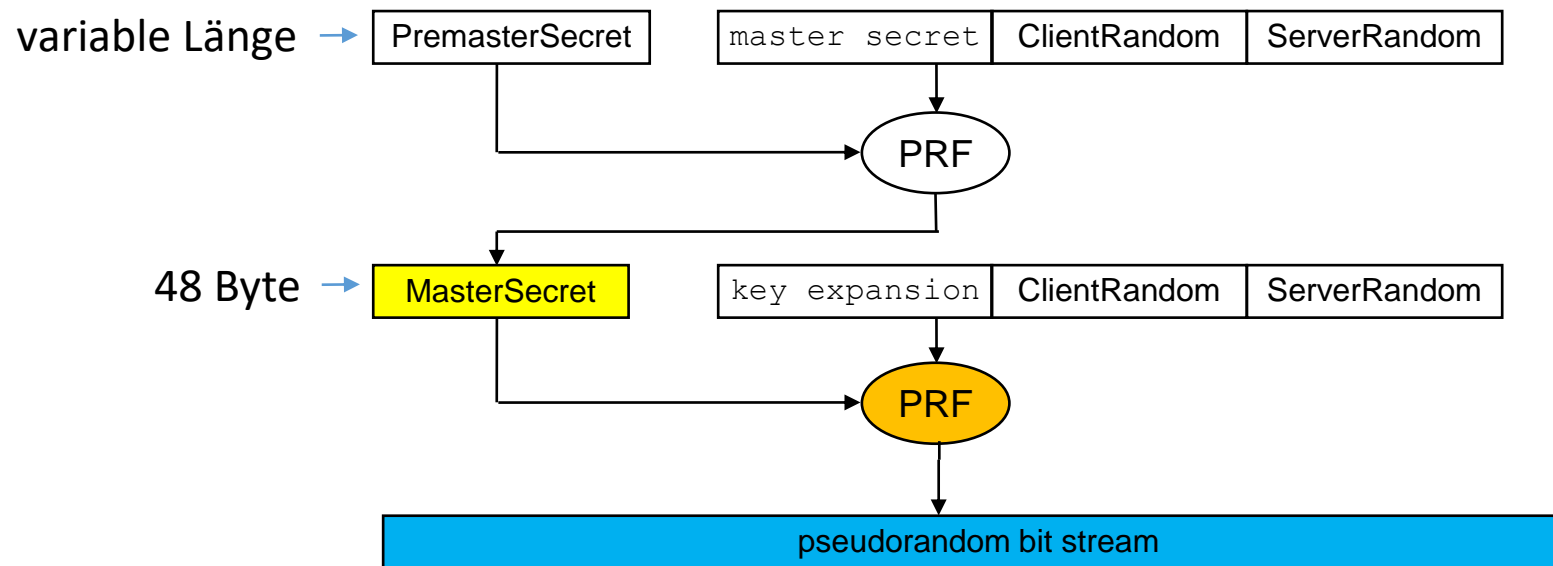
RFC 5246: `key_block = PRF(SecurityParameters.master_secret, "key expansion", SecurityParameters.server_random + SecurityParameters.client_random);`

Schlüsselmaterial für Record Layer



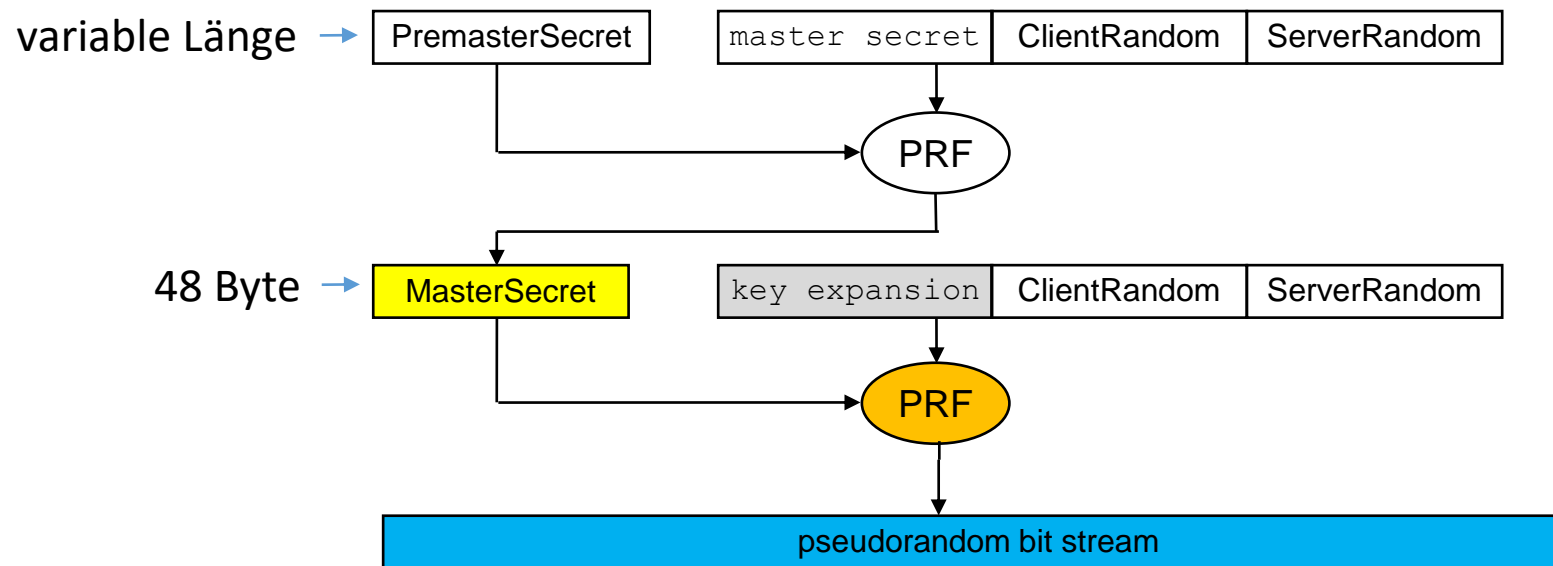
RFC 5246: `key_block` = `PRF`(`SecurityParameters.master_secret`, "key expansion",
`SecurityParameters.server_random` + `SecurityParameters.client_random`);

Schlüsselmaterial für Record Layer



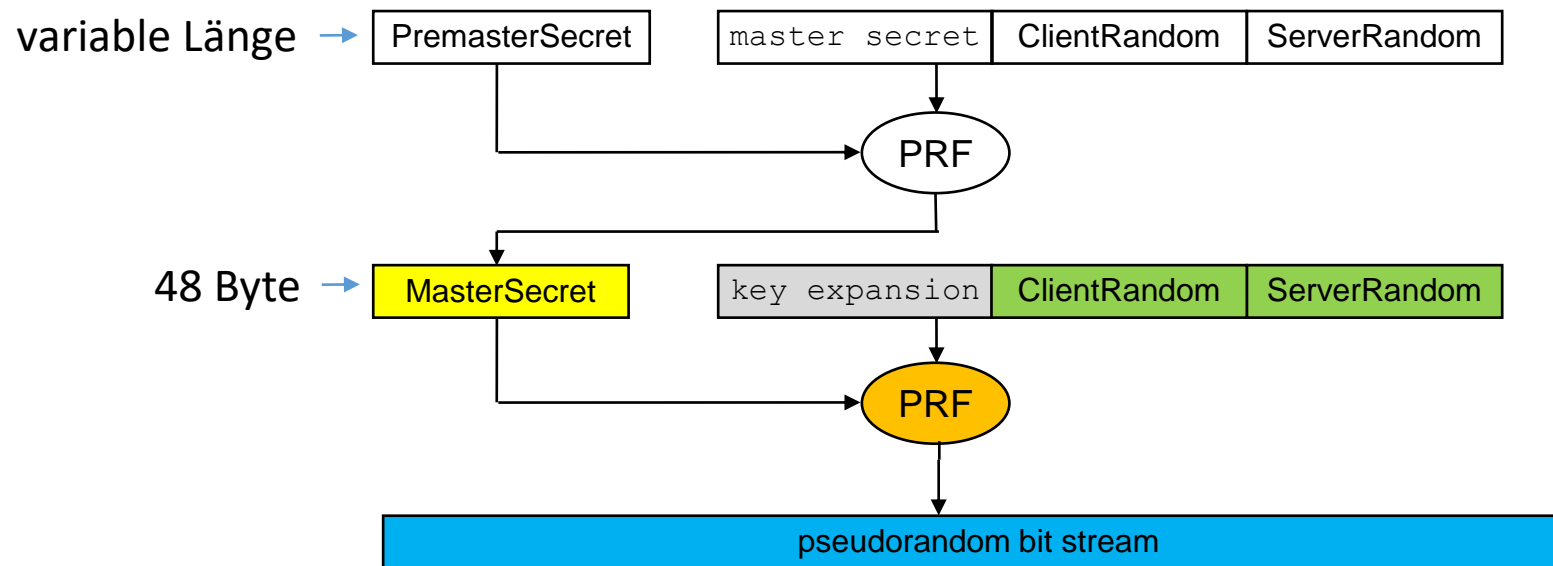
RFC 5246: `key_block = PRF(SecurityParameters.master_secret, "key expansion", SecurityParameters.server_random + SecurityParameters.client_random);`

Schlüsselmaterial für Record Layer



RFC 5246: `key_block = PRF(SecurityParameters.master_secret, "key expansion", SecurityParameters.server_random + SecurityParameters.client_random);`

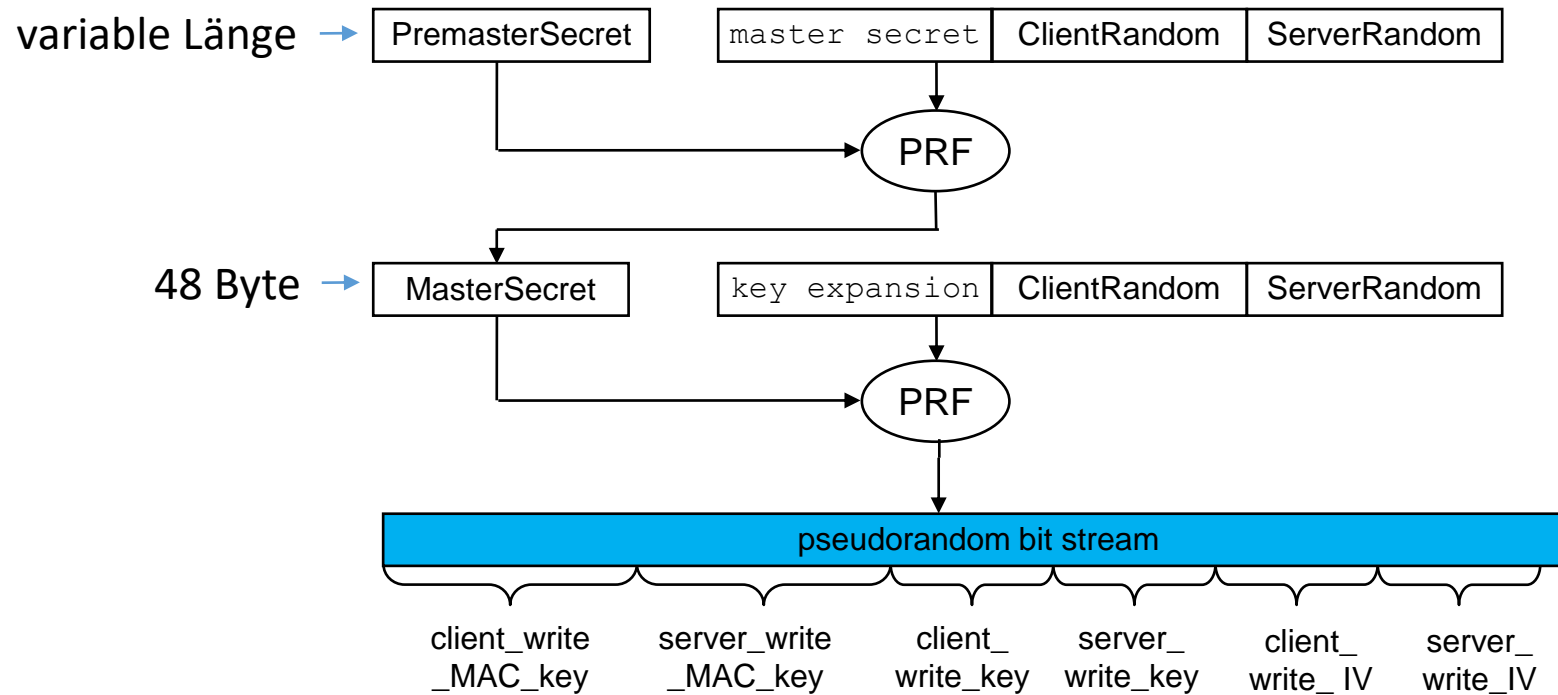
Schlüsselmaterial für Record Layer



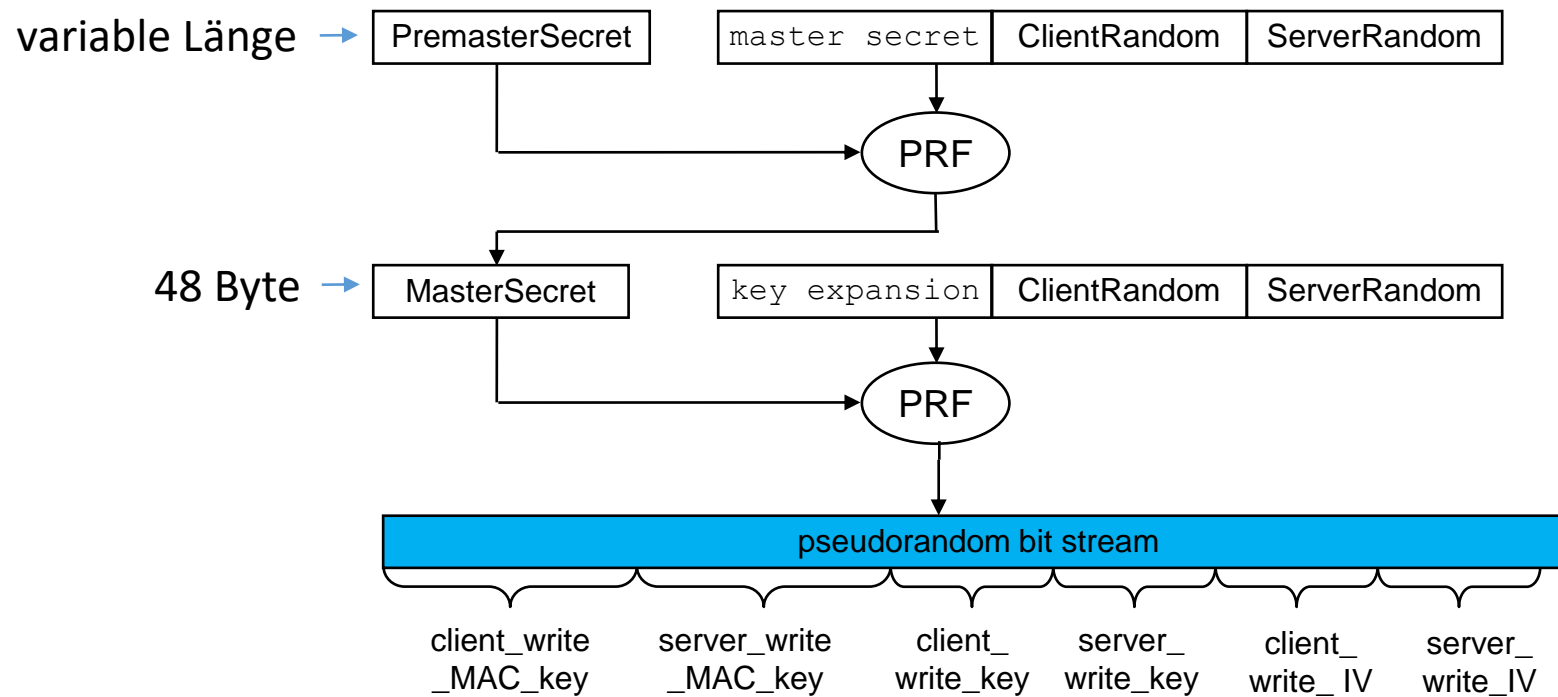
RFC 5246:

```
key_block = PRF(SecurityParameters.master_secret, "key expansion",  
SecurityParameters.server_random + SecurityParameters.client_random);
```


Schlüsselmaterial für Record Layer



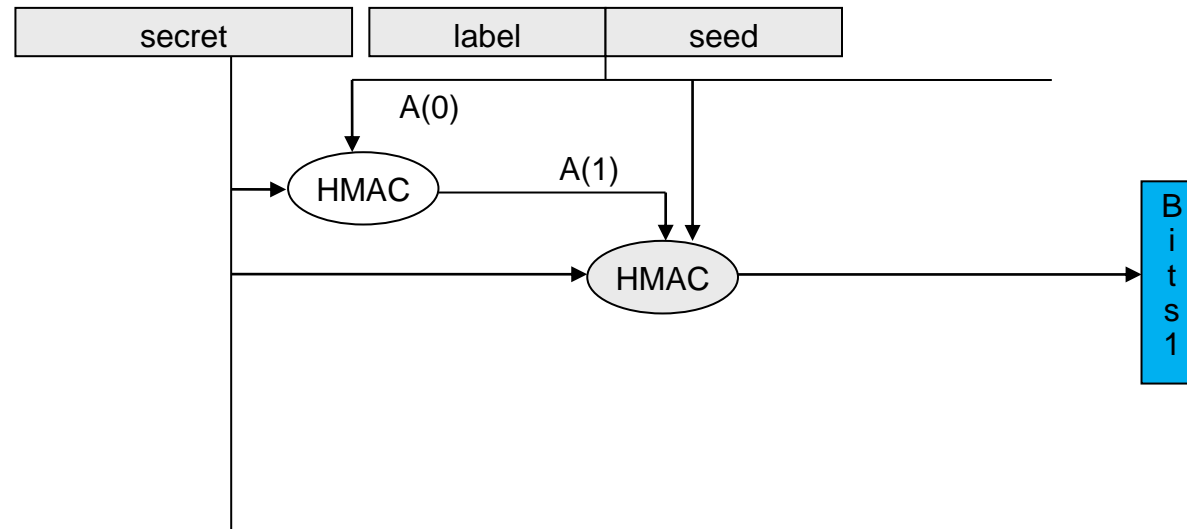
Schlüsselmaterial für Record Layer



RFC 5246:

```
client_write_MAC_key[SecurityParameters.mac_key_length]
server_write_MAC_key[SecurityParameters.mac_key_length]
client_write_key[SecurityParameters.enc_key_length]
server_write_key[SecurityParameters.enc_key_length]
client_write_IV[SecurityParameters.fixed_iv_length]
server_write_IV[SecurityParameters.fixed_iv_length]
```

Die TLS-PRF



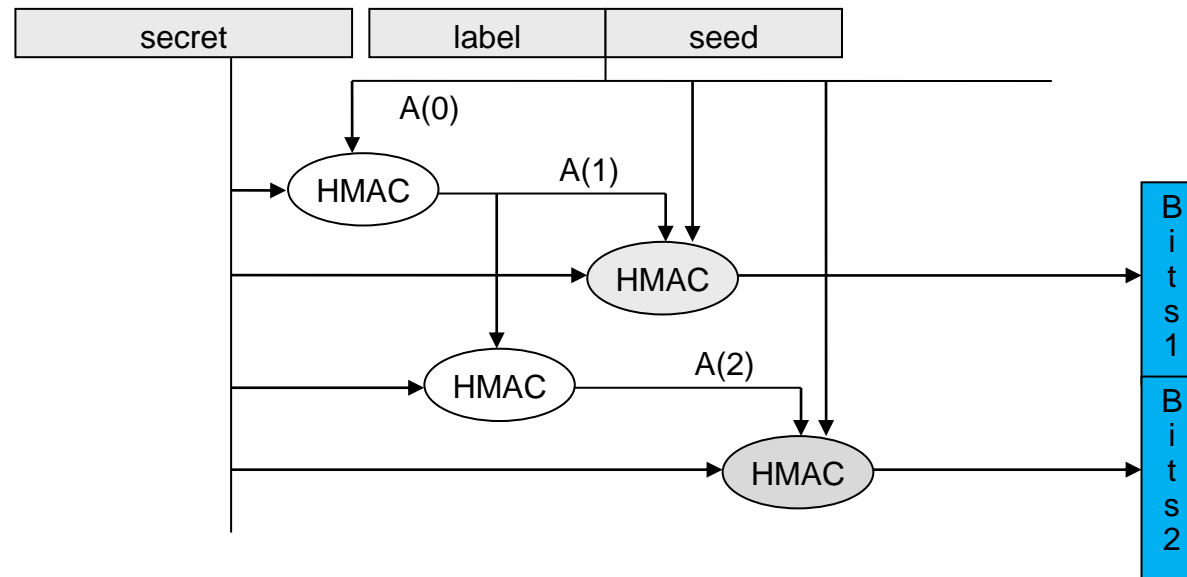
RFC 5246:
$$P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) + \\ HMAC_hash(secret, A(2) + seed) + \\ \dots$$

$A()$ is defined as:

$A(0) = seed$

$A(i) = HMAC_hash(secret, A(i-1))$

Die TLS-PRF



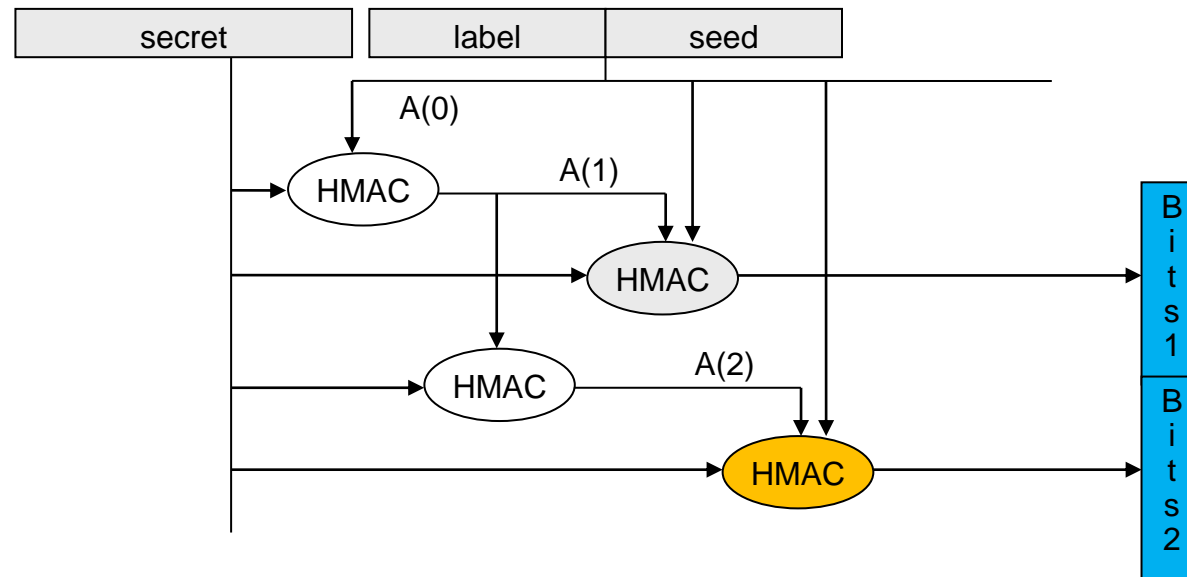
RFC 5246: $P_hash(secret, seed) = \text{HMAC_hash}(secret, A(1) + seed) + \text{HMAC_hash}(secret, A(2) + seed) + \dots$

$A()$ is defined as:

$A(0) = seed$

$A(i) = \text{HMAC_hash}(secret, A(i-1))$

Die TLS-PRF



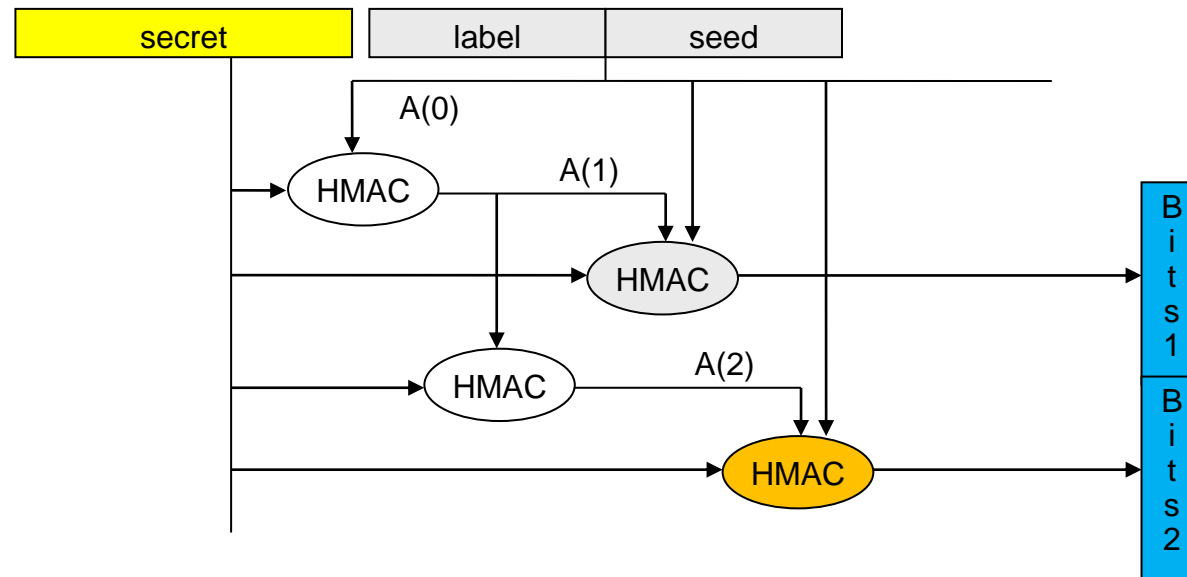
RFC 5246: $P_hash(secret, seed) = \text{HMAC_hash}(secret, A(1) + seed) + \text{HMAC_hash}(secret, A(2) + seed) + \dots$

$A()$ is defined as:

$A(0) = seed$

$A(i) = \text{HMAC_hash}(secret, A(i-1))$

Die TLS-PRF



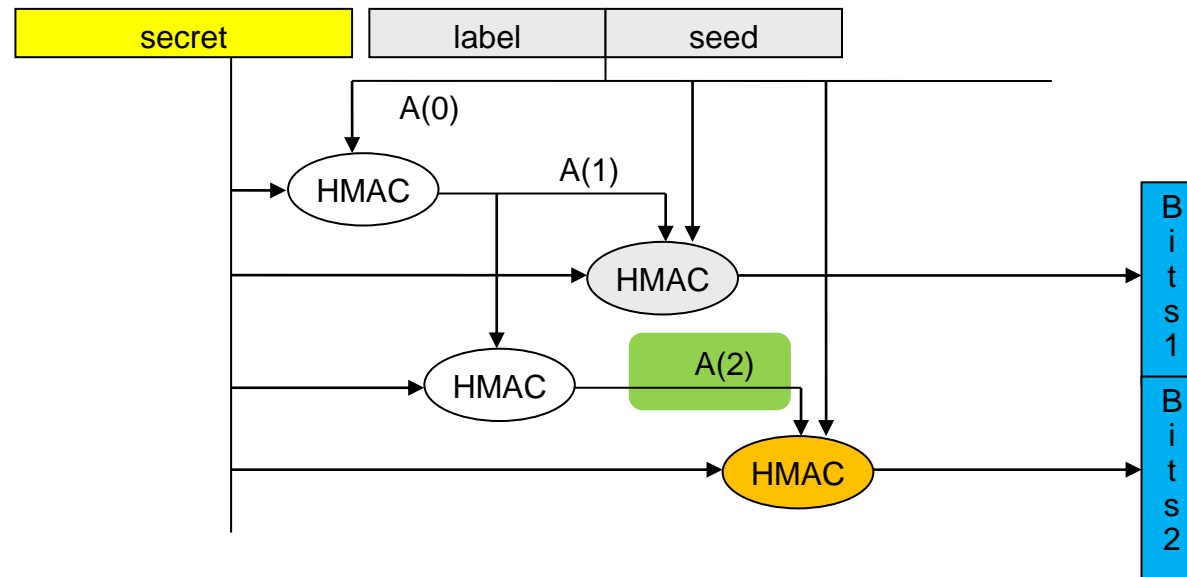
RFC 5246: $P_hash(secret, seed) = \text{HMAC_hash}(secret, A(1) + seed) + \text{HMAC_hash}(secret, A(2) + seed) + \dots$

A() is defined as:

$A(0) = seed$

$A(i) = \text{HMAC_hash}(secret, A(i-1))$

Die TLS-PRF



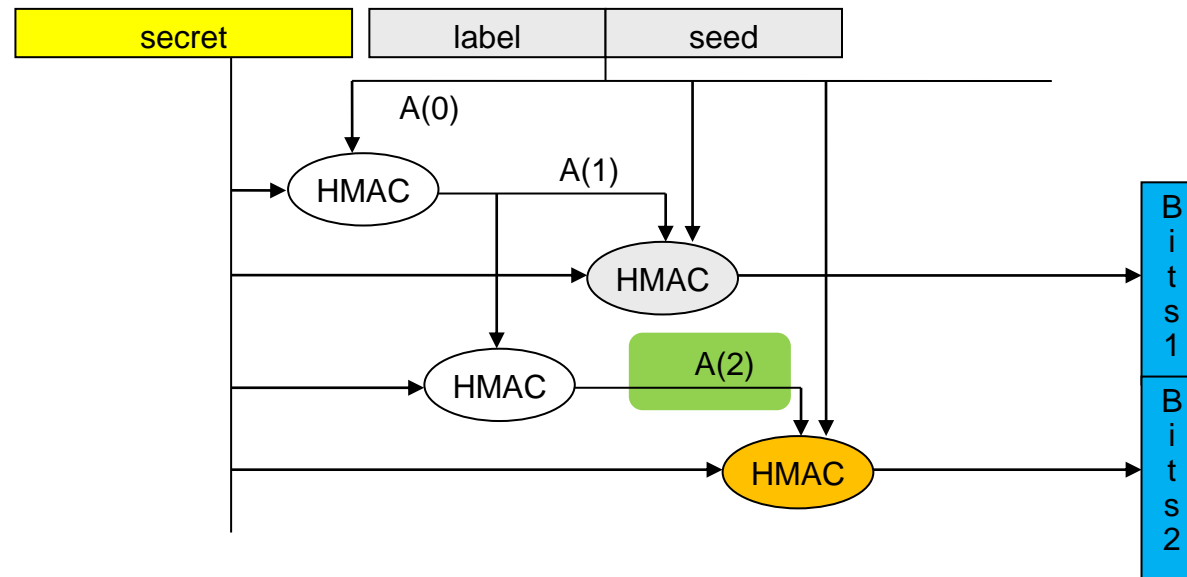
RFC 5246: $P_hash(secret, seed) = \text{HMAC_hash}(secret, A(1) + seed) + \text{HMAC_hash}(secret, A(2) + seed) + \dots$

$A()$ is defined as:

$$A(0) = seed$$

$$A(i) = \text{HMAC_hash}(secret, A(i-1))$$

Die TLS-PRF



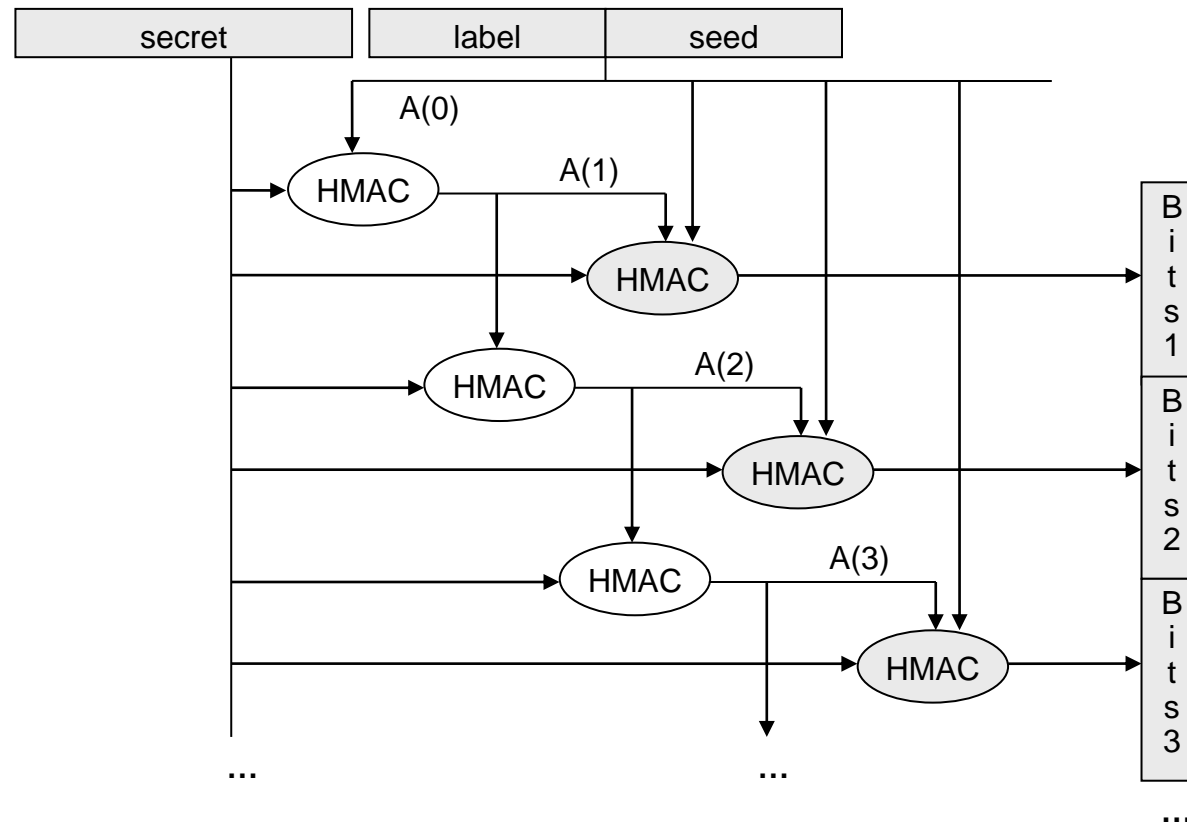
RFC 5246:
$$\text{P_hash}(\text{secret}, \text{seed}) = \text{HMAC_hash}(\text{secret}, A(1) + \text{seed}) + \text{HMAC_hash}(\text{secret}, A(2) + \text{seed}) + \dots$$

A() is defined as:

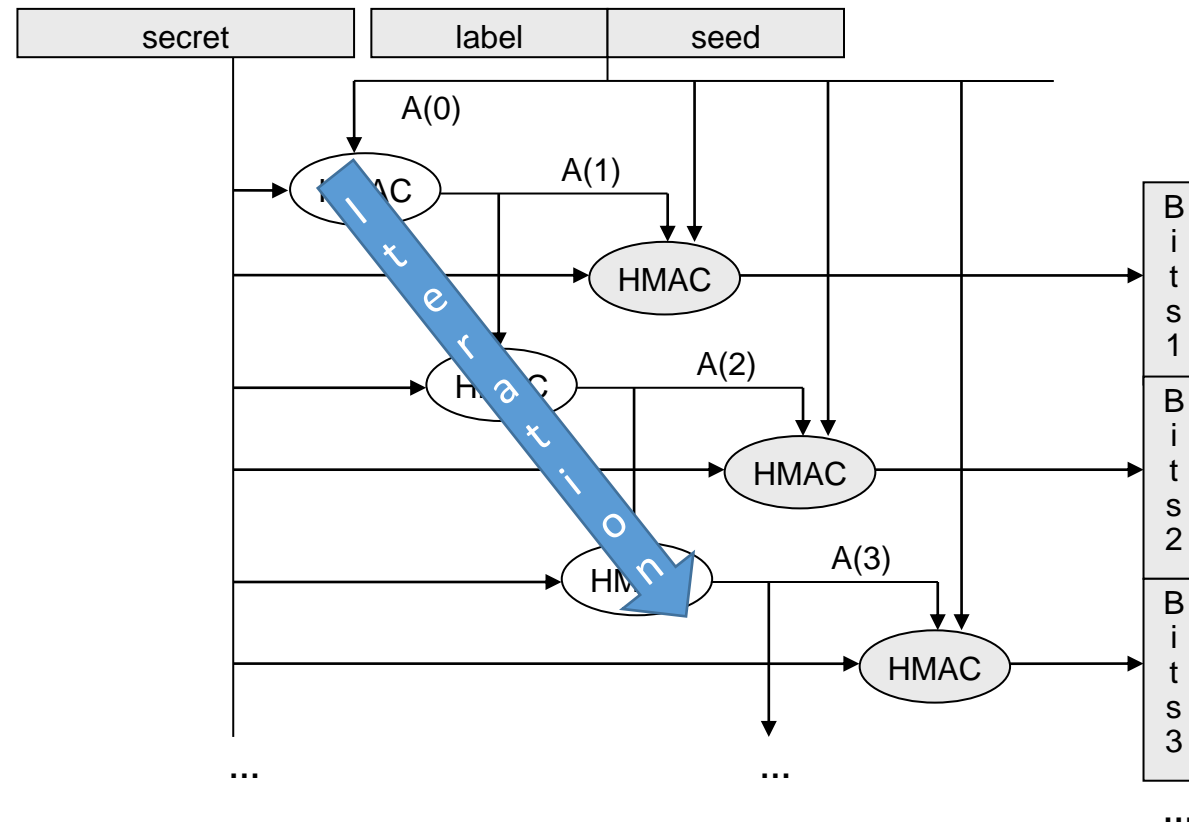
$A(0) = \text{seed}$

$A(i) = \text{HMAC_hash}(\text{secret}, A(i-1))$

Die TLS-PRF



Die TLS-PRF



Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
- Schlüssel

Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
 - Basis für weitere Schlüsselableitung
- Schlüssel

Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
 - Basis für weitere Schlüsselableitung
 - Schlüssel für MAC in den Finished-Nachrichten
- Schlüssel

Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
 - Basis für weitere Schlüsselableitung
 - Schlüssel für MAC in den Finished-Nachrichten
 - Basis für TLS Session Resumption
- Schlüssel

Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
 - Basis für weitere Schlüsselableitung
 - Schlüssel für MAC in den Finished-Nachrichten
 - Basis für TLS Session Resumption
- Schlüssel
 - 2 Encryption Keys: Für jede Richtung einer

Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
 - Basis für weitere Schlüsselableitung
 - Schlüssel für MAC in den Finished-Nachrichten
 - Basis für TLS Session Resumption
- Schlüssel
 - 2 Encryption Keys: Für jede Richtung einer
 - 2 MAC Keys: Für jede Richtung einer

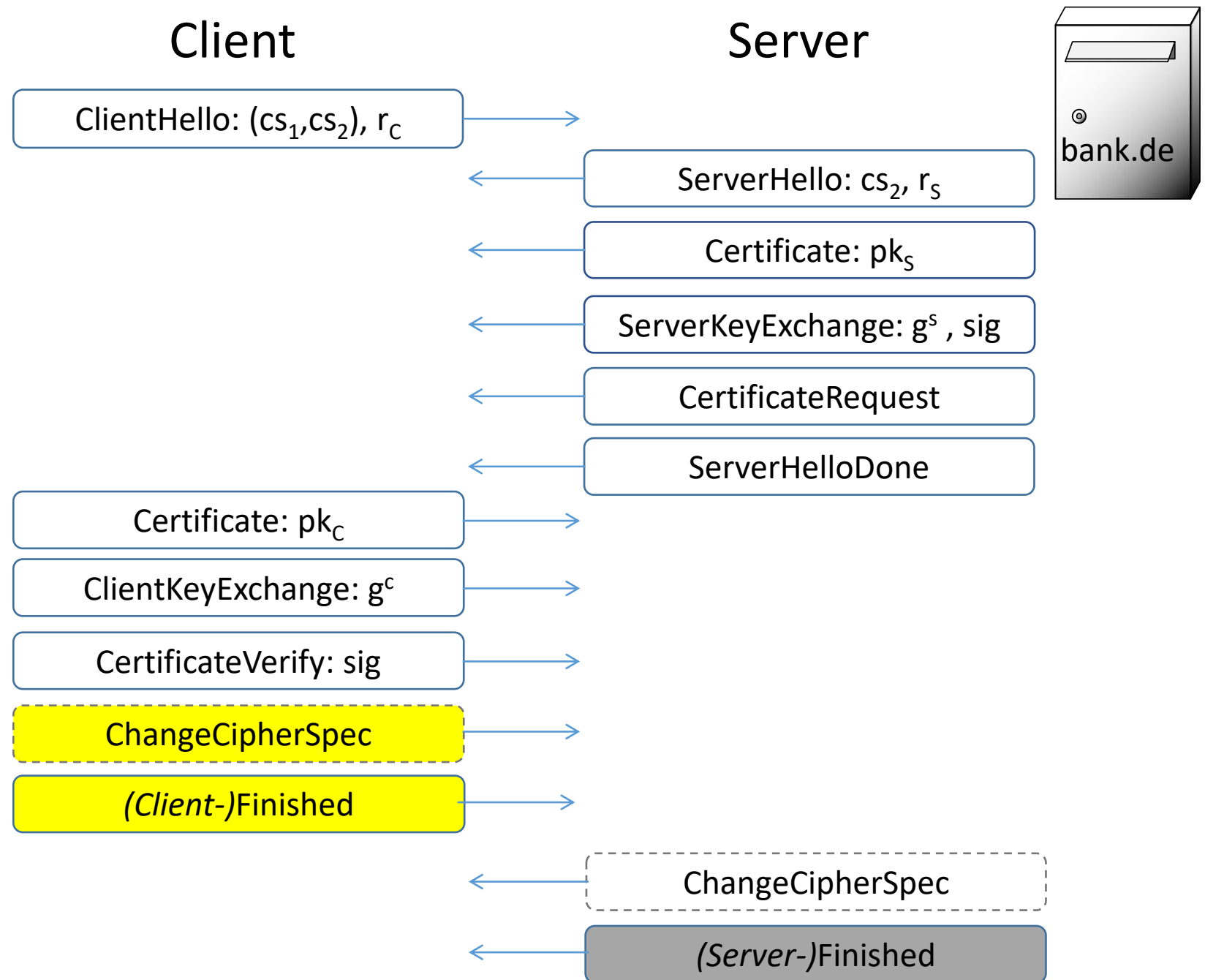
Ergebnis der Schlüsselableitung

- PremasterSecret: Kann gelöscht werden
- MasterSecret: Wird gespeichert
 - Basis für weitere Schlüsselableitung
 - Schlüssel für MAC in den Finished-Nachrichten
 - Basis für TLS Session Resumption
- Schlüssel
 - 2 Encryption Keys: Für jede Richtung einer
 - 2 MAC Keys: Für jede Richtung einer
 - weiteres Schlüsselmaterial: z.B. IVs für CBC, nonces für GCM, ...

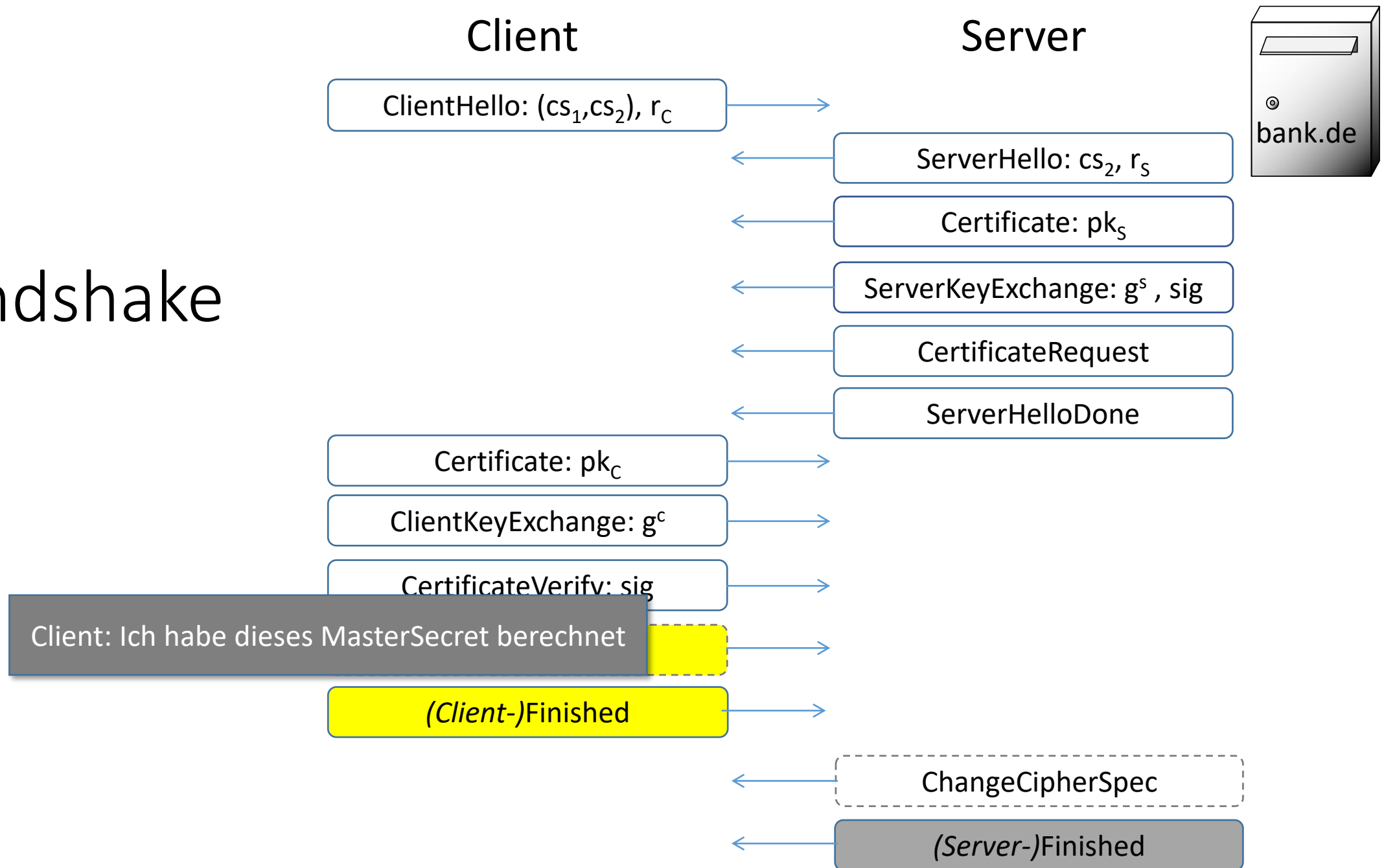
2.3 TLS Handshake

2.3.6 Bestätigung der Schlüssel, Authentifikation des Servers

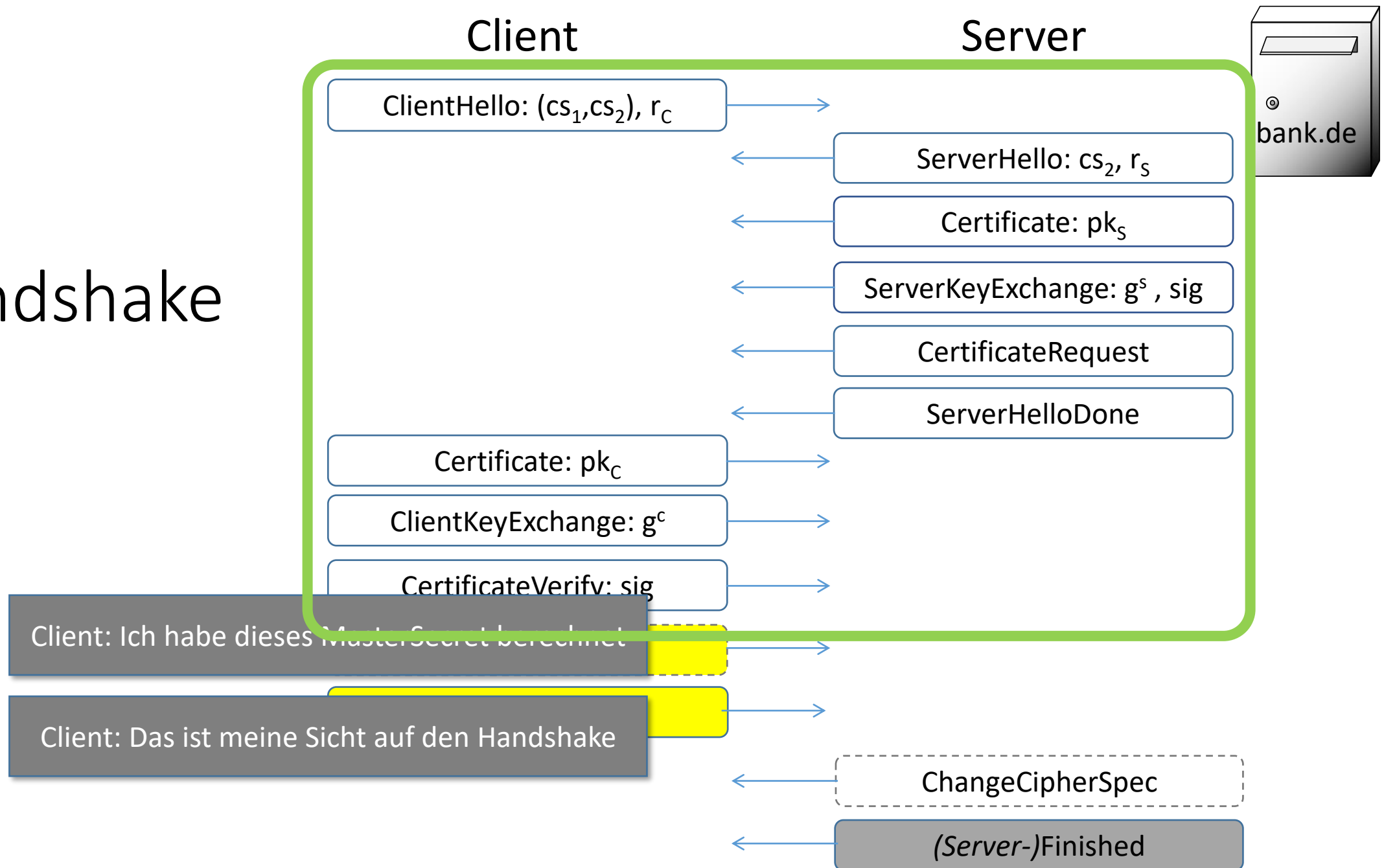
TLS Handshake



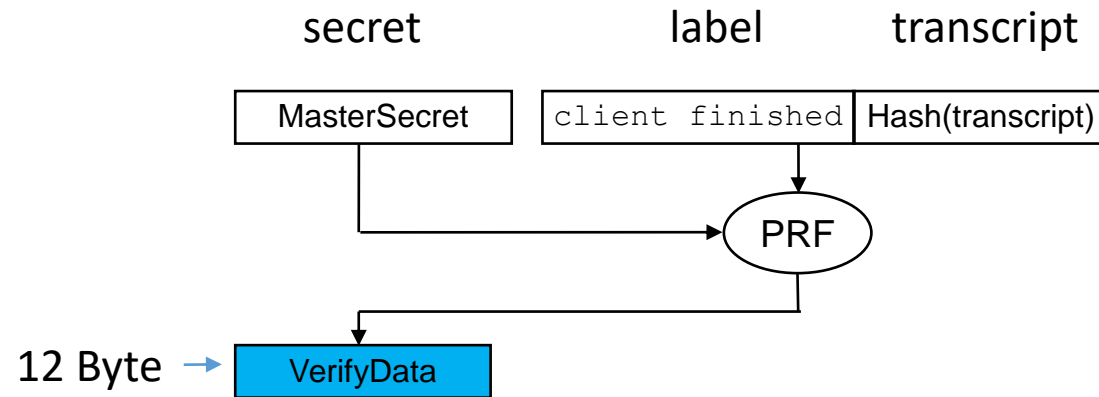
TLS Handshake



TLS Handshake



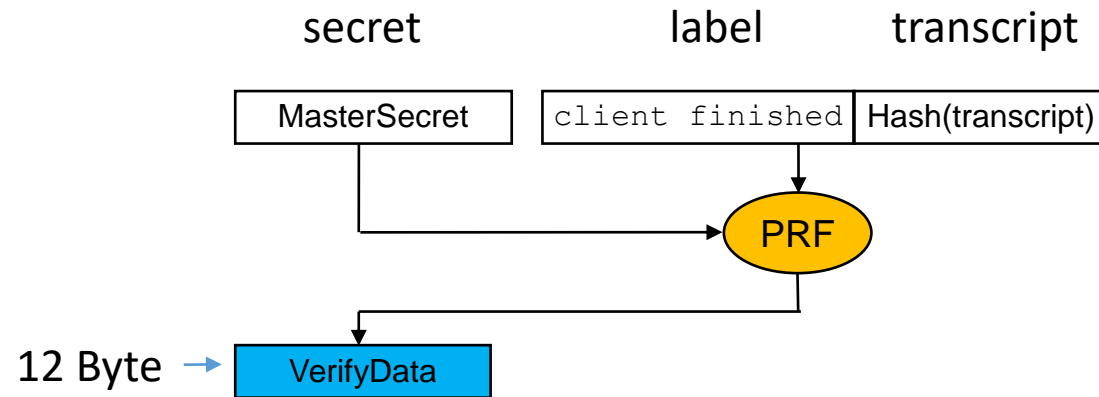
Finished



RFC 5246: `verify_data <- PRF(master_secret, finished_label, Hash(handshake_messages))`
`[0..verify_data_length-1];`

`finished_label` For Finished messages sent by the client, the string "client finished". For Finished messages sent by the server, the string "server finished".

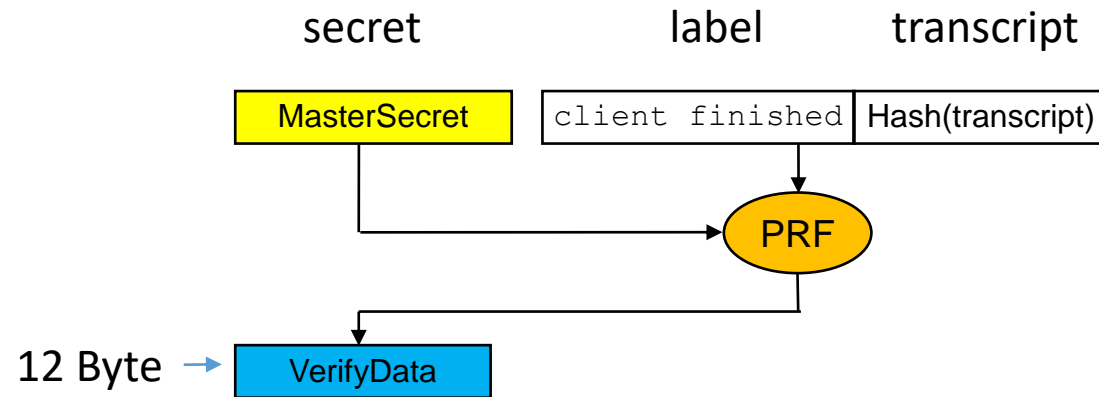
Finished



RFC 5246: `verify_data <- PRF(master_secret, finished_label, Hash(handshake_messages))`
`[0..verify_data_length-1];`

`finished_label` For Finished messages sent by the client, the string "client finished". For Finished messages sent by the server, the string "server finished".

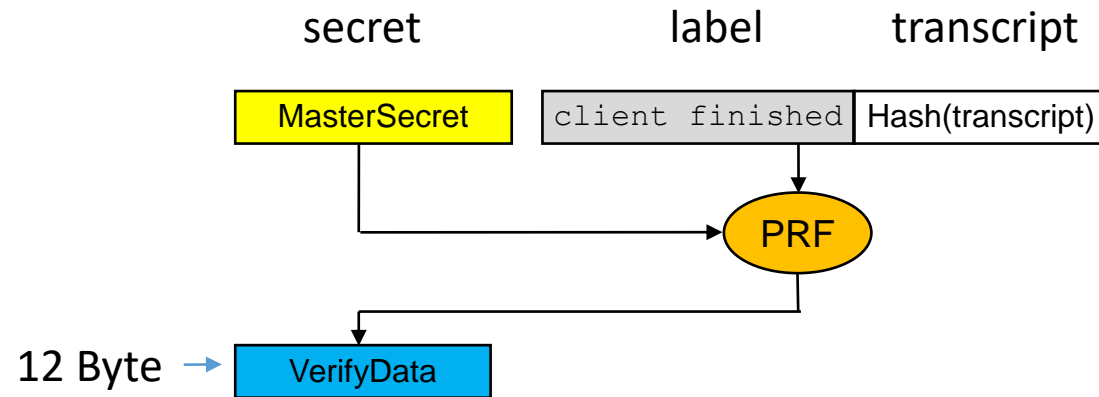
Finished



RFC 5246: `verify_data <- PRF(master_secret, finished_label, Hash(handshake_messages))`
`[0..verify_data_length-1];`

`finished_label` For Finished messages sent by the client, the string "client finished". For Finished messages sent by the server, the string "server finished".

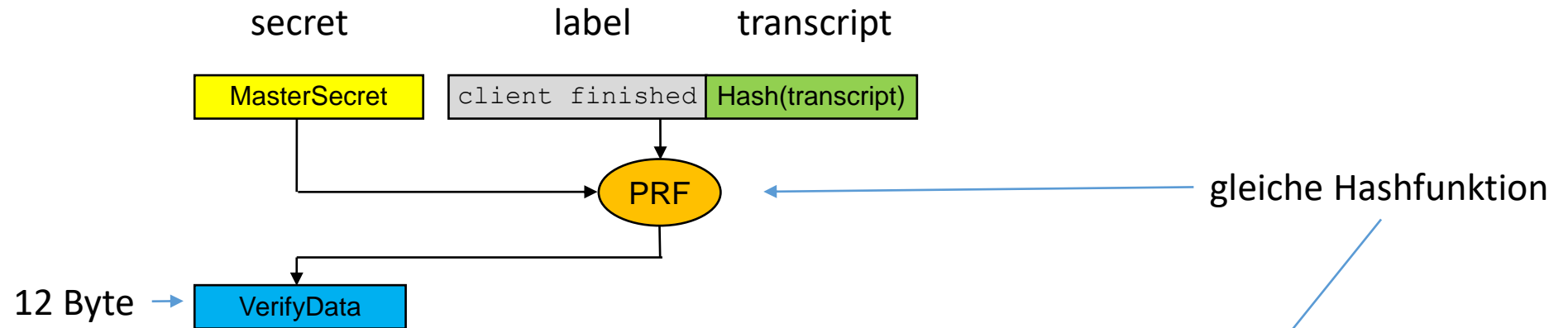
Finished



RFC 5246: `verify_data <- PRF(master_secret, finished_label, Hash(handshake_messages))`
`[0..verify_data_length-1];`

`finished_label` For Finished messages sent by the client, the string "client finished". For Finished messages sent by the server, the string "server finished".

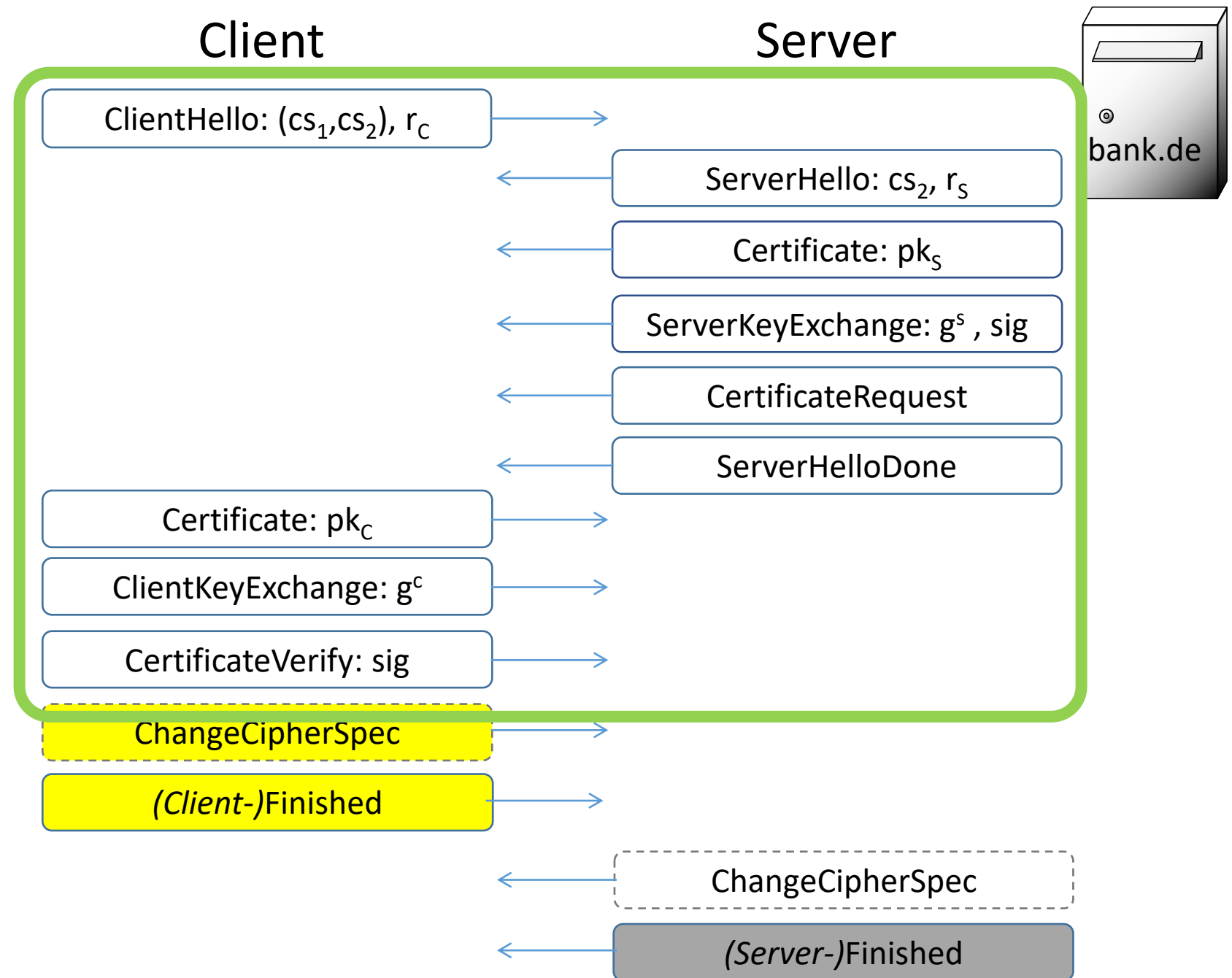
Finished



RFC 5246: `verify_data <- PRF(master_secret, finished_label, Hash(handshake_messages))`
`[0..verify_data_length-1];`

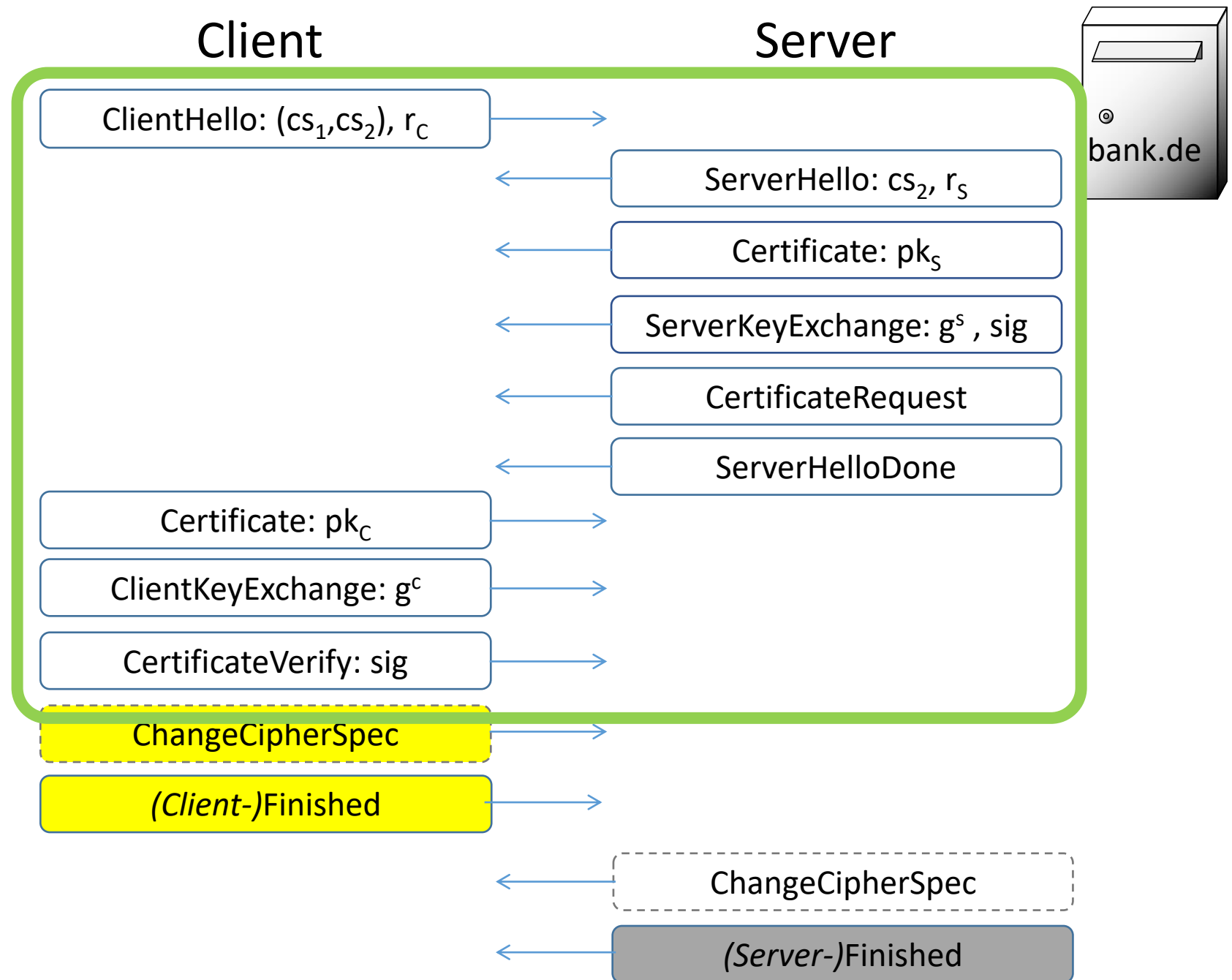
`finished_label` For Finished messages sent by the client, the string "client finished". For Finished messages sent by the server, the string "server finished".

TLS Handshake



TLS Handshake

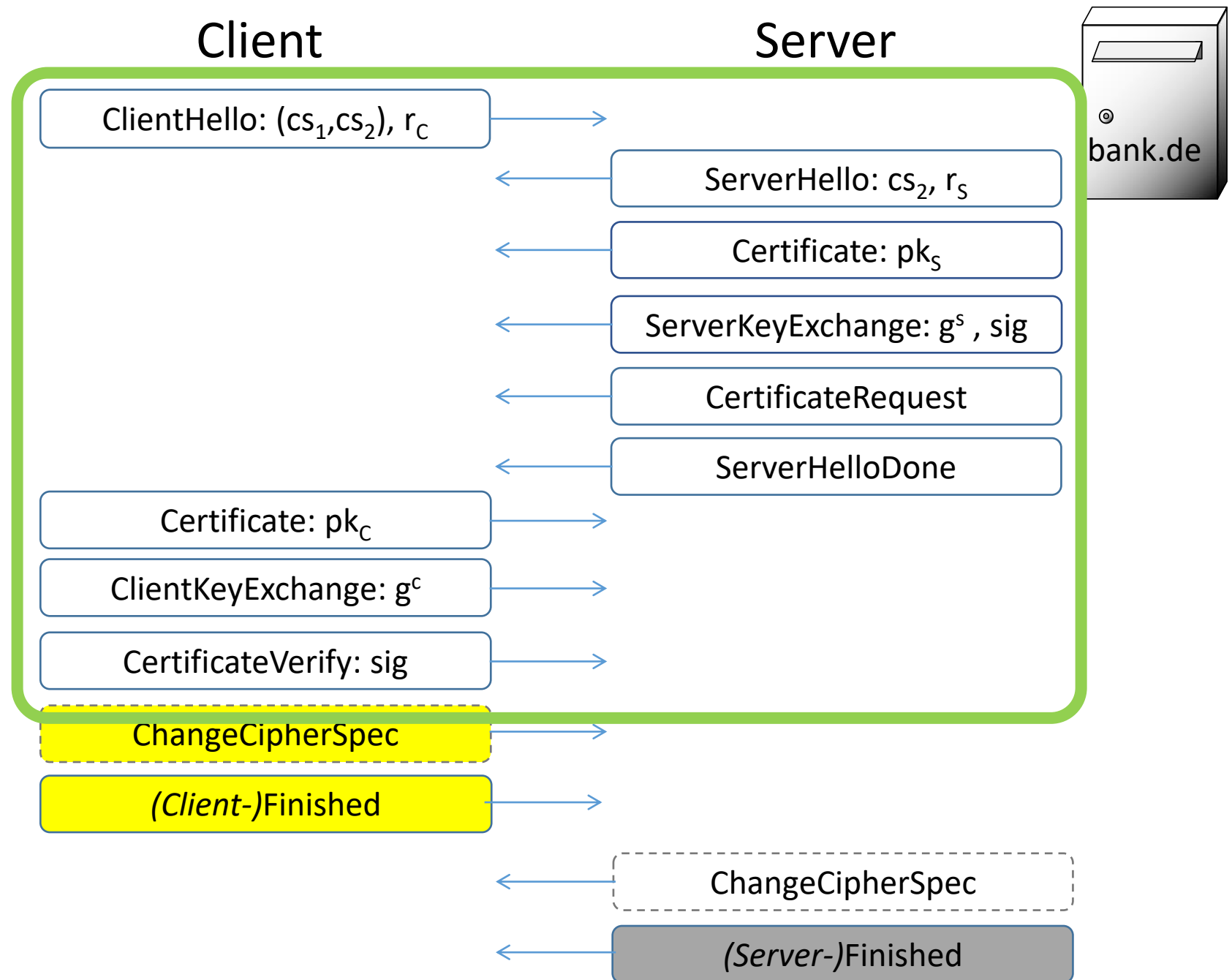
OHNE Record Layer Header!



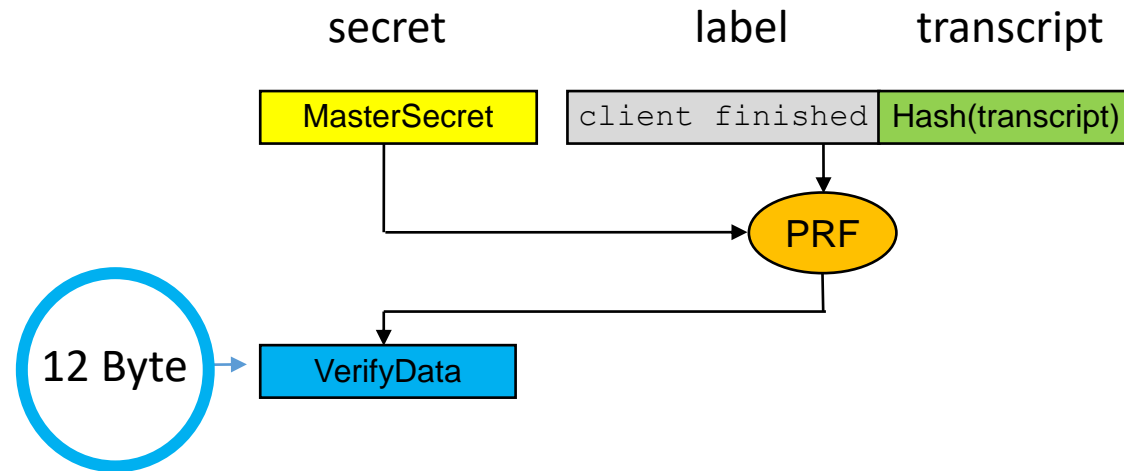
TLS Handshake

OHNE Record Layer Header!

OHNE ChangeCipherSpec!



Finished



RFC 5246:

```
verify_data <- PRF(master_secret, finished_label, Hash(handshake_messages))  
[0..verify_data_length-1];
```

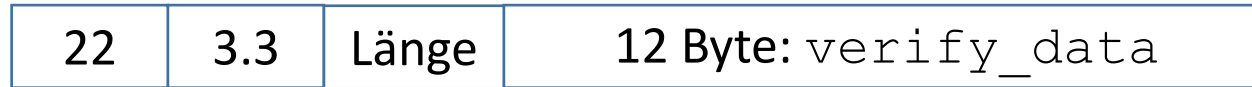
`finished_label` For Finished messages sent by the client, the string "client finished". For Finished messages sent by the server, the string "server finished".

12 Byte: verify_data

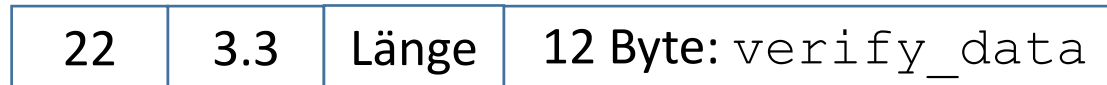
Handshakedaten

TLS 1.2

Fragmentierung



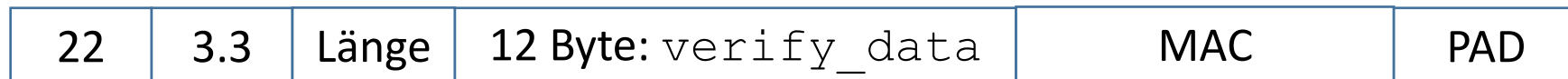
optional: Kompression



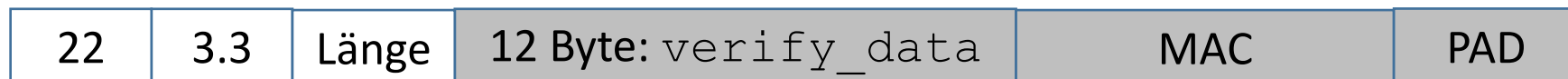
MAC



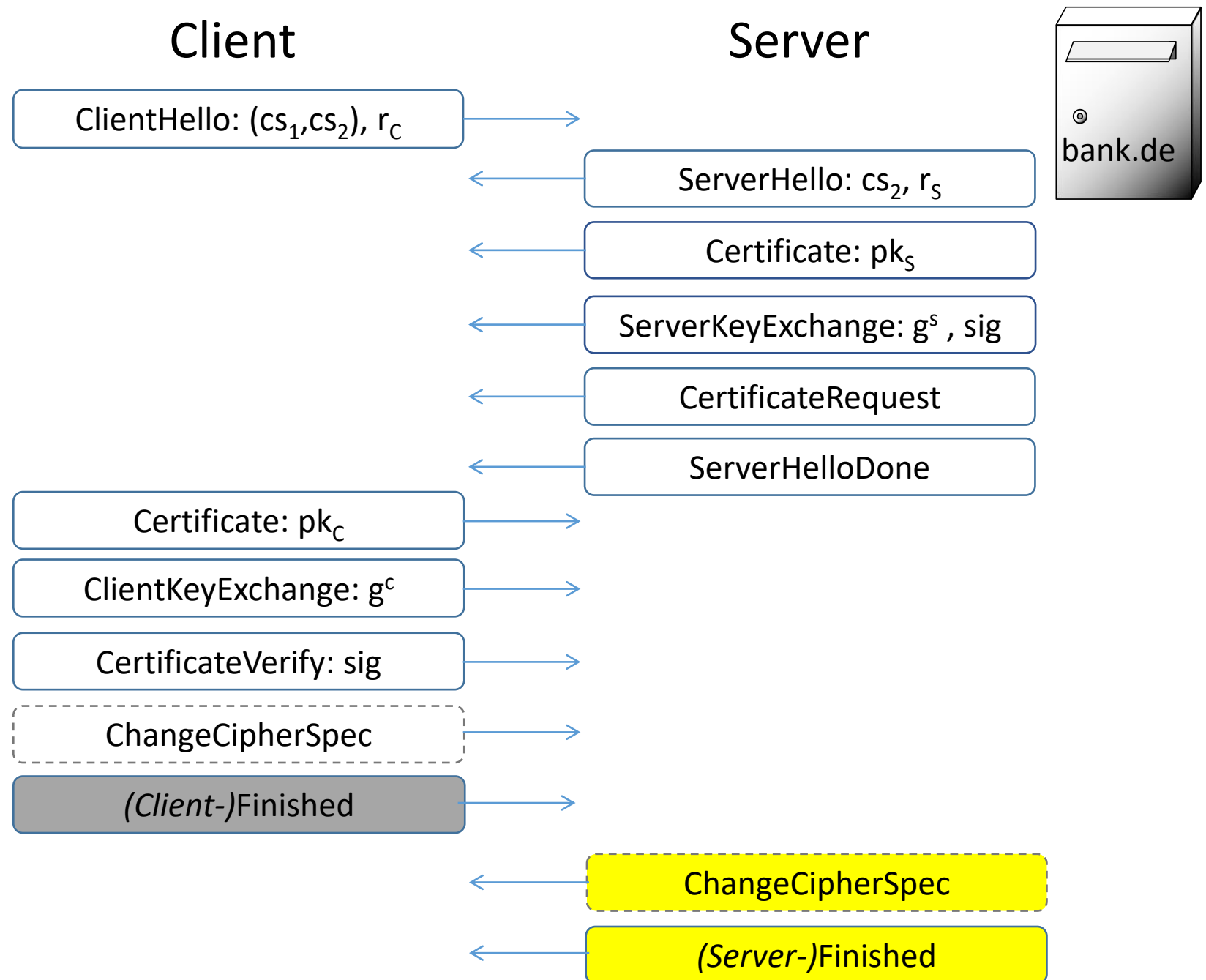
PAD



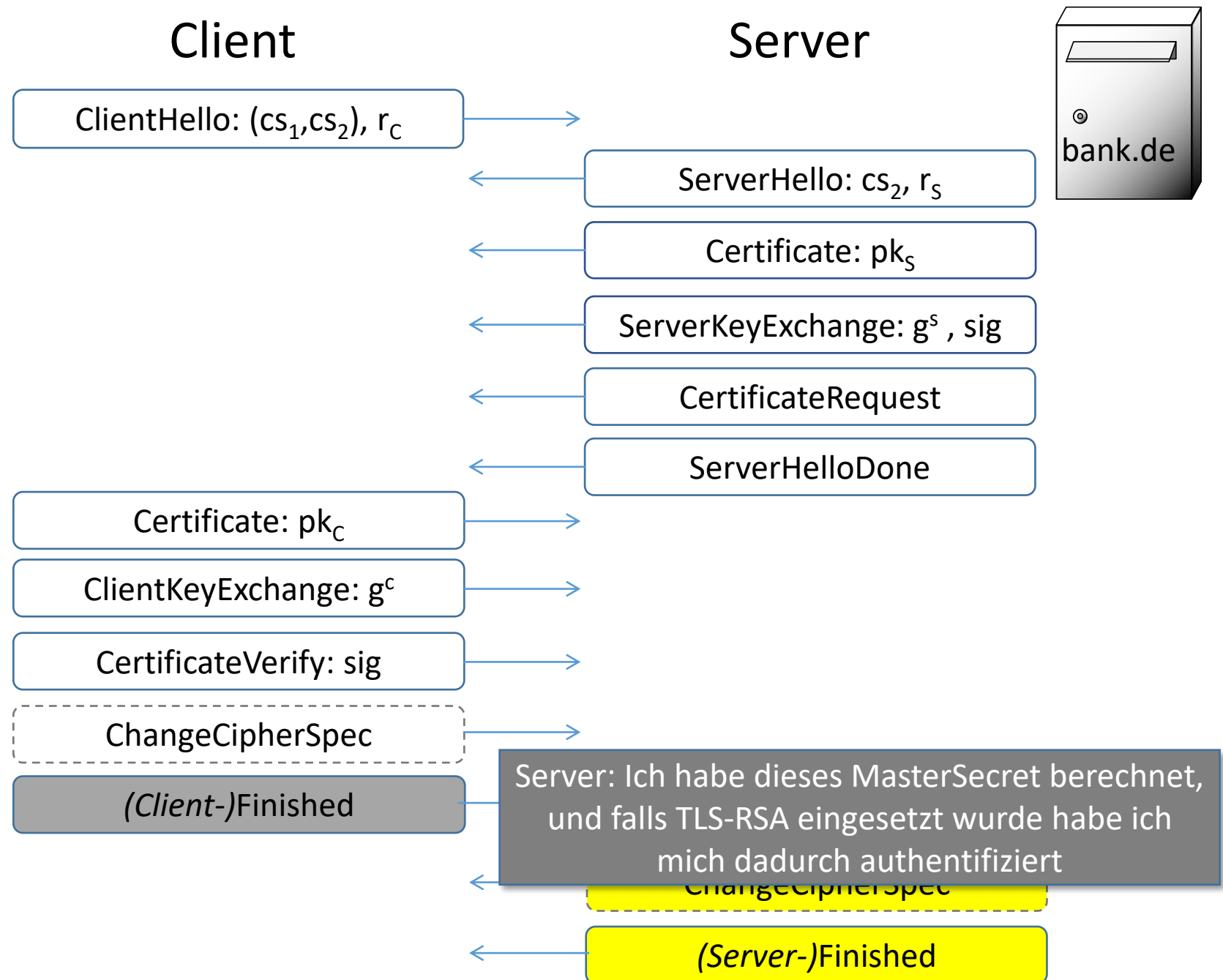
ENCRYPT



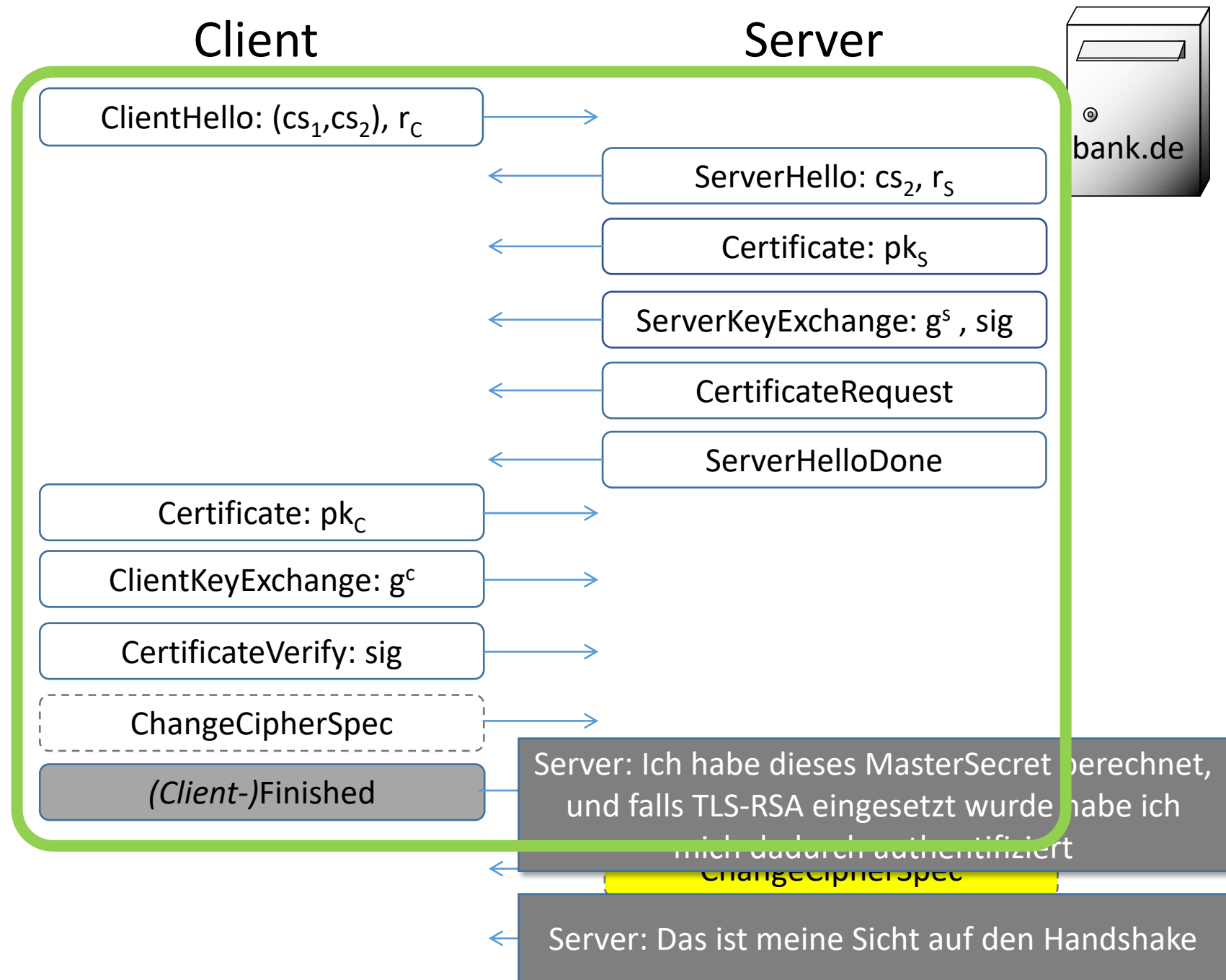
TLS Handshake



TLS Handshake

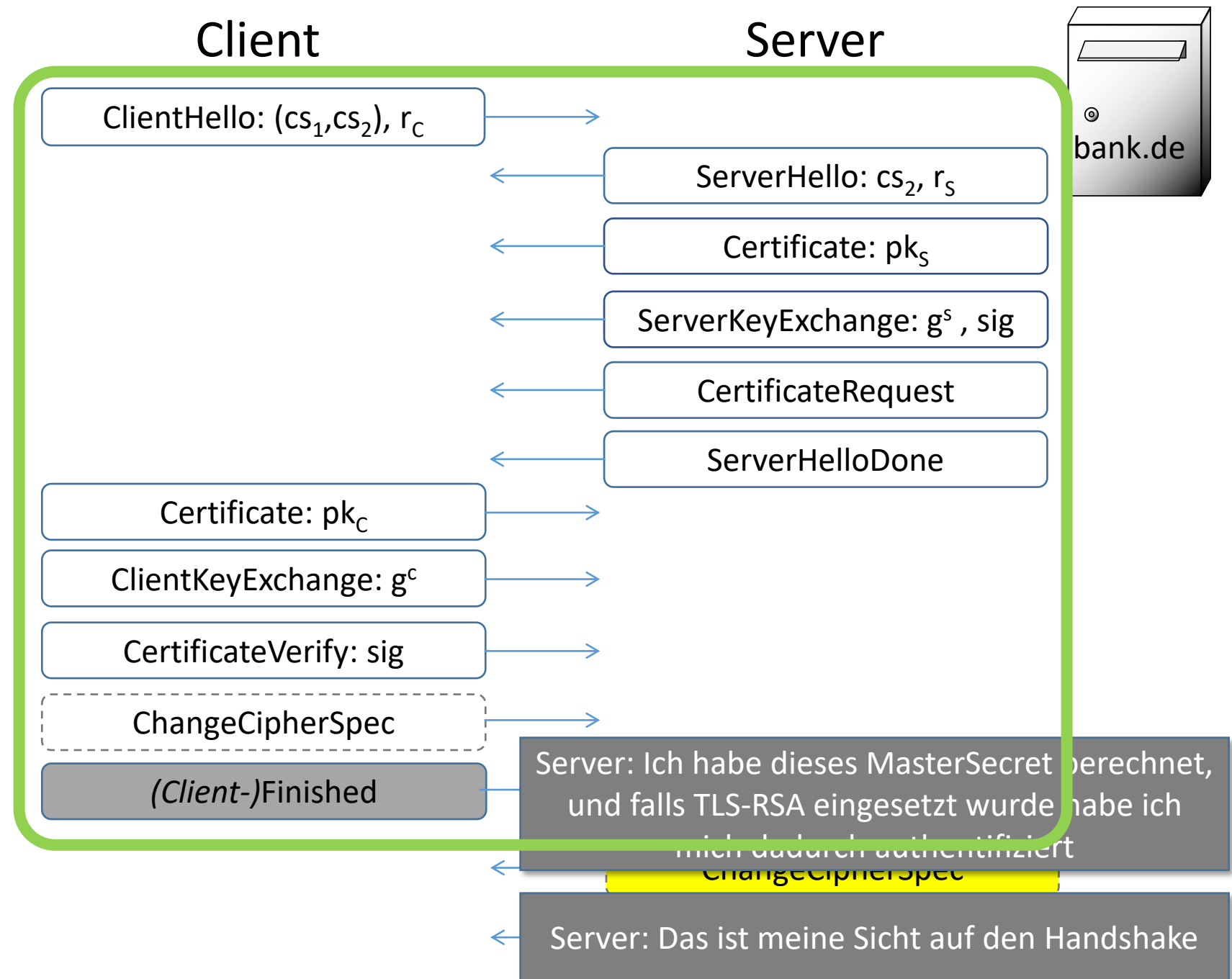


TLS Handshake



TLS Handshake

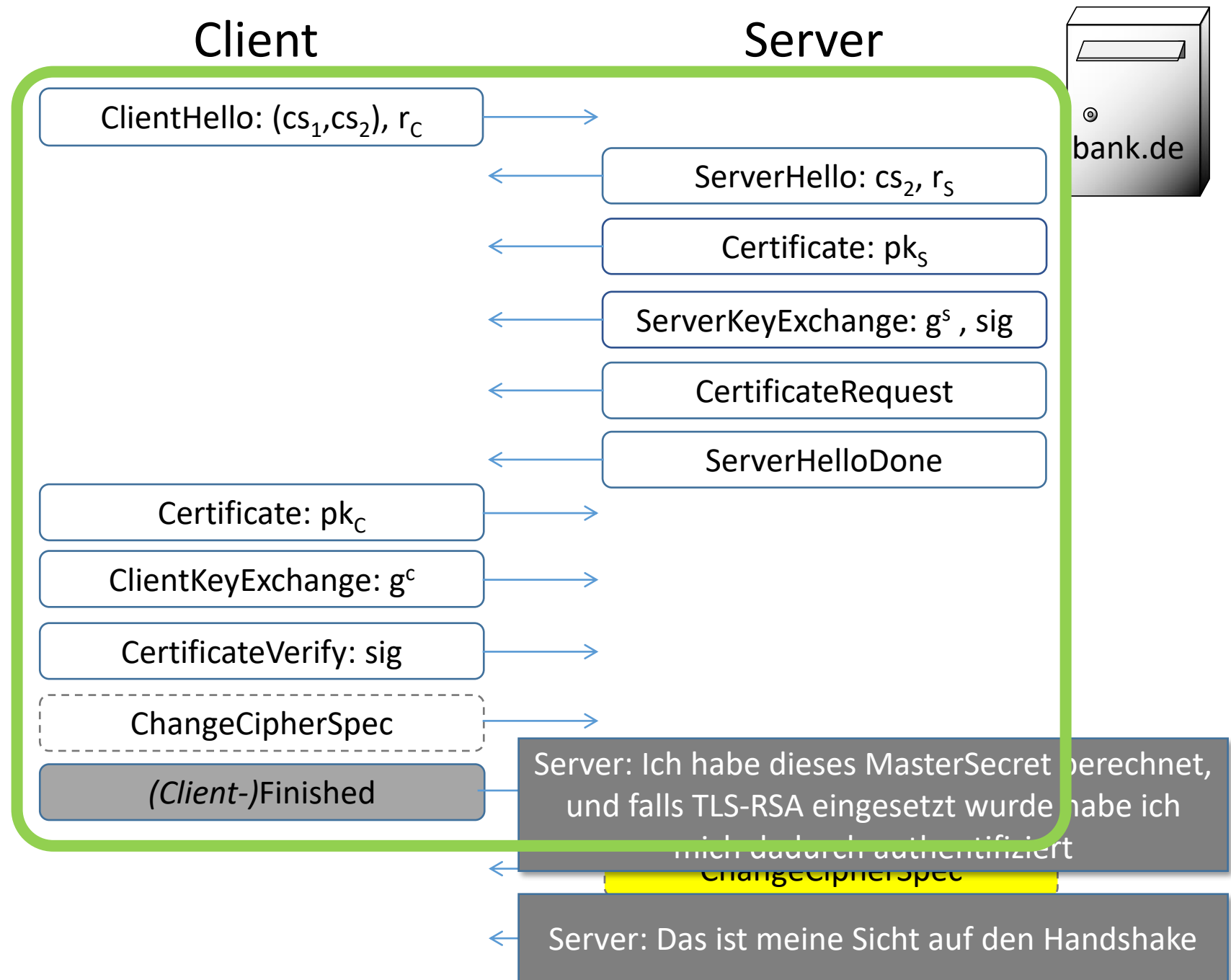
OHNE Record Layer Header!



TLS Handshake

OHNE Record Layer Header!

OHNE ChangeCipherSpec!

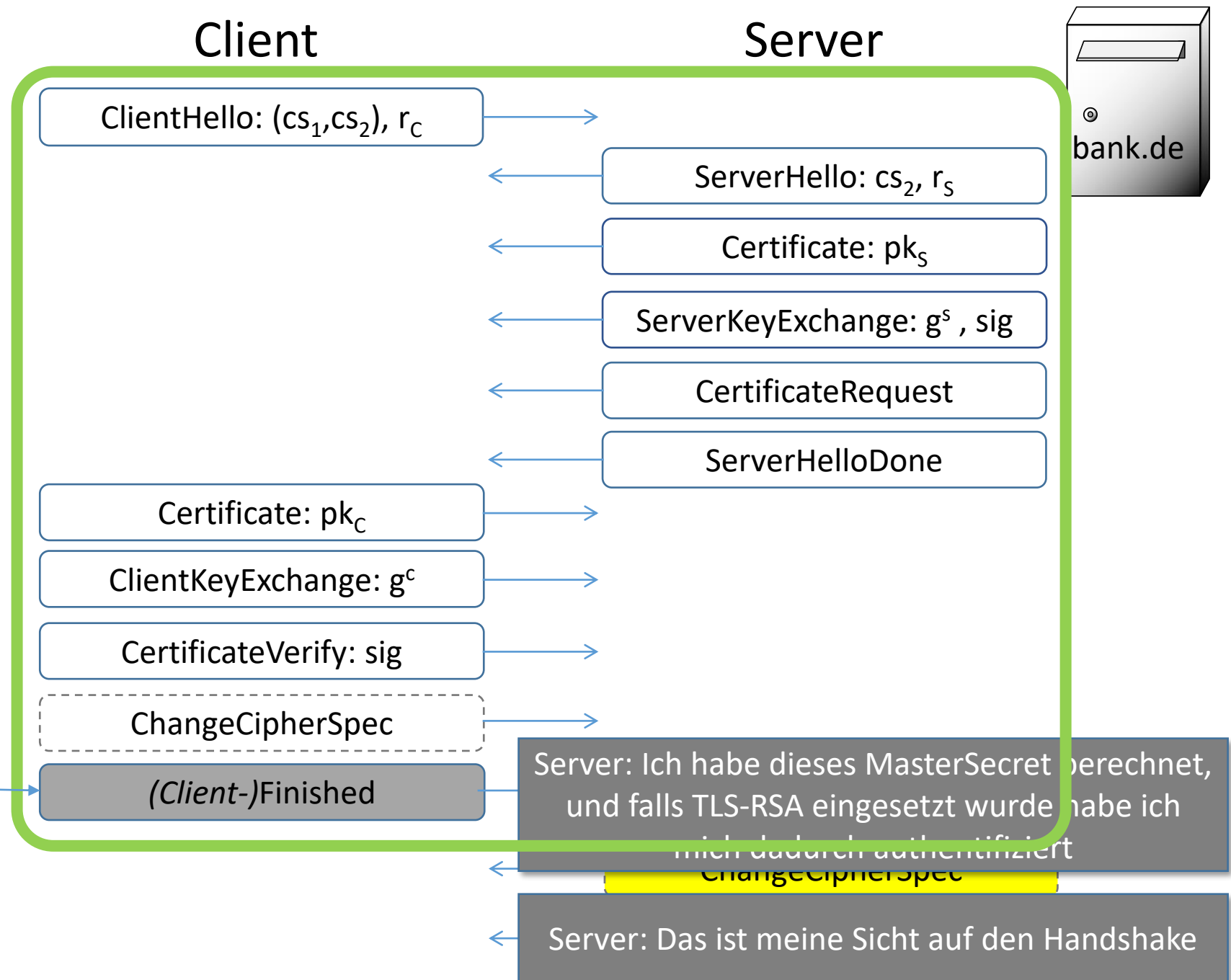


TLS Handshake

OHNE Record Layer Header!

OHNE ChangeCipherSpec!

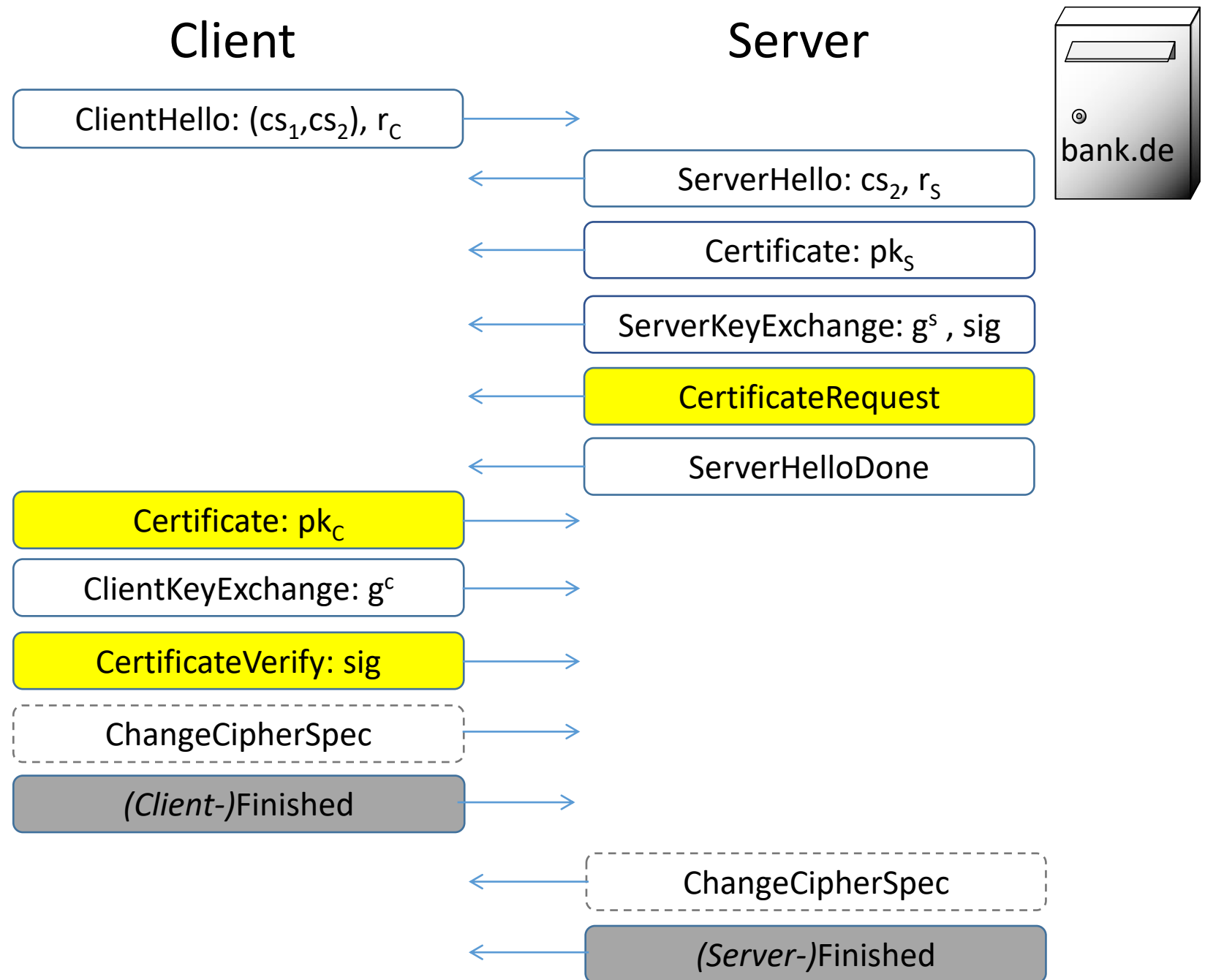
Unverschlüsselter Inhalt!



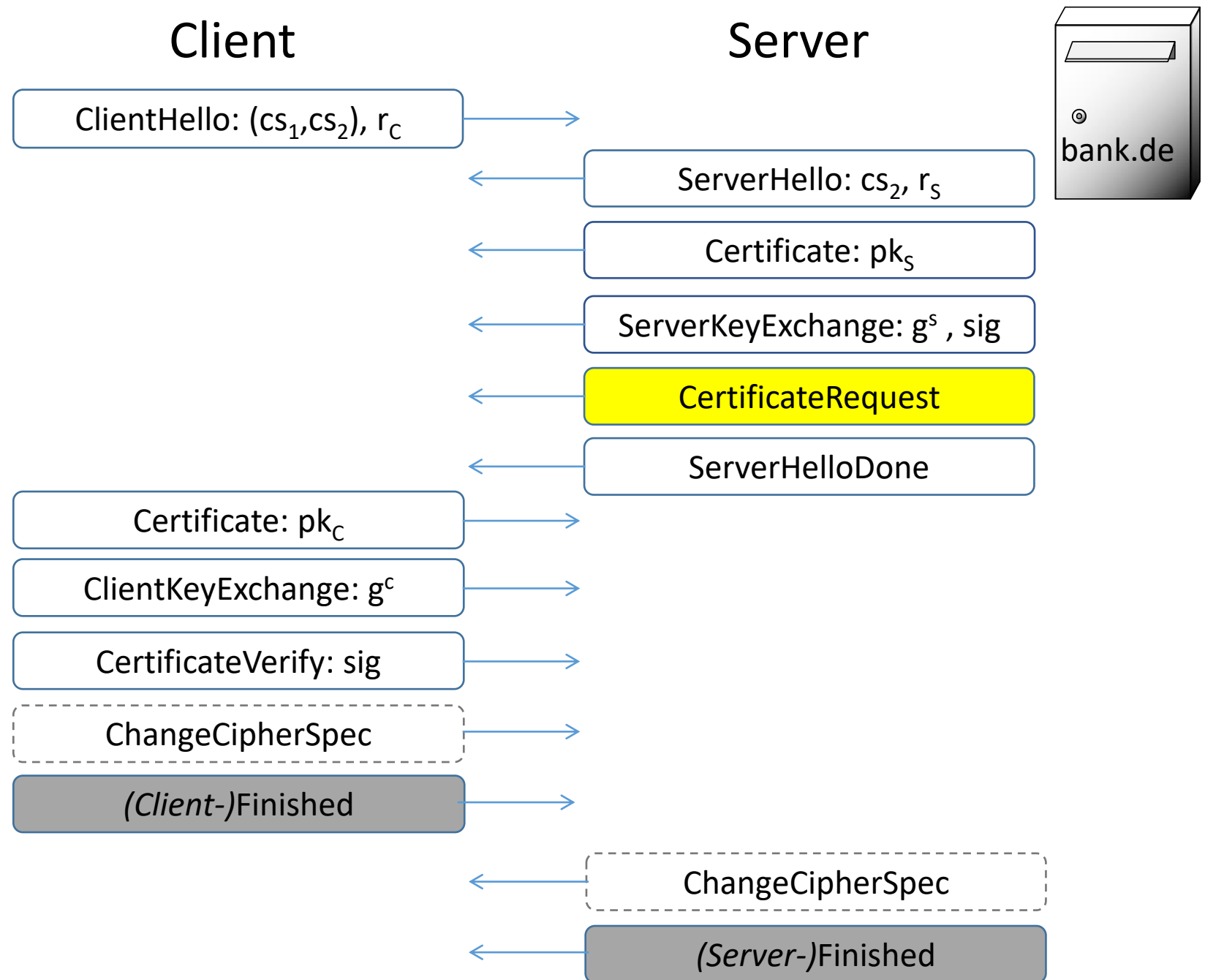
2.3 TLS Handshake

2.3.7 Optionale Authentifikation des Client

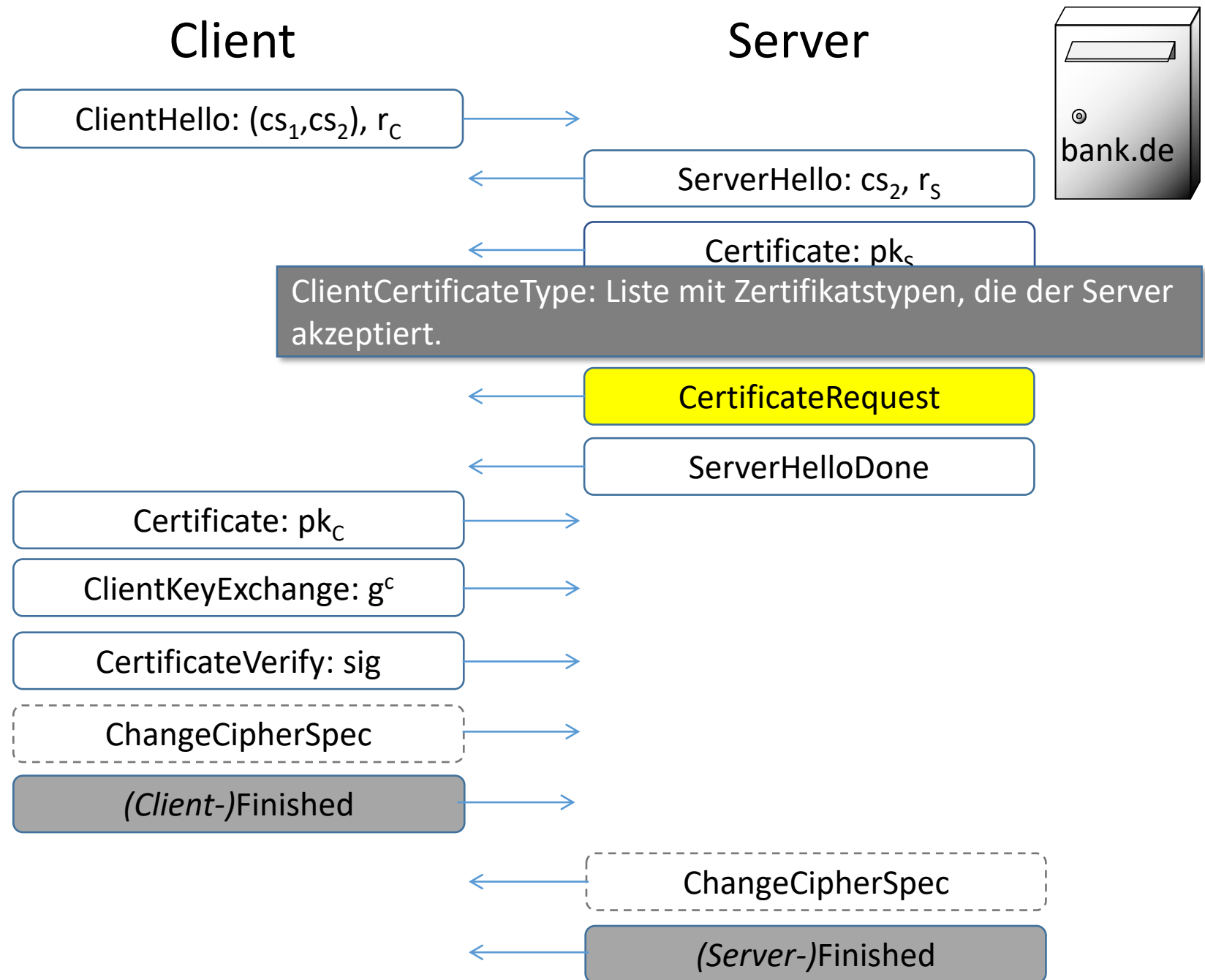
TLS Handshake



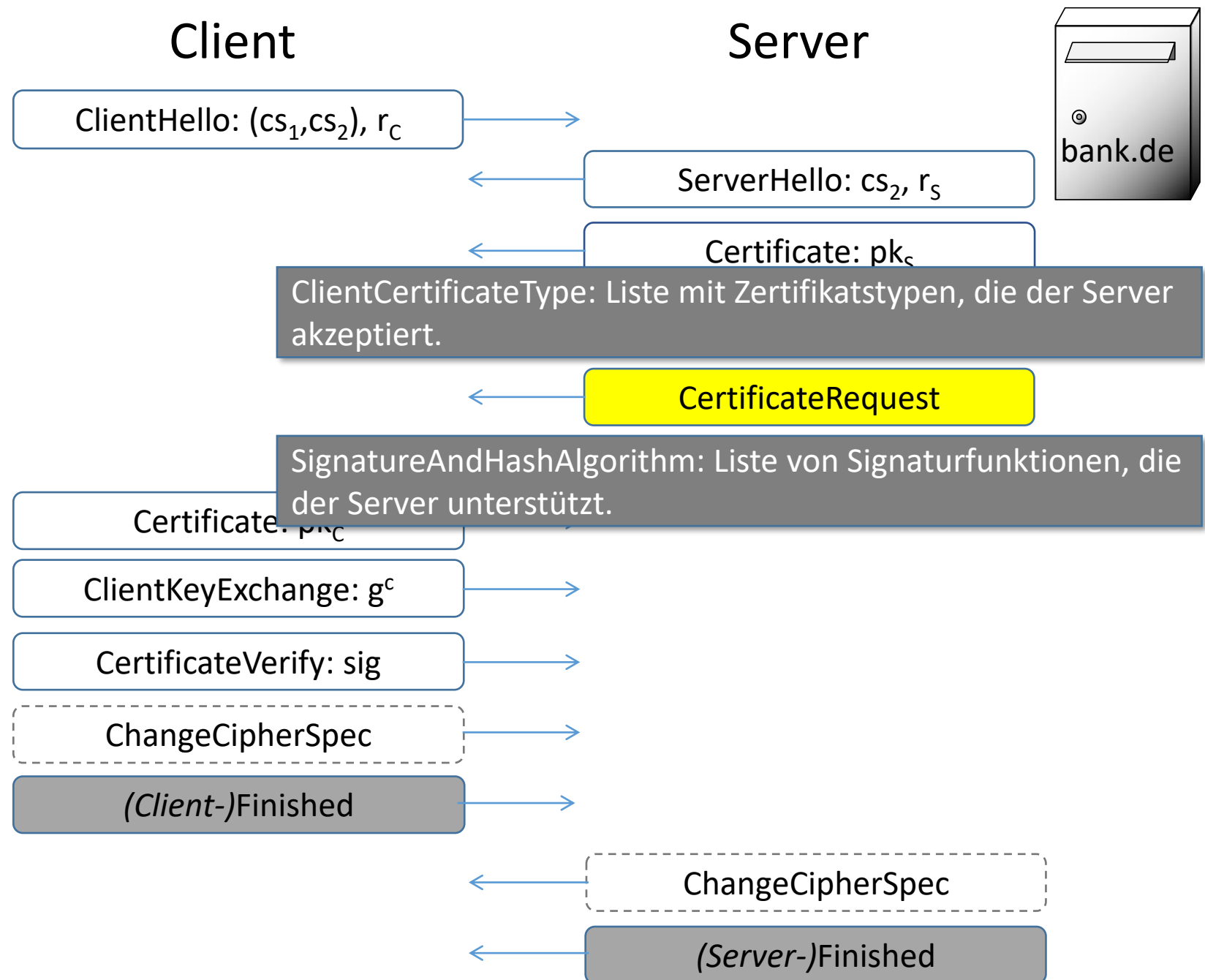
TLS Handshake



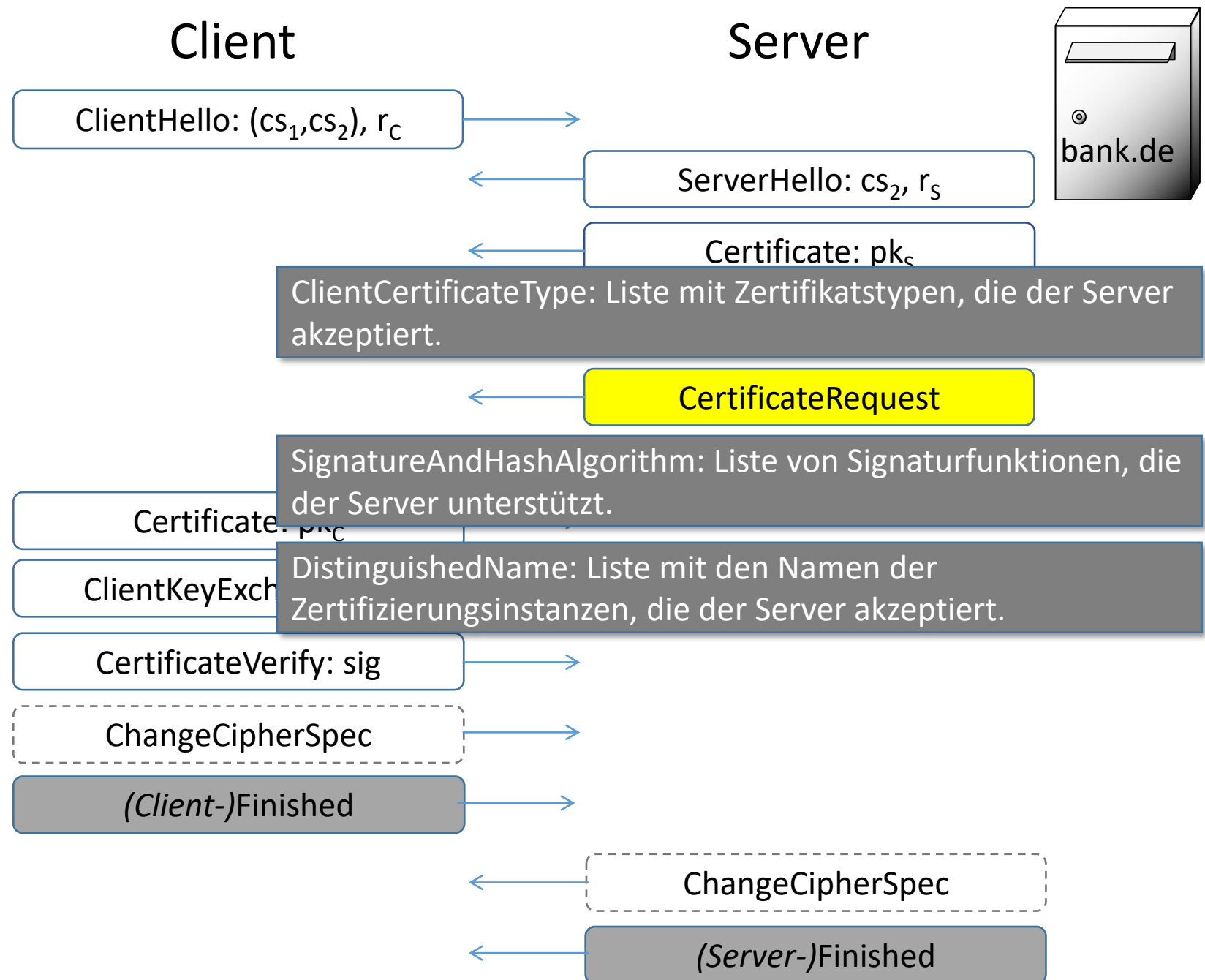
TLS Handshake



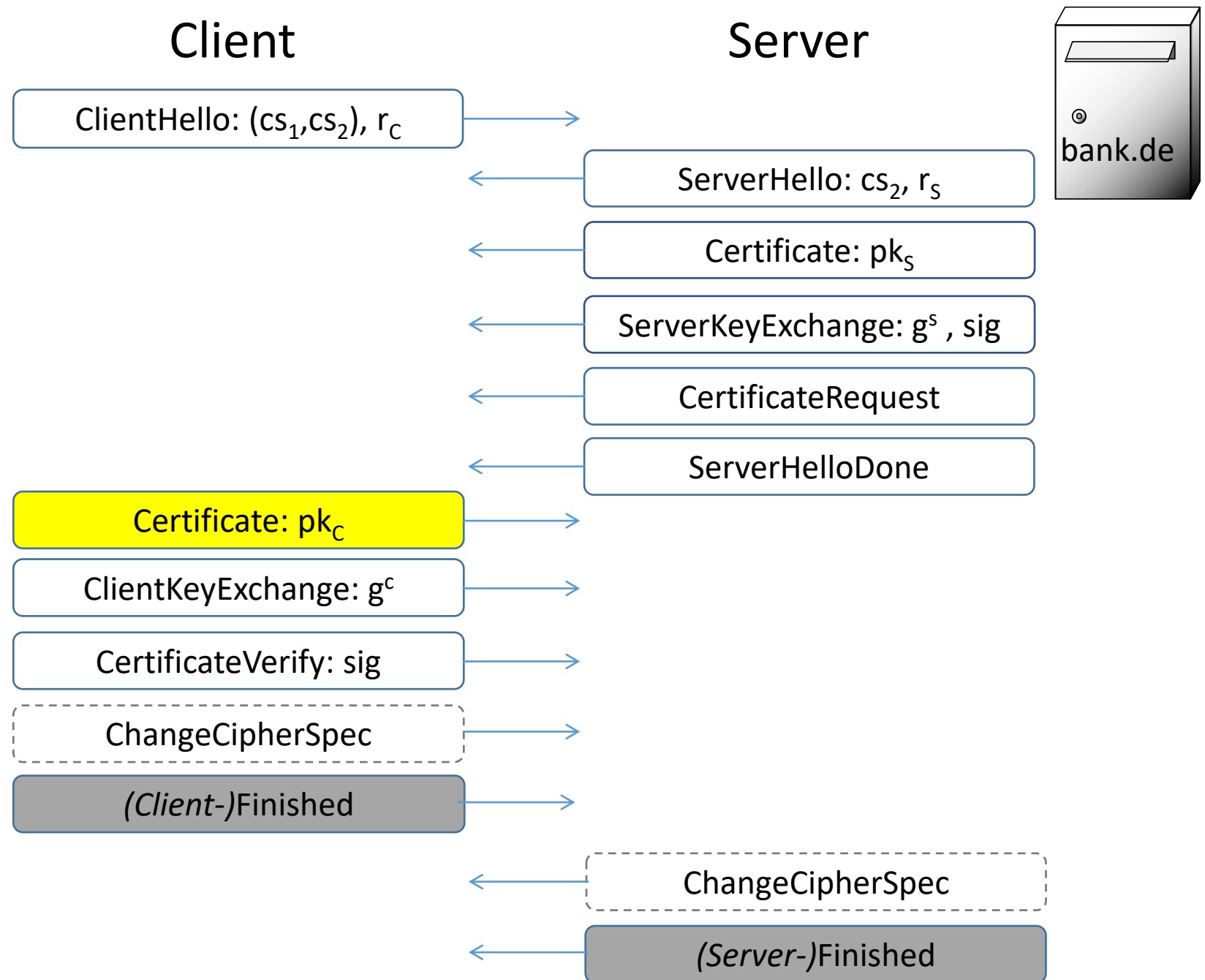
TLS Handshake



TLS Handshake

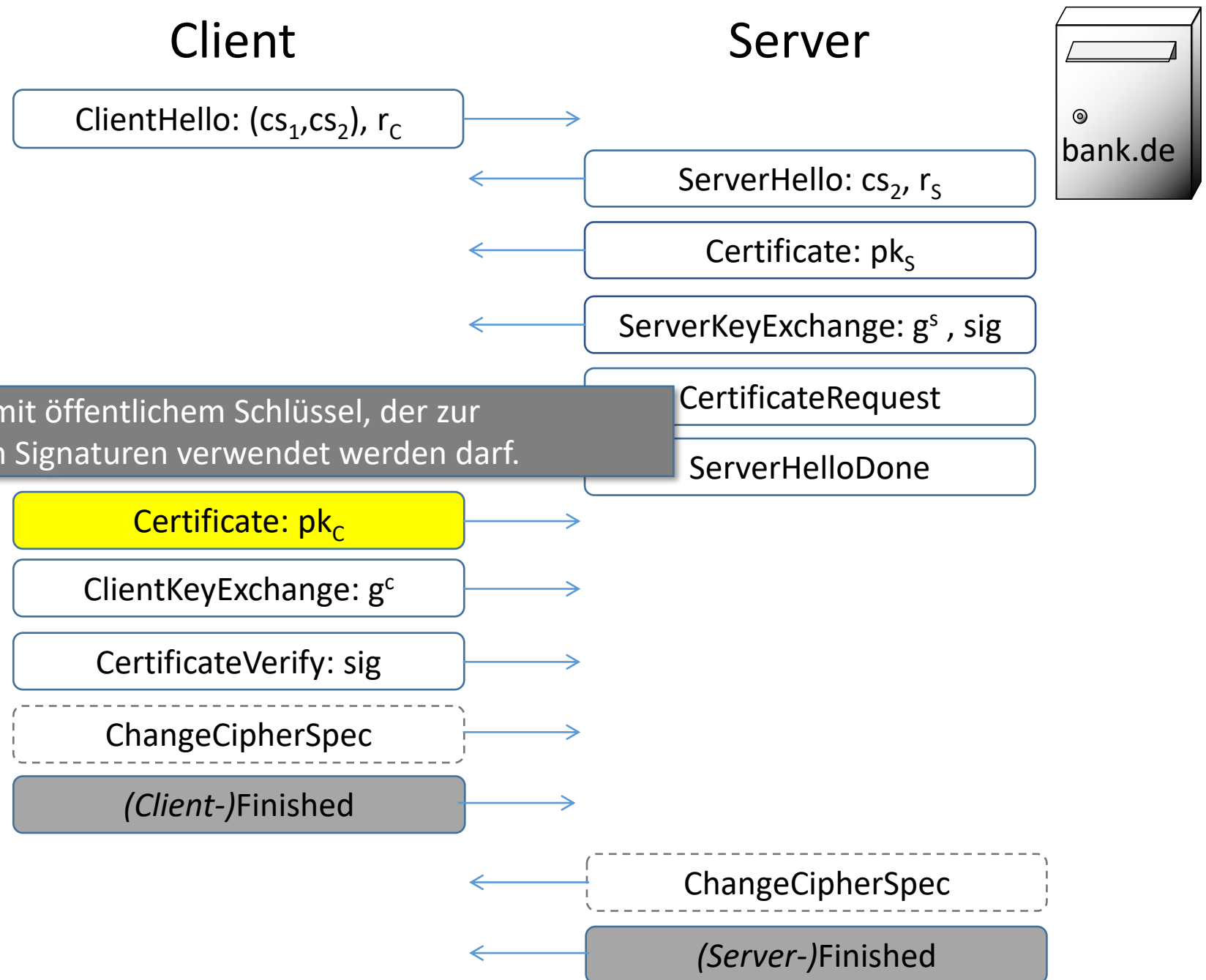


TLS Handshake

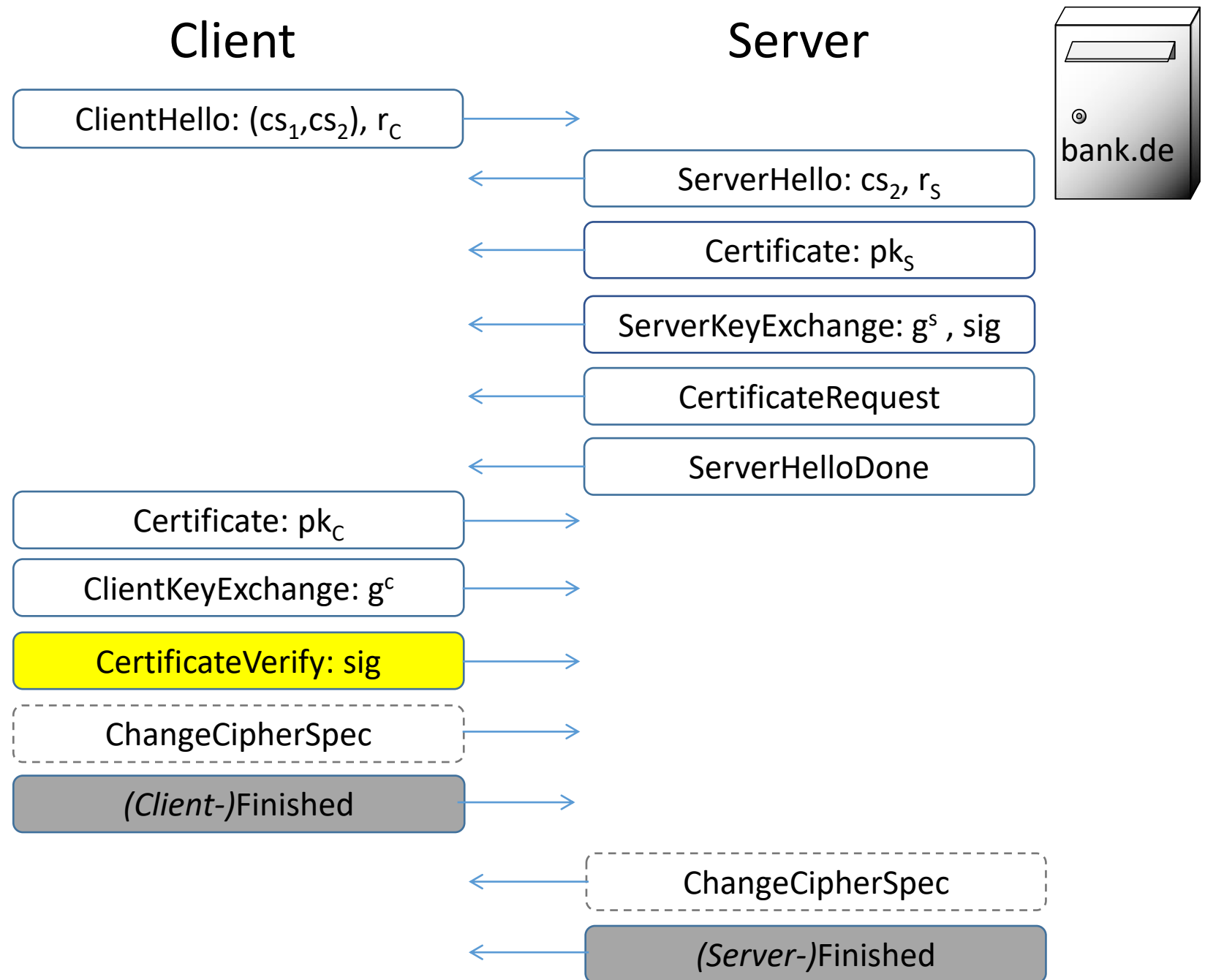


TLS Handshake

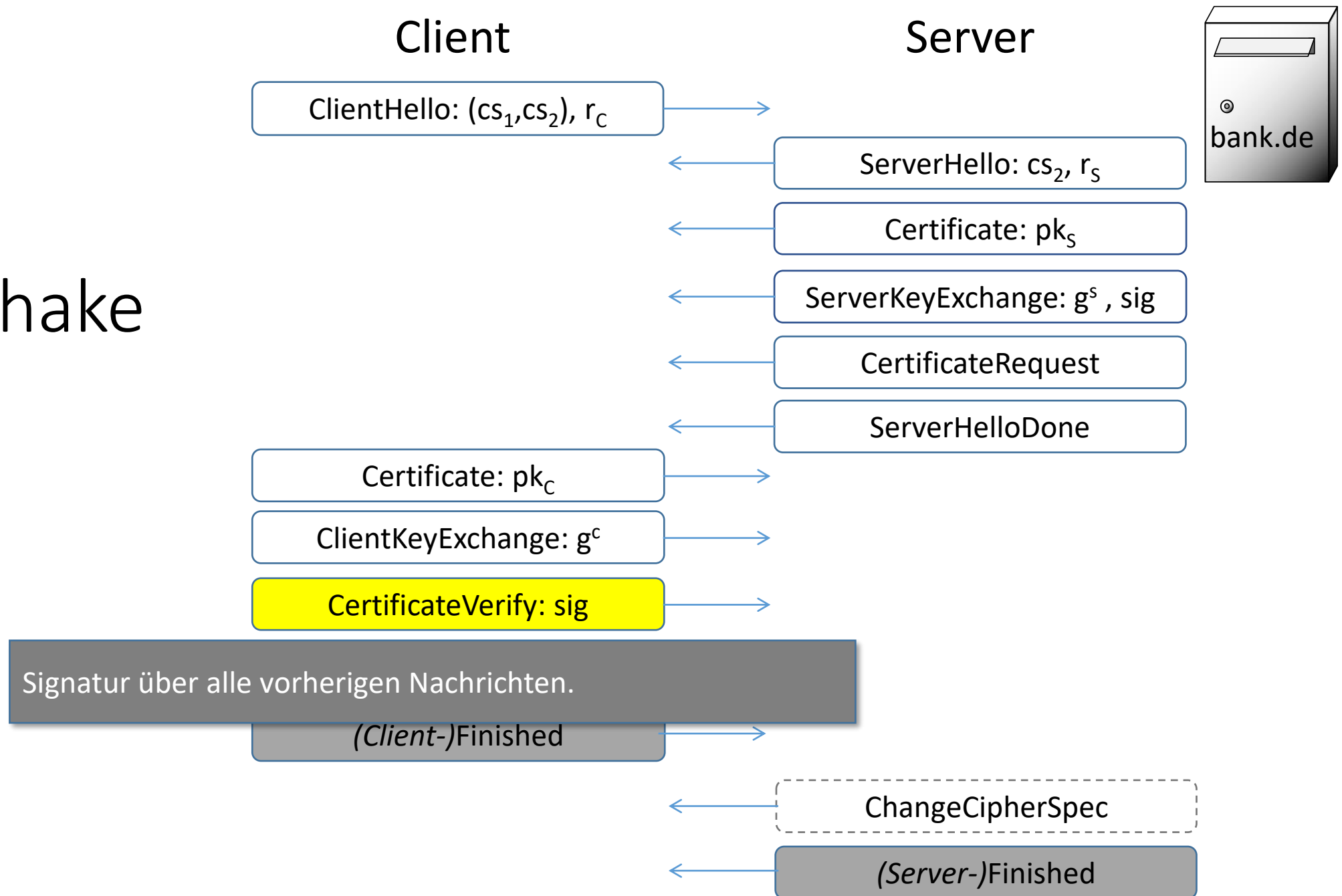
X.509-Zertifikat mit öffentlichem Schlüssel, der zur Überprüfung von Signaturen verwendet werden darf.



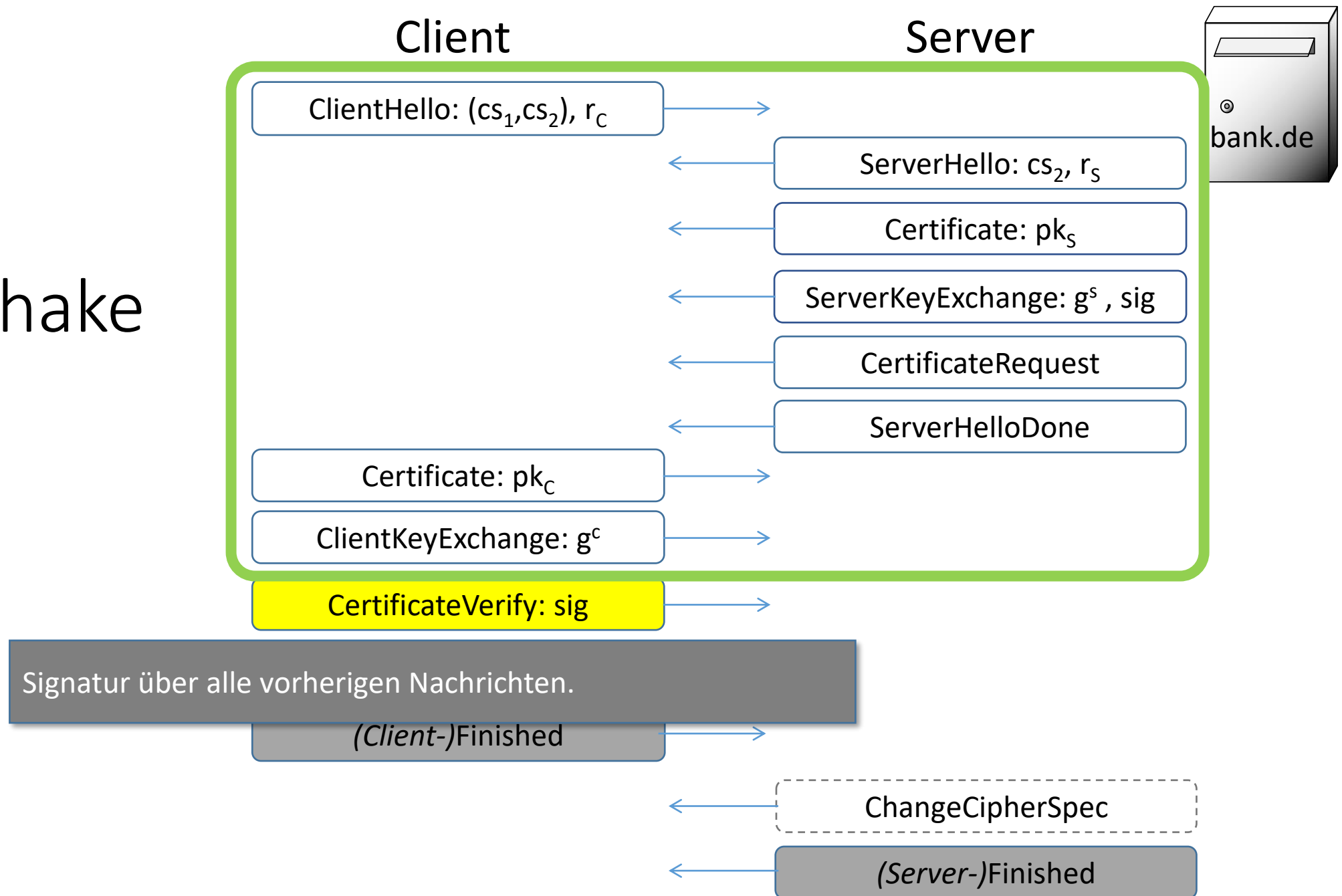
TLS Handshake



TLS Handshake

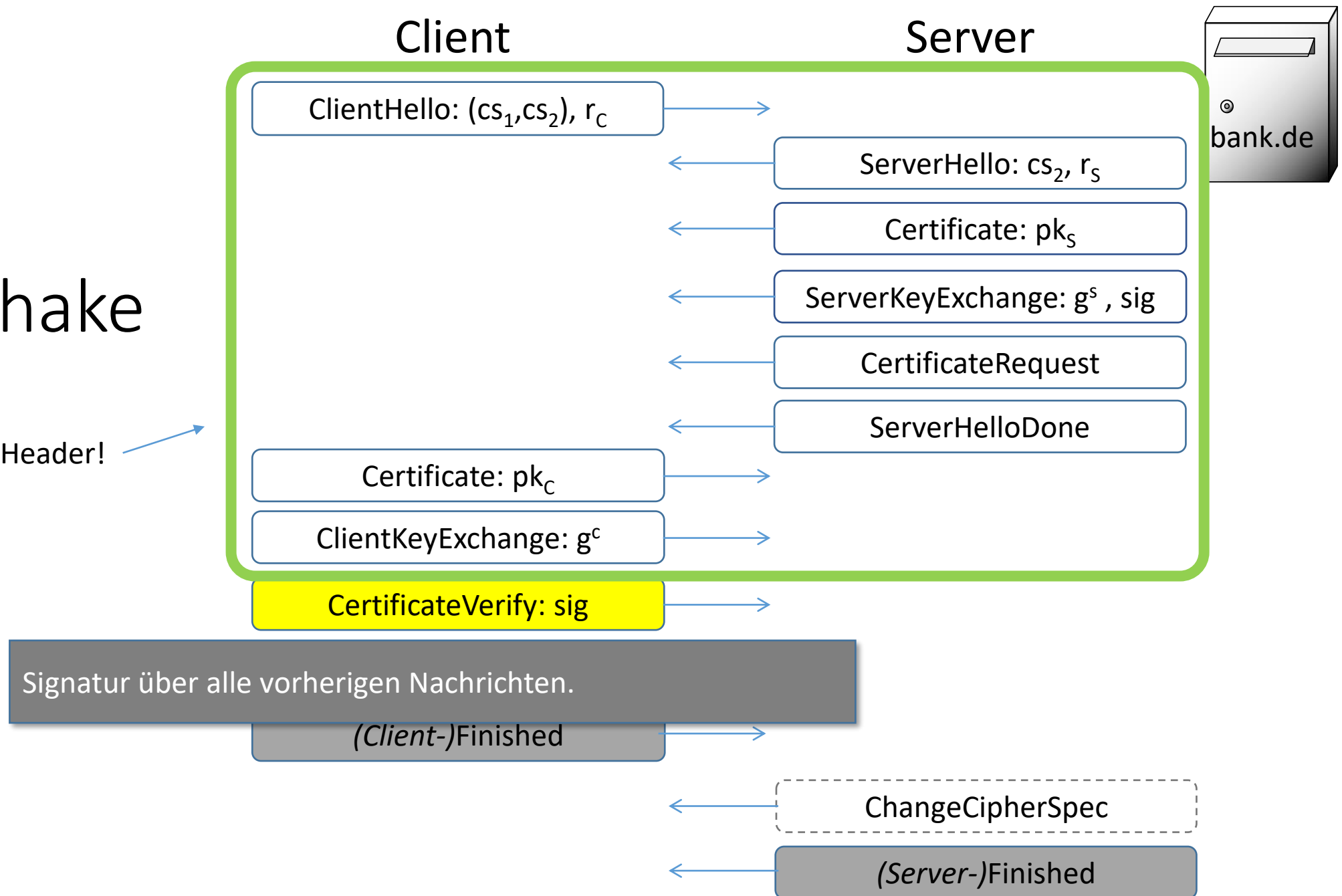


TLS Handshake



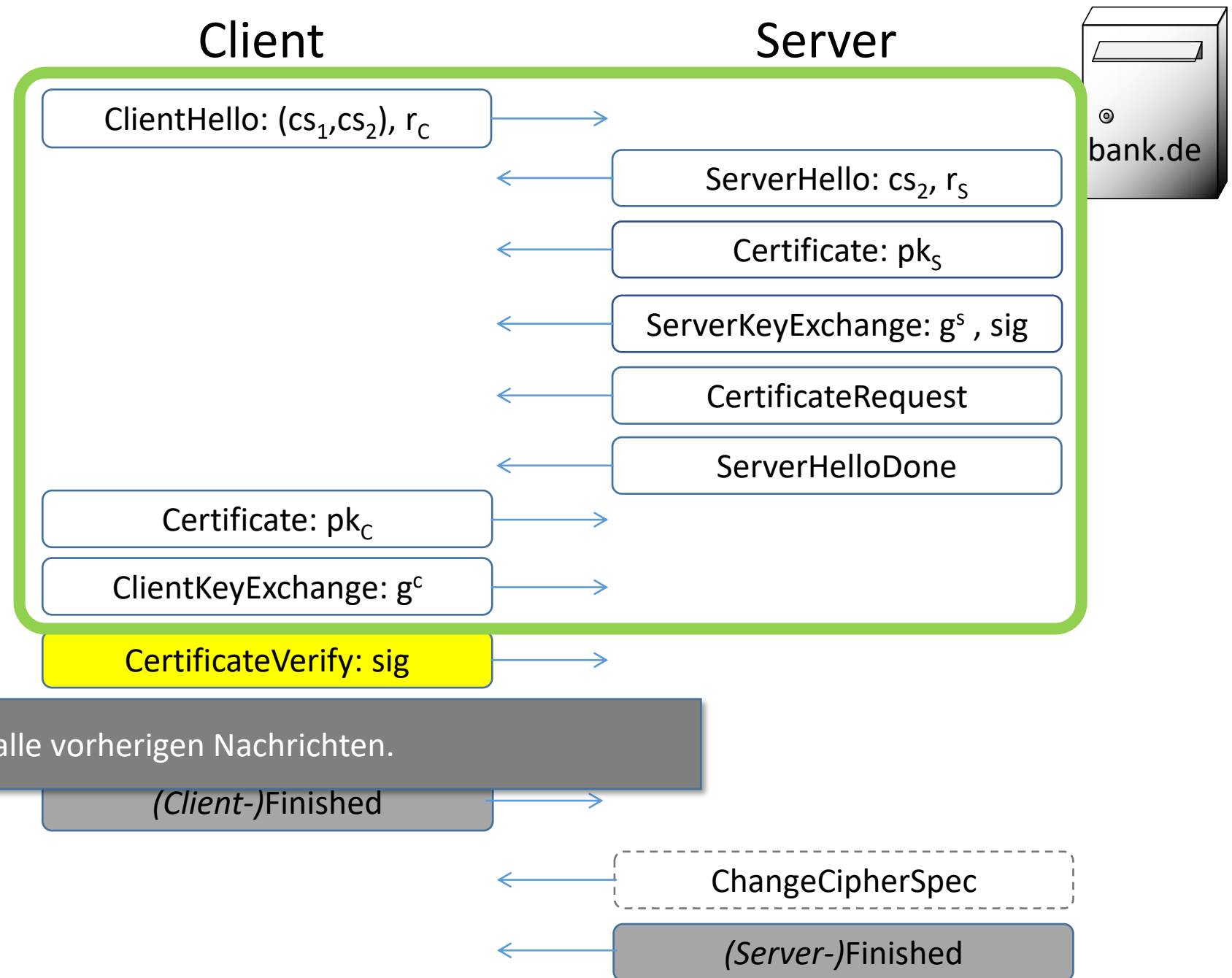
TLS Handshake

OHNE Record Layer Header!



TLS Handshake

Insbesondere fließen auch die
beiden Zufallszahlen r_C und r_S ein!



2.4 TLS-Hilfsprotokolle: ChangeCipherSpec und Alert

Alert: TLS 1.2

close_notify	0	illegal_parameter	47
unexpected_message	10	unknown_ca	48
bad_record_mac	20	access_denied	49
decryption_failed_RESERVED	21	decode_error	50
record_overflow	22	decrypt_error	51
decompression_failure	30	export_restrictions_RESERVED	60
handshake_failure	40	protocol_version	70
no_certificate_RESERVED	41	insufficient_security	71
bad_certificate	42	internal_error	80
unsupported_certificate	43	user_canceled	90
certificate_revoked	44	no_renegotiation	100
certificate_expired	45	unsupported_extension	110
certificate_unknown	46		

Alert: TLS 1.2

close_notify	0	illegal_parameter	47
unexpected_message	10	unknown_ca	48
bad_record_mac	20	access_denied	49
decryption_failed_RESERVED	21	decode_error	50
record_overflow	22	decrypt_error	51
decompression_failure	30	export_restrictions_RESERVED	60
handshake_failure	40	protocol_version	70
no_certificate_RESERVED	41	insufficient_security	71
bad_certificate	42	internal_error	80
unsupported_certificate	43	user_canceled	90
certificate_revoked	44	no_renegotiation	100
certificate_expired	45	unsupported_extension	110
certificate_unknown	46		

Schwere des Fehlers (Byte 1)

- fatal: TLS Session muss beendet werden; Schlüsselmaterial (incl. MasterSecret) wird ungültig

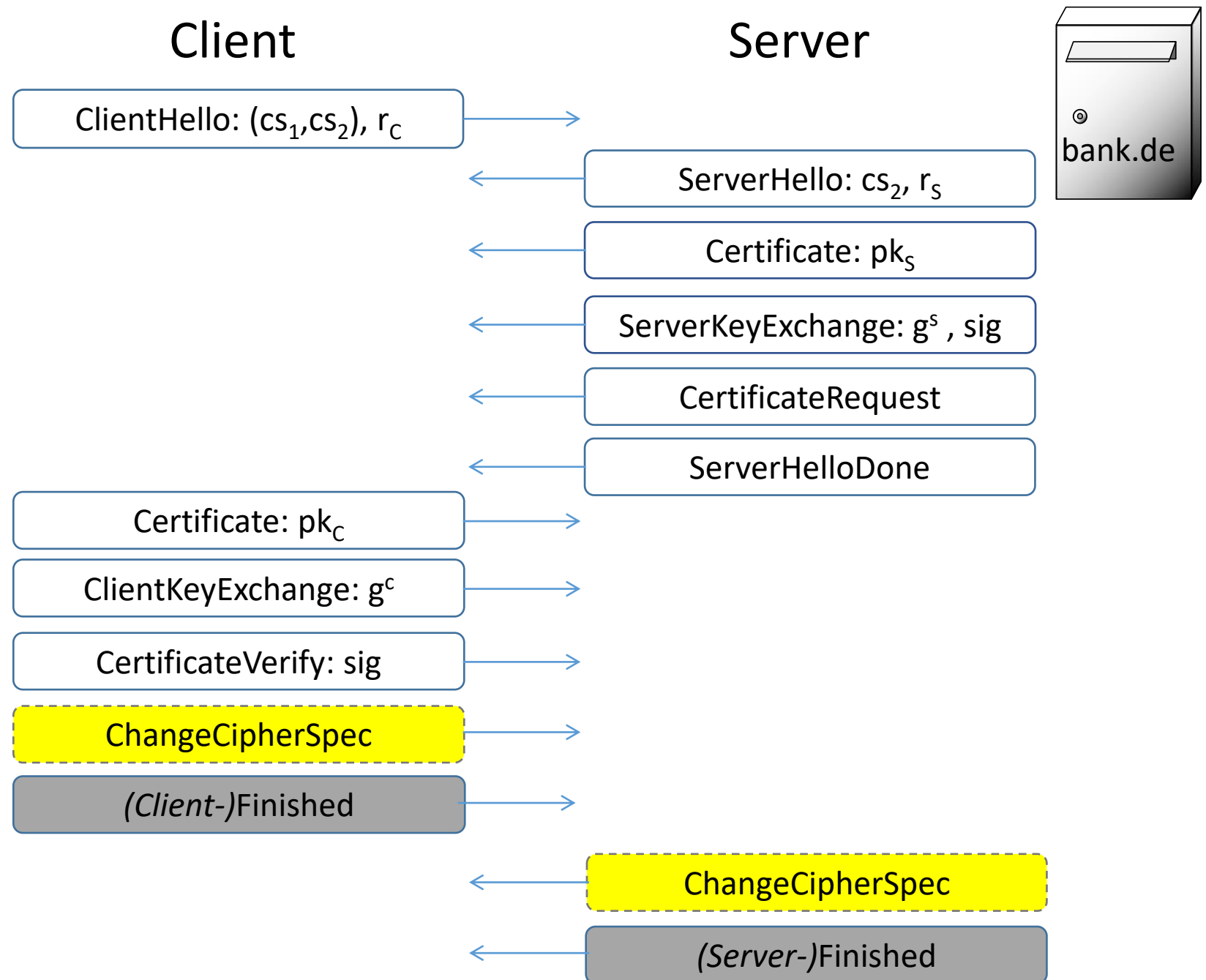
Alert: TLS 1.2

close_notify	0	illegal_parameter	47
unexpected_message	10	unknown_ca	48
bad_record_mac	20	access_denied	49
decryption_failed_RESERVED	21	decode_error	50
record_overflow	22	decrypt_error	51
decompression_failure	30	export_restrictions_RESERVED	60
handshake_failure	40	protocol_version	70
no_certificate_RESERVED	41	insufficient_security	71
bad_certificate	42	internal_error	80
unsupported_certificate	43	user_canceled	90
certificate_revoked	44	no_renegotiation	100
certificate_expired	45	unsupported_extension	110
certificate_unknown	46		

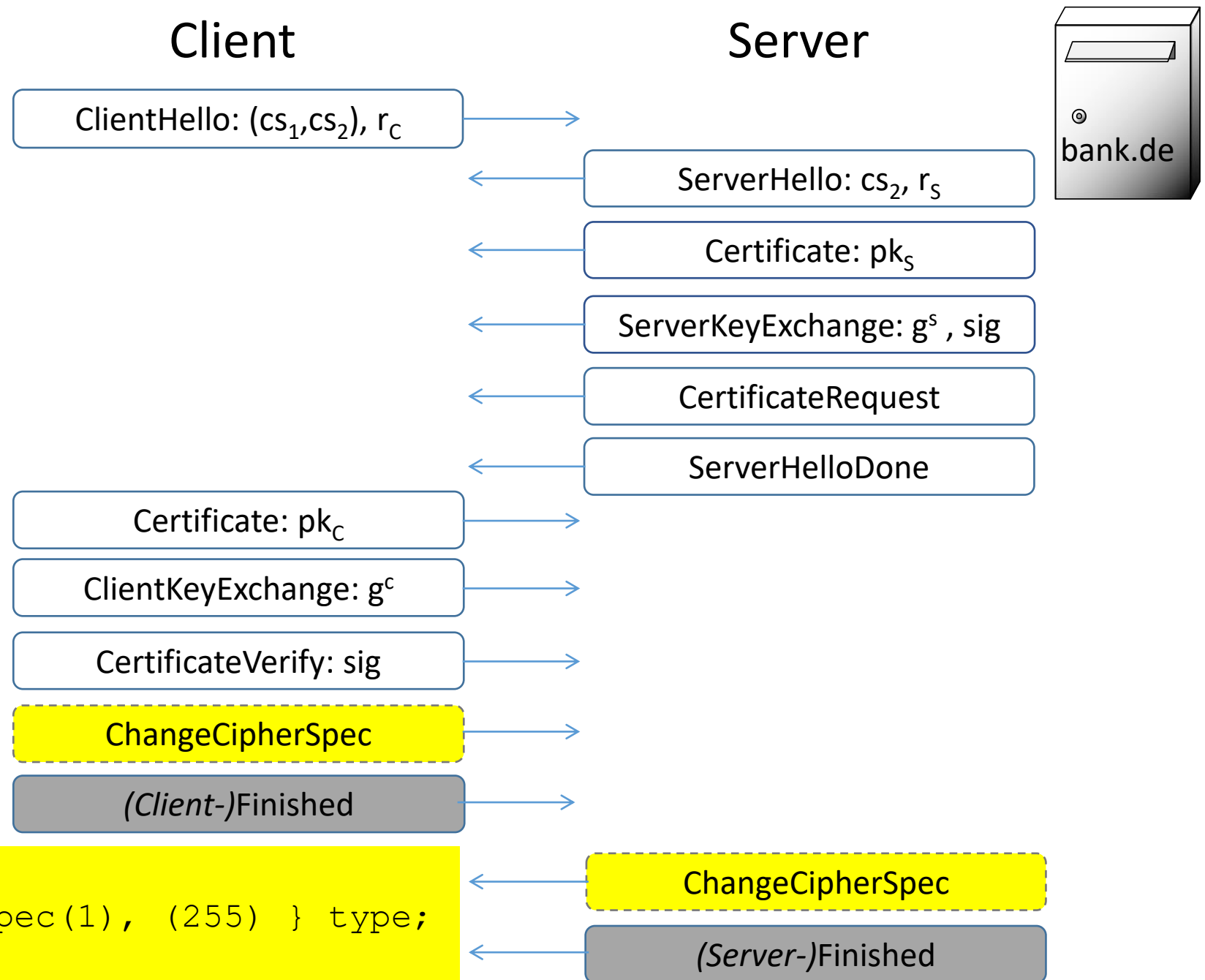
Schwere des Fehlers (Byte 1)

- **fatal:** TLS Session muss beendet werden; Schlüsselmaterial (incl. MasterSecret) wird ungültig
- **warning:** Kommunikationspartner darf selbst entscheiden, wie er reagieren will

Change- CipherSpec



Change-CipherSpec



```

struct {
    enum { change_cipher_spec(1), (255) } type;
} ChangeCipherSpec;
  
```


2.5 TLS Session Resumption

TLS
Handshake:
Session
Resumption

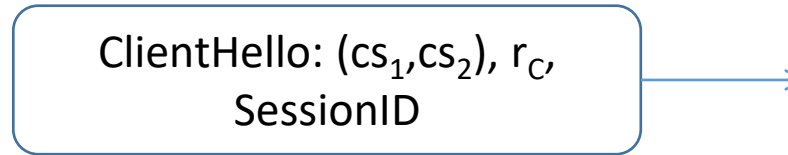
Client:
(SessionID, ms)

Server:
(SessionID, ms)

TLS Handshake: Session Resumption

Client:
(SessionID, ms)

Server:
(SessionID, ms)




TLS Handshake: Session Resumption

Client:
(SessionID, ms)

Server:
(SessionID, ms)

ClientHello: $(cs_1, cs_2), r_C,$
SessionID

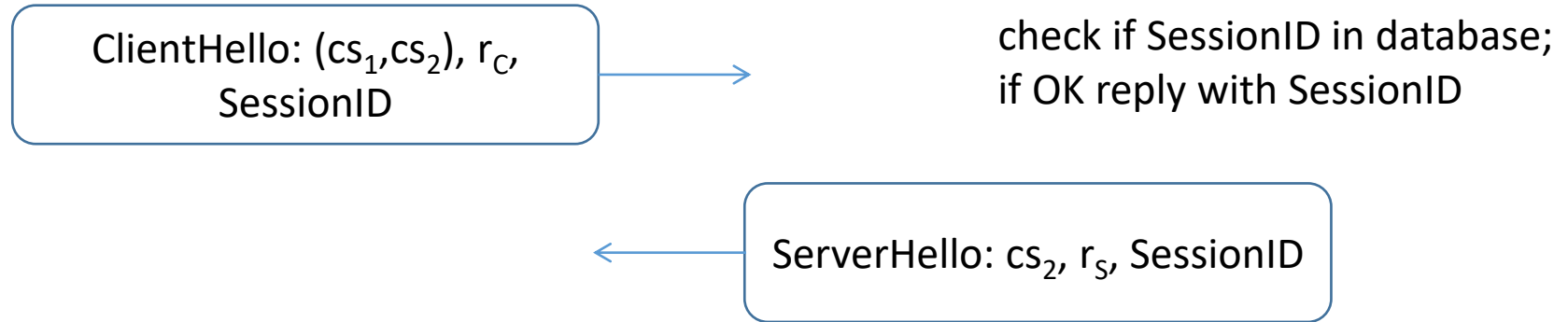


check if SessionID in database;
if OK reply with SessionID

TLS Handshake: Session Resumption

Client:
(SessionID, ms)

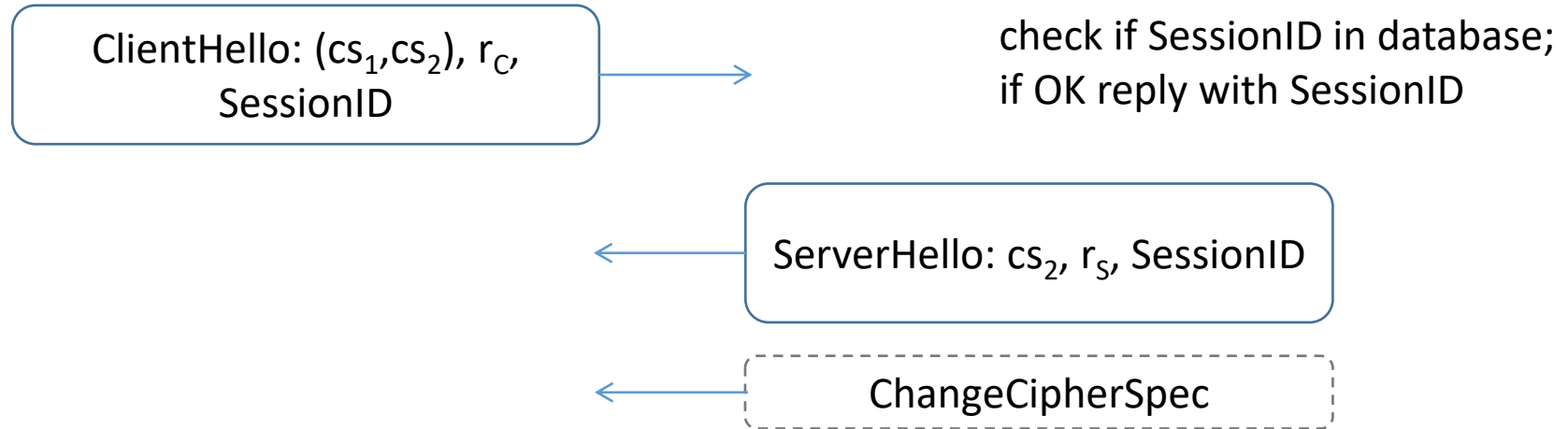
Server:
(SessionID, ms)



TLS Handshake: Session Resumption

Client:
(SessionID, ms)

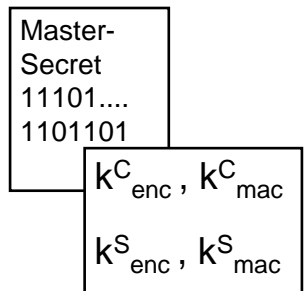
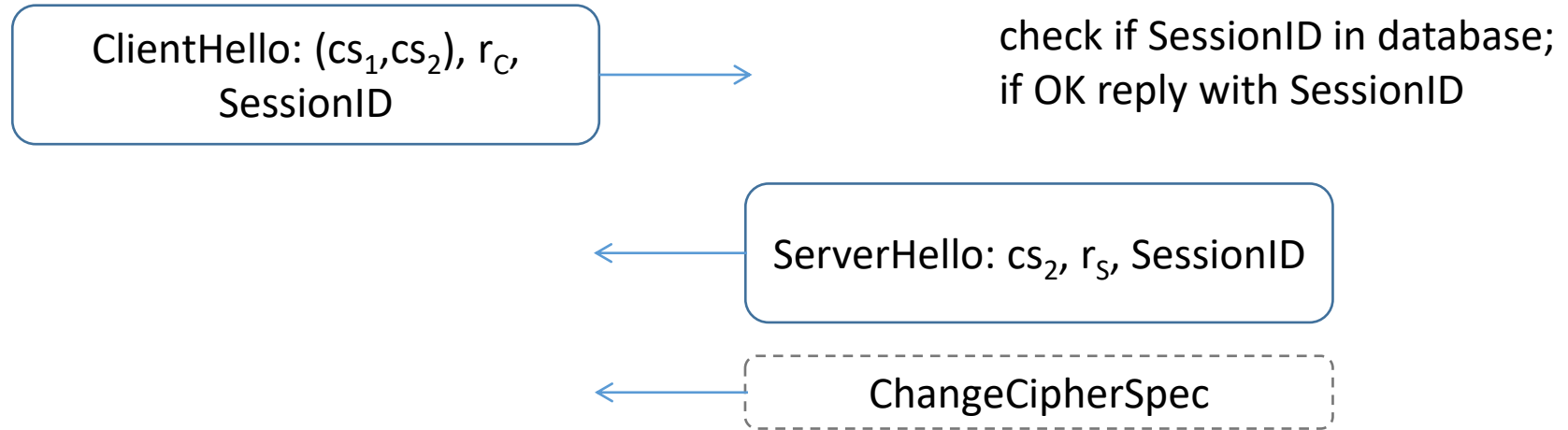
Server:
(SessionID, ms)



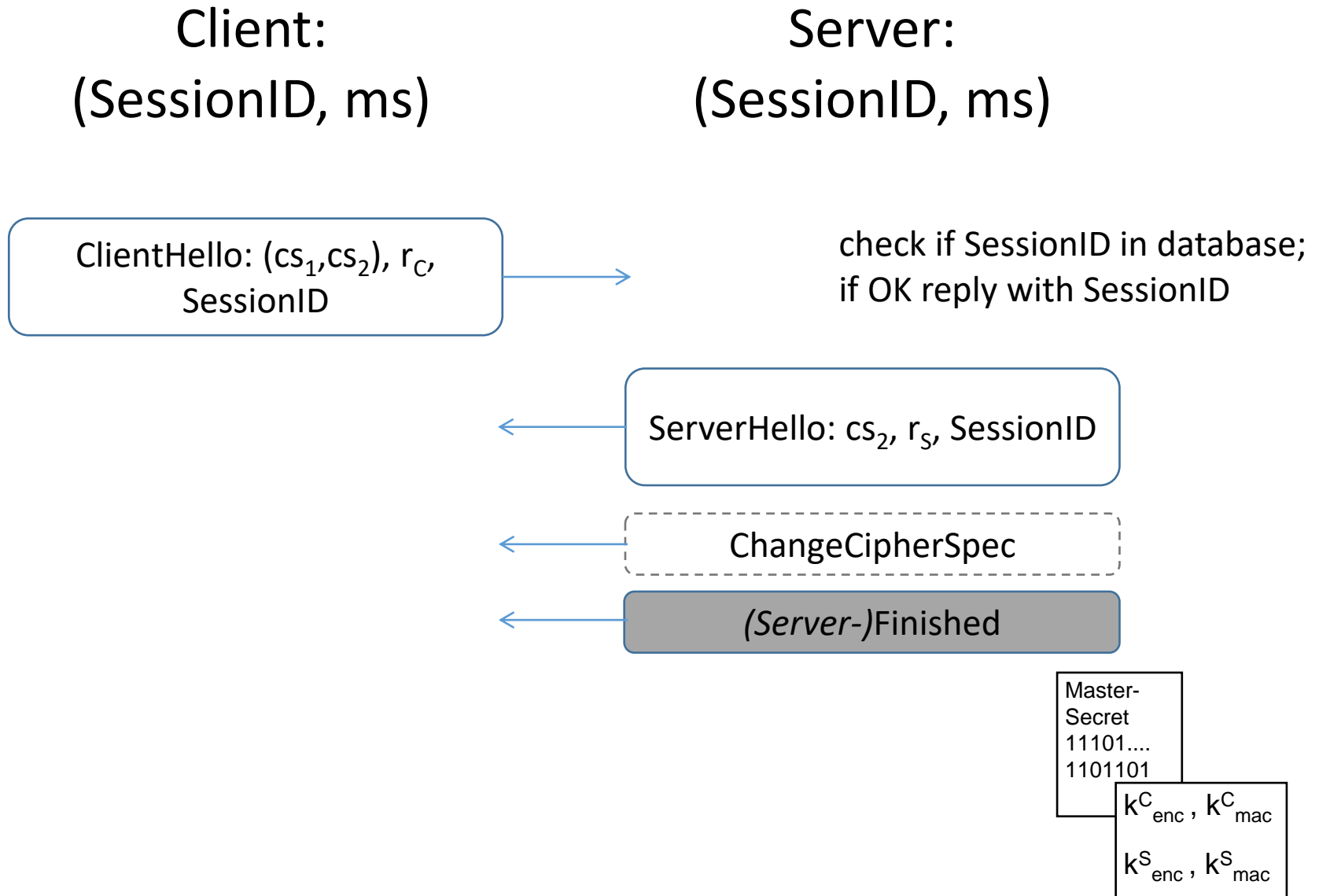
TLS Handshake: Session Resumption

Client:
(SessionID, ms)

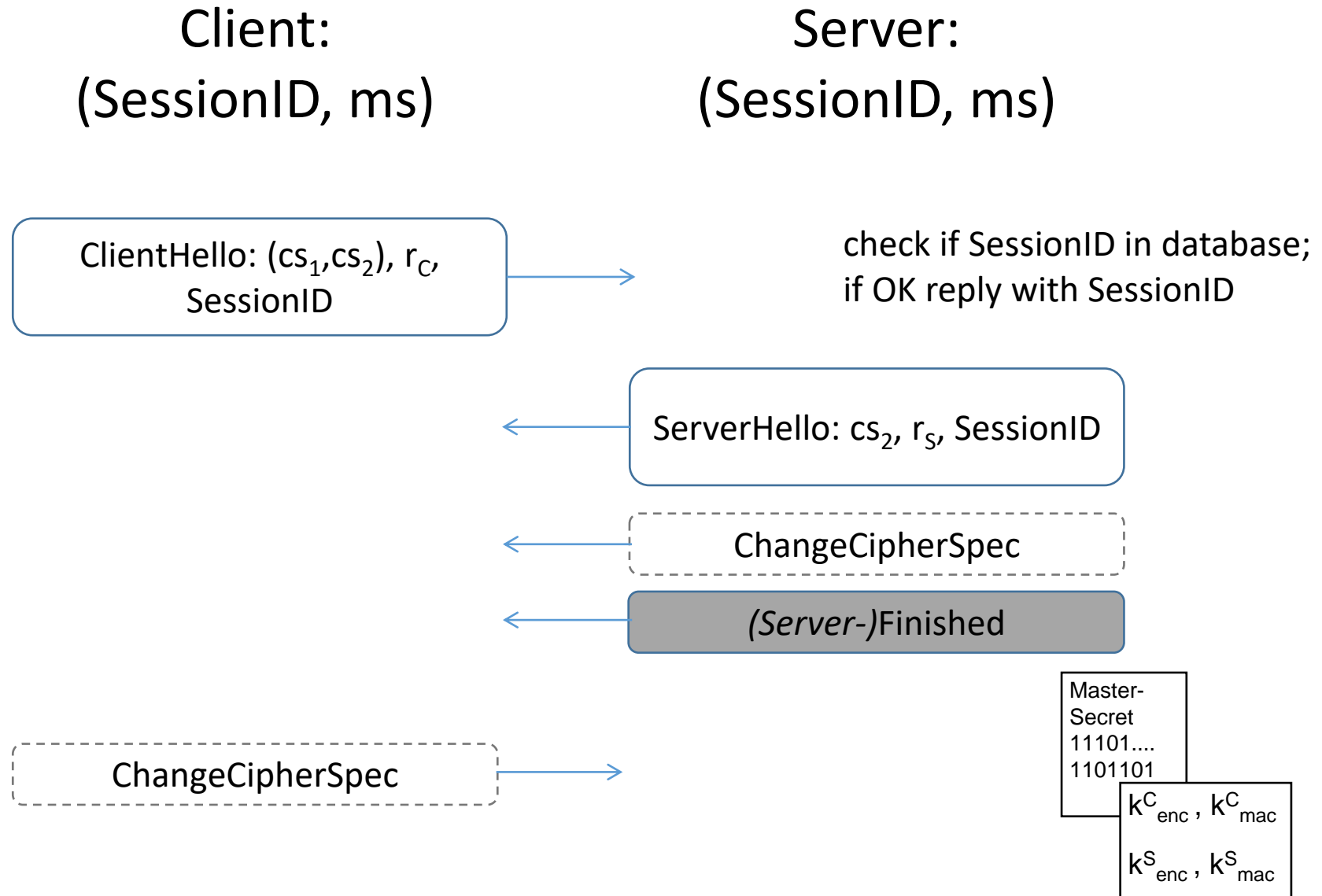
Server:
(SessionID, ms)



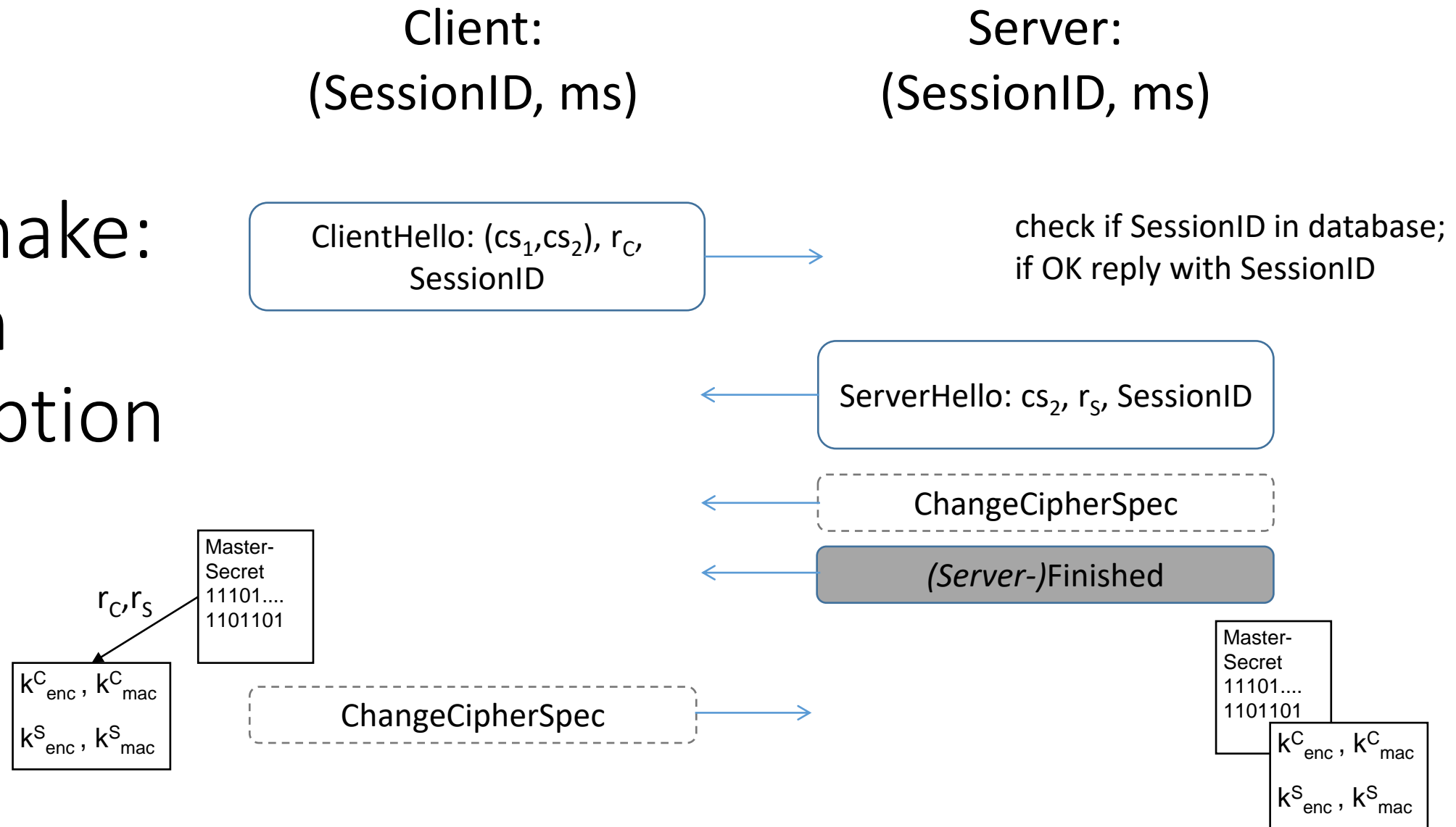
TLS Handshake: Session Resumption



TLS Handshake: Session Resumption

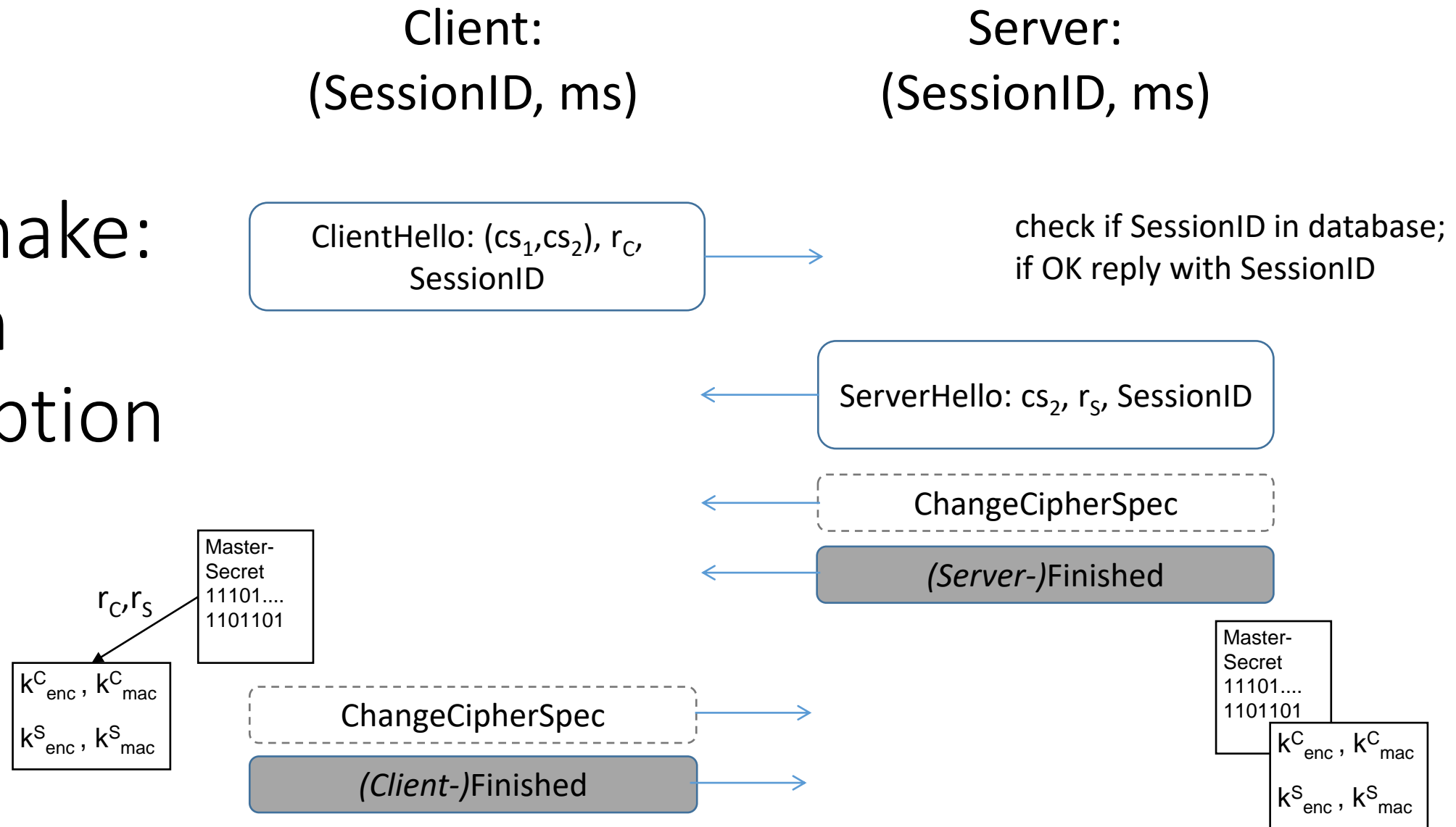


TLS Handshake: Session Resumption



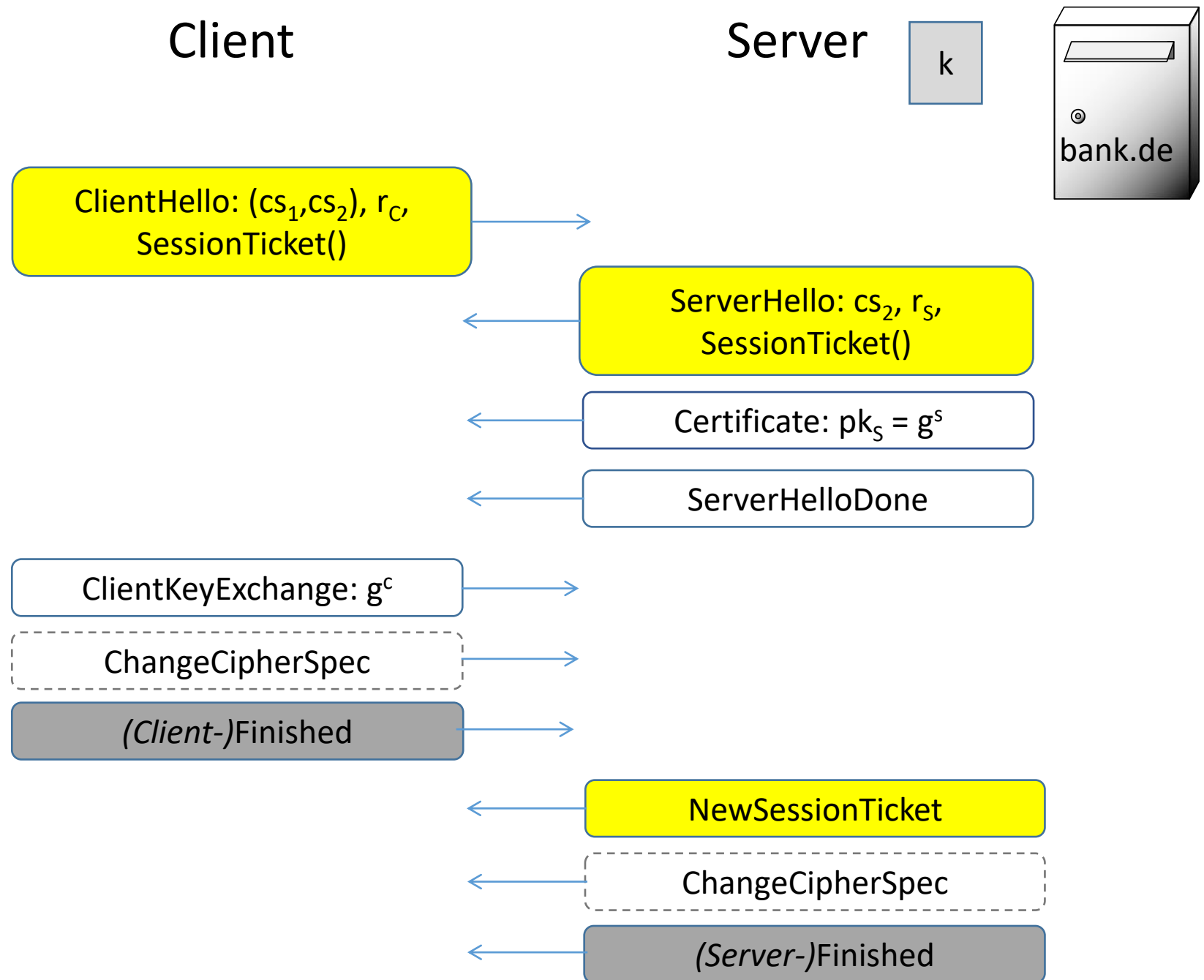
TLS

Handshake: Session Resumption

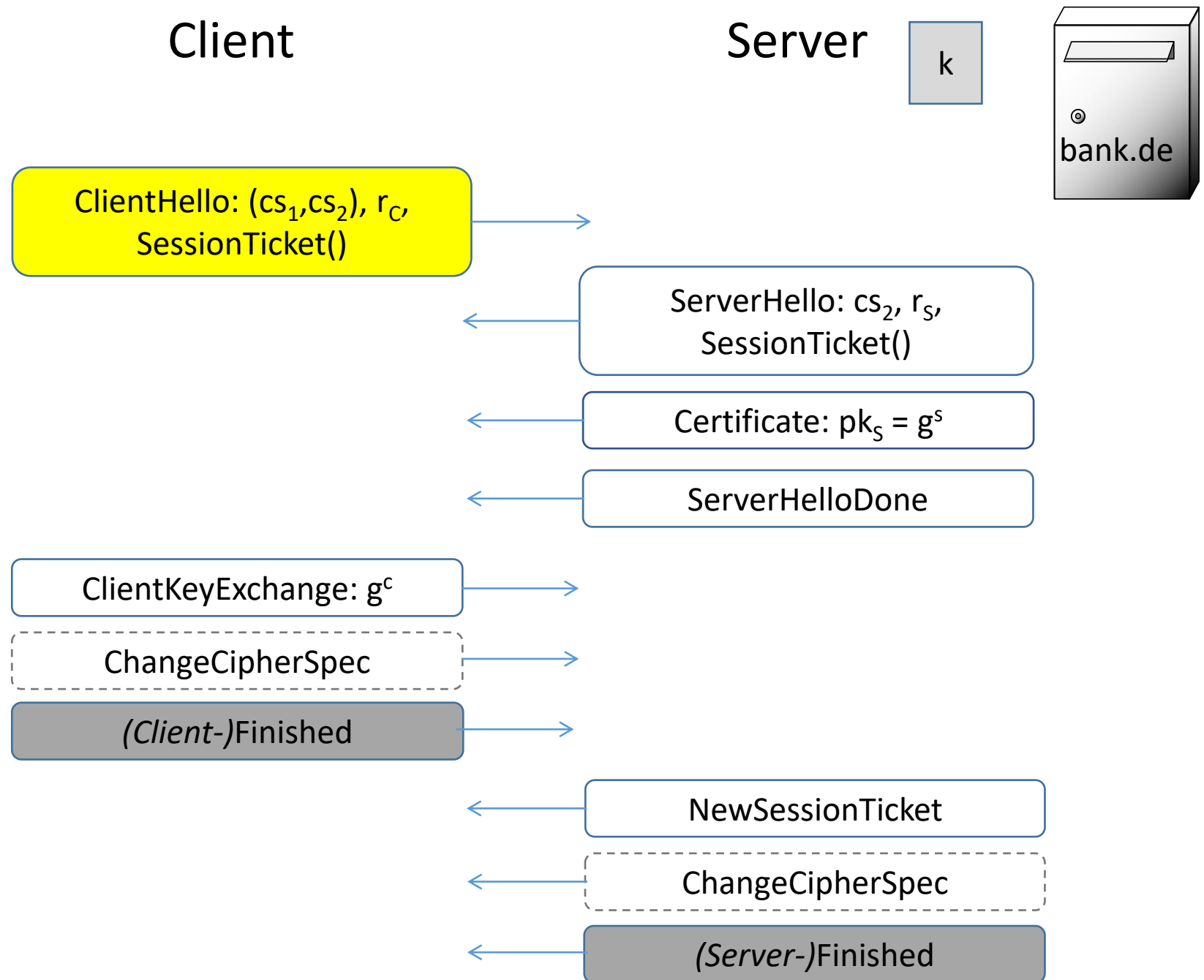


TLS Handshake: Session Resumption with Session Ticket

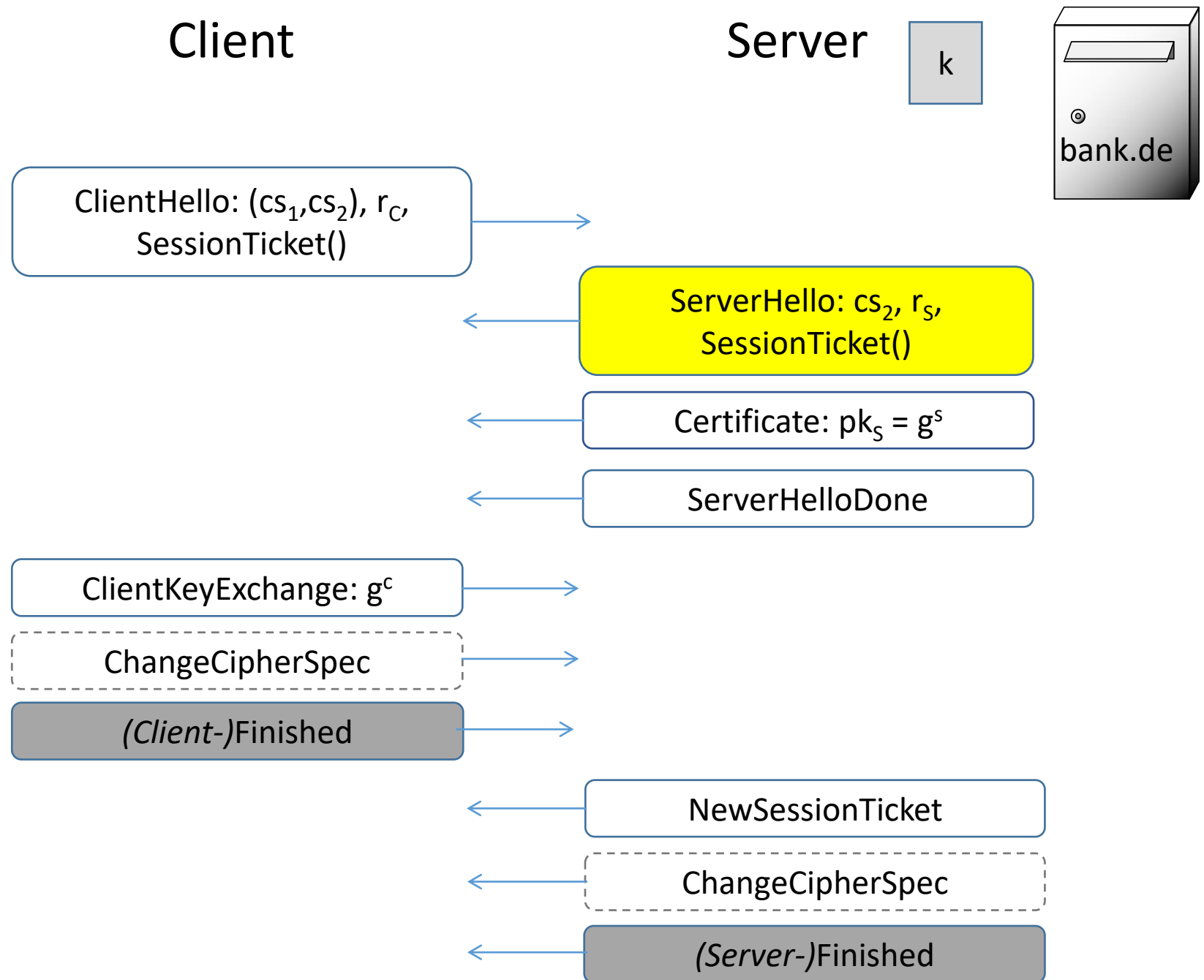
TLS Handshake: Setting a Session Ticket



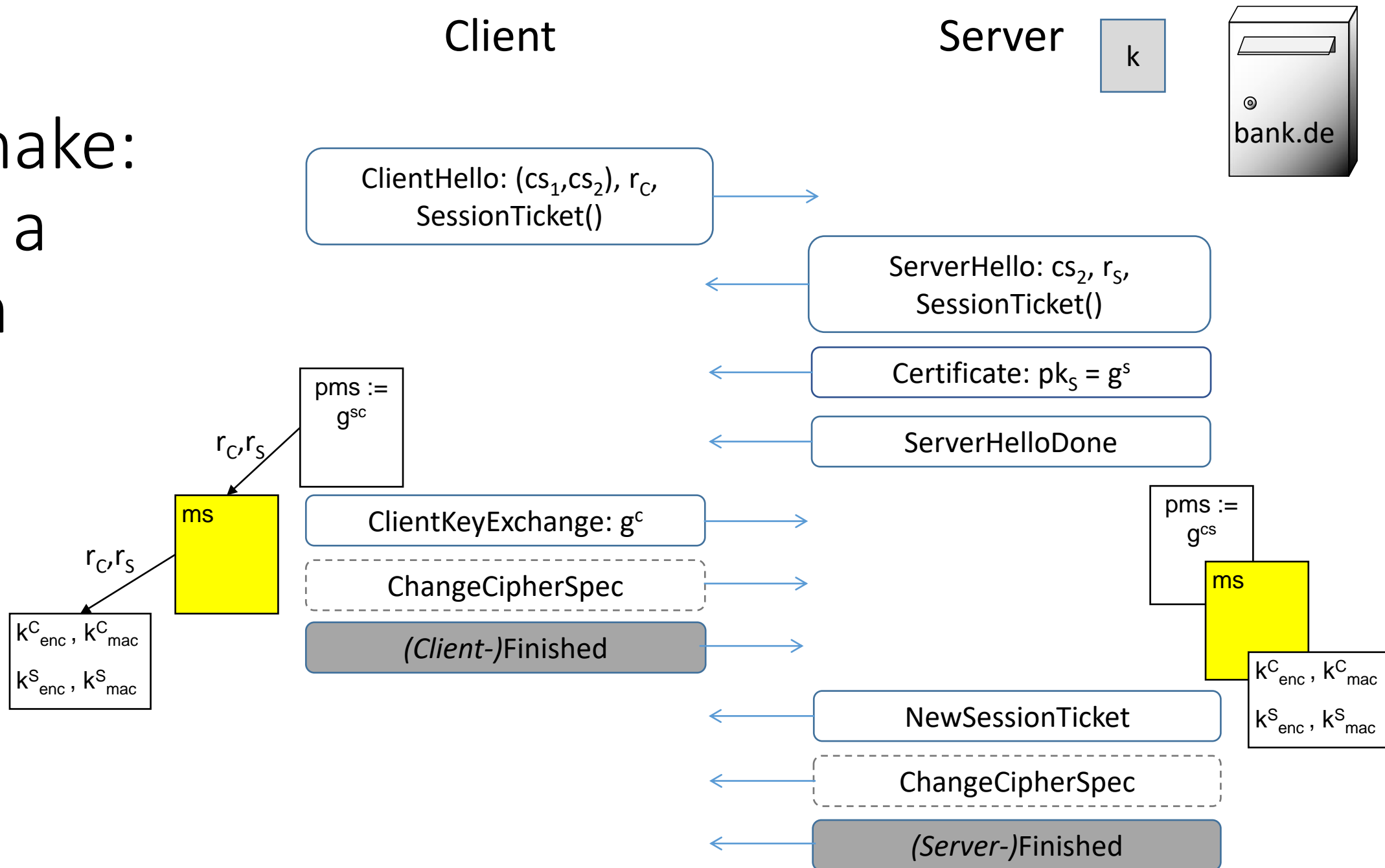
TLS Handshake: Setting a Session Ticket



TLS Handshake: Setting a Session Ticket



TLS Handshake: Setting a Session Ticket

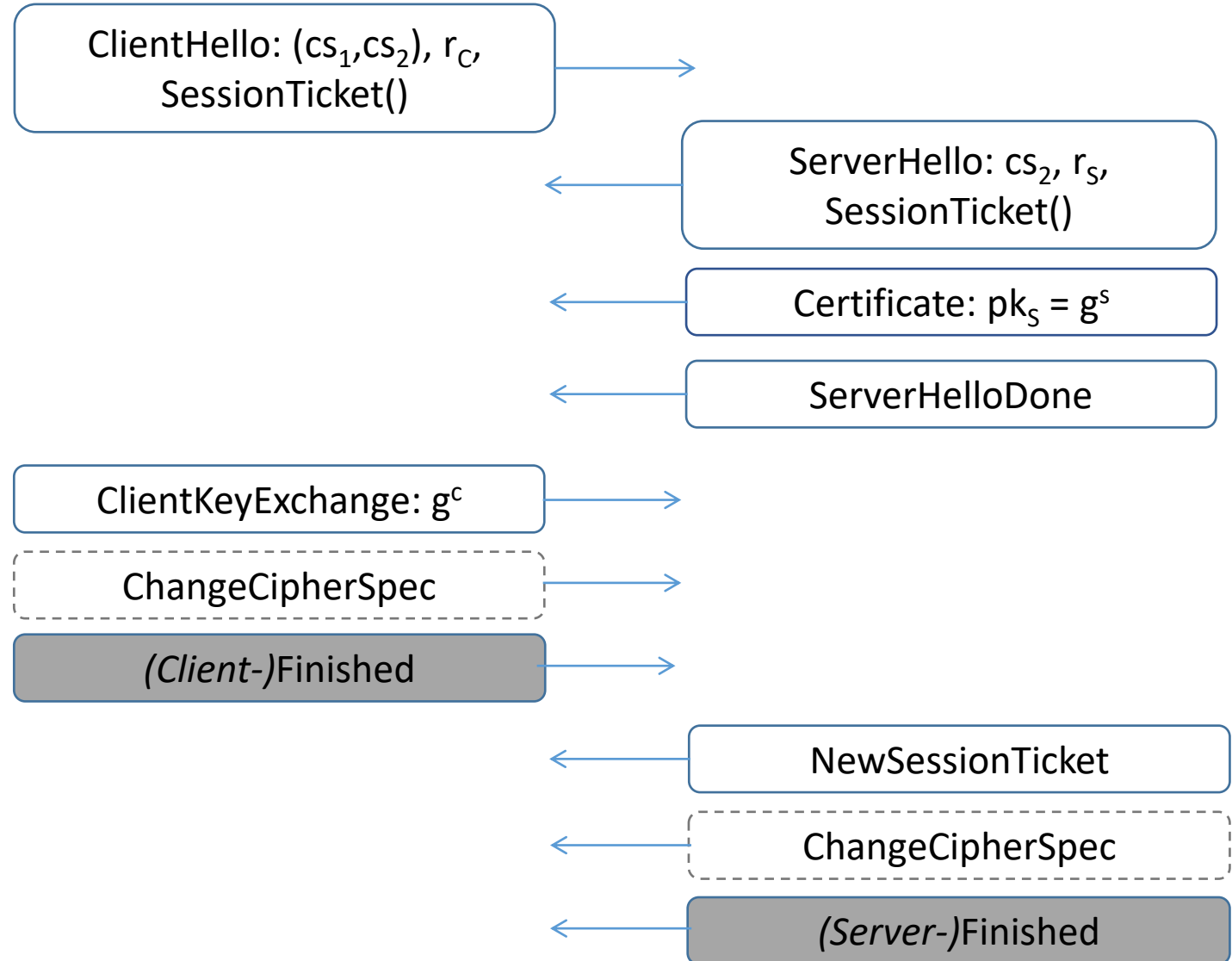
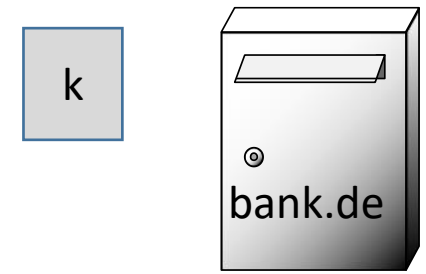


TLS Handshake: Setting a Session Ticket

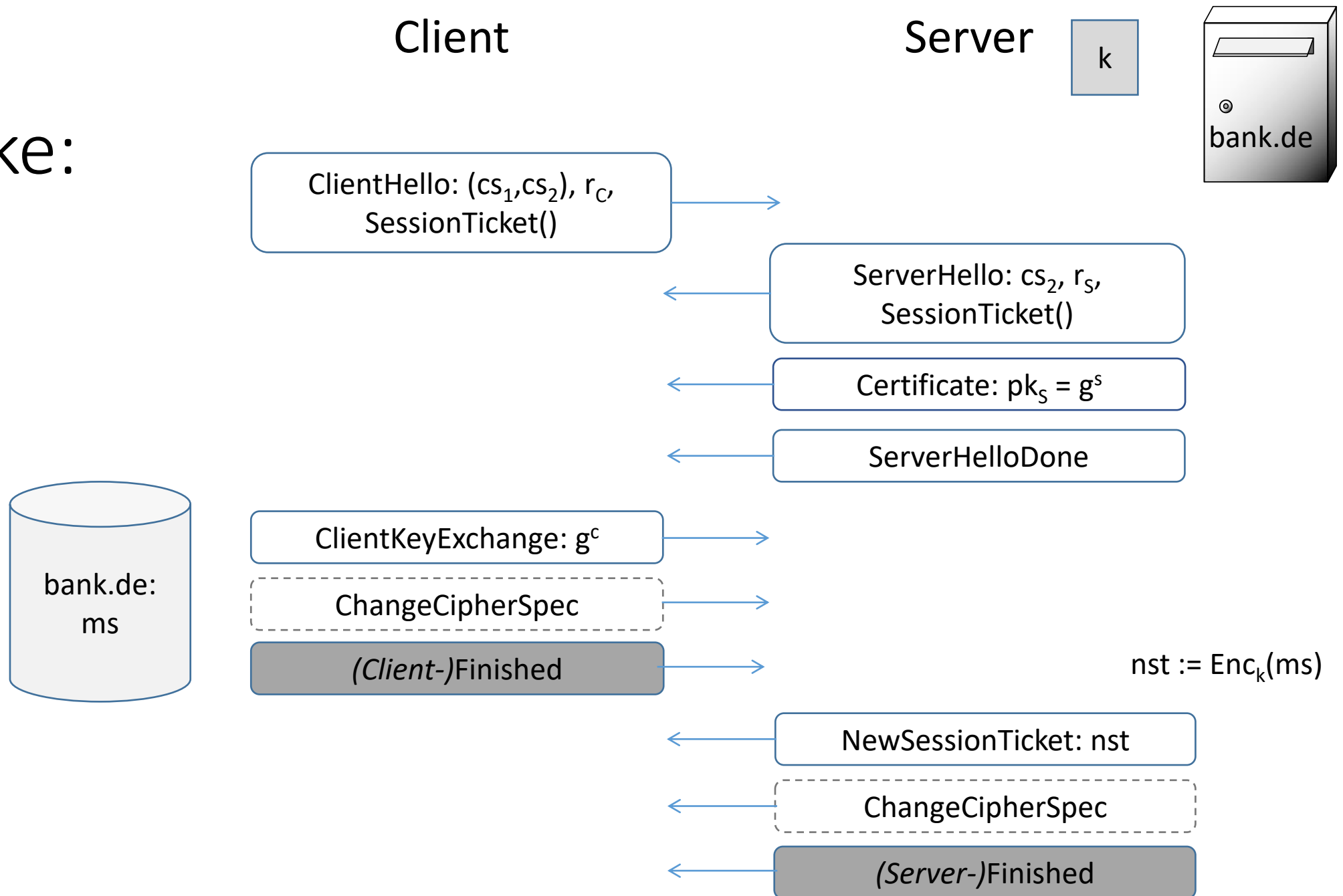


Client

Server



TLS Handshake: Setting a Session Ticket

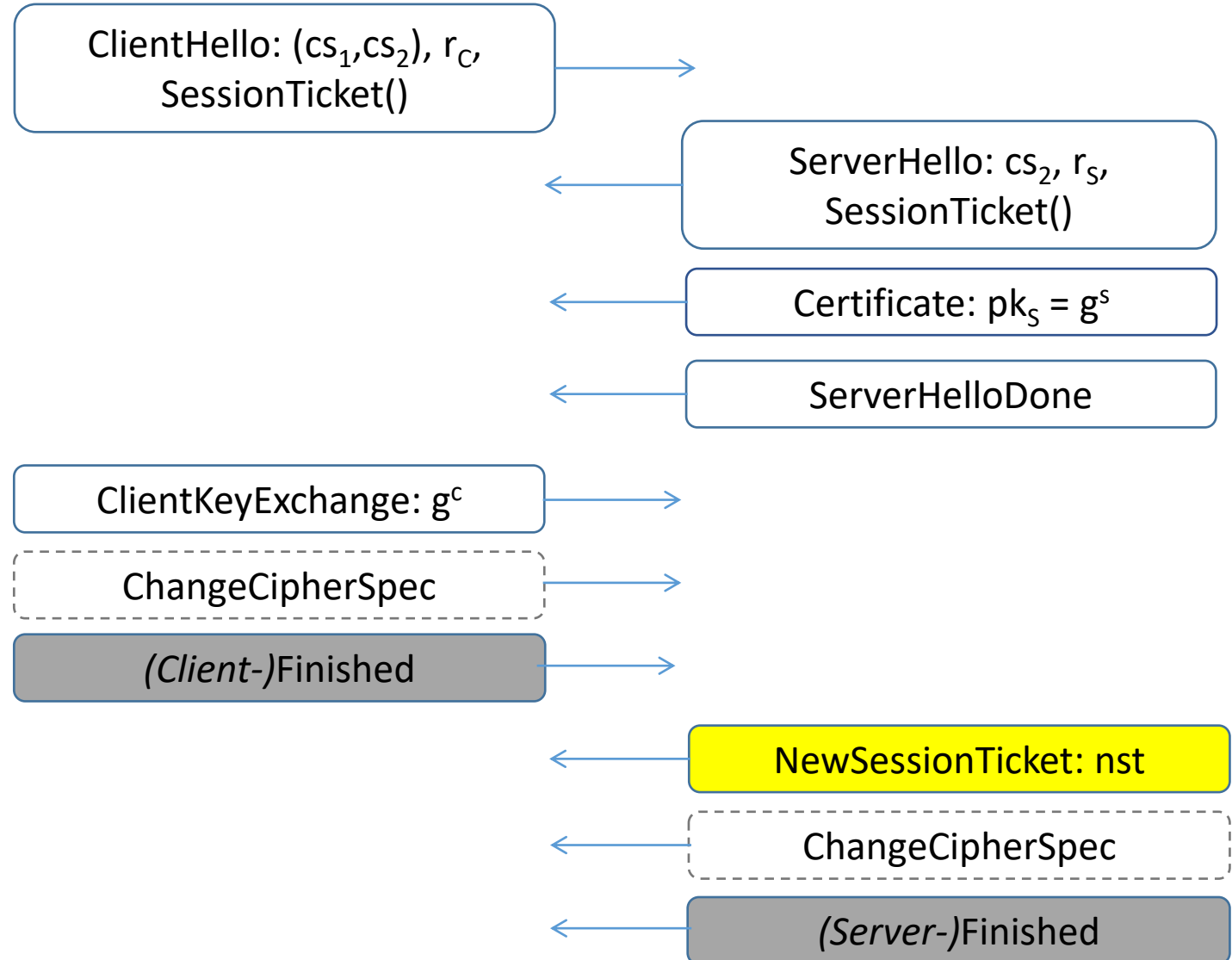
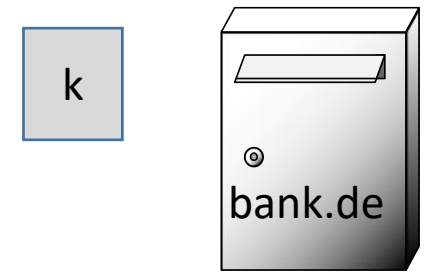


TLS Handshake: Setting a Session Ticket

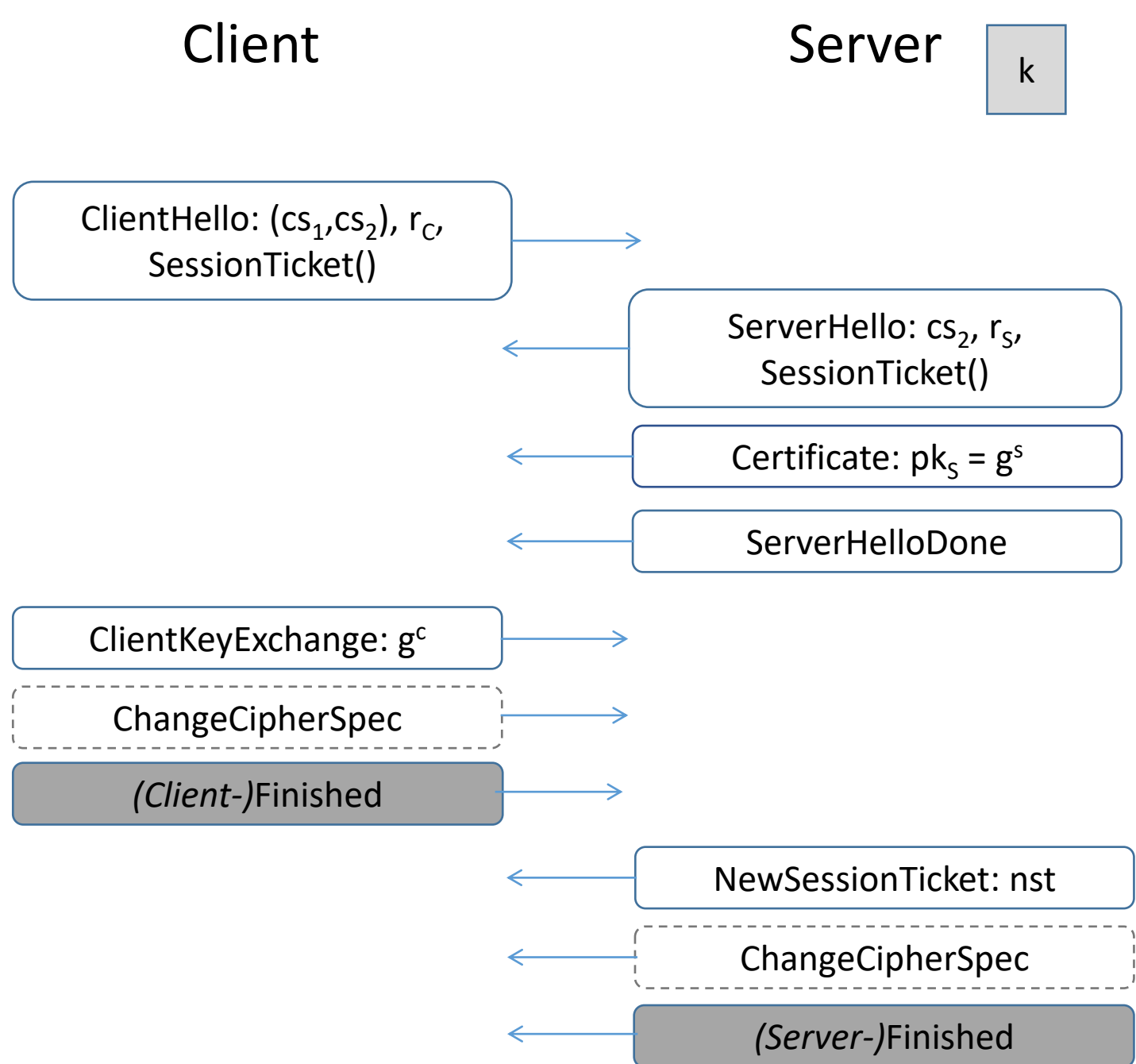


Client

Server



TLS Handshake: Setting a Session Ticket



TLS

Handshake:

Session

Resumption

with

Session

Ticket

Client:

(bank.de: ms, $t = E_k(ms)$)

Server:

bank.de


k

TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

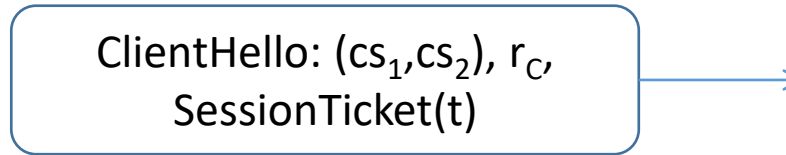
Server: 
bank.de

ClientHello: $(cs_1, cs_2), r_C,$
SessionTicket(t)



TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)



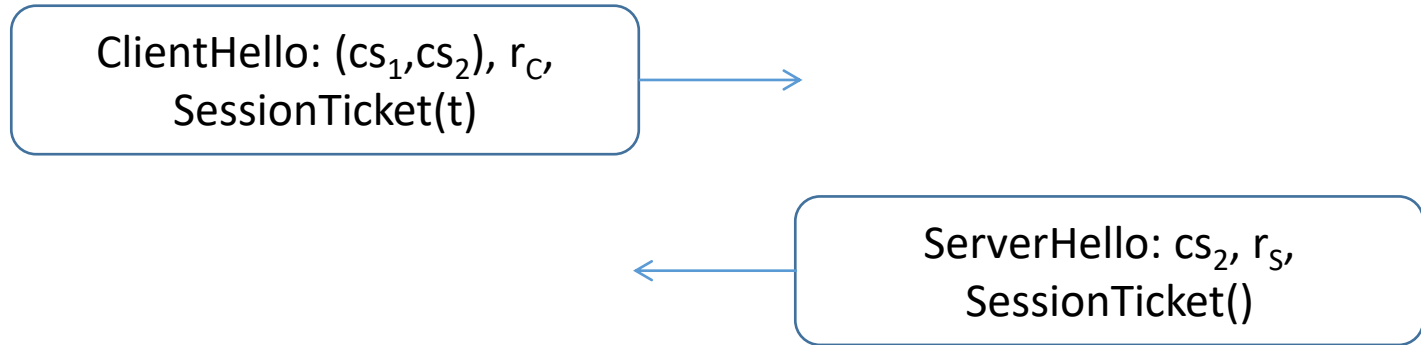
Server: 
bank.de

$ms := Dec_k(t)$
if OK agree on Session
Resumption

TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

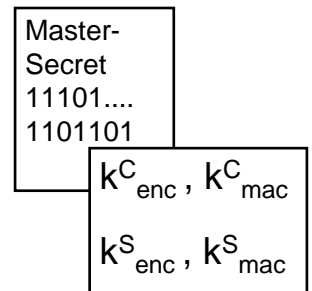
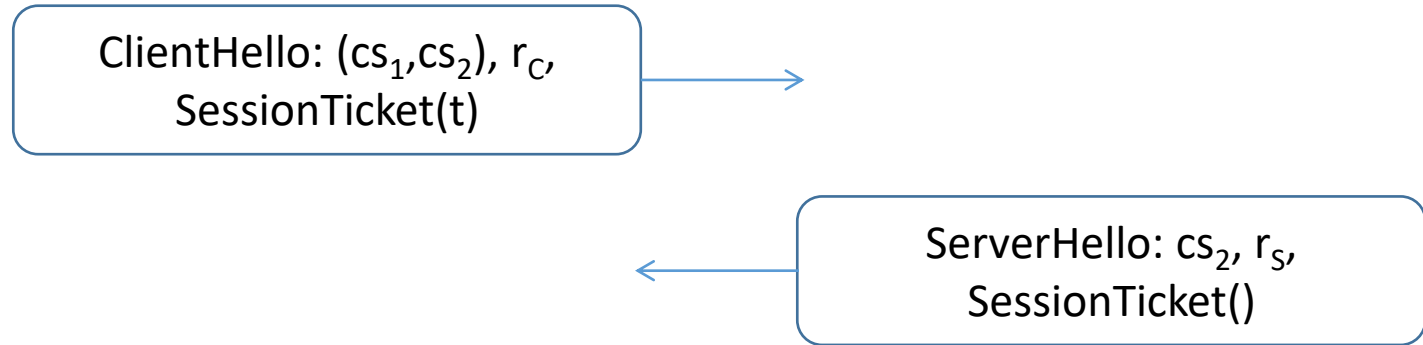
Server: bank.de k ms



TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

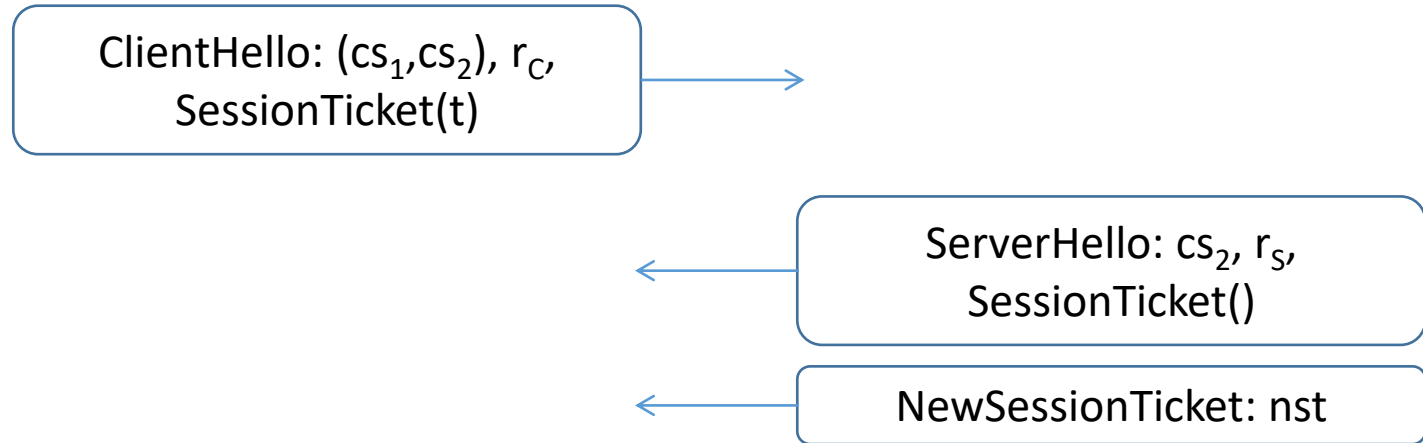
Server: bank.de k ms



TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

Server: bank.de k ms

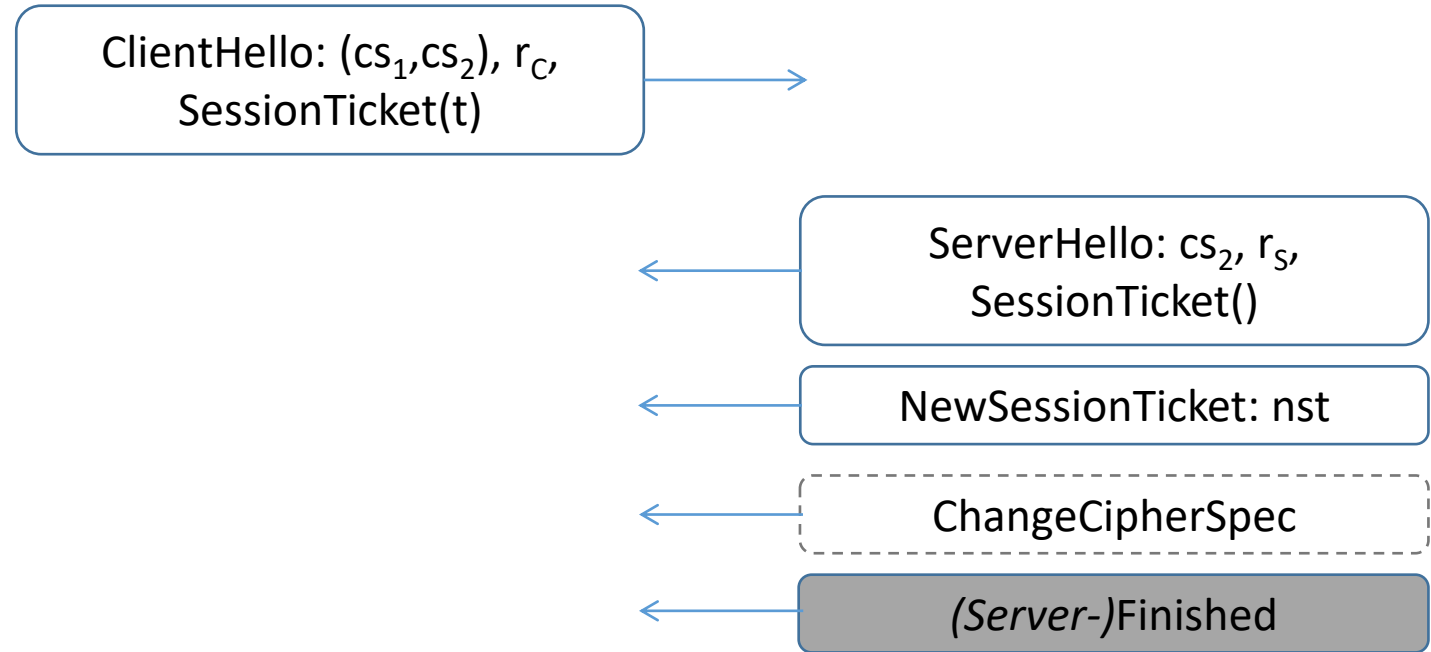


TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

Server: bank.de

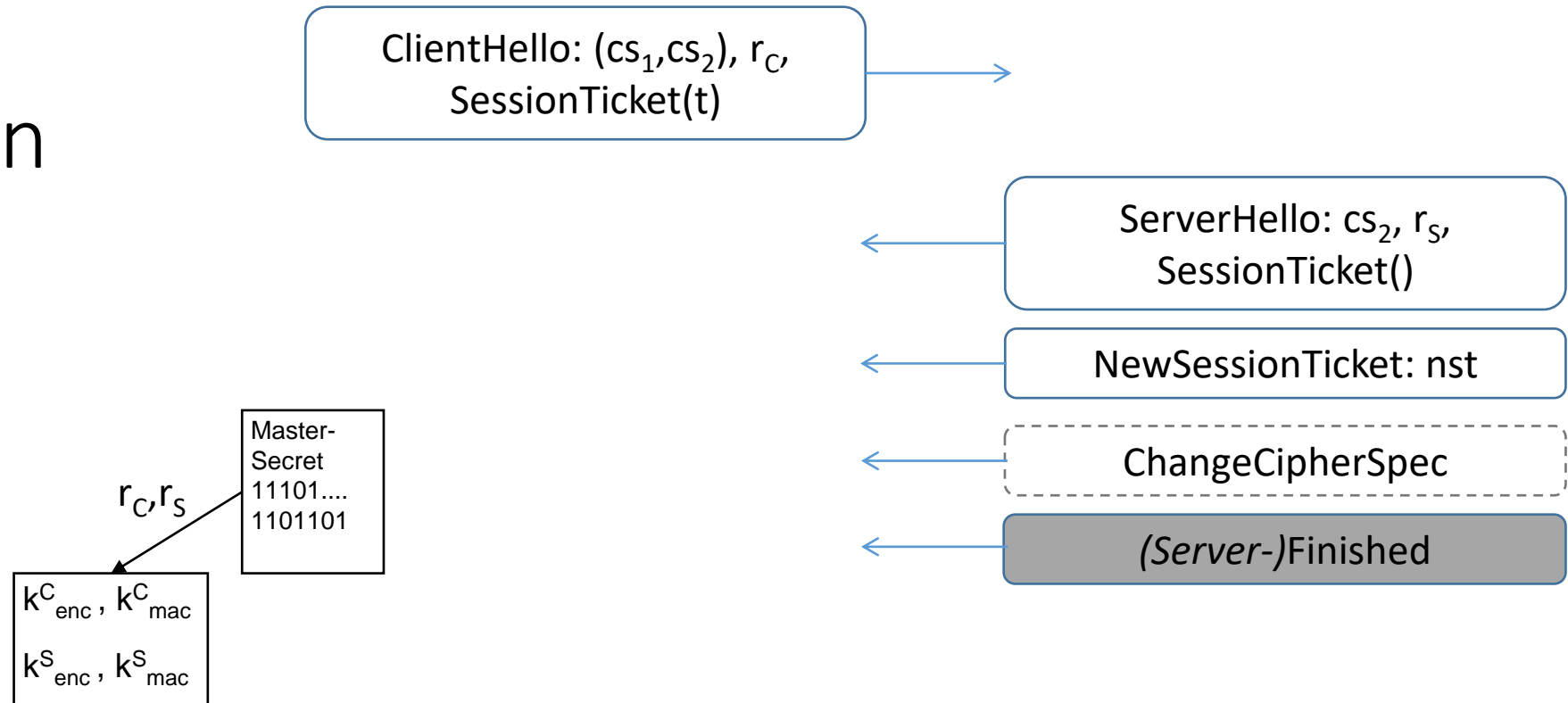
k	ms
---	----



TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

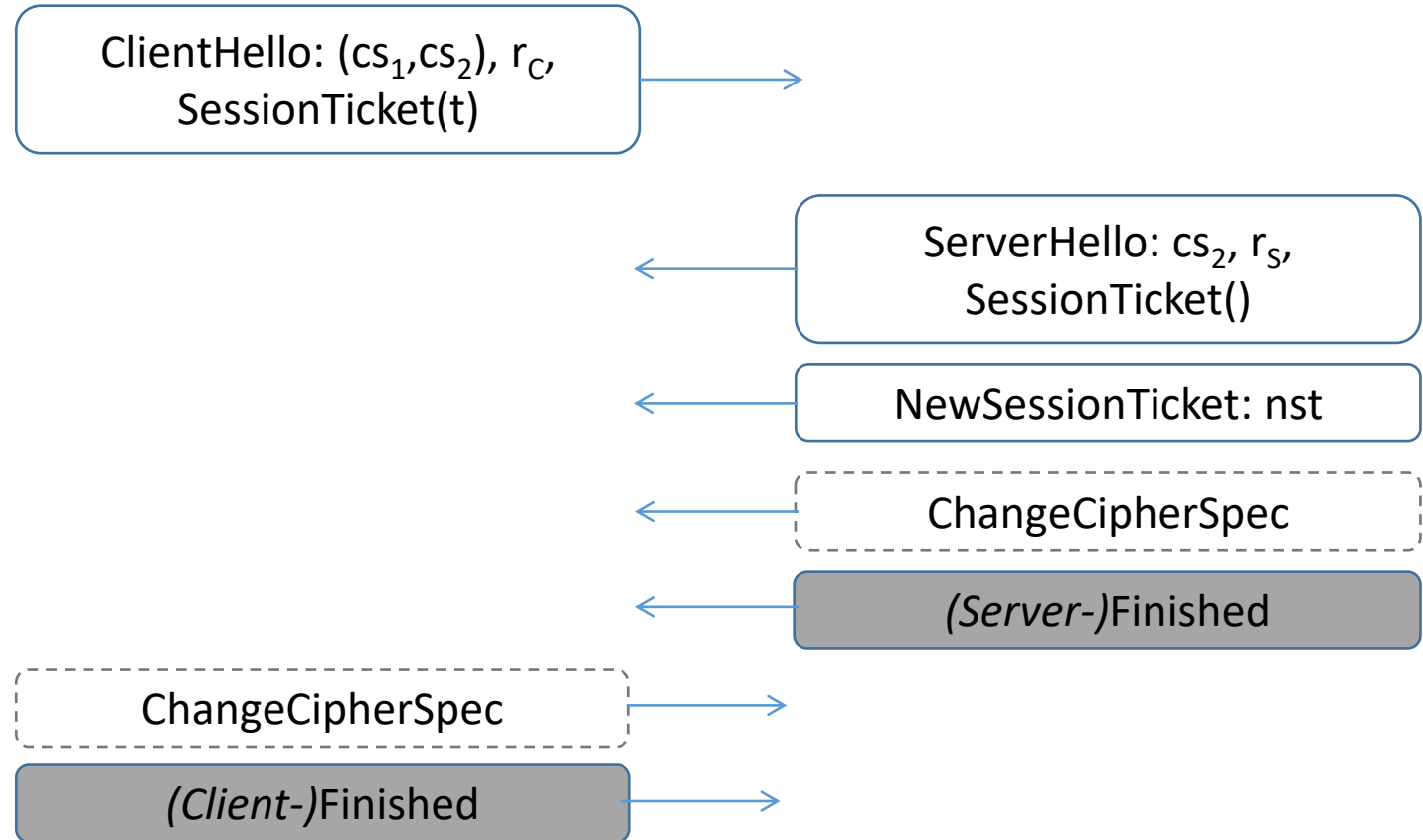
Server: bank.de k ms



TLS Handshake: Session Resumption with Session Ticket

Client:
(bank.de: ms, $t = E_k(ms)$)

Server: bank.de k ms



2.6 TLS Renegotiation

TLS Renegotiation

2. Handshake

TLS Renegotiation

2. Handshake

- Handshake-Nachrichten werden verschlüsselt mit den im 1. Handshake ausgehandelten Schlüsseln übertragen

TLS Renegotiation

2. Handshake

- Handshake-Nachrichten werden verschlüsselt mit den im 1. Handshake ausgehandelten Schlüsseln übertragen
- Mit ChangeCipherSpec wird der Record Layer auf die neu ausgehandelten Schlüssel umgeschaltet

TLS Renegotiation

2. Handshake

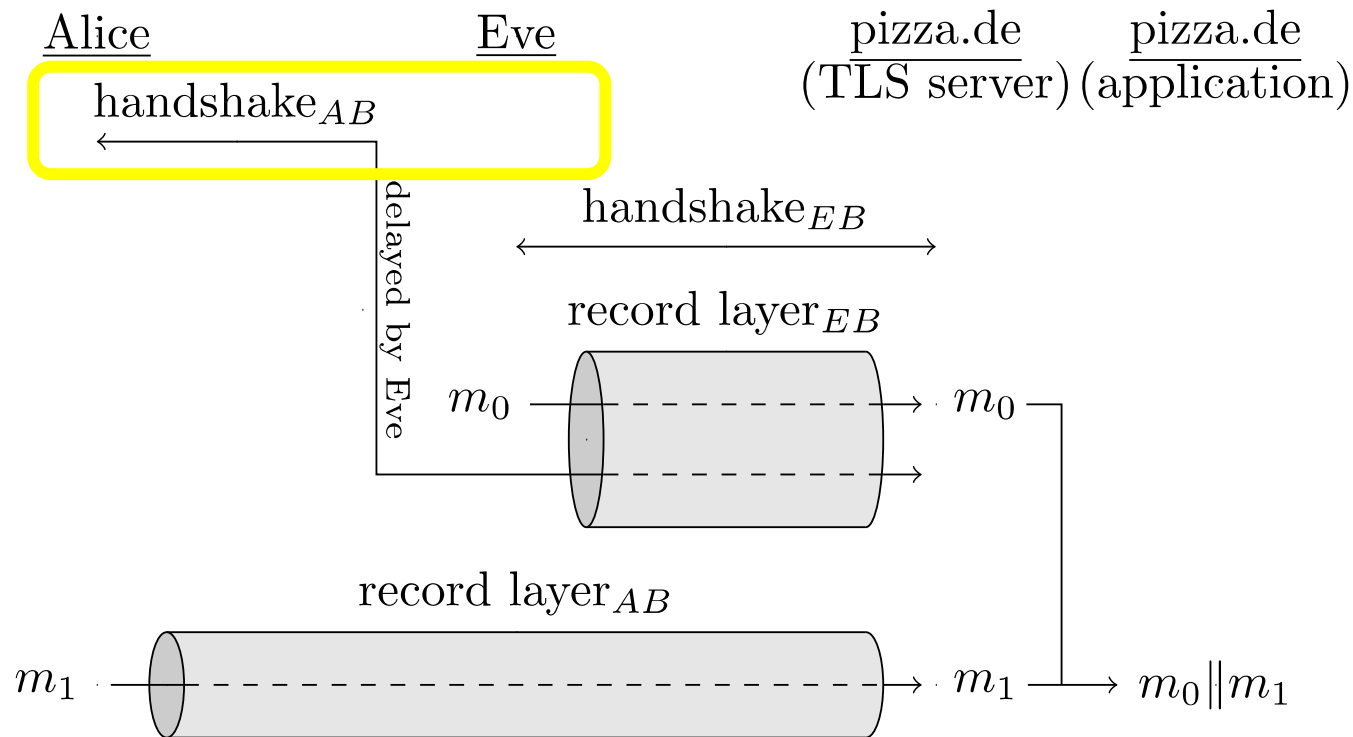
- Handshake-Nachrichten werden verschlüsselt mit den im 1. Handshake ausgehandelten Schlüsseln übertragen
- Mit ChangeCipherSpec wird der Record Layer auf die neu ausgehandelten Schlüssel umgeschaltet
- Typischer Einsatz:

TLS Renegotiation

2. Handshake

- Handshake-Nachrichten werden verschlüsselt mit den im 1. Handshake ausgehandelten Schlüsseln übertragen
- Mit ChangeCipherSpec wird der Record Layer auf die neu ausgehandelten Schlüssel umgeschaltet
- Typischer Einsatz:
 1. TLS-Handshake ohne Client-Authentifizierung
 2. TLS-Handshake mit Client-Authentifizierung

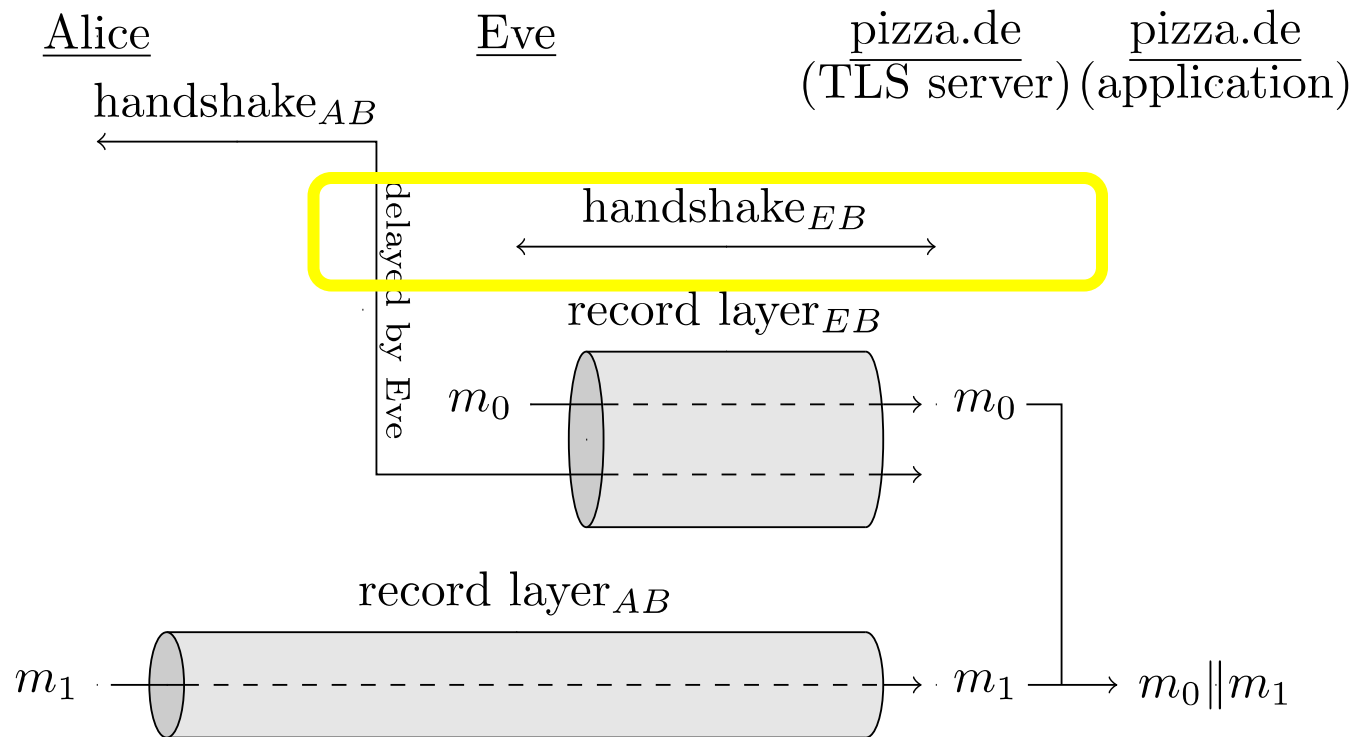
Angriff von Ray und Dispensa



Eve ist TCP-MitM

1. Eve beobachtet den Versuch von Alice, eine TLS-Verbindung zu pizza.de aufzubauen

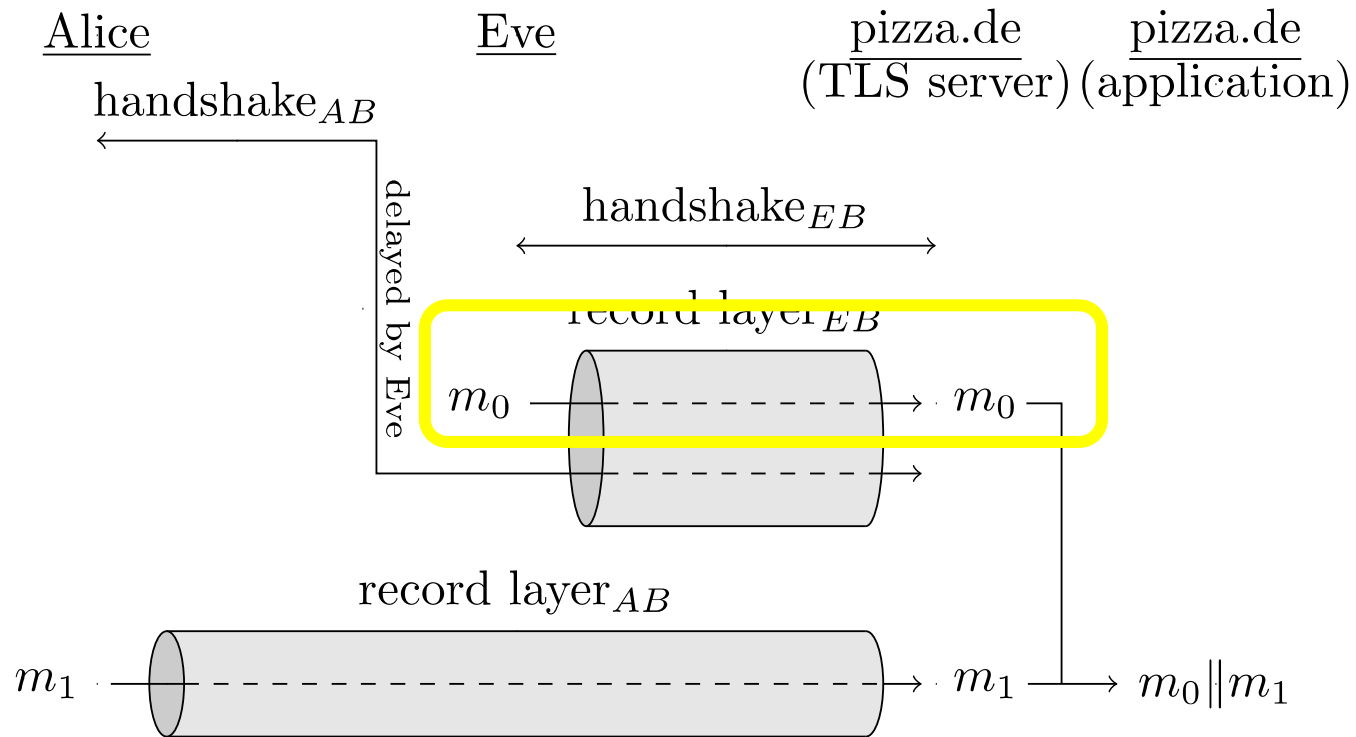
Angriff von Ray und Dispensa



Eve ist TCP-MitM

2. Eve hält die TCP-Pakete von Alice zurück und baut selbst eine Verbindung zu pizza.de auf

Angriff von Ray und Dispensa



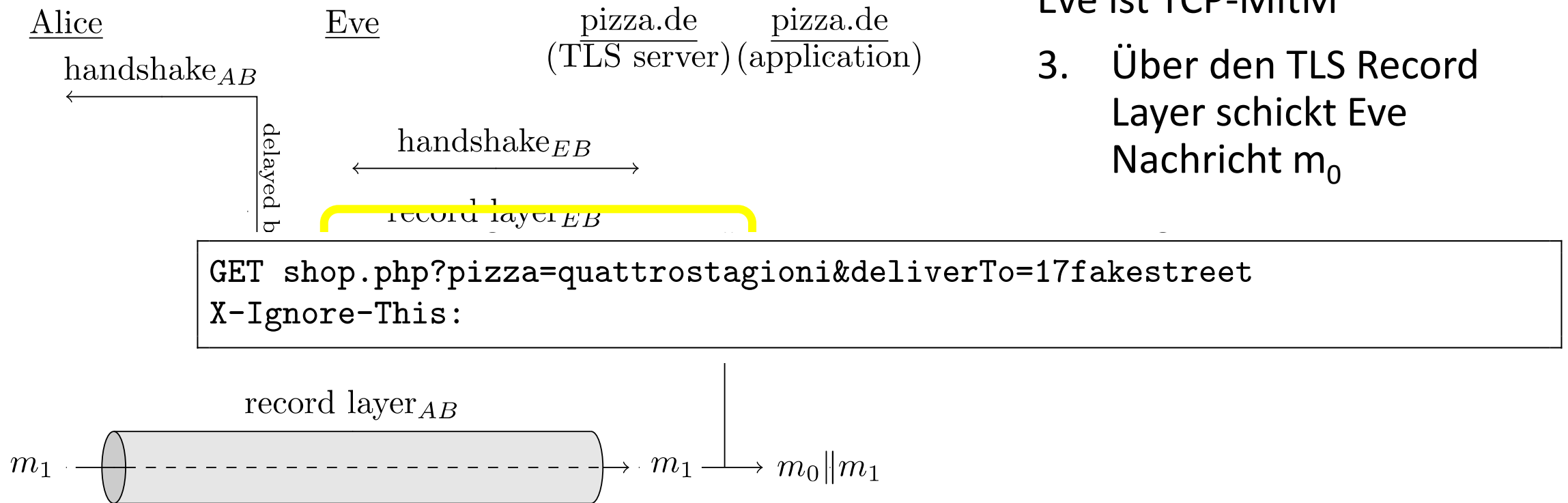
Eve ist TCP-MitM

3. Über den TLS Record Layer schickt Eve Nachricht m_0

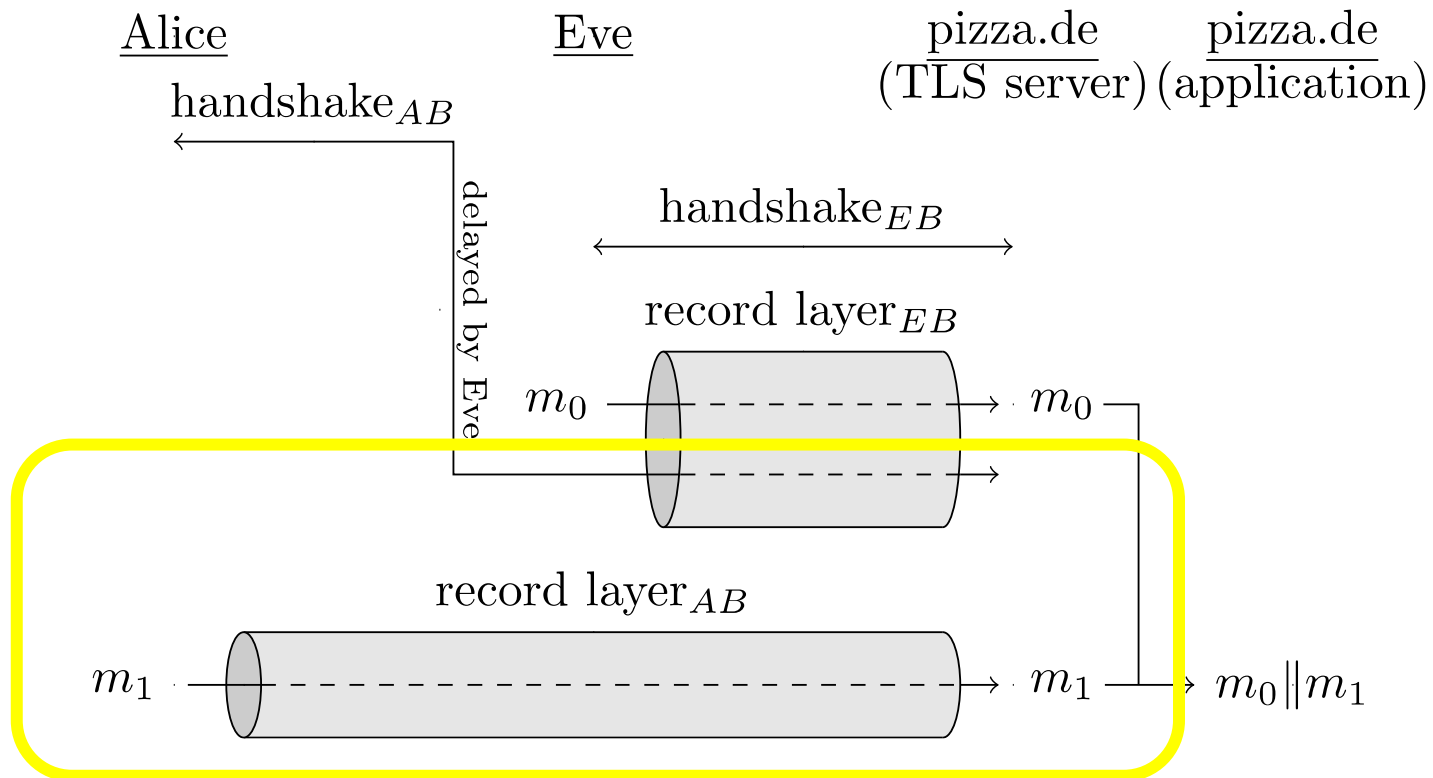
Angriff von Ray und Dispensa

Eve ist TCP-MitM

3. Über den TLS Record Layer schickt Eve Nachricht m_0



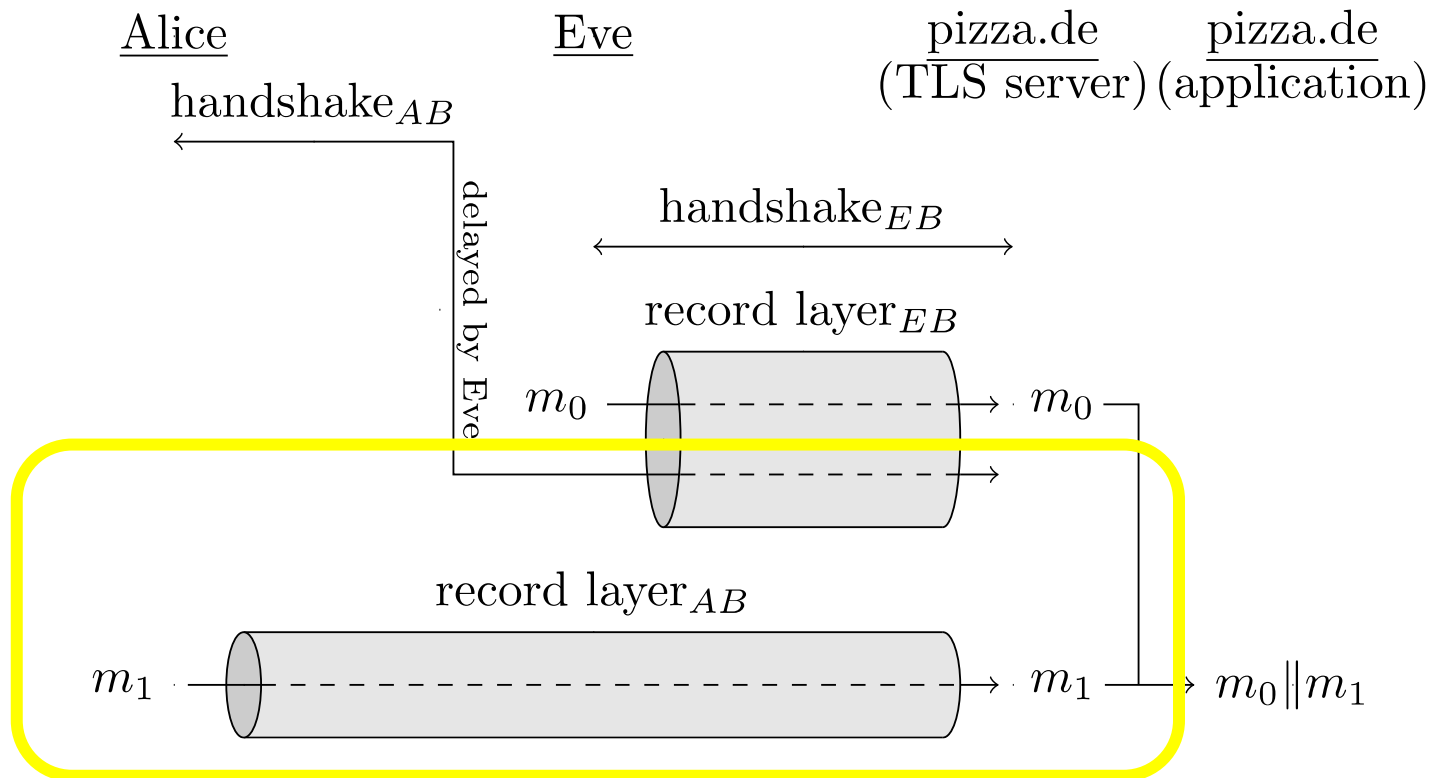
Angriff von Ray und Dispensa



Eve ist TCP-MitM

4. Dann leitet Eve alle TCP-Pakete zwischen Alice und pizza.de weiter

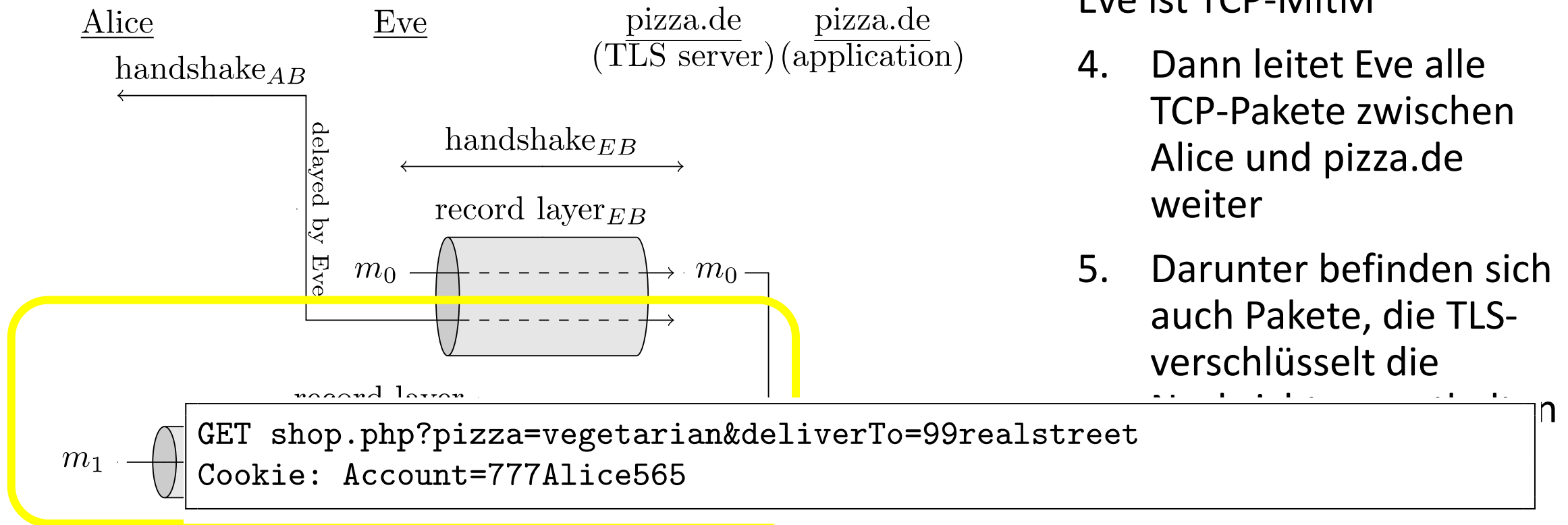
Angriff von Ray und Dispensa



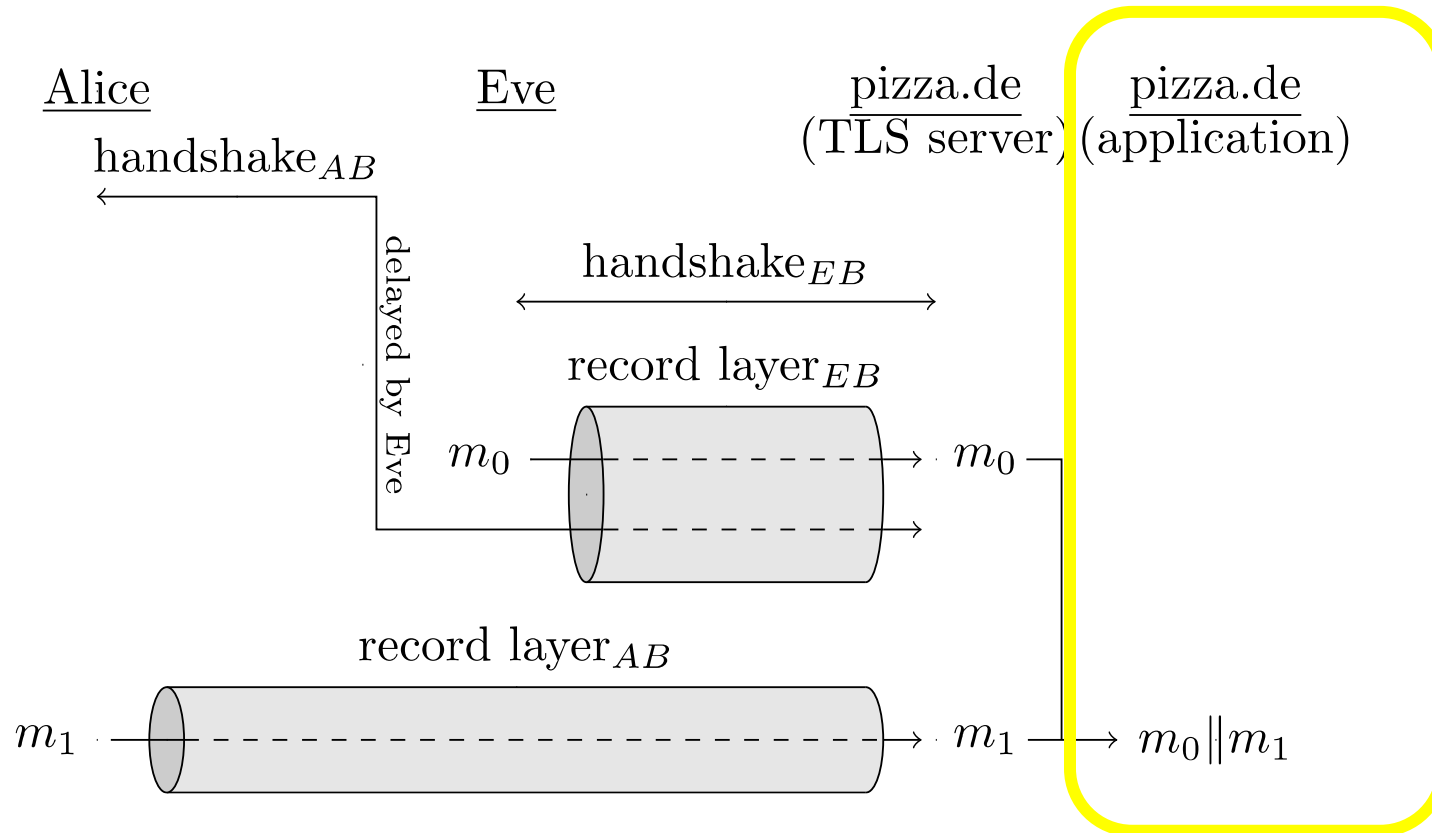
Eve ist TCP-MitM

4. Dann leitet Eve alle TCP-Pakete zwischen Alice und pizza.de weiter
5. Darunter befinden sich auch Pakete, die TLS-verschlüsselt die Nachricht m_1 enthalten

Angriff von Ray und Dispensa



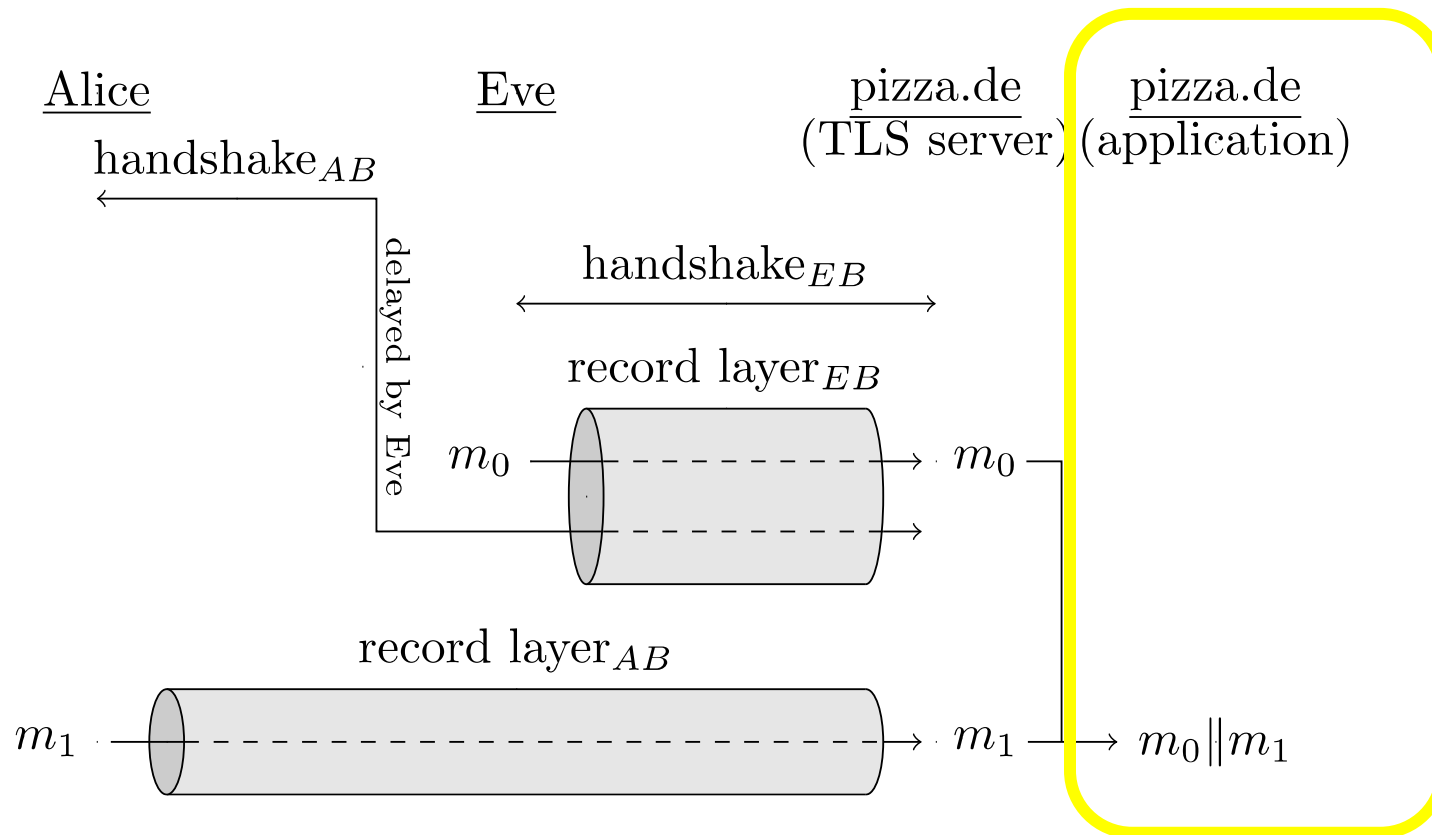
Angriff von Ray und Dispensa



Eve ist TCP-MitM

6. Der HTTP-Webserver sieht nicht, dass m_0 und m_1 über unterschiedliche TLS-Verbindungen geschickt wurden, da die TLS-Verschlüsselung für ihn transparent ist

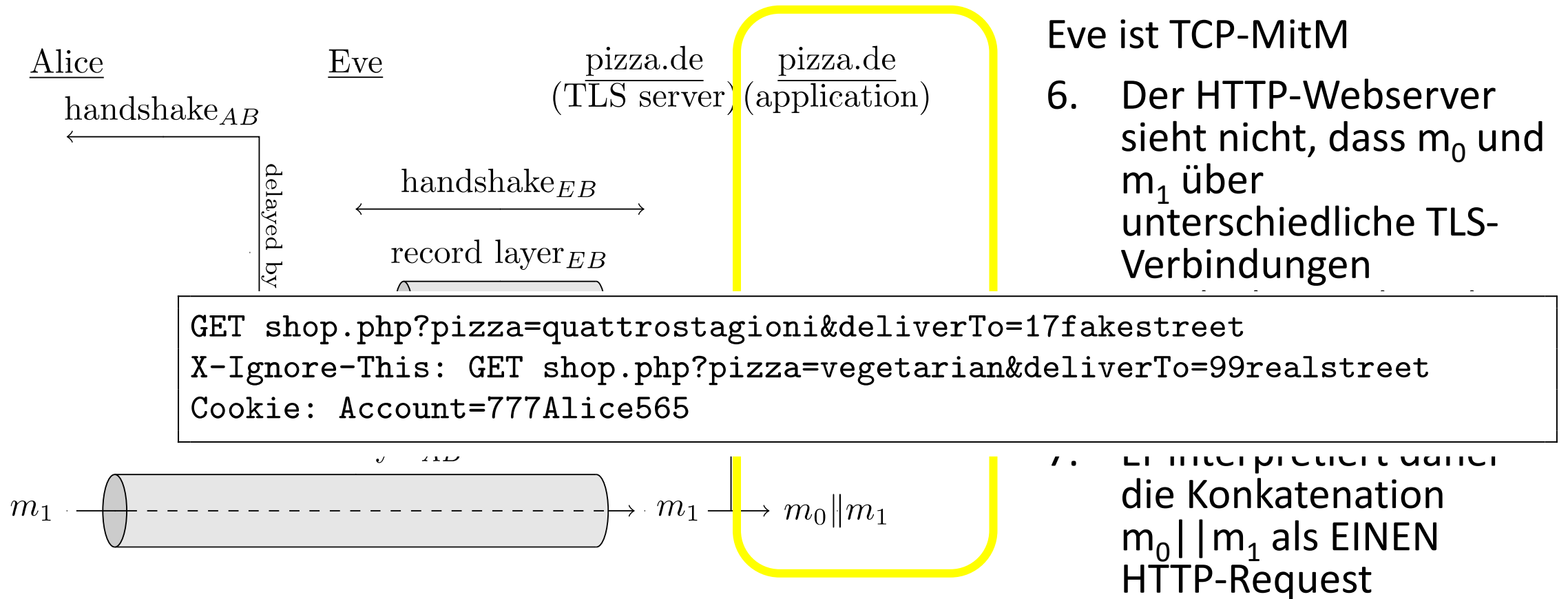
Angriff von Ray und Dispensa



Eve ist TCP-MitM

6. Der HTTP-Webserver sieht nicht, dass m_0 und m_1 über unterschiedliche TLS-Verbindungen geschickt wurden, da die TLS-Verschlüsselung für ihn transparent ist
7. Er interpretiert daher die Konkatenation $m_0 || m_1$ als EINEN HTTP-Request

Angriff von Ray und Dispensa



Angriff von Ray und Dispensa

```
GET shop.php?pizza=quattrostagioni&deliverTo=17fakestreet  
X-Ignore-This: GET shop.php?pizza=vegetarian&deliverTo=99realstreet  
Cookie: Account=777Alice565
```

Angriff von Ray und Dispensa

```
GET shop.php?pizza=quattrostagioni&deliverTo=17fakestreet  
X-Ignore-This: GET shop.php?pizza=vegetarian&deliverTo=99realstreet  
Cookie: Account=777Alice565
```

Eve ist TCP-MitM

8. Die von Eve bestellte Pizza wird somit an Eve's Adresse geliefert ...

Angriff von Ray und Dispensa

```
GET shop.php?pizza=quattrostagioni&deliverTo=17fakestreet  
X-Ignore-This: GET shop.php?pizza=vegetarian&deliverTo=99realstreet  
Cookie: Account=777Alice565
```

Eve ist TCP-MitM

8. Die von Eve bestellte Pizza wird somit an Eve's Adresse geliefert ...
9. ... die Rechnung geht aber an Alice, die sich mit ihrem Session Cookie
Account=777Alice565 authentifiziert hat

Angriff von Ray und Dispensa

Gründe für den Erfolg des Angriffs

Angriff von Ray und Dispensa

Gründe für den Erfolg des Angriffs

- TLS ist für Anwendungsprotokolle transparent, d.h. 'unsichtbar'

Angriff von Ray und Dispensa

Gründe für den Erfolg des Angriffs

- TLS ist für Anwendungsprotokolle transparent, d.h. 'unsichtbar'
- Der TLS-Client kann nicht erkennen, ob er einen 1. oder einen 2. Handshake mit dem Server durchführt

Angriff von Ray und Dispensa

Schutzmaßnahmen

- Server: TLS Renegotiation in der Server-Konfiguration verbietet

Angriff von Ray und Dispensa

Schutzmaßnahmen

- Server: TLS Renegotiation in der Server-Konfiguration verbietet
- Client oder Server: Sender der TLS-Extension *RenegotiationInfo* (0xFF01)

Angriff von Ray und Dispensa

Schutzmaßnahmen

- Server: TLS Renegotiation in der Server-Konfiguration verbietet
- Client oder Server: Sender der TLS-Extension *RenegotiationInfo* (0xFF01)

```
struct {  
    opaque renegotiated_connection<0..255>;  
} RenegotiationInfo;
```

Angriff von Ray und Dispensa

Schutzmaßnahmen

- Server: TLS Renegotiation in der Server-Konfiguration verbietet
- Client oder Server: Sender der TLS-Extension *RenegotiationInfo* (0xFF01)
 - 1. Handshake:
 - 2. Handshake:

Angriff von Ray und Dispensa

Schutzmaßnahmen

- Server: TLS Renegotiation in der Server-Konfiguration verbietet
- Client oder Server: Sender der TLS-Extension *RenegotiationInfo* (0xFF01)
 - 1. Handshake:
 - ClientHello enthält diese Extension mit leerem Inhalt der Länge 0
 - ServerHello enthält diese Extension mit leerem Inhalt der Länge 0
 - 2. Handshake:

Angriff von Ray und Dispensa

Schutzmaßnahmen

- Server: TLS Renegotiation in der Server-Konfiguration verbietet
- Client oder Server: Sender der TLS-Extension *RenegotiationInfo* (0xFF01)
 - 1. Handshake:
 - ClientHello enthält diese Extension mit leerem Inhalt der Länge 0
 - ServerHello enthält diese Extension mit leerem Inhalt der Länge 0
 - 2. Handshake:
 - ClientHello enthält diese Extension mit client_verify_data (MAC aus dem 1. Handshake)
 - ServerHello enthält diese Extension mit server_verify_data (MAC aus dem 1. Handshake)

2.7 TLS Extensions

TLS Extensions

- erweitern oder modifizieren das Verhalten von TLS

TLS Extensions

- erweitern oder modifizieren das Verhalten von TLS
- können in ClientHello oder ServerHello enthalten sein

TLS Extensions

- erweitern oder modifizieren das Verhalten von TLS
- können in ClientHello oder ServerHello enthalten sein
- Übersicht: <https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml>

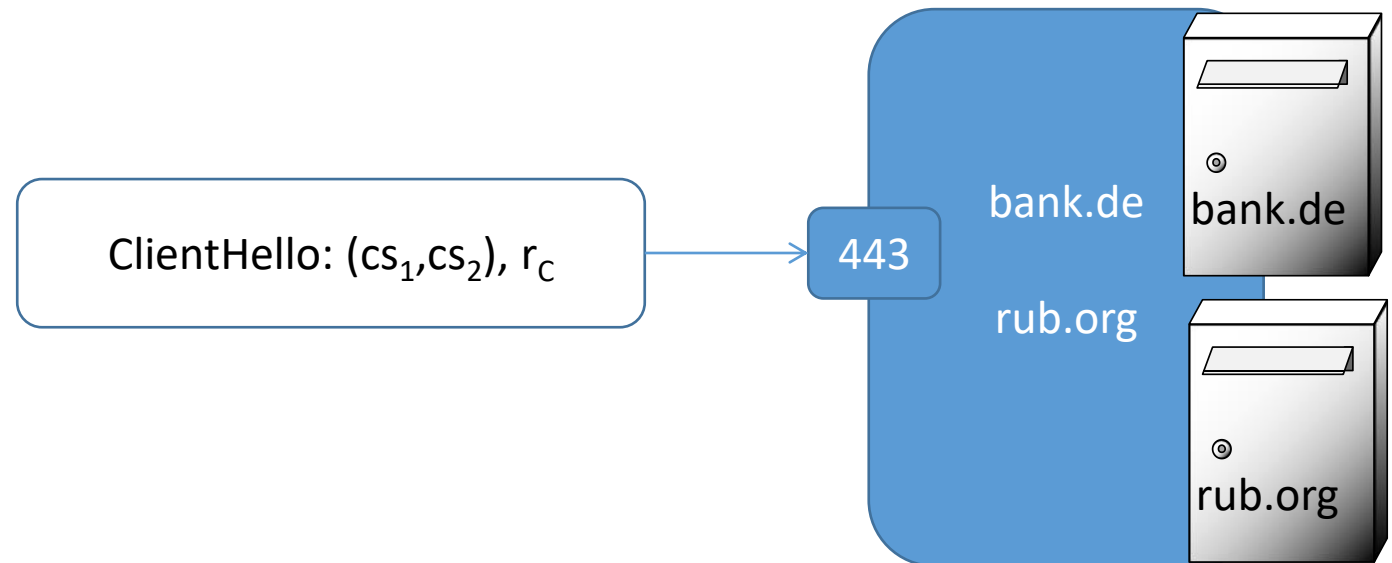
TLS Extensions: SNI

Server Name Indication (SNI)

TLS Extensions: SNI

Server Name Indication (SNI)

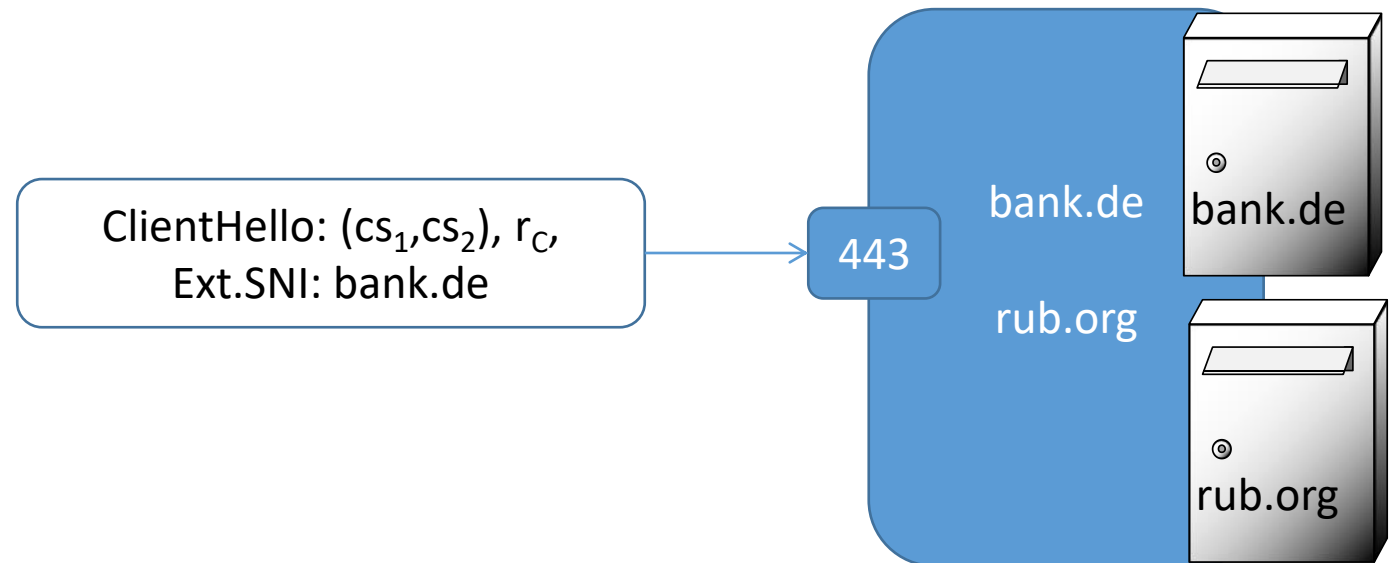
- Werden auf einer IP-Adresse mehrere HTTPS-Webserver gehostet, so weiß der TLS-Server nicht, welches Serverzertifikat er im Handshake verwenden soll



TLS Extensions: SNI

Server Name Indication (SNI)

- Werden auf einer IP-Adresse mehrere HTTPS-Webserver gehostet, so weiß der TLS-Server nicht, welches Serverzertifikat er im Handshake verwenden soll
- Daher wird in der SNI-Extension der Domainname des HTTPS-Webserver mitgesendet



TLS Extensions: Erweiterung Ciphersuites

TLS Extensions: Erweiterung Ciphersuites

Supported Groups

- Die von Client oder Server unterstützten DHKE-Gruppen.

TLS Extensions: Erweiterung Ciphersuites

Supported Groups

- Die von Client oder Server unterstützten DHKE-Gruppen.

Signature Algorithms

- Die von Client oder Server unterstützten Signaturalgorithmen
- diese sind nicht eindeutig aus der Liste der Ciphersuites ermittelbar

TLS Extensions: ALPN

Aktivierung von TLS

- TCP-Verbindungsaufbau auf Well-Known Port (z.B. 443 für HTTPS)

TLS Extensions: ALPN

Aktivierung von TLS

- TCP-Verbindungsaufbau auf Well-Known Port (z.B. 443 für HTTPS)
- Client empfängt STARTTLS-Befehl auf ungeschützter TCP-Verbindung

TLS Extensions: ALPN

Aktivierung von TLS

- TCP-Verbindungsaufbau auf Well-Known Port (z.B. 443 für HTTPS)
- Client empfängt STARTTLS-Befehl auf ungeschützter TCP-Verbindung
- Client baut immer TLS-Verbindung auf und handelt Anwendungsprotokoll aus (ALPN)

TLS Extensions: ALPN

Application Layer Protocol Negotiation (ALPN)

- Nach Aufbau der TCP-Verbindung startet der Client sofort den TLS-Handshake

TLS Extensions: ALPN

Application Layer Protocol Negotiation (ALPN)

- Nach Aufbau der TCP-Verbindung startet der Client sofort den TLS-Handshake
- Er teilt dem Server in der ALPN-Extension mit, welches Anwendungsprotokoll er über TLS sprechen möchte

TLS Extensions: Sicherheit

Encrypt-then-Mac

- Nutze andere Verschlüsselung im TLS-1.2-Record-Layer

TLS Extensions: Sicherheit

Encrypt-then-Mac

- Nutze andere Verschlüsselung im TLS-1.2-Record-Layer

Extended Master Secret:

- Berechne das MasterSecret anders

TLS Extensions

Session Ticket (s.o.)

TLS Extensions

Session Ticket (s.o.)

Secure Renegotiation (s.o.)

TLS Extensions

Session Ticket (s.o.)

Secure Renegotiation (s.o.)

Heartbeat

- Ist der DTLS-Server noch online?
- Basis für den Heartbleed-Angriff

2.8 HTTP-Header mit Auswirkungen auf TLS

HTTP Header: HSTS

HTTP Strict Transport Security (HSTS)

- Wenn ein Client diesen HTTP-Header vom Server empfängt, darf er in Zukunft nur noch HTTPS verwenden

HTTP Header: HSTS

HTTP Strict Transport Security (HSTS)

- Wenn ein Client diesen HTTP-Header vom Server empfängt, darf er in Zukunft nur noch HTTPS verwenden

Beispiel: `Strict-Transport-Security: max-age=31536000`

HTTP Header: HSTS

HTTP Strict Transport Security (HSTS)

- Wenn ein Client diesen HTTP-Header vom Server empfängt, darf er in Zukunft nur noch HTTPS verwenden

Beispiel: `Strict-Transport-Security: max-age=31536000`

- Weist den Browser an, nur noch HTTPS zu verwenden und auch alle Hyperlinks innerhalb eines anderen HTML-Dokuments in HTTPS-Hyperlinks umzuwandeln

HTTP Header: HSTS

HTTP Strict Transport Security (HSTS)

- Wenn ein Client diesen HTTP-Header vom Server empfängt, darf er in Zukunft nur noch HTTPS verwenden

Beispiel: `Strict-Transport-Security: max-age=31536000`

- Weist den Browser an, nur noch HTTPS zu verwenden und auch alle Hyperlinks innerhalb eines anderen HTML-Dokuments in HTTPS-Hyperlinks umzuwandeln
- Policy gilt ein Jahr lang

HTTP Header: HSTS

HTTP Strict Transport Security (HSTS)

- Wenn ein Client diesen HTTP-Header vom Server empfängt, darf er in Zukunft nur noch HTTPS verwenden

Beispiel: `Strict-Transport-Security: max-age=31536000`

- Weist den Browser an, nur noch HTTPS zu verwenden und auch alle Hyperlinks innerhalb eines anderen HTML-Dokuments in HTTPS-Hyperlinks umzuwandeln
- Policy gilt ein Jahr lang
- Bei Zertifikatsfehler für ein Jahr kein Verbindungsaufbau mehr möglich

HTTP Header: HPKP (RFC 7469)

HTTP Public Key Pinning (HPKP)

HTTP Header: HPKP (RFC 7469)

HTTP Public Key Pinning (HPKP)

- Google konnte Angriffe auf Zertifizierungsstellen wie DigiNotar erkennen, da Google Chrome die Public Keys der Google-Server fest im Sourcecode gespeichert hatte ('Pinning')

HTTP Header: HPKP (RFC 7469)

HTTP Public Key Pinning (HPKP)

- Google konnte Angriffe auf Zertifizierungsstellen wie DigiNotar erkennen, da Google Chrome die Public Keys der Google-Server fest im Sourcecode gespeichert hatte ('Pinning')
- Idee: Mit HPKP soll das jede Webanwendung können

HTTP Header: HPKP

HTTP Public Key Pinning (HPKP)

```
Public-Key-Pins: max-age=2592000;  
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";  
pin-sha256="LPJNul+wow4m6DsqxbninhSWhlwfp0JecwQzYpOLmCQ=";  
report-uri="http://example.com/pkp-report"; includeSubDomains
```

- Dauer des Pinning

HTTP Header: HPKP

HTTP Public Key Pinning (HPKP)

```
Public-Key-Pins: max-age=2592000;
```

```
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";
```

```
pin-sha256="LPJNul+wow4m6DsqxbninhSWhlwfp0JecwQzYpOLmCQ=";
```

```
report-uri="http://example.com/pkp-report"; includeSubDomains
```

- SHA-256-Hahswert des Public Key des aktuellen TLS-Zertifikats

HTTP Header: HPKP

HTTP Public Key Pinning (HPKP)

```
Public-Key-Pins: max-age=2592000;  
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";  
pin-sha256="LPJNul+wow4m6DsqxnbnihsWHlwfp0JecwQzYpOLmCQ=";  
report-uri="http://example.com/pkp-report"; includeSubDomains
```

- SHA-256-Hahswert des Public Key des Reserve-TLS-Zertifikats

HTTP Header: HPKP

HTTP Public Key Pinning (HPKP)

```
Public-Key-Pins: max-age=2592000;  
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g=";  
pin-sha256="LPJNul+wow4m6DsqxbninhSWhlwfp0JecwQzYpOLmCQ=";  
report-uri="http://example.com/pkp-report"; includeSubDomains
```

- URI, an die ein Bericht gesandt werden soll wenn etwas schiefgeht
- auch für alle Subdomains sollen diese beiden Schlüssel gepinnt werden

2.9 Datagram TLS (DTLS)

DTLS

Datagram TLS = TLS über UDP (statt TCP)

DTLS

Datagram TLS = TLS über UDP (statt TCP)

Warum funktioniert TLS über UDP nicht?

DTLS

Datagram TLS = TLS über UDP (statt TCP)

Warum funktioniert TLS über UDP nicht?

- UDP garantiert nicht die zuverlässige Übertragung von Handshake-Nachrichten:
Handshake funktioniert nicht

DTLS

Datagram TLS = TLS über UDP (statt TCP)

Warum funktioniert TLS über UDP nicht?

- UDP garantiert nicht die zuverlässige Übertragung von Handshake-Nachrichten: Handshake funktioniert nicht
- DoS: Absender der ClientHello-Nachricht kann IP Spoofing verwenden

DTLS

Datagram TLS = TLS über UDP (statt TCP)

Warum funktioniert TLS über UDP nicht?

- UDP garantiert nicht die zuverlässige Übertragung von Handshake-Nachrichten: Handshake funktioniert nicht
- DoS: Absender der ClientHello-Nachricht kann IP Spoofing verwenden
- Record Layer: Implizite Sequenznummern funktionieren nicht mehr, wenn UDP-Pakete verloren gehen oder die Reihenfolge vertauscht wird

DTLS

Datagram TLS = TLS über UDP (statt TCP)

Warum funktioniert TLS über UDP nicht?

- UDP garantiert nicht die zuverlässige Übertragung von Handshake-Nachrichten: Handshake funktioniert nicht
- DoS: Absender der ClientHello-Nachricht kann IP Spoofing verwenden
- Record Layer: Implizite Sequenznummern funktionieren nicht mehr, wenn UDP-Pakete verloren gehen oder die Reihenfolge vertauscht wird
- Stromchiffren: Entschlüsselung funktioniert nur, wenn Pakete in korrekter Reihenfolge eintreffen

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Record Layer:

- Übertragung expliziter 48-Bit-Sequenznummern in jedem Record-Header

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Record Layer:

- Übertragung expliziter 48-Bit-Sequenznummern in jedem Record-Header
- Stromchiffren sind verboten

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Record Layer:

- Übertragung expliziter 48-Bit-Sequenznummern in jedem Record-Header
- Stromchiffren sind verboten
- Epoche: Ein 16-Bit-Wert gibt an, aus welcher 'Epoche' die zu verwendenden Schlüssel sind

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Record Layer:

- Übertragung expliziter 48-Bit-Sequenznummern in jedem Record-Header
- Stromchiffren sind verboten
- Epoche: Ein 16-Bit-Wert gibt an, aus welcher 'Epoche' die zu verwendenden Schlüssel sind
 - epoch=0: Unverschlüsselt
 - epoch=1: Schlüssel aus 1. Handshake
 - epoch=2: Schlüssel aus 2. Handshake

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

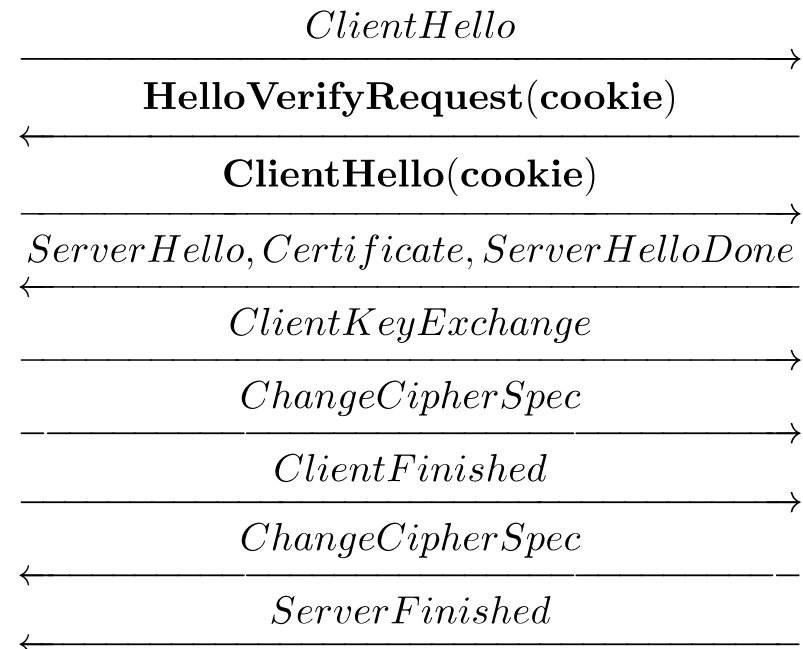
DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Handshake:

- Anti-DoS-Cookie

Client C

Server S
($sk_S, pk_S, cert_S$)



DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Handshake:

- Anti-DoS-Cookie
- Verlust/Fehlerhafte Übertragung von Handshake-Nachrichten: Nach Ablauf eines Timeouts kann jede Partei durch Senden der letzten empfangenen Handshake-Nachricht eine erneute Übertragung der darauf folgenden Nachricht anfordern

DTLS: Anpassungen

DTLS 1.0 (RFC 4347): Anpassung von TLS 1.1 an UDP

DTLS 1.2 (RFC 6347): Anpassung von TLS 1.2 an UDP

Handshake:

- Anti-DoS-Cookie
- Verlust/Fehlerhafte Übertragung von Handshake-Nachrichten: Nach Ablauf eines Timeouts kann jede Partei durch Senden der letzten empfangenen Handshake-Nachricht eine erneute Übertragung der darauf folgenden Nachricht anfordern
- 16-Bit-Sequenznummern und Fragment-Identifizierer, um Handshake-Nachrichten korrekt zusammensetzen zu können

DTLS: Einsatz

Zur Übertragung von Audio/Video-Daten, und zum Tunneln von IP-Paketen in VPNs, ist UDP besser geeignet als TCP

Einsatz:

DTLS: Einsatz

Zur Übertragung von Audio/Video-Daten, und zum Tunneln von IP-Paketen in VPNs, ist UDP besser geeignet als TCP

Einsatz:

- In TLS-VPNs von Cisco, F5, Citrix

DTLS: Einsatz

Zur Übertragung von Audio/Video-Daten, und zum Tunneln von IP-Paketen in VPNs, ist UDP besser geeignet als TCP

Einsatz:

- In TLS-VPNs von Cisco, F5, Citrix
- Opera, Chrome, FF: DTLS-SRTP für WebRTC