Exercise Session IV: TLS II – Attacks and 0-RTT

Network Security

Matteo Scarlata

15/10/2020

ETH Zurich

Context

TLS version $\leq$ 1.2

Haunted by numerous:

TLS version $\leq$ 1.2

Haunted by numerous:

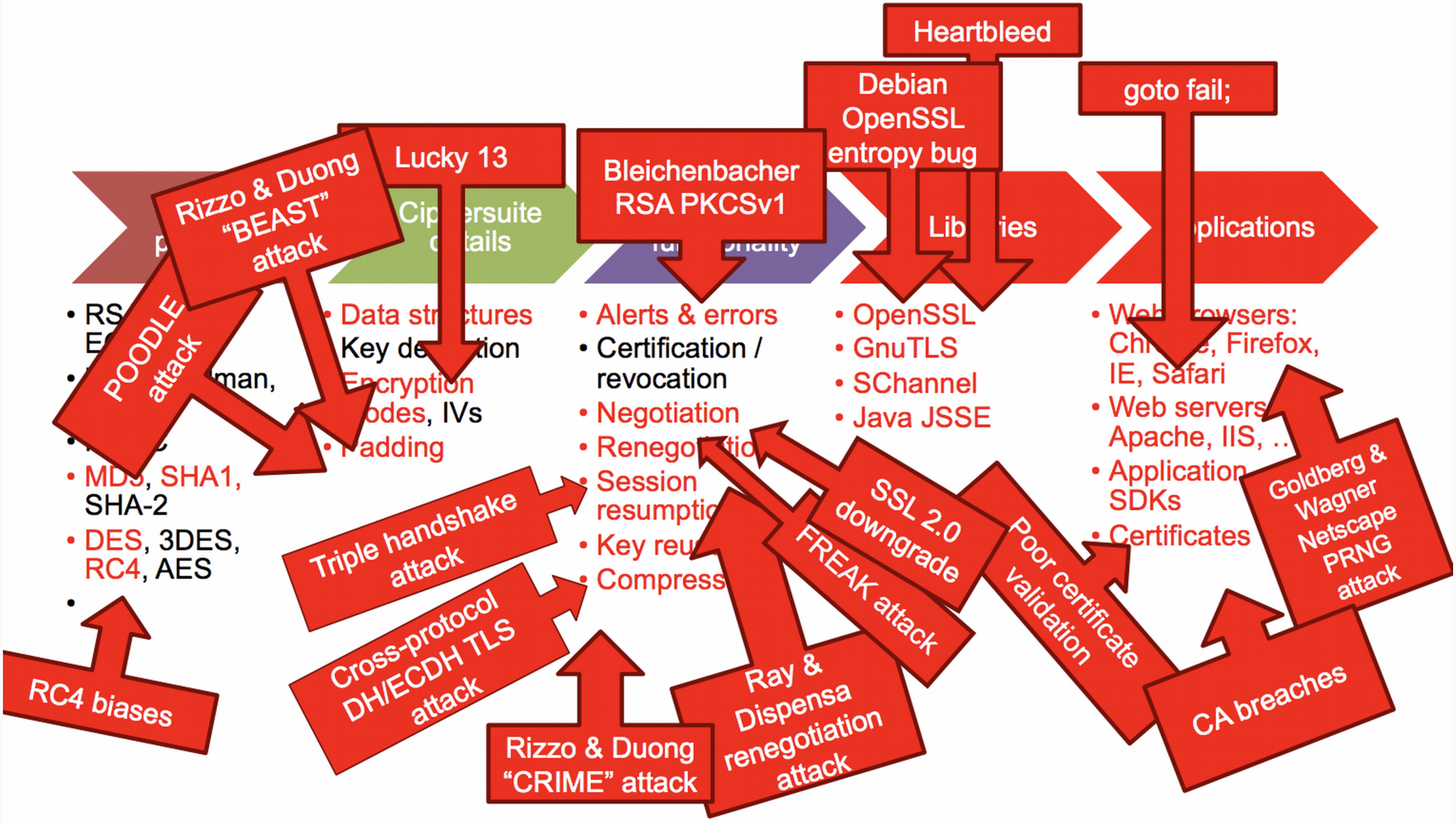
- Cryptographic attacks

TLS version $\leq$ 1.2

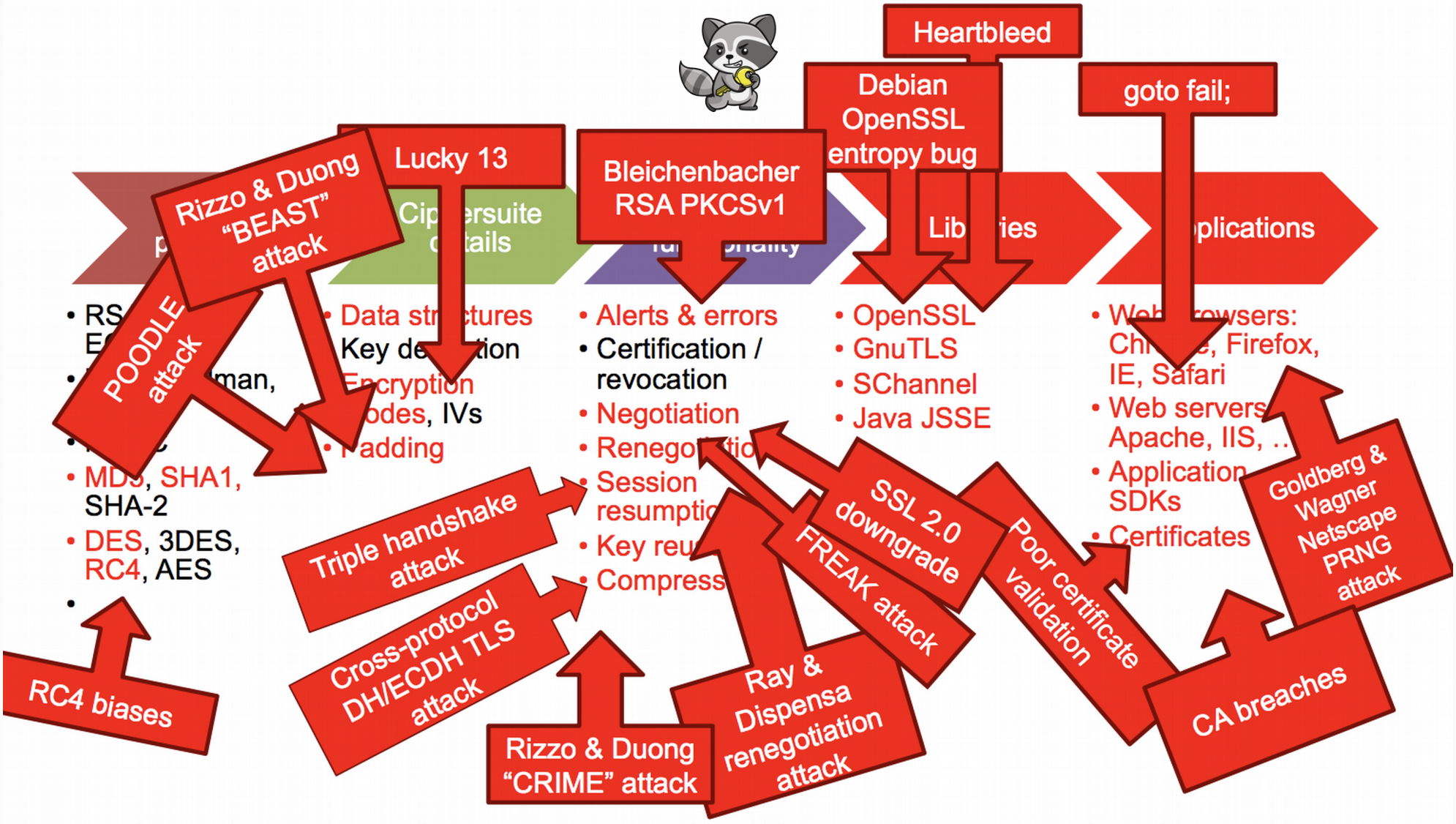
Haunted by numerous:

- Cryptographic attacks
- Implementation attacks

TLS version $\leq$ 1.2



TLS version $\leq$ 1.2



Douglas Stebila

TLS 1.3

TLS 1.3

https://unsplash.com/@benwhitephotography

TLS 1.3

https://unsplash.com/@benwhitephotography

TLS 1.3

[DFGS16] Dowling, B., Fischlin, M., Guenther, F., and D. Stebi "A Cryptographic Analysis of the TLS 1.3 Full and Pre-shared Key Handshake Protocol", TRON 2016, February 2016, <https://eprint.iacr.org/2016/081>.

[DOW92] Diffie, W., van Oorschot, P., and M. Wiener, "Authentication and authenticated key exchanges", Des Codes and Cryptography, DOI 10.1007/BF00124891, June

[DSS] National Institute of Standards and Technology, U.S. Department of Commerce, "Digital Signature Standard (DSS)", NIST FIPS PUB 186-4, DOI 10.6028/NIST.FIPS.18 July 2013.

[FG17] Fischlin, M. and F. Guenther, "Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates", Proceedings of EuroS&P 2017, April 2017, <https://eprint.iacr.org/2017/082>.

[FGSW16] Fischlin, M., Guenther, F., Schmidt, B., and B. Warin "Key Confirmation in Key Exchange: A Formal Treatment Implications for TLS 1.3", Proceedings of IEEE Sympos on Security and Privacy (San Jose), DOI 10.1109/SP.2016.34, May 2016, <https://ieeexplore.ieee.org/document/7546517/>.

[FW15] Weimer, F., "Factoring RSA Keys With TLS Perfect Forw Secrecy", September 2015.

Exercises

TLS 1.3: Resumption

TLS 1.3 provides three handshake modes:

- Full Handshake

TLS 1.3: Resumption

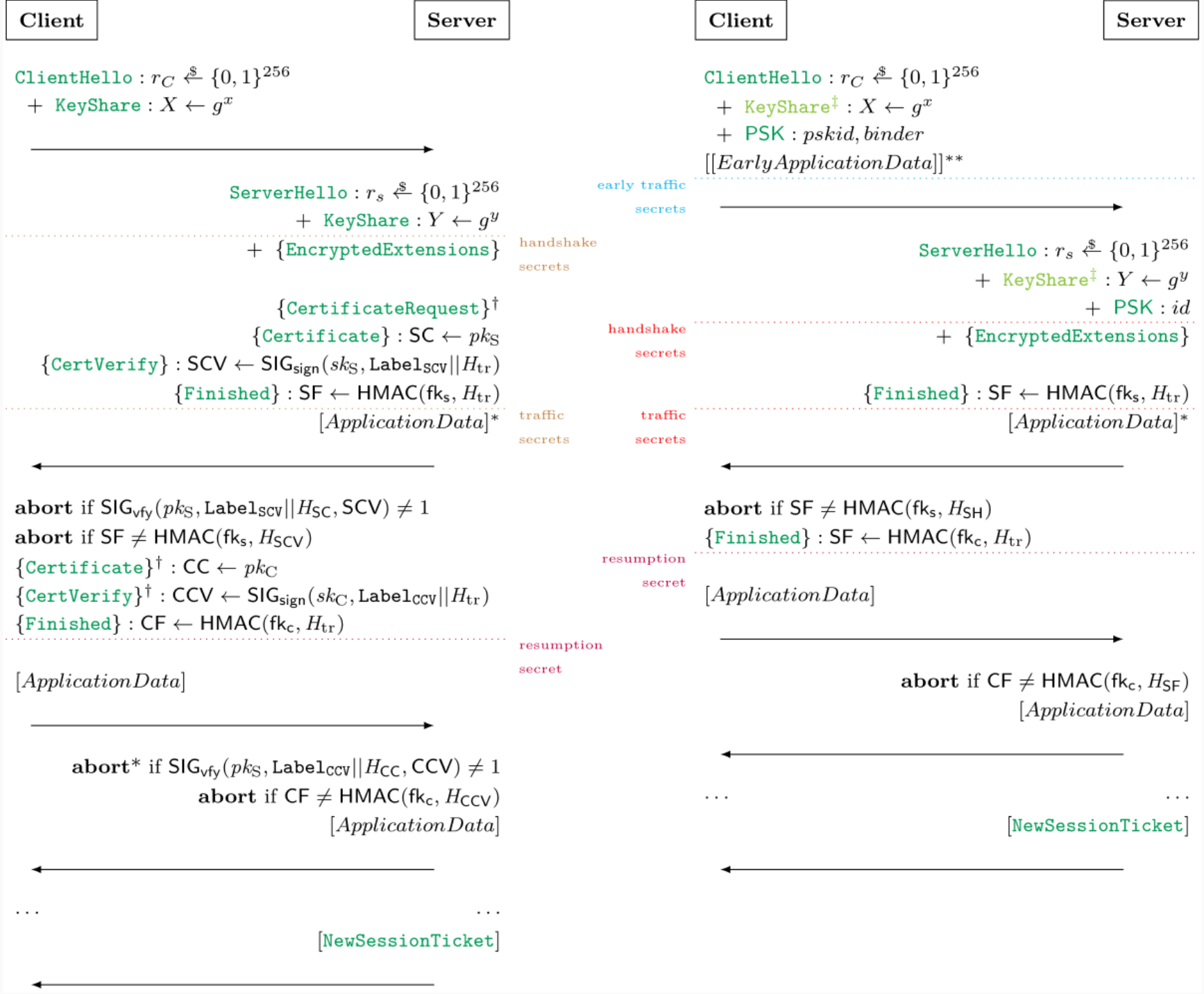
TLS 1.3 provides three handshake modes:

- Full Handshake
- PSK-DHE Handshake

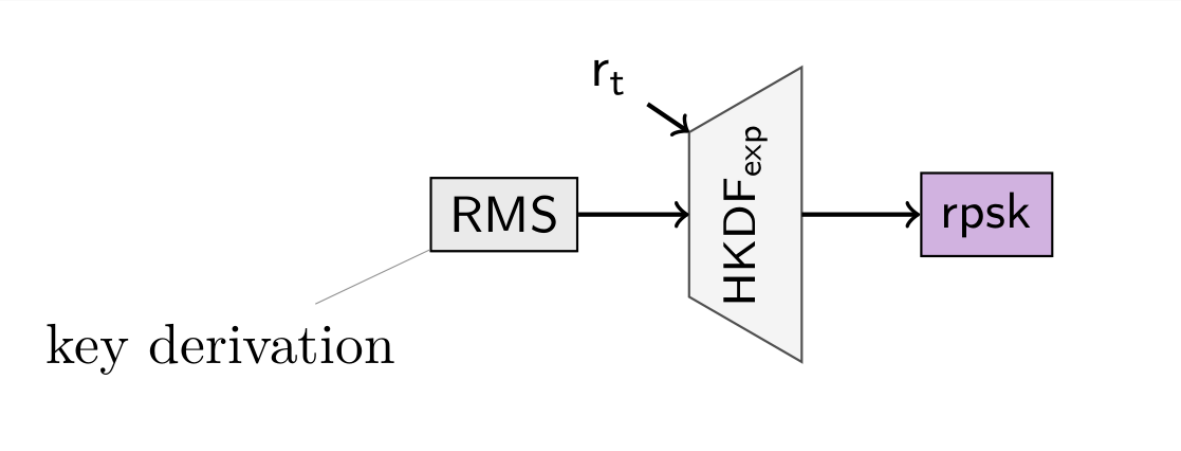
TLS 1.3: Resumption

TLS 1.3 provides three handshake modes:

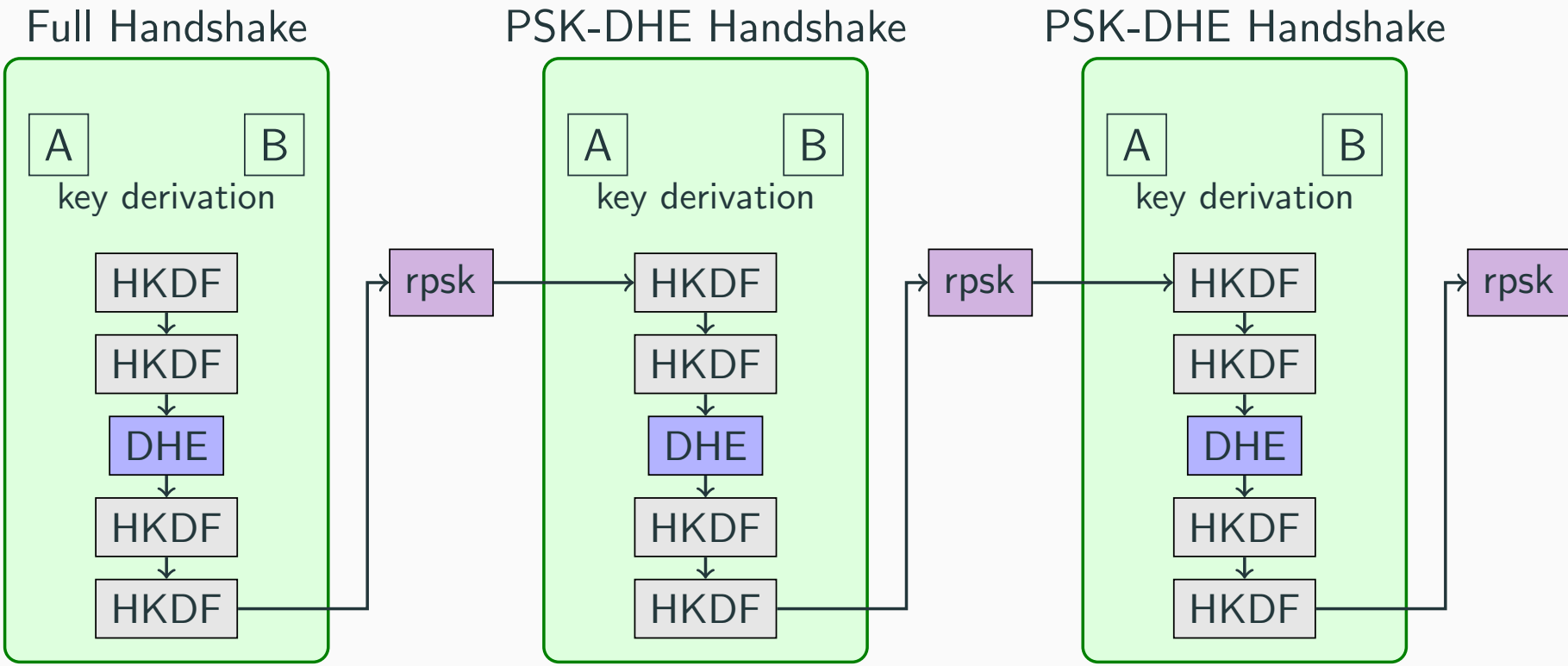
- Full Handshake
- PSK-DHE Handshake
- PSK-Only Handshake


Client
Server
ClientHello : $r_C \xleftarrow{\$} \{0,1\}^{256}$
+ KeyShare : $X \leftarrow g^x$
ServerHello : $r_s \xleftarrow{\$} \{0,1\}^{256}$
+ KeyShare : $Y \leftarrow g^y$
+ {EncryptedExtensions}
{CertificateRequest}†
{Certificate} : SC ← pk_S
{CertVerify} : SCV ← $\mathsf{SIG}_{\mathsf{sign}}(sk_S, \mathtt{Label}_{\mathtt{SCV}} || H_{tr})$
{Finished} : SF ← $\mathsf{HMAC}(\mathsf{fk_s}, H_{tr})$
[ApplicationData]*
handshake secrets
traffic secrets
abort if $\mathsf{SIG}_{\mathsf{vfy}}(pk_S, \mathtt{Label}_{\mathtt{SCV}} || H_{SC}, \mathsf{SCV}) \neq 1$
abort if SF ≠ $\mathsf{HMAC}(\mathsf{fk_s}, H_{SCV})$
{Certificate}† : CC ← pk_C
{CertVerify}† : CCV ← $\mathsf{SIG}_{\mathsf{sign}}(sk_C, \mathtt{Label}_{\mathtt{CCV}} || H_{tr})$
{Finished} : CF ← $\mathsf{HMAC}(\mathsf{fk_c}, H_{tr})$
resumption secret
[ApplicationData]
abort* if $\mathsf{SIG}_{\mathsf{vfy}}(pk_S, \mathtt{Label}_{\mathtt{CCV}} || H_{CC}, \mathsf{CCV}) \neq 1$
abort if CF ≠ $\mathsf{HMAC}(\mathsf{fk_c}, H_{CCV})$
[ApplicationData]
...
[NewSessionTicket]
Client
Server
ClientHello : $r_C \xleftarrow{\$} \{0,1\}^{256}$
+ KeyShare‡ : $X \leftarrow g^x$
+ PSK : pskid, binder
[[EarlyApplicationData]]**
ServerHello : $r_s \xleftarrow{\$} \{0,1\}^{256}$
+ KeyShare‡ : $Y \leftarrow g^y$
+ PSK : id
+ {EncryptedExtensions}
{Finished} : SF ← $\mathsf{HMAC}(\mathsf{fk_s}, H_{tr})$
[ApplicationData]*
early traffic secrets
handshake secrets
traffic secrets
abort if SF ≠ $\mathsf{HMAC}(\mathsf{fk_s}, H_{SH})$
{Finished} : SF ← $\mathsf{HMAC}(\mathsf{fk_c}, H_{tr})$
resumption secret
[ApplicationData]
abort if CF ≠ $\mathsf{HMAC}(\mathsf{fk_c}, H_{SF})$
[ApplicationData]
...
[NewSessionTicket]

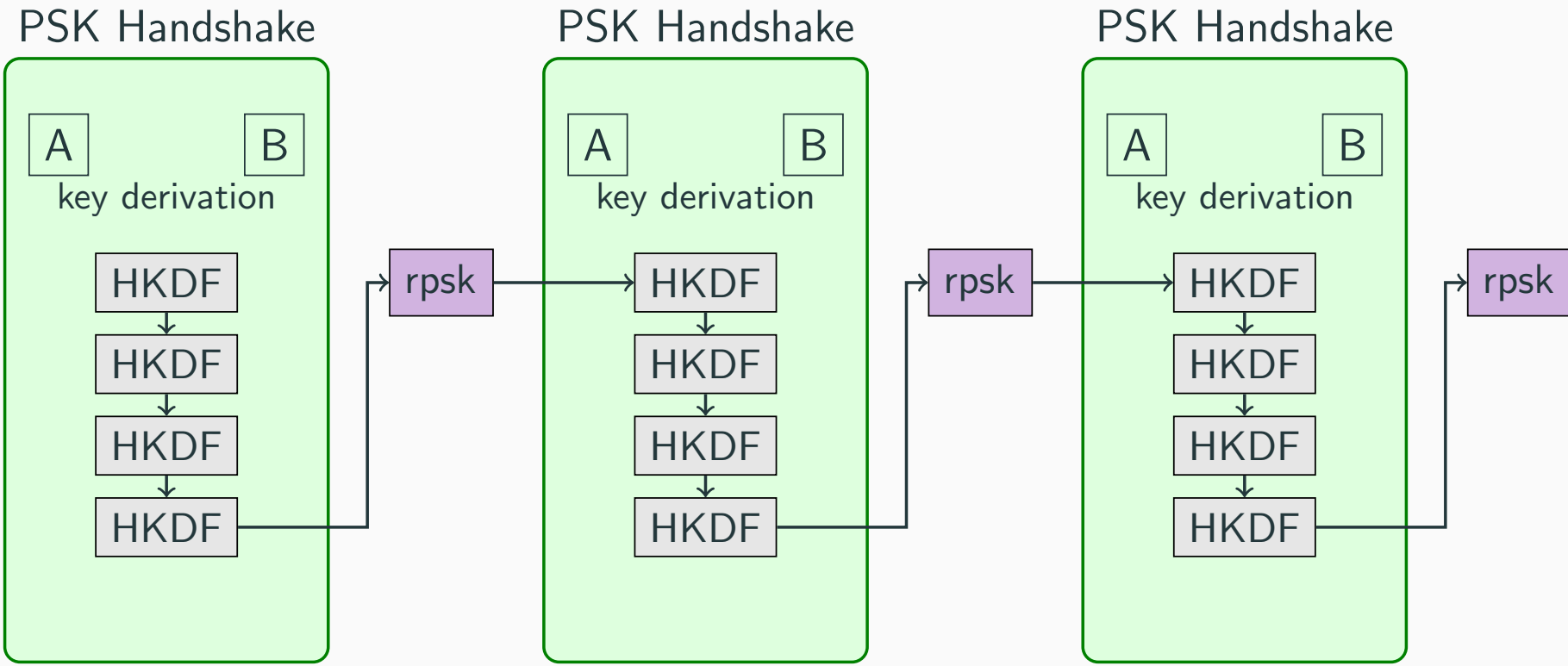
Resumption PSK

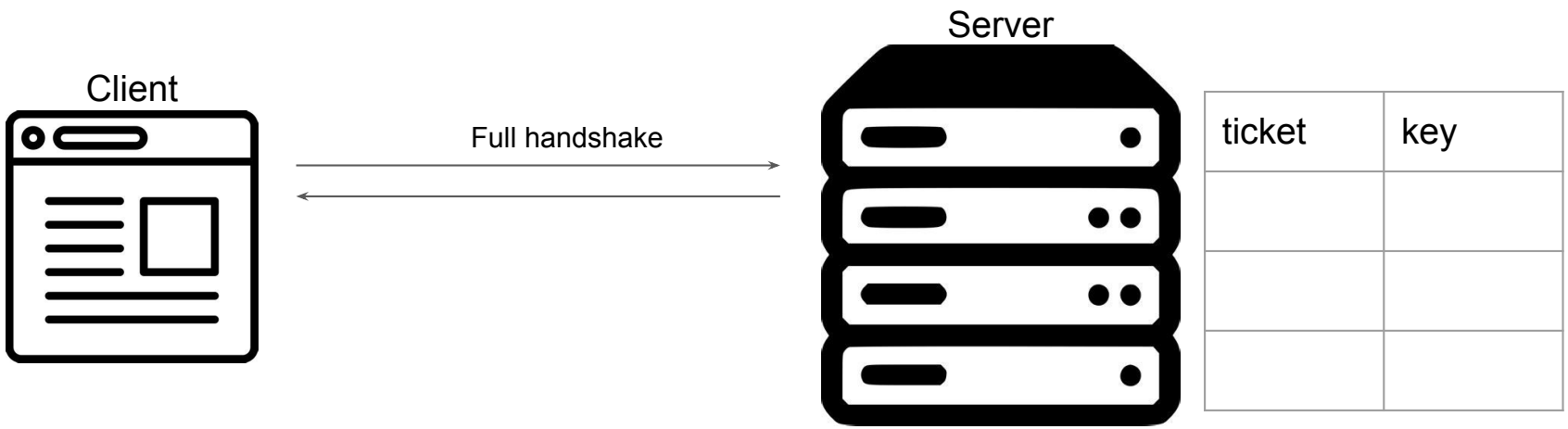


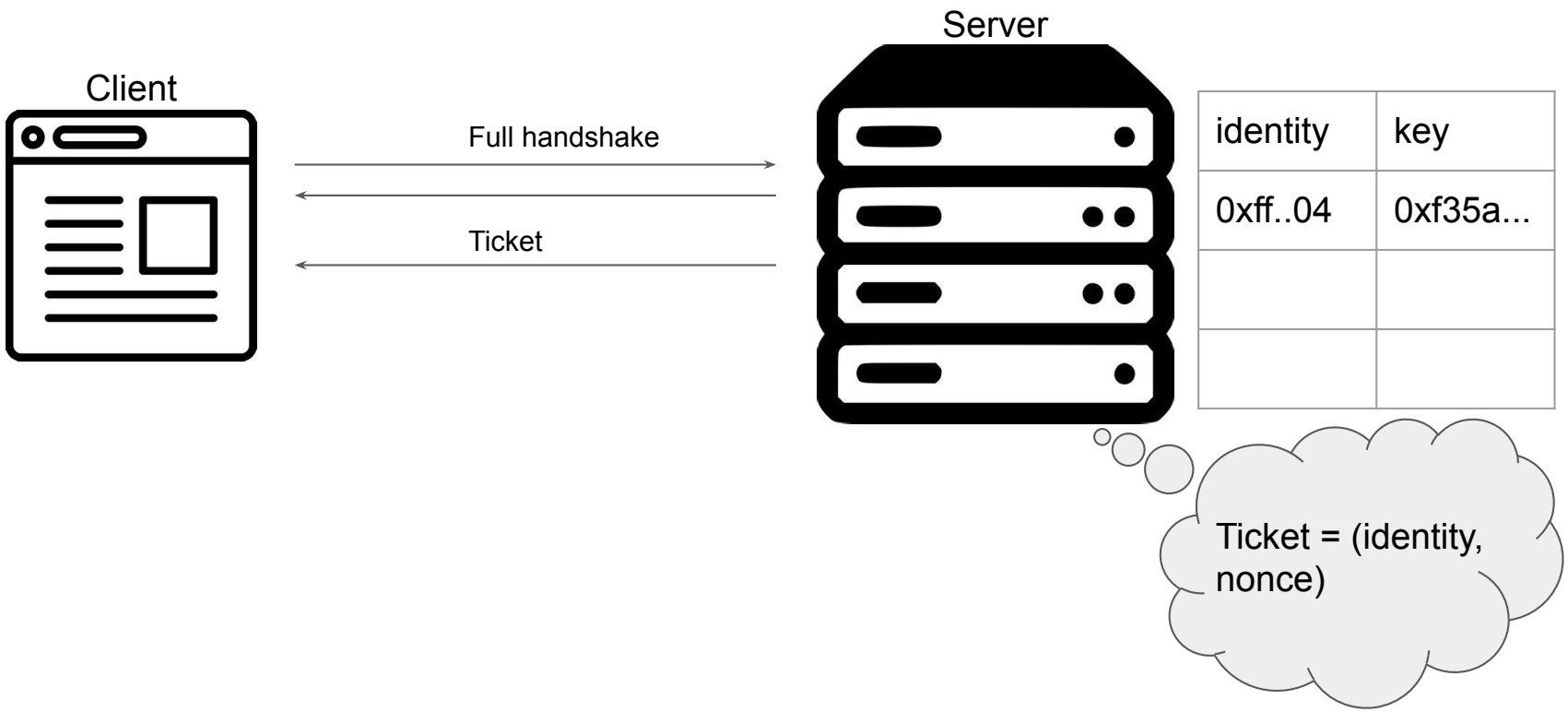
TLS 1.3: Resumption

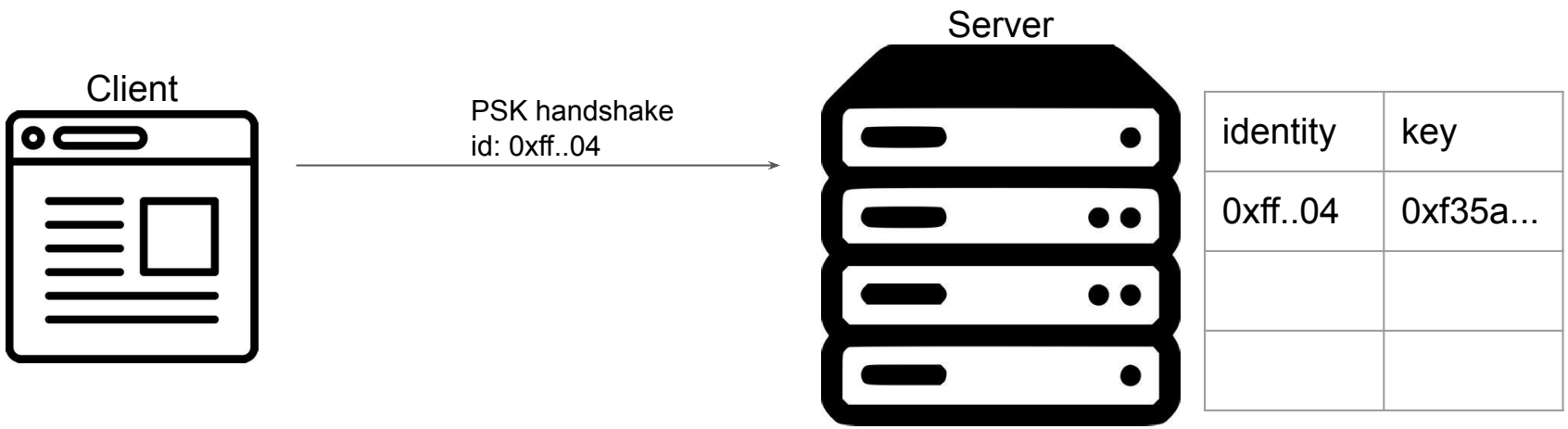


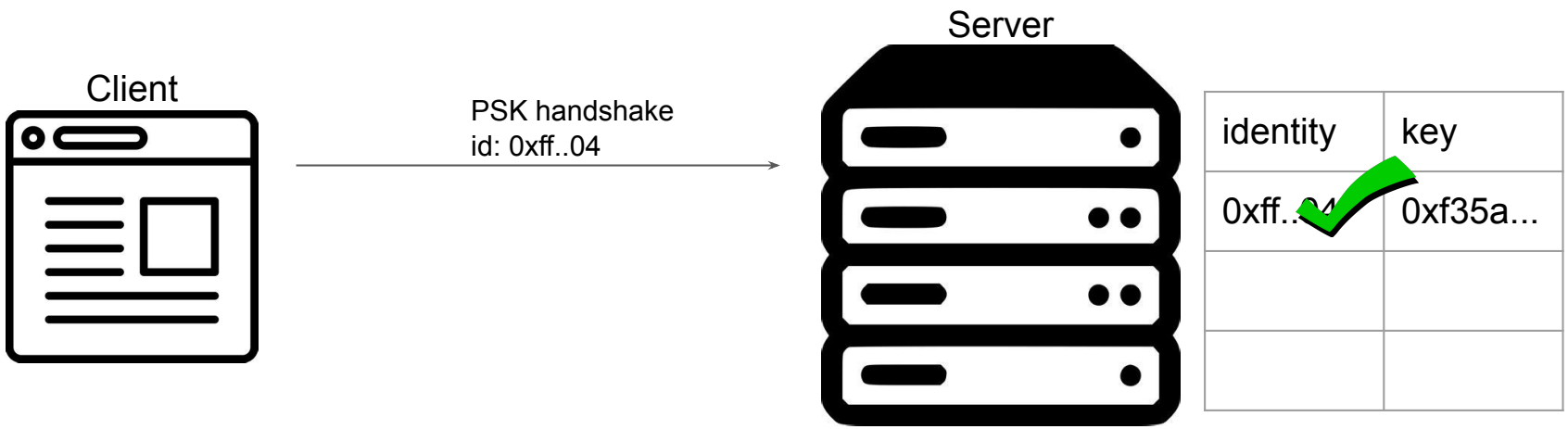
TLS 1.3: Resumption

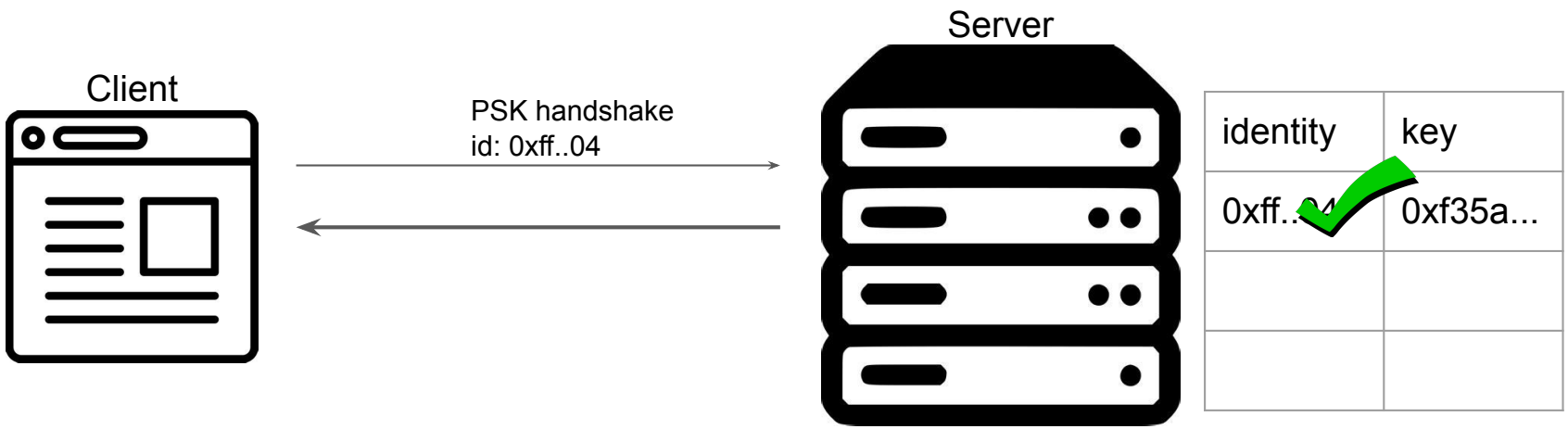


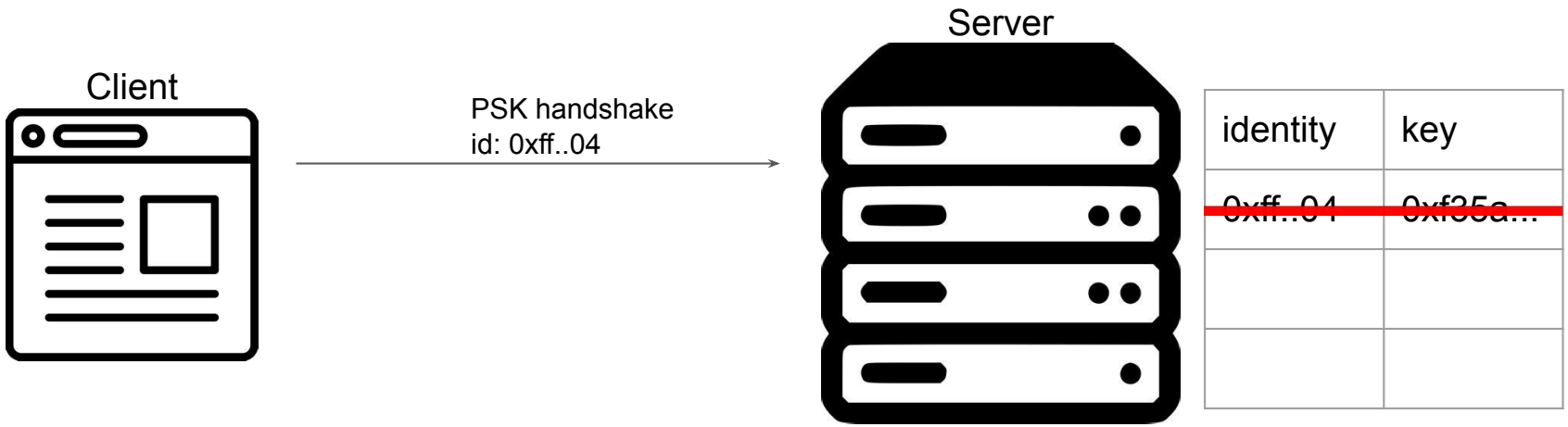

Client
Full handshake
Server
ticket
key

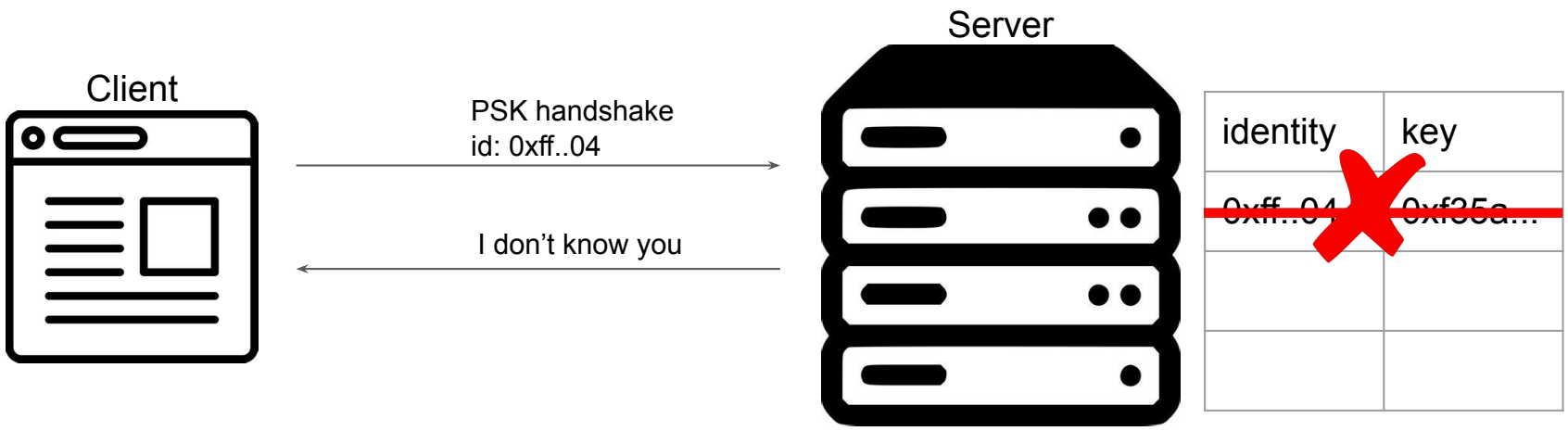

Client
Server
Full handshake
Ticket
identity
key
0xff..04
0xf35a...
Ticket = (identity, nonce)


Client
PSK handshake
id: 0xff..04
Server
identity
key
0xff..04
0xf35a...

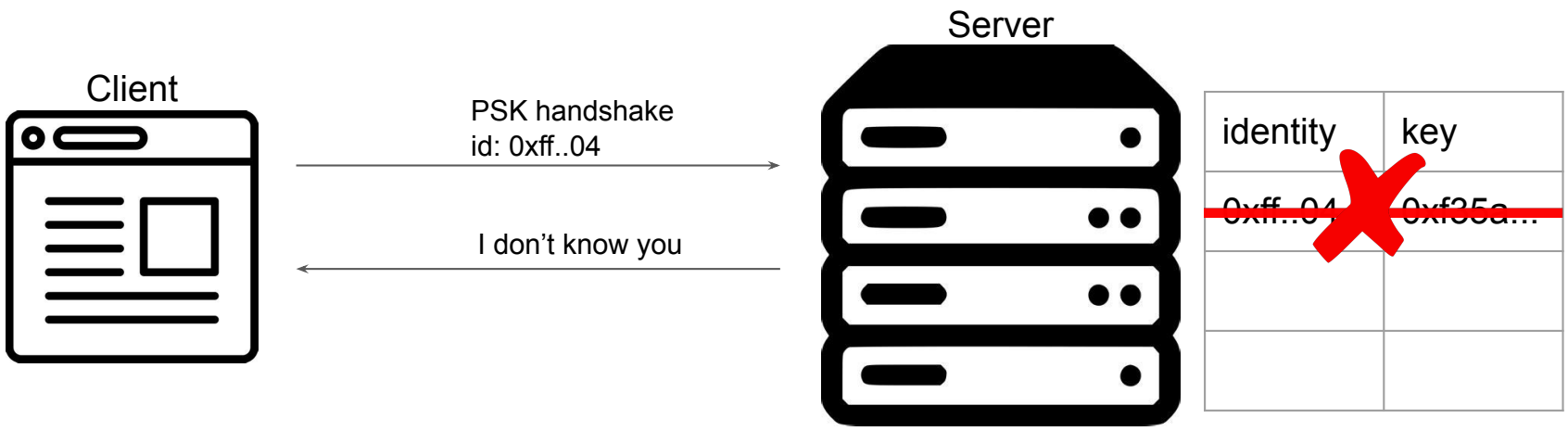

Client
PSK handshake
id: 0xff..04
Server
identity
key
0xff..04
0xf35a...


Client
PSK handshake
id: 0xff..04
Server
identity
key
0xff..04
0xf35a...

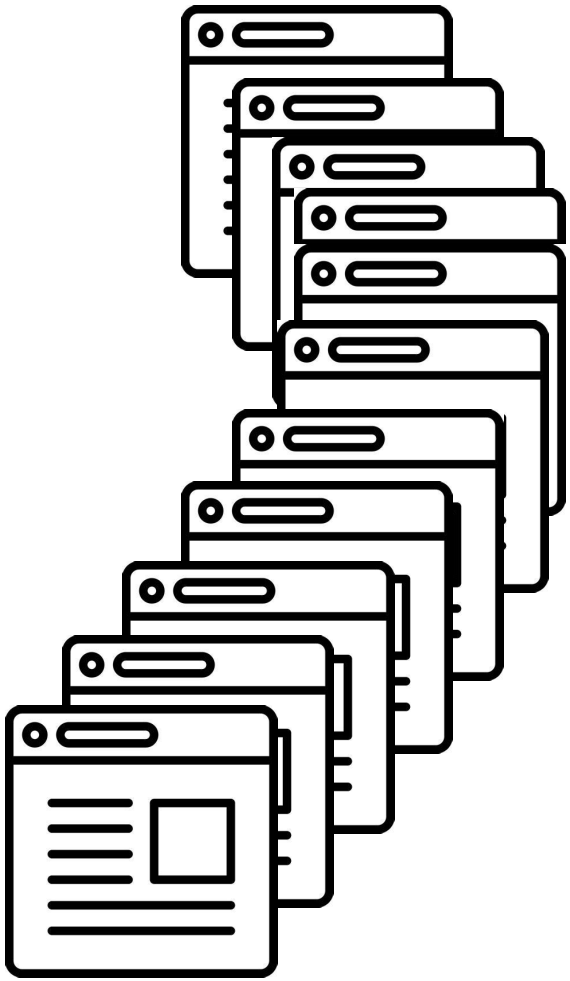

Client
PSK handshake
id: 0xff..04
Server
identity
key
0xff..04
0xf35a...


Client
PSK handshake
id: 0xff..04
I don’t know you
Server
identity
key
0xff..04
0xf35a...

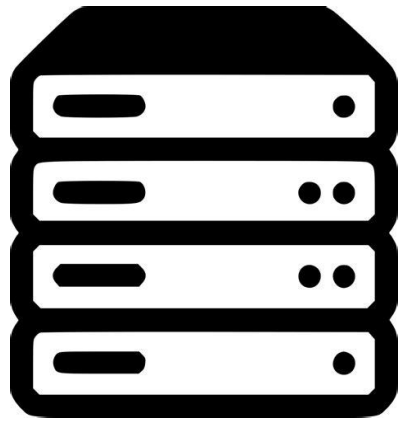
Question 4.1


Client
PSK handshake
id: 0xff..04
I don’t know you
Server
identity
key
0xff..04
0xf35a...

Clients

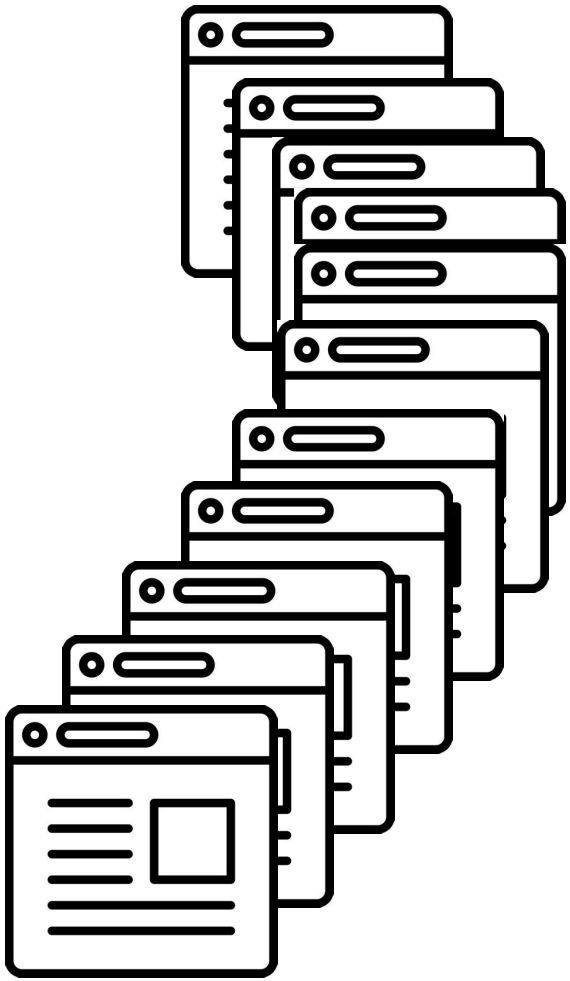


Server

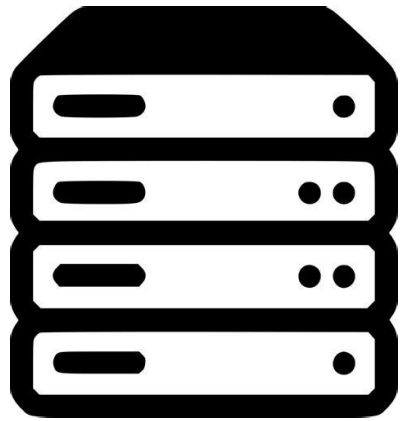


identity	key
0xff..04	0xf35a...
0xa5..31	0x522...
...	...
...	...
0xb3..3f	0xd34...
0x13..37	0xbab...
0xab..ba	0xcaf3...
0xf3..31	0xbad...

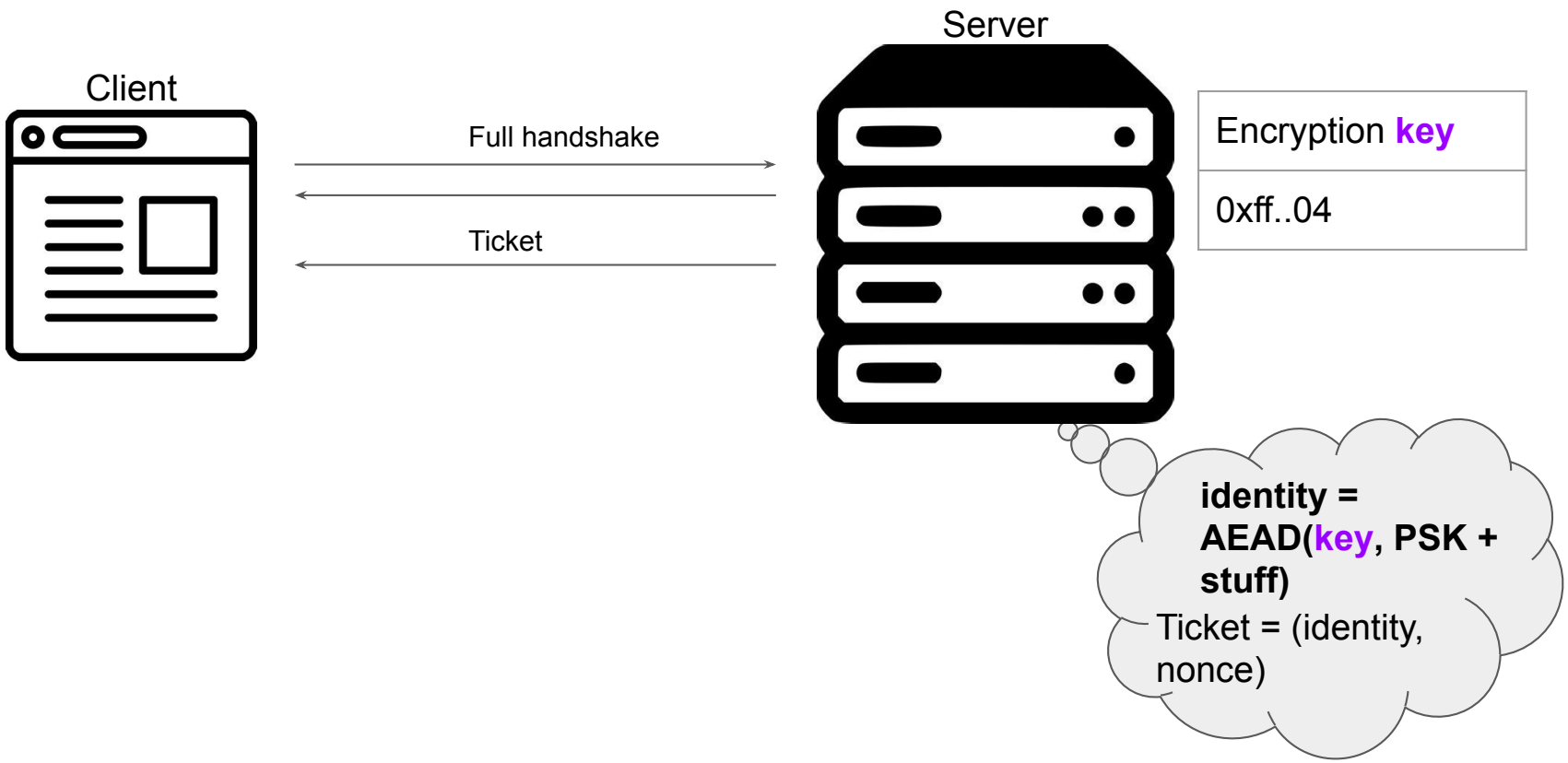
Clients

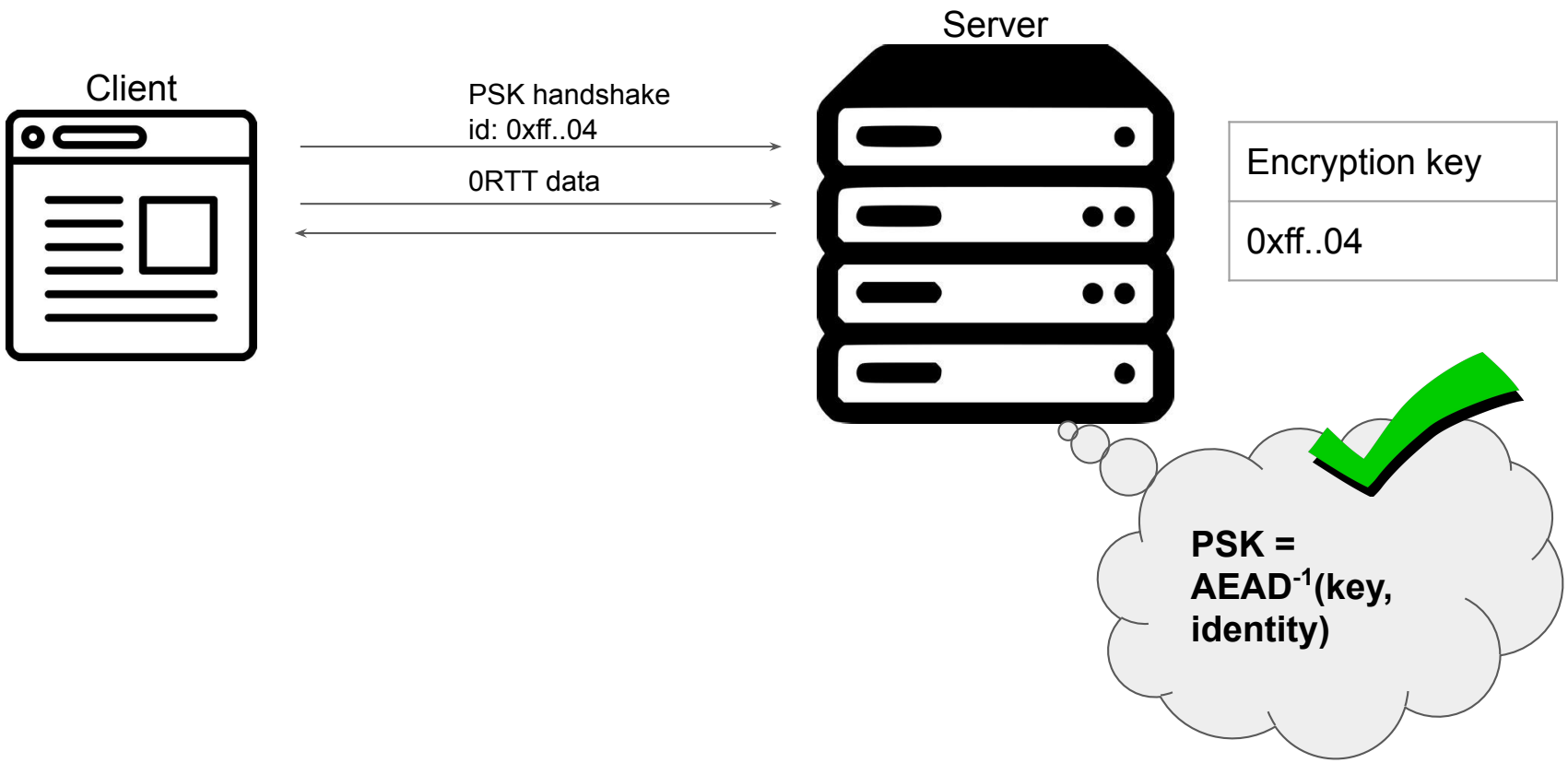


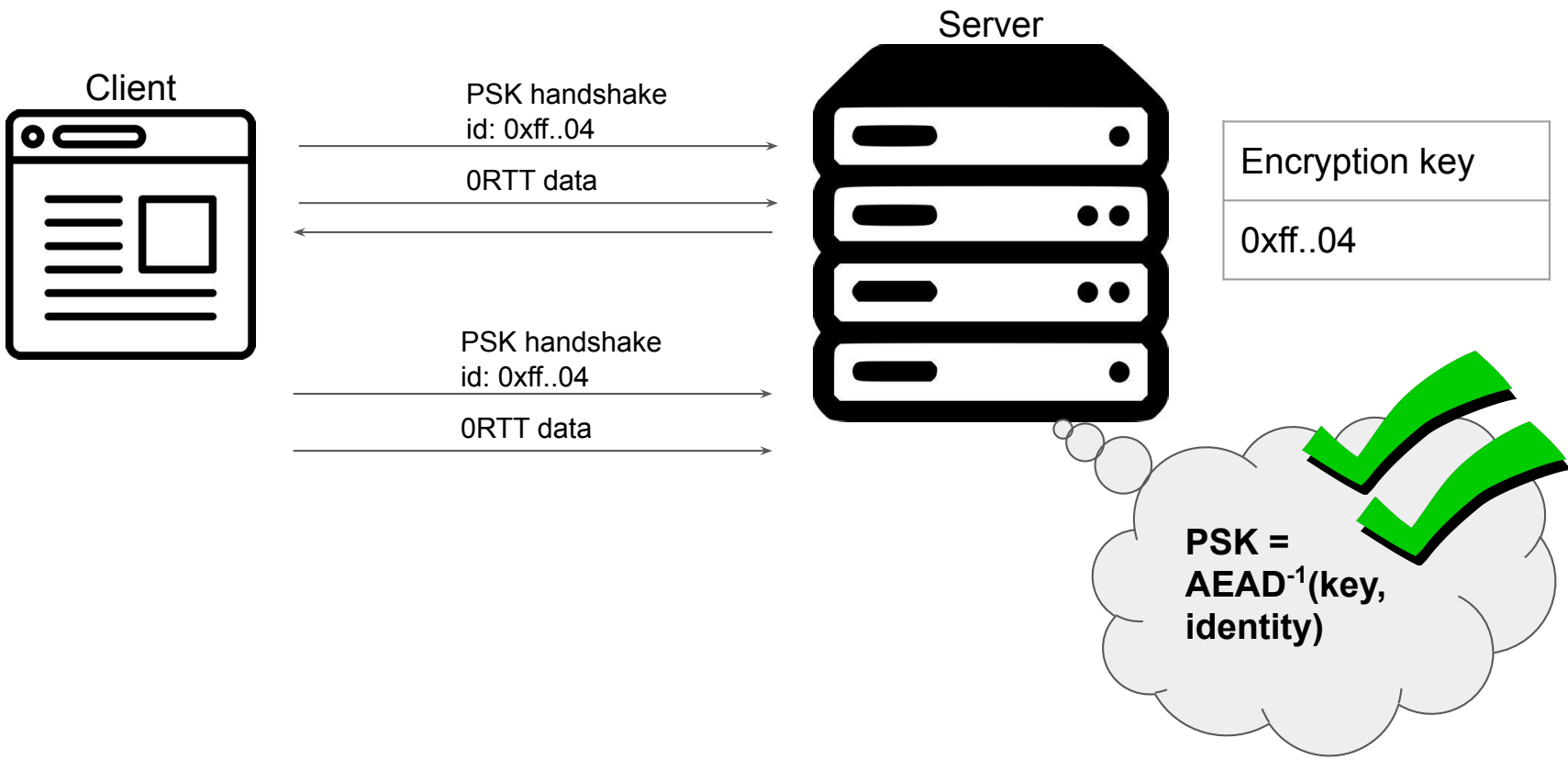
Server

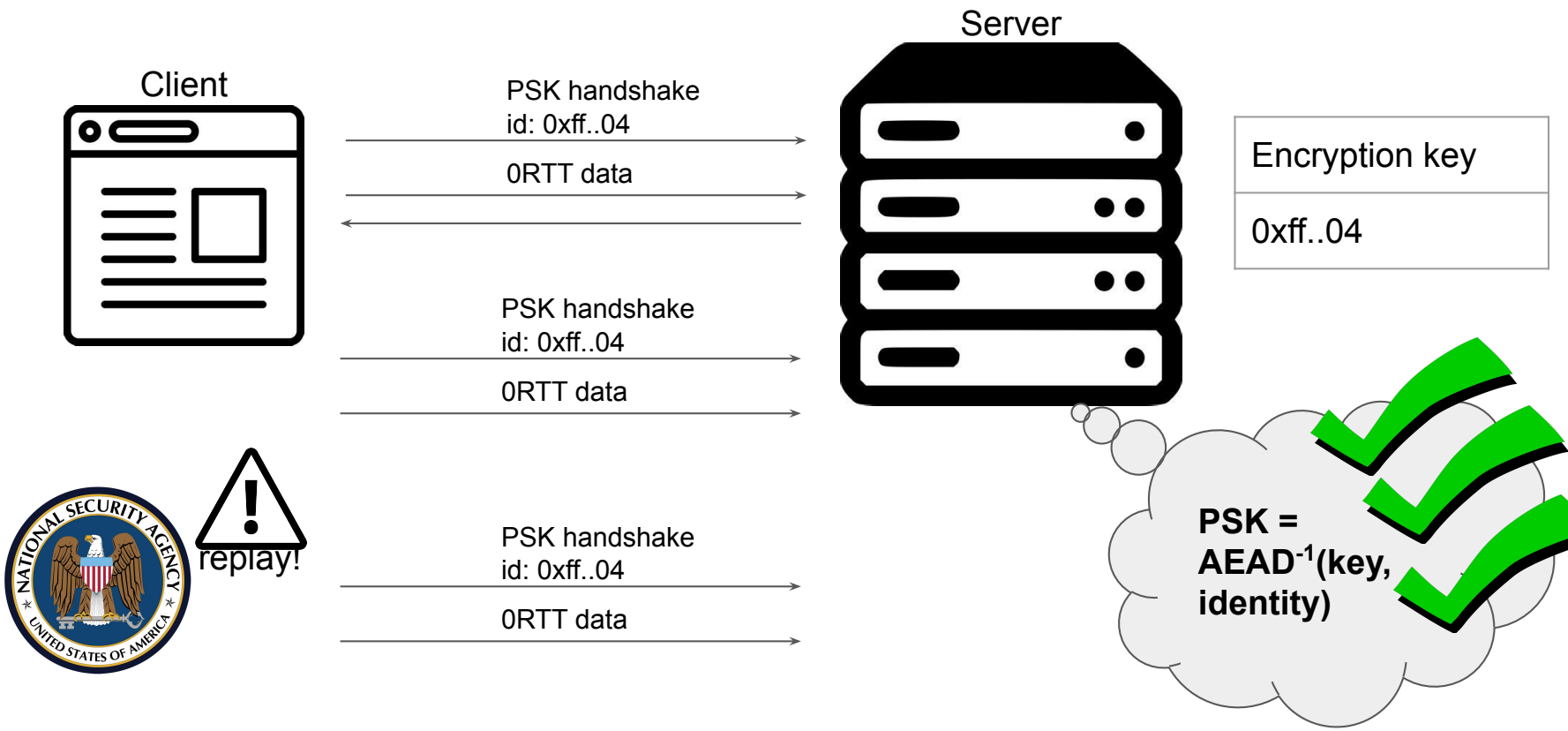


identity	key
0xff..04	0xf35a...
0xa5..31	0x522...
...	...
...	...
0xb3..3f	0xd34...
0x13..37	0xbab...
0xab..ba	0xcaf3...
0xf3..31	0xbad...

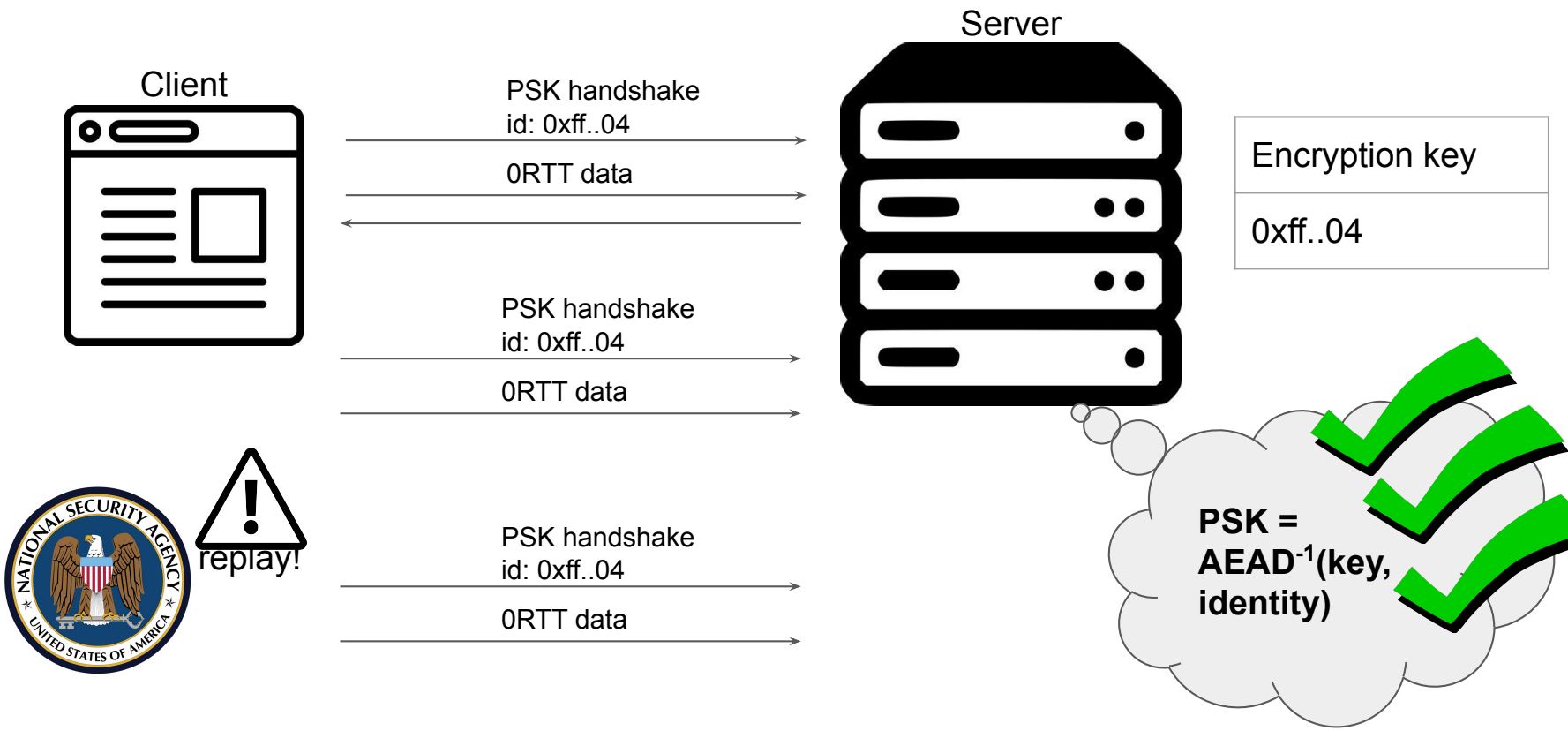

Client
Server
Full handshake
Ticket
Encryption key
0xff..04
identity = AEAD(key, PSK + stuff)
Ticket = (identity, nonce)


Client
Server
PSK handshake
id: 0xff..04
0RTT data
Encryption key
0xff..04
PSK =
AEAD-1(key,
identity)

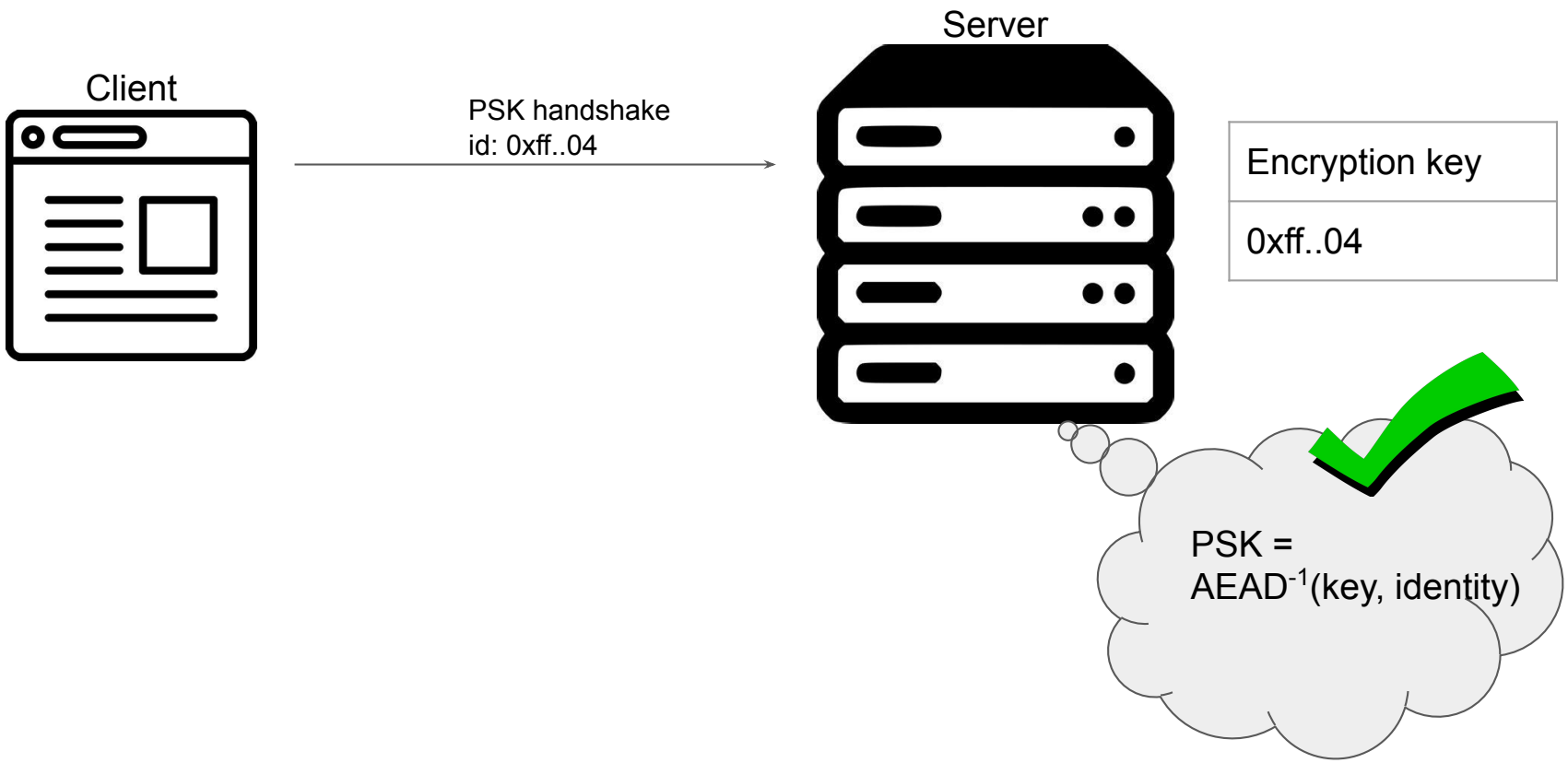

Client
Server
PSK handshake
id: 0xff..04
0RTT data
PSK handshake
id: 0xff..04
0RTT data
Encryption key
0xff..04
PSK =
AEAD-1(key,
identity)


Server
Client
PSK handshake
id: 0xff..04
0RTT data
Encryption key
0xff..04
PSK handshake
id: 0xff..04
0RTT data
replay!
PSK handshake
id: 0xff..04
0RTT data
PSK =
AEAD-1(key,
identity)

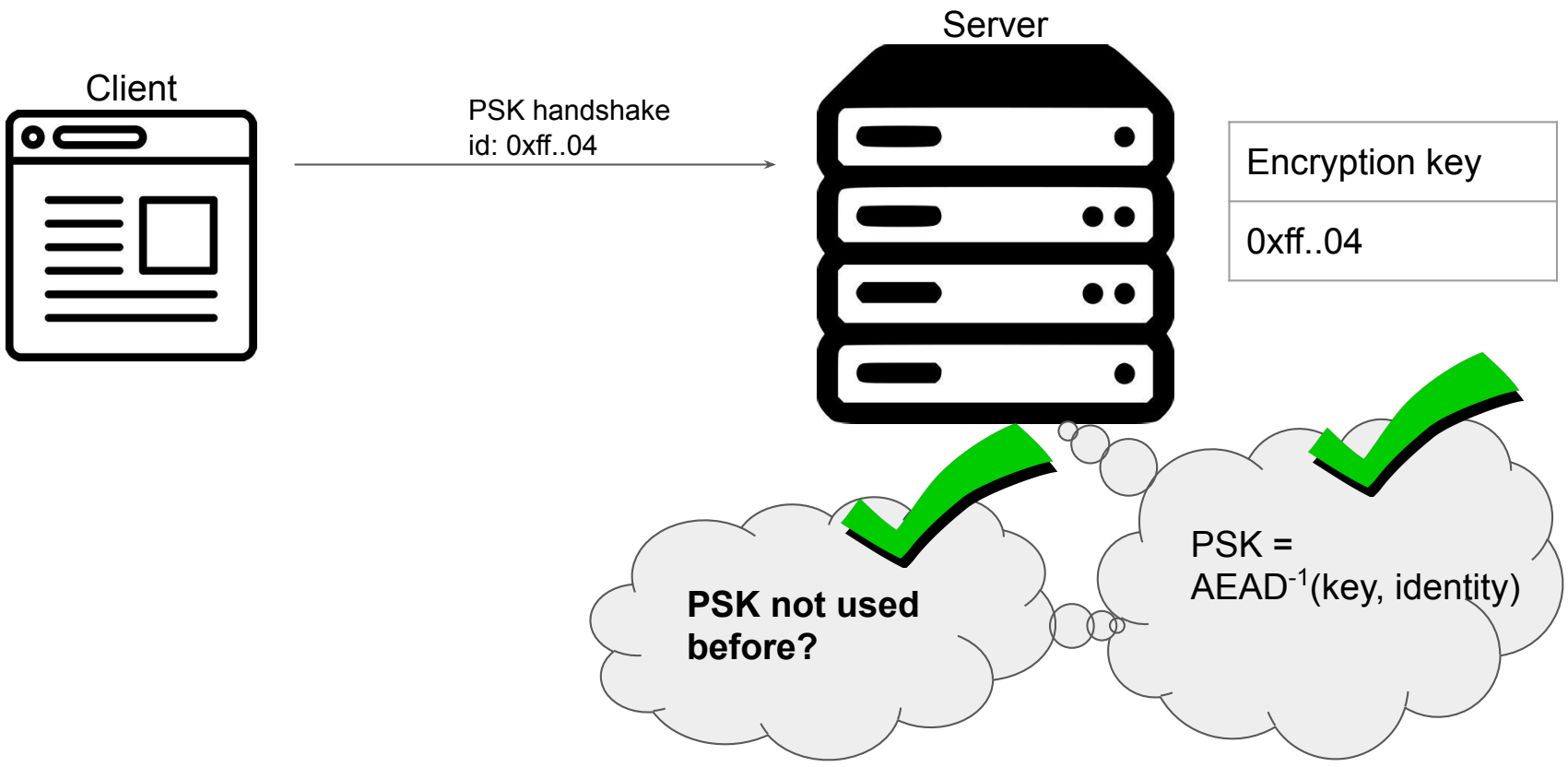
Question 4.2


Server
Client
PSK handshake
id: 0xff..04
0RTT data
Encryption key
0xff..04
PSK handshake
id: 0xff..04
0RTT data
replay!
PSK handshake
id: 0xff..04
0RTT data
PSK =
AEAD-1(key,
identity)

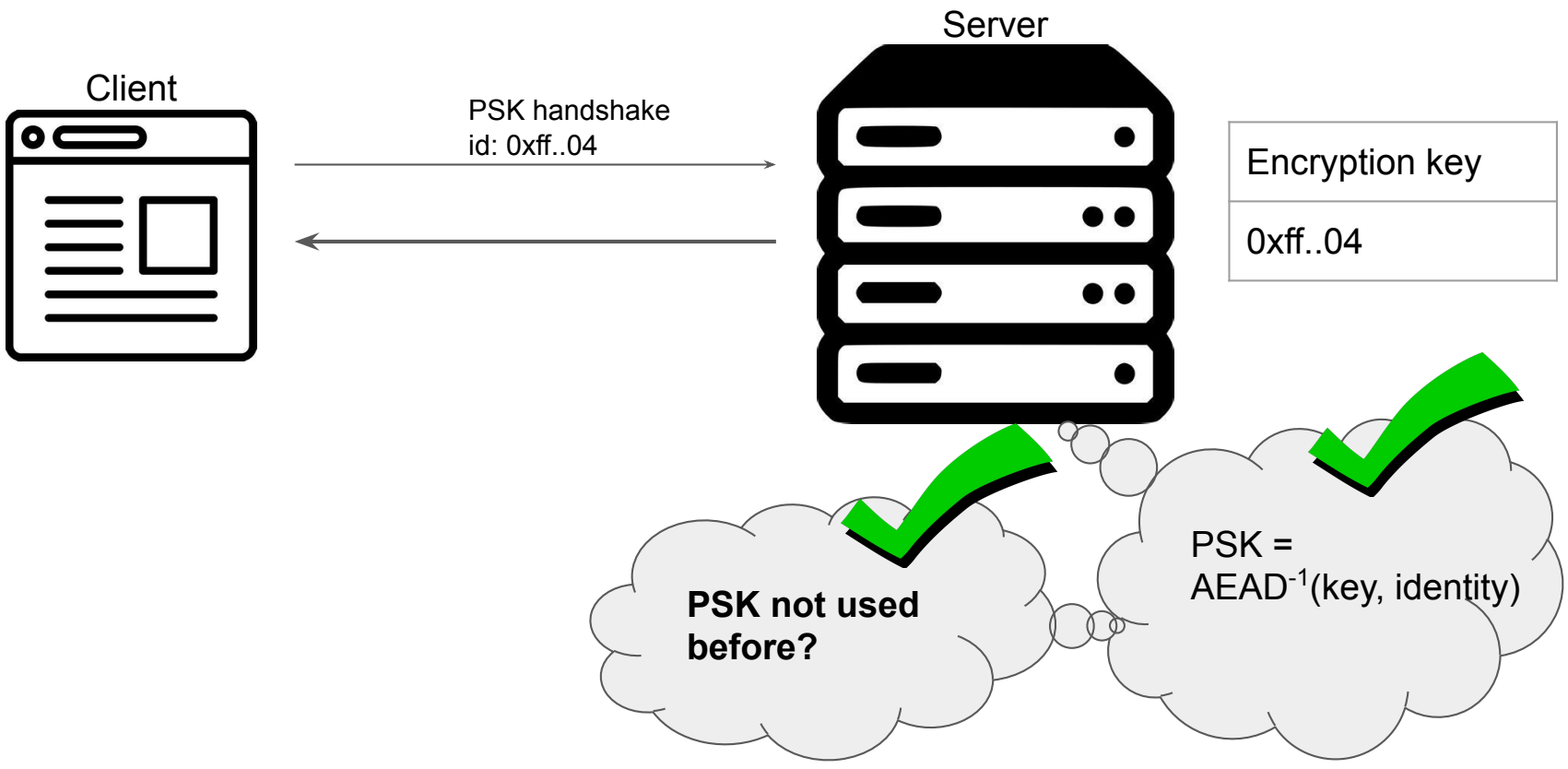
Fight...


Client
PSK handshake
id: 0xff..04
Server
Encryption key
0xff..04
PSK =
AEAD-1(key, identity)

Fight...

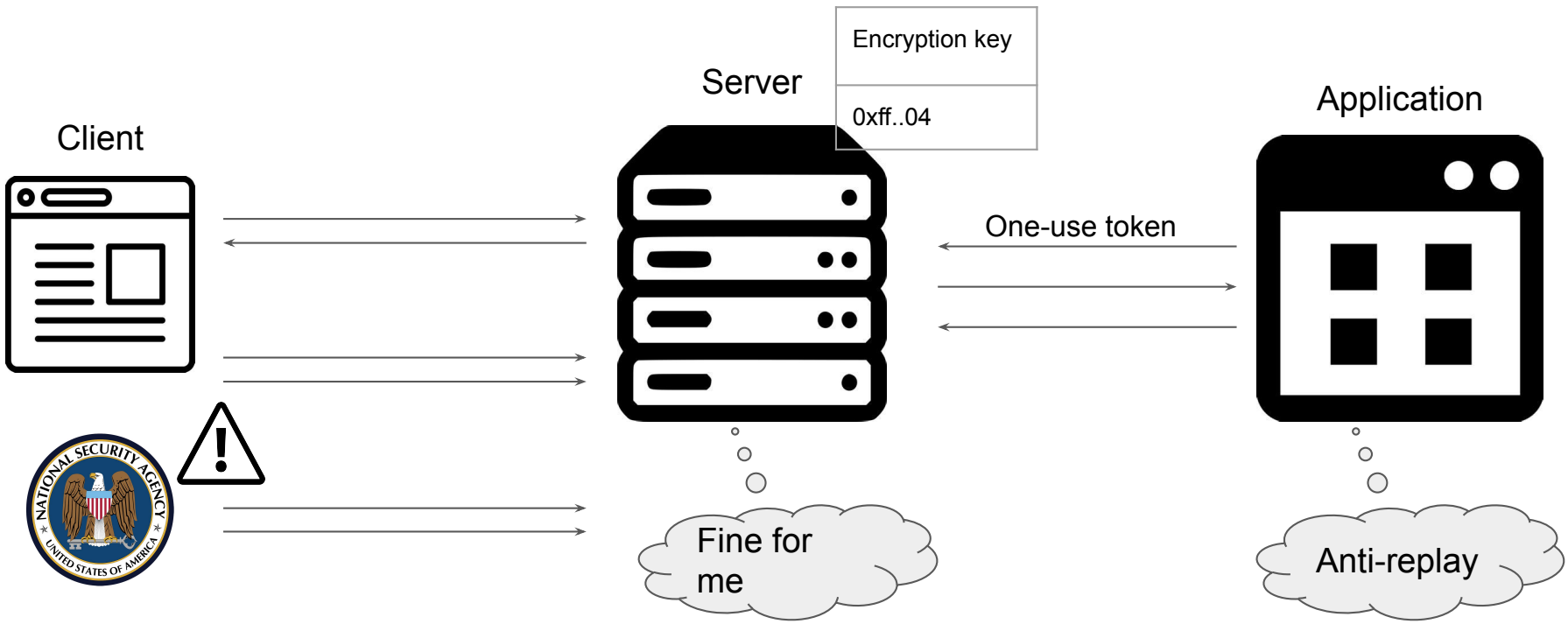

Client
PSK handshake
id: 0xff..04
Server
Encryption key
0xff..04
PSK not used before?
PSK = AEAD-1(key, identity)

Fight...


Server
Client
PSK handshake
id: 0xff..04
Encryption key
0xff..04
PSK not used before?
PSK = AEAD-1(key, identity)

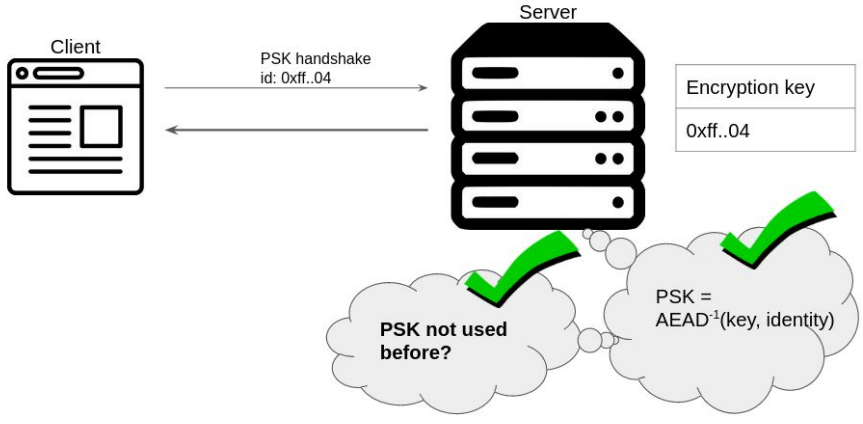
...or give up

...or give up

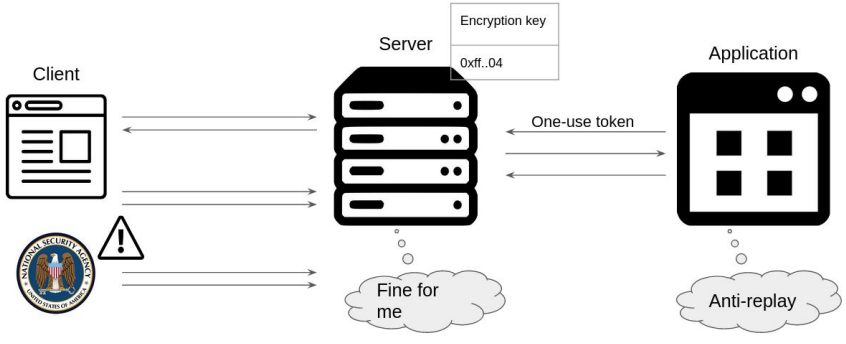

Client
Server
Encryption key
0xff..04
Application
One-use token
Fine for me
Anti-replay
NATIONAL SECURITY AGENCY
UNITED STATES OF AMERICA

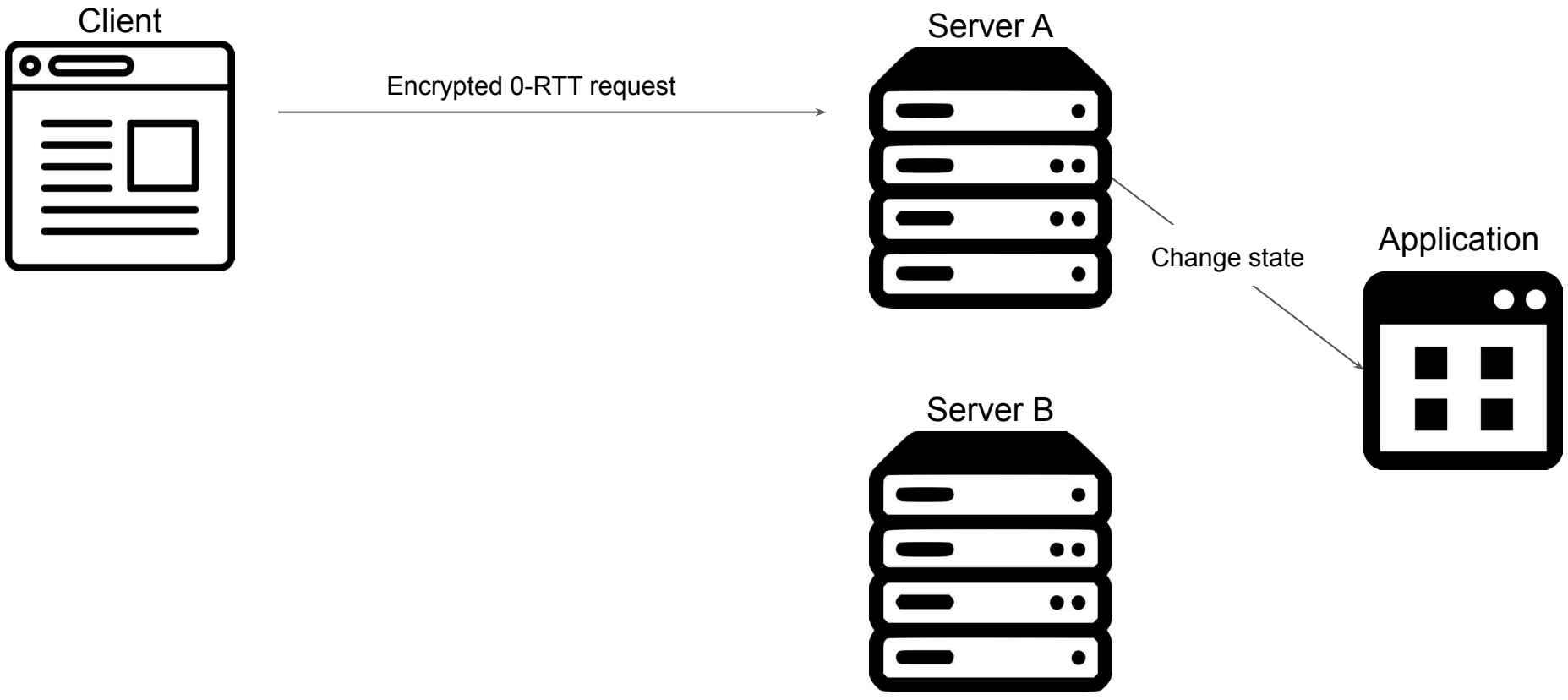
Question 4.3

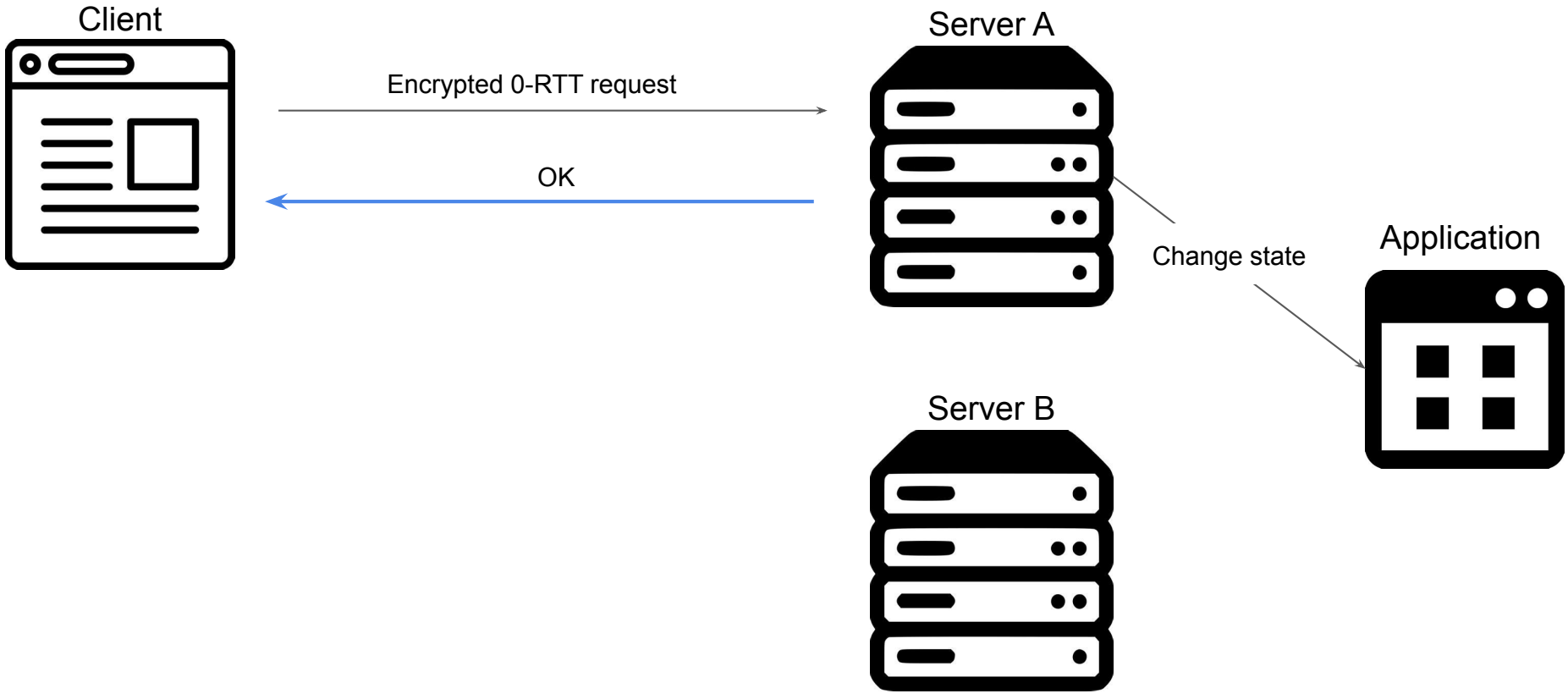
Fight...



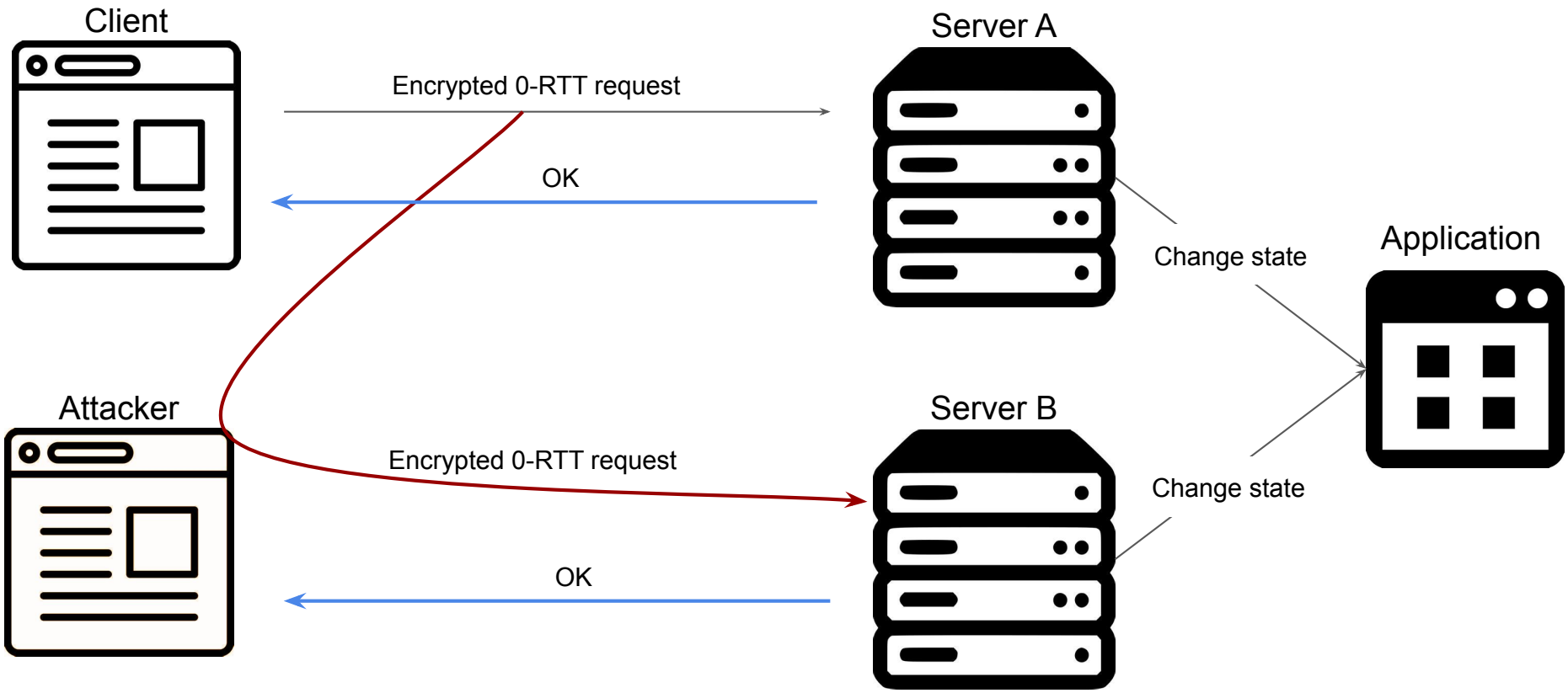
...or give up



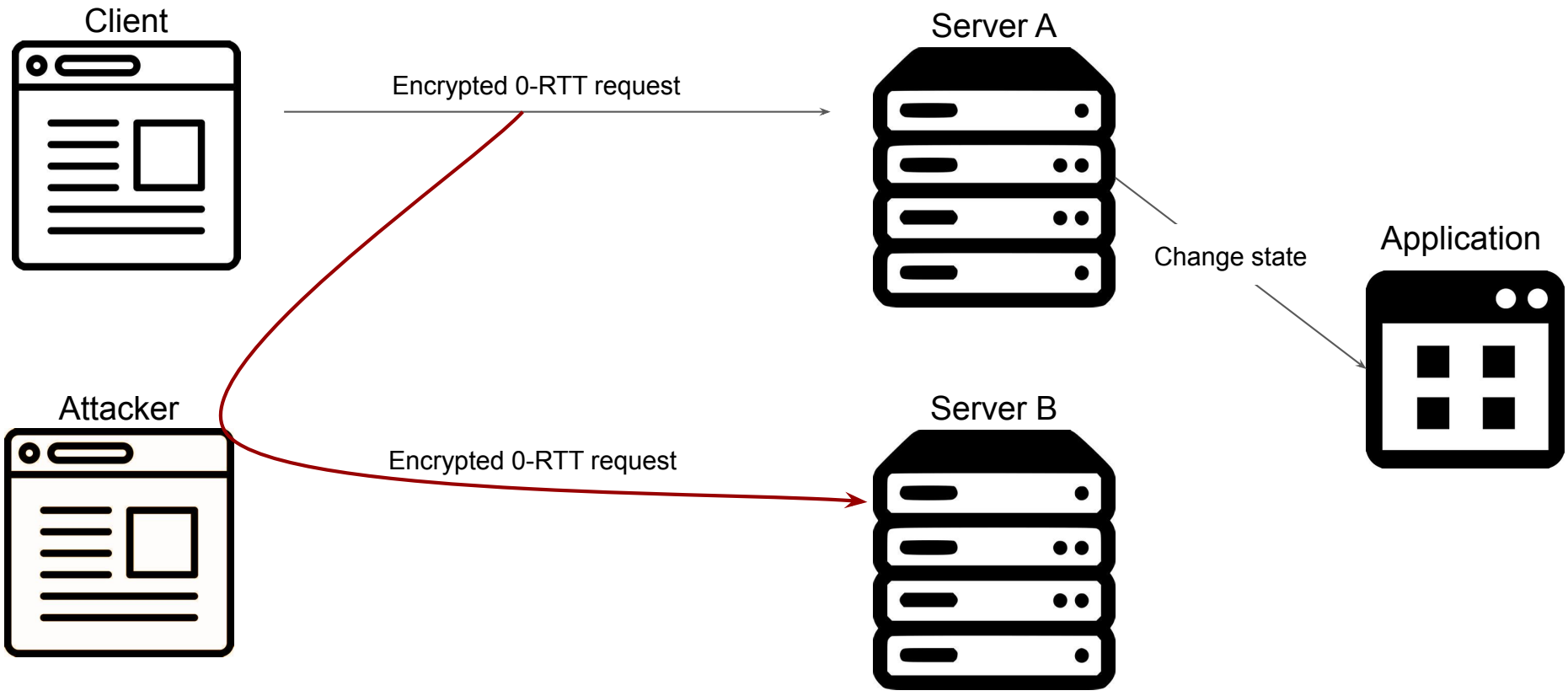

Client
Encrypted 0-RTT request
Server A
Change state
Application
Server B


Client
Encrypted 0-RTT request
OK
Server A
Change state
Application
Server B

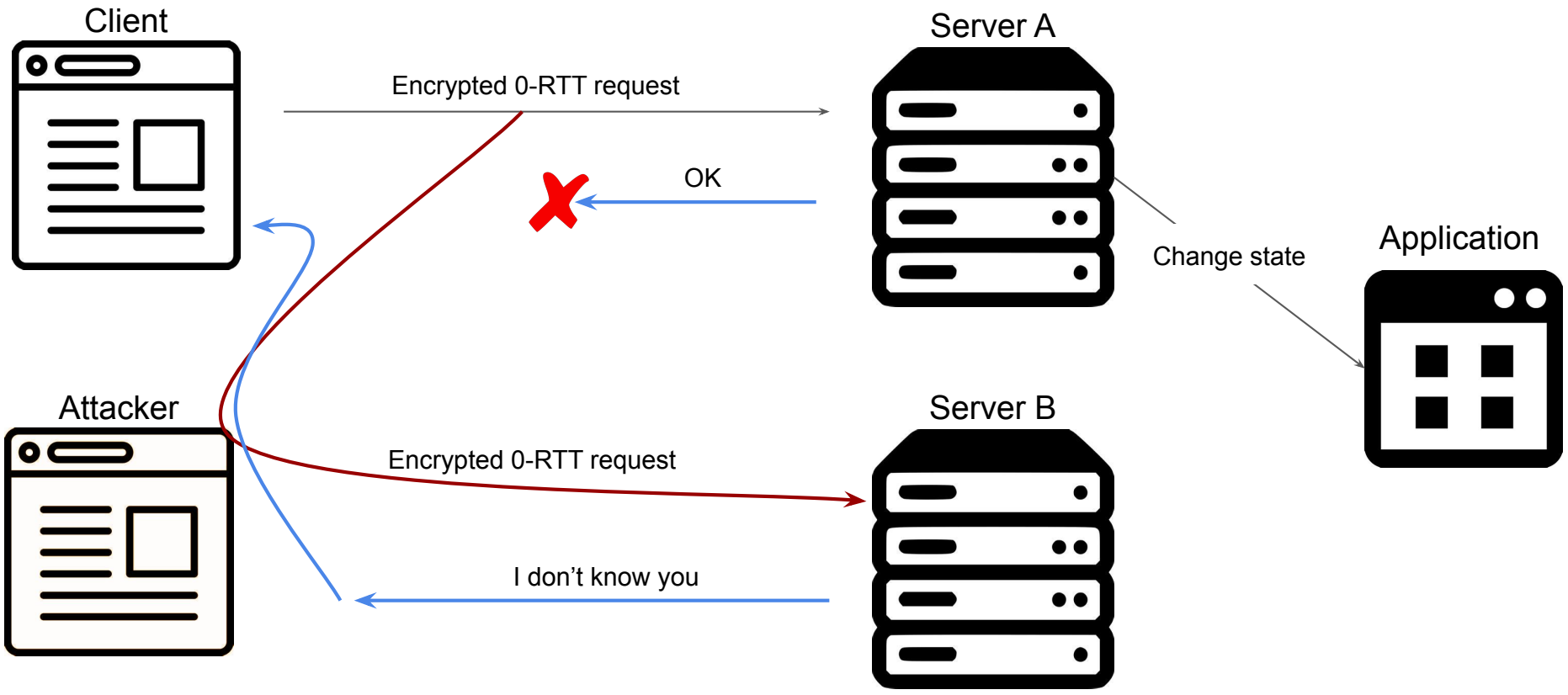
No replay protection



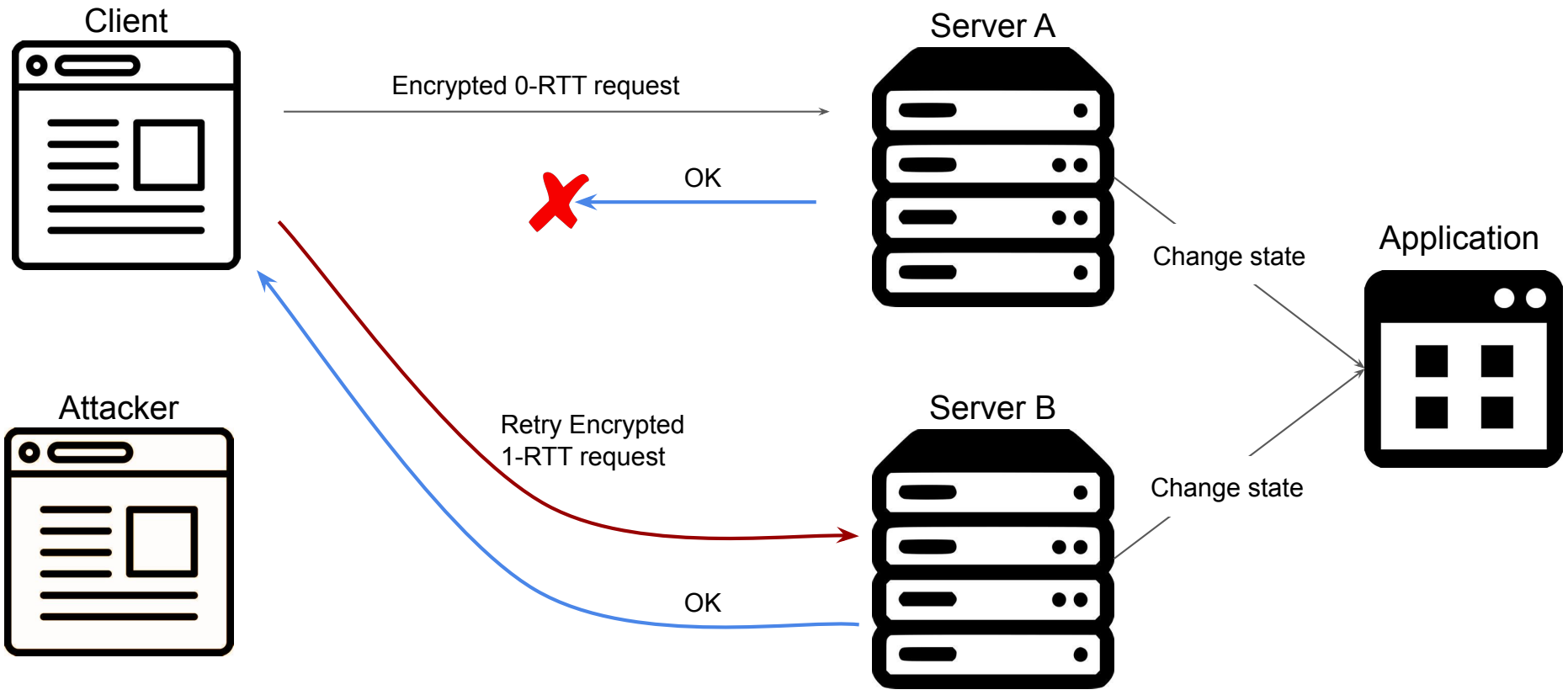
With TLS-level replay protection



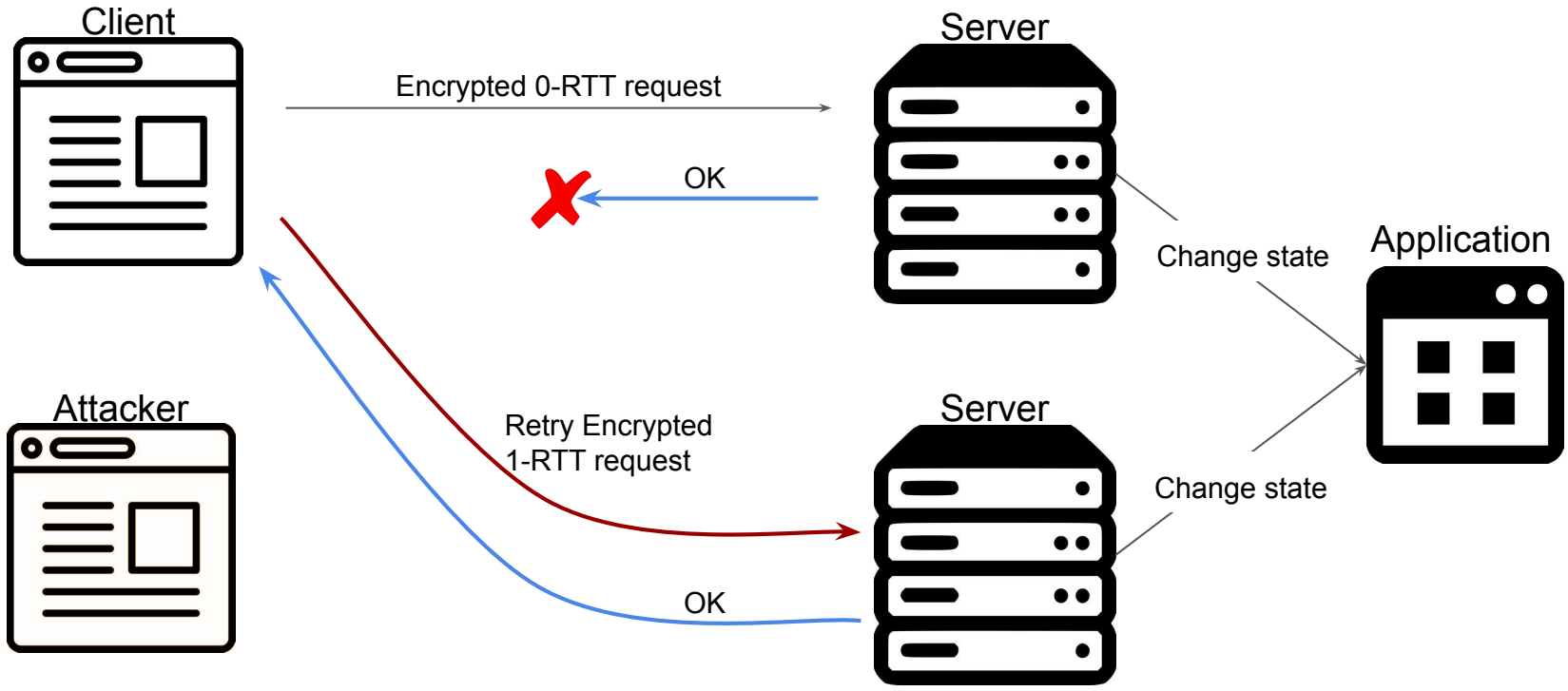
With TLS-level replay protection



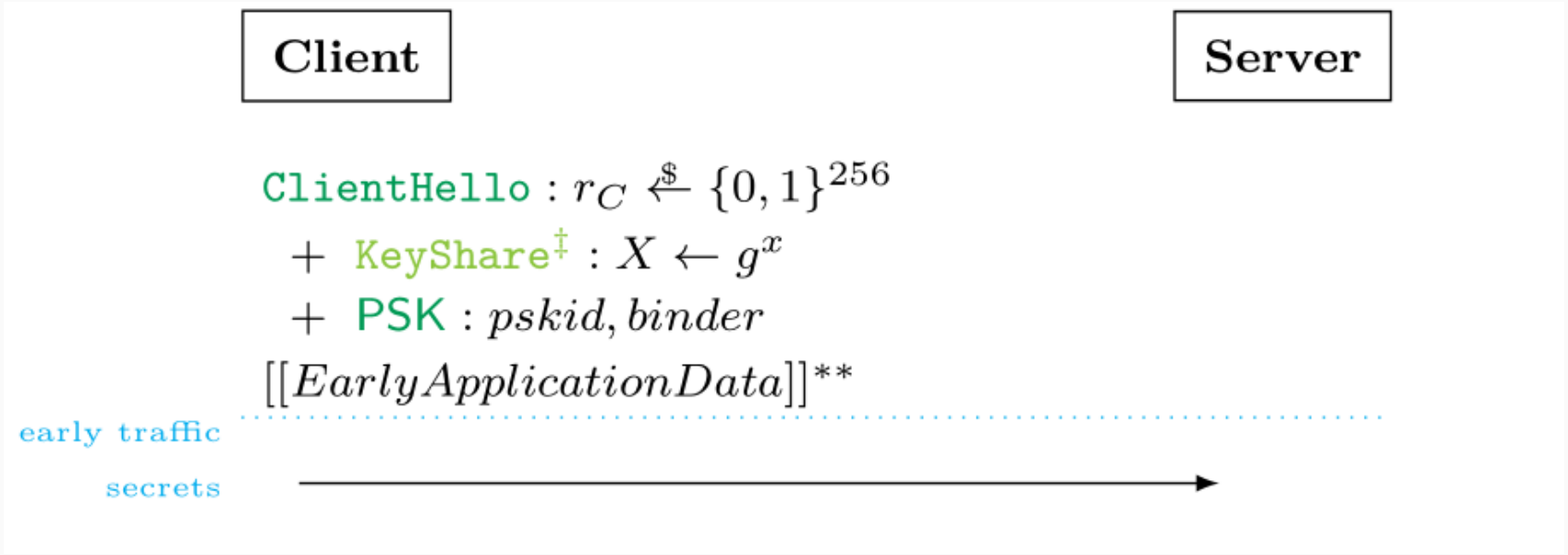
With TLS-level replay protection



Question 4.4: you can't win with TLS-level replay protection

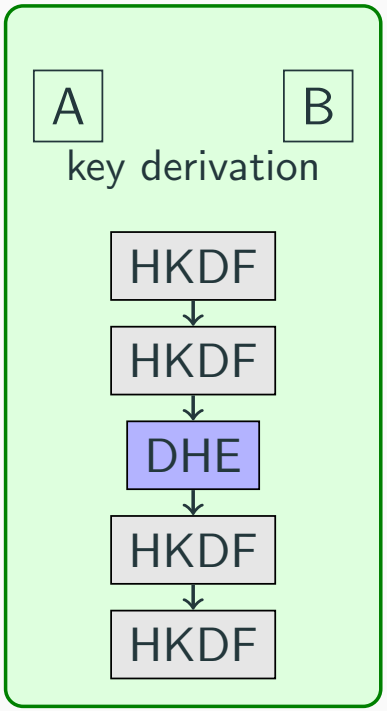


TLS 1.3: PSK Handshakes Early Data

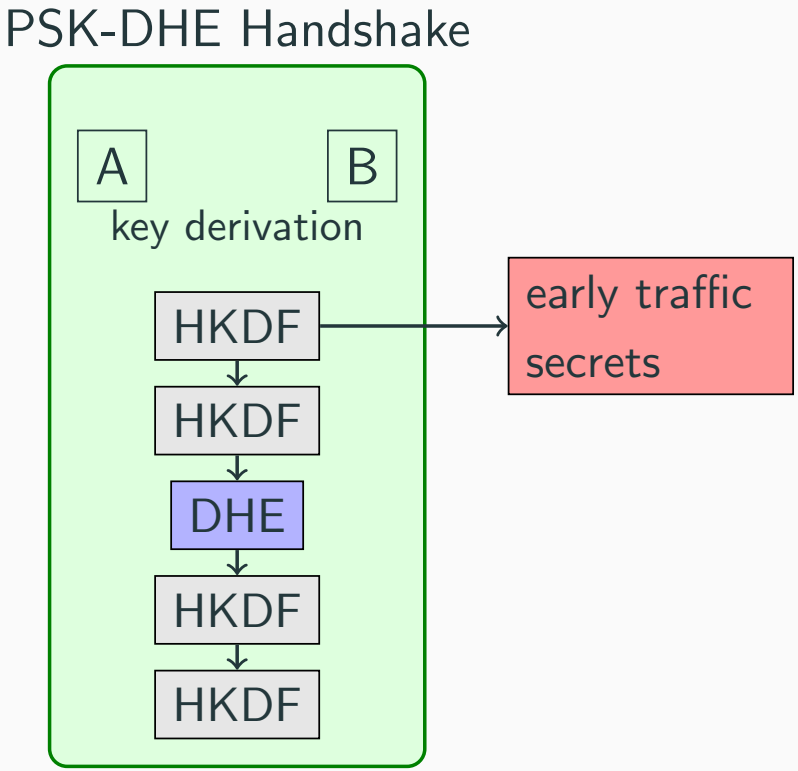


TLS 1.3: PSK Forward Secrecy

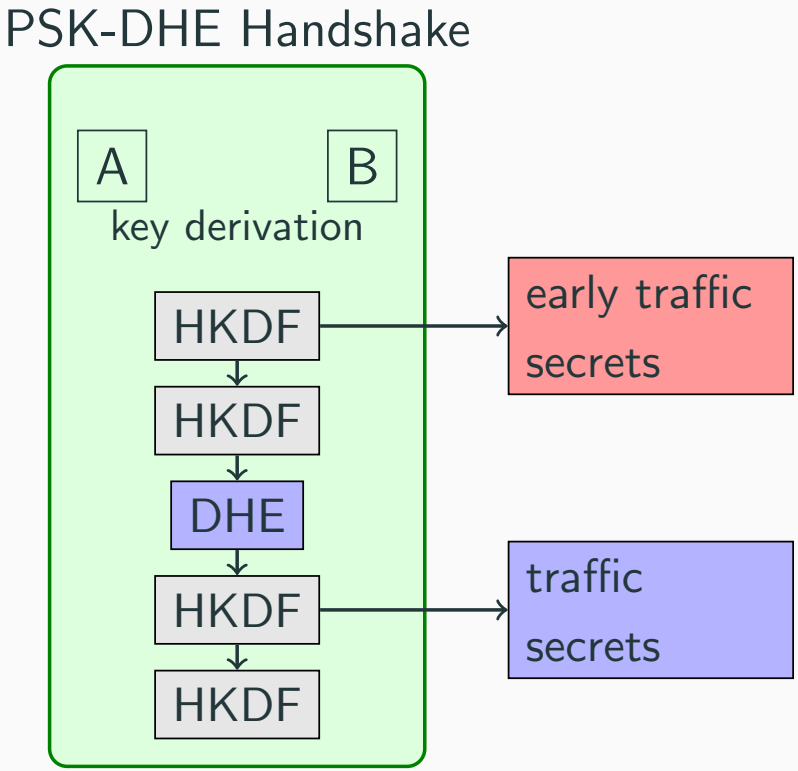
PSK-DHE Handshake



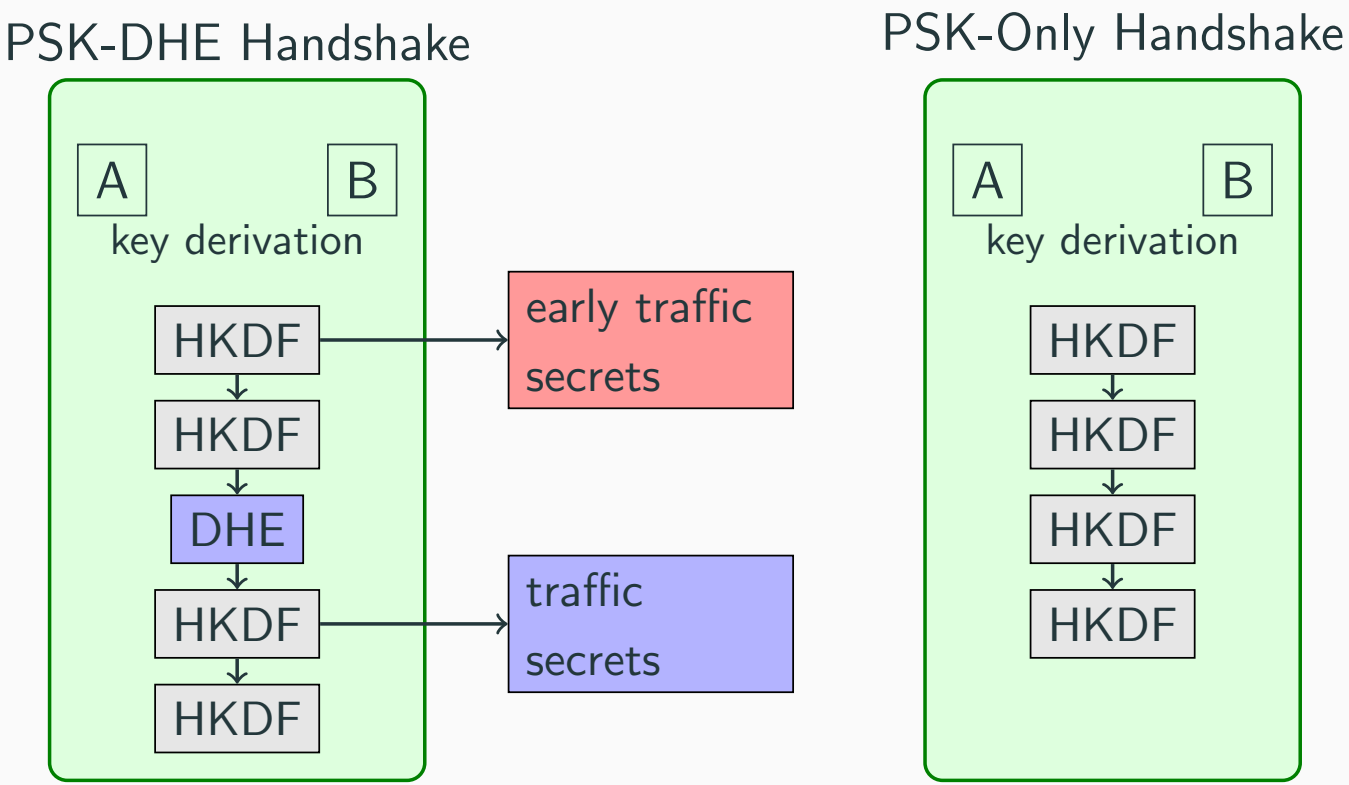
TLS 1.3: PSK Forward Secrecy



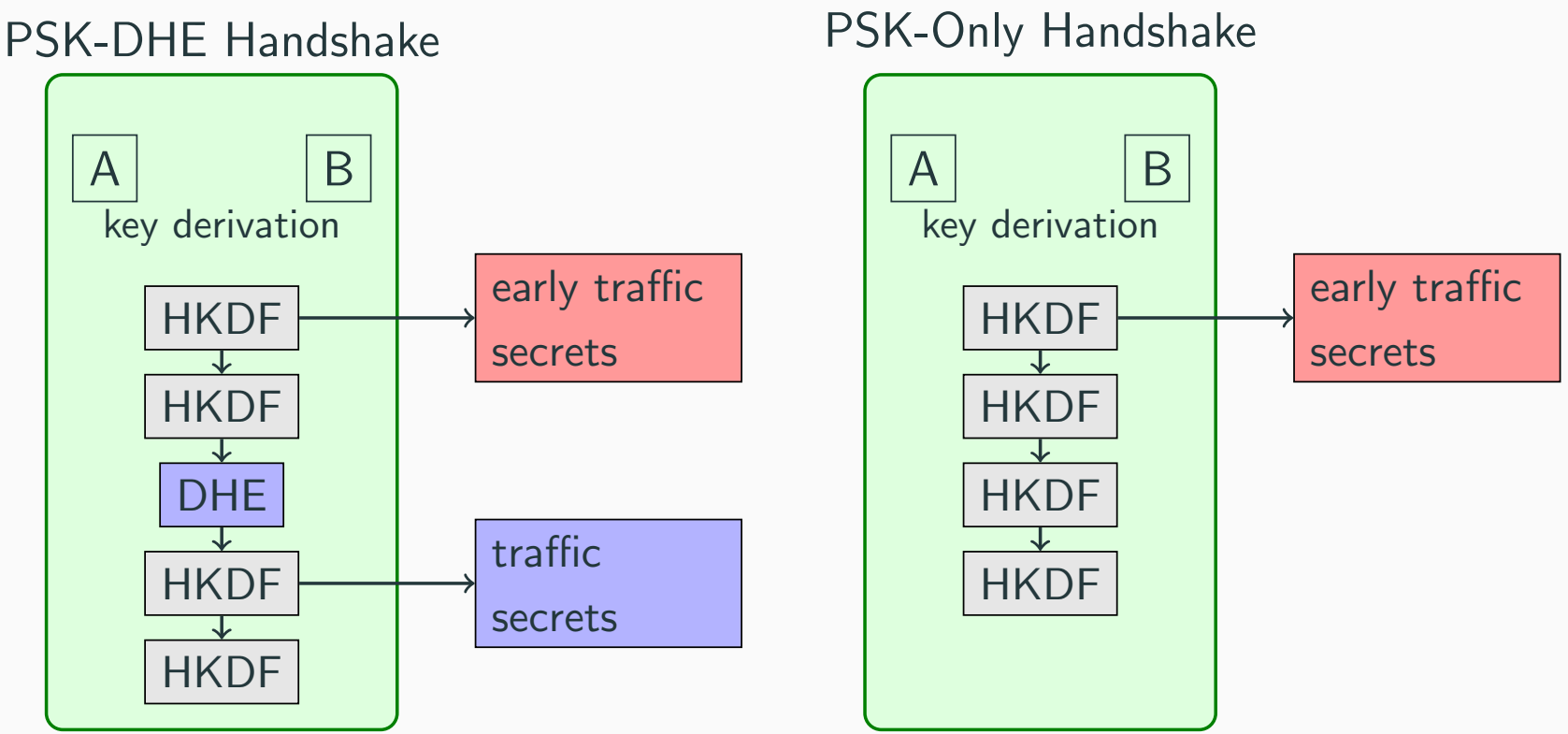
TLS 1.3: PSK Forward Secrecy



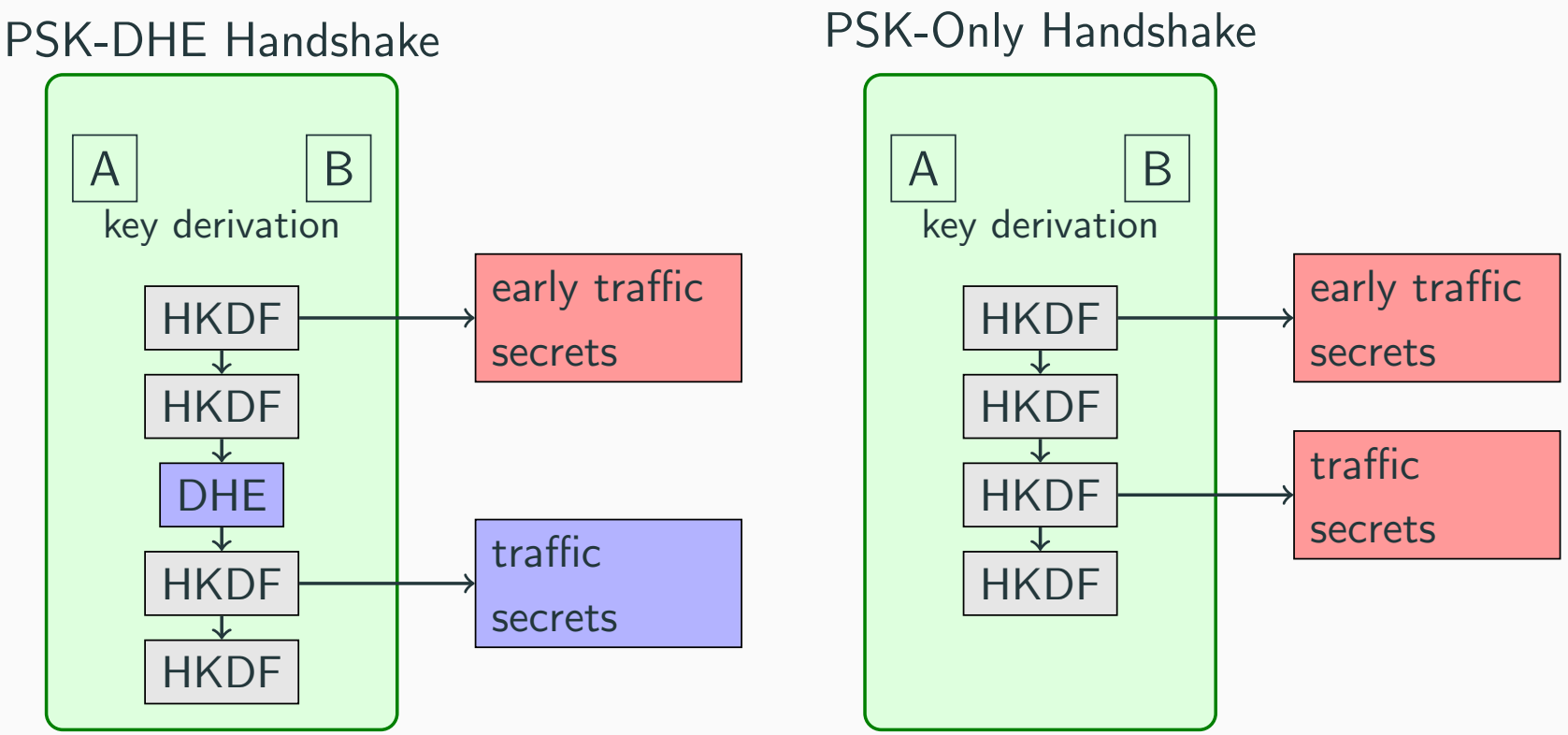
TLS 1.3: PSK Forward Secrecy



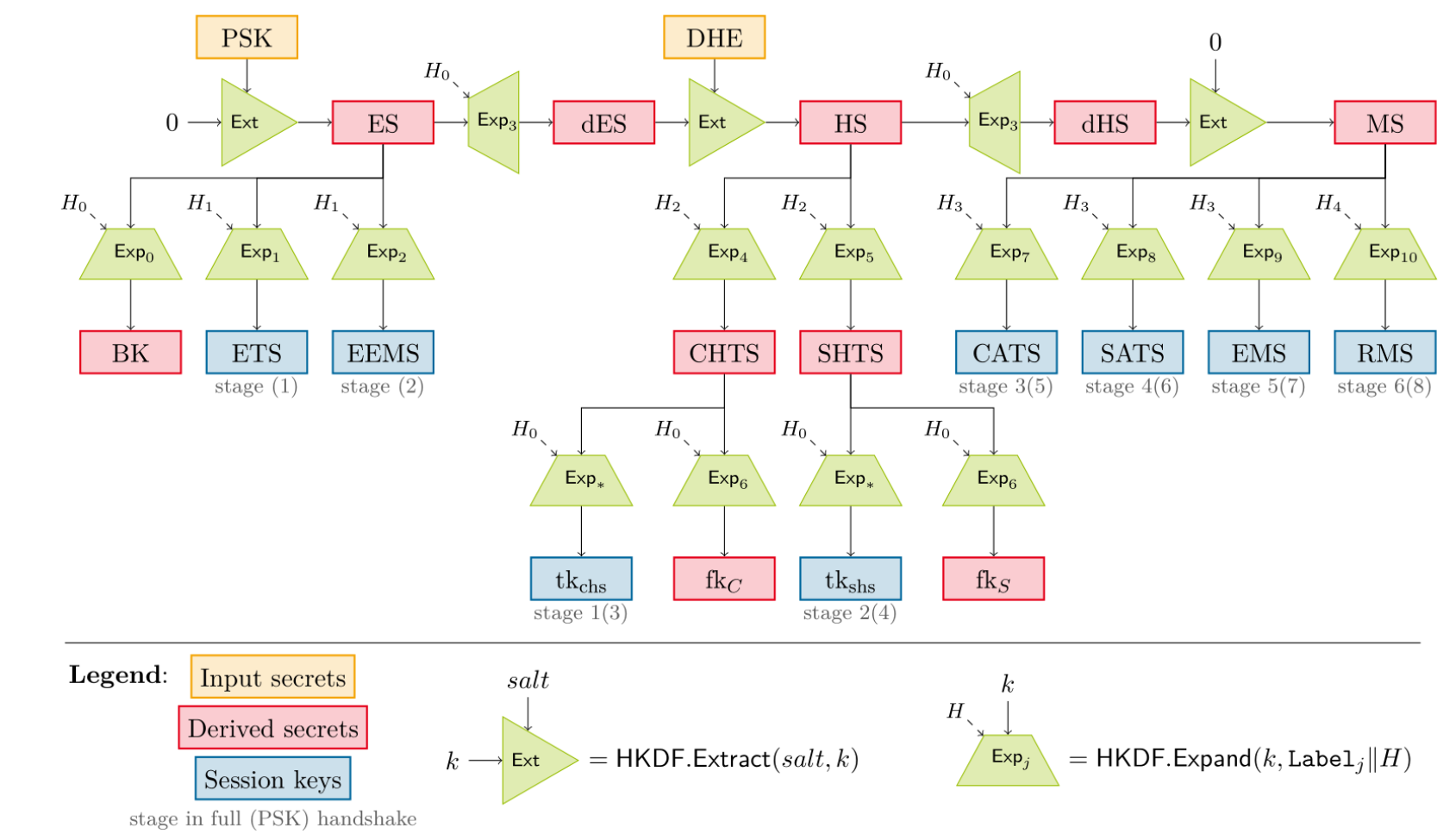
TLS 1.3: PSK Forward Secrecy



TLS 1.3: PSK Forward Secrecy

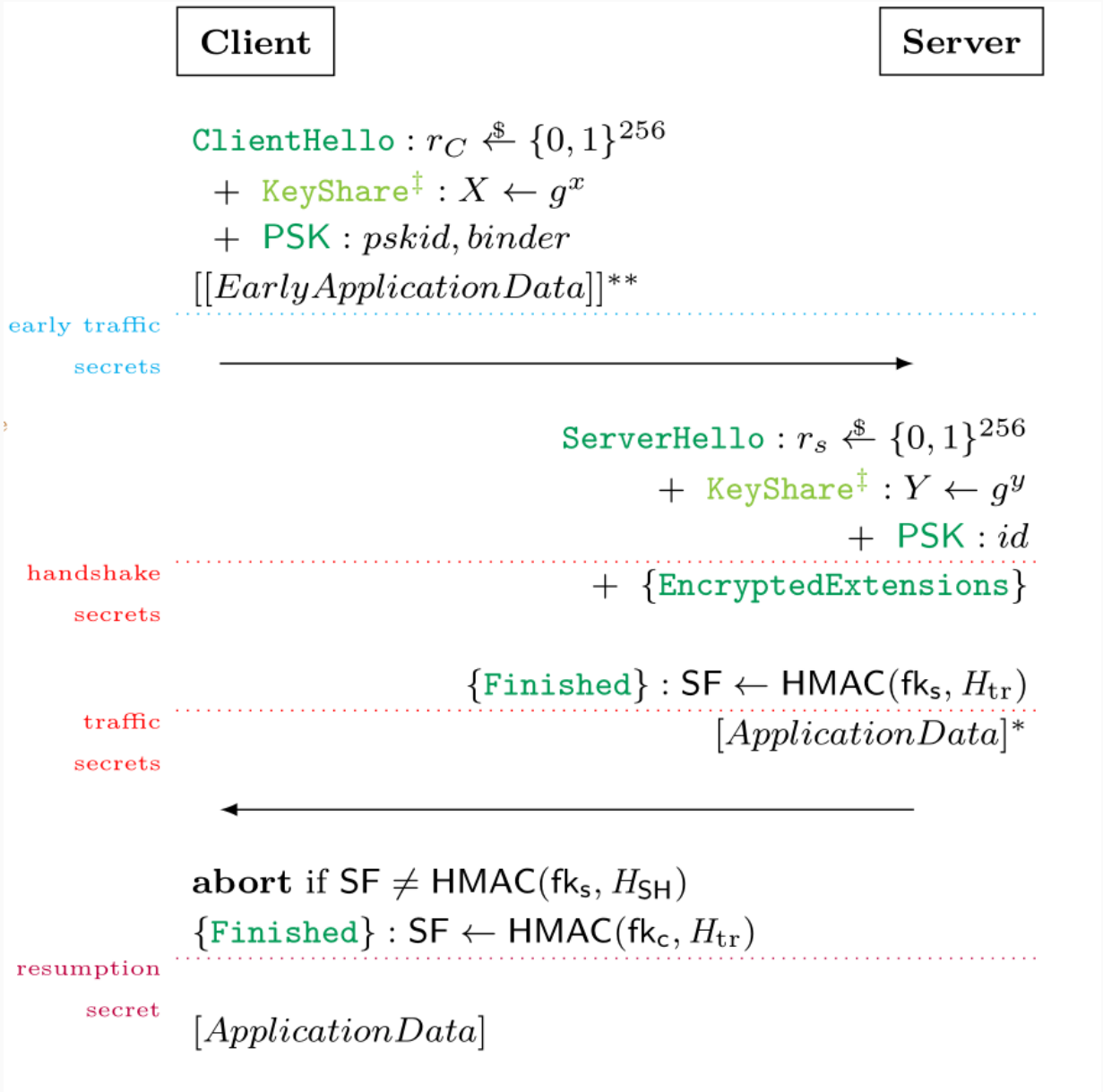


TLS 1.3: Key Schedule

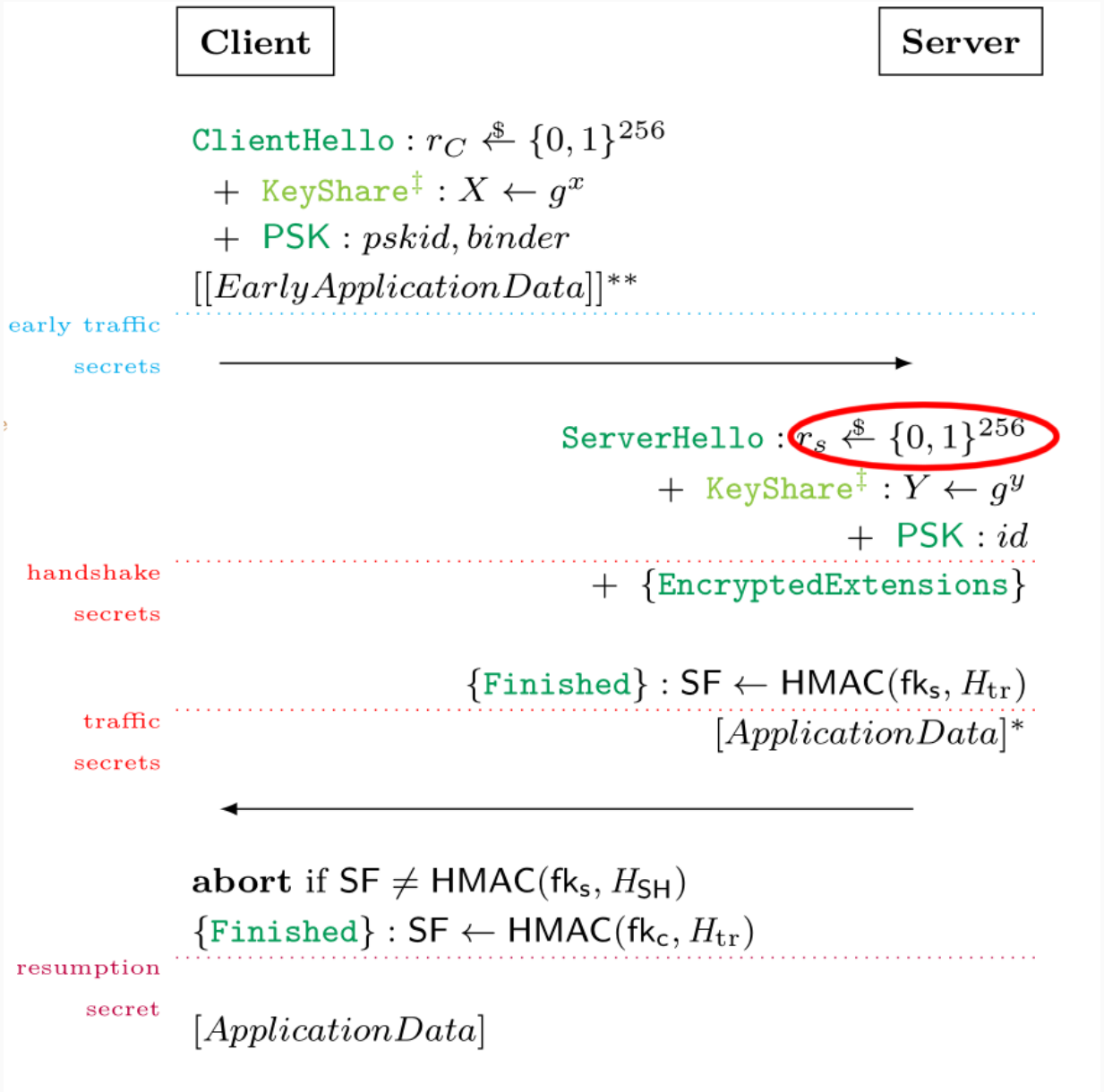


Dowling et al., `https://eprint.iacr.org/2020/1044`

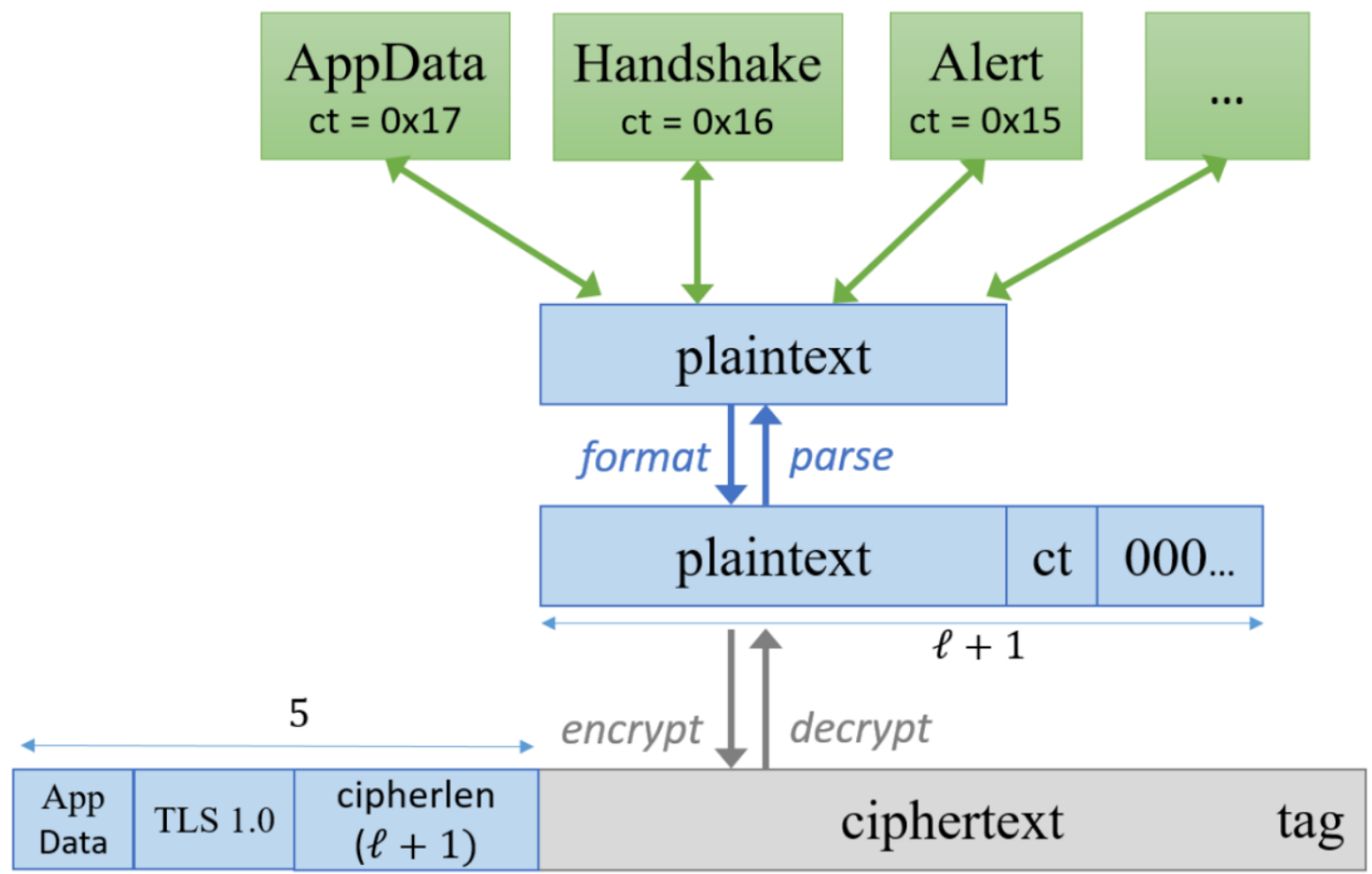
Can you replay a full TLS 1.3 handshake?



Can you replay a full TLS1.3 handshake? No!



TLS 1.3 Record Layer: Padding



Bhargavan et al., `https://eprint.iacr.org/2016/1178`

TLS 1.3 Record Layer: Padding

```
type len = n:nat {n ≤ 2^14} (* valid record length in TLS *)
type fragment (ℓ:len) = {ct:byte; data:bbytes ℓ}
val parse: ℓ:len → lbytes (ℓ+1) → Tot (option (fragment ℓ))
val format: ℓ:len → f:fragment ℓ → Tot (p:lbytes (ℓ+1))
  (ensures parse ℓ p = Some f)
```

These functions must be carefully implemented to prevent any side channel. We also construct and parse records into headers and payloads using functions

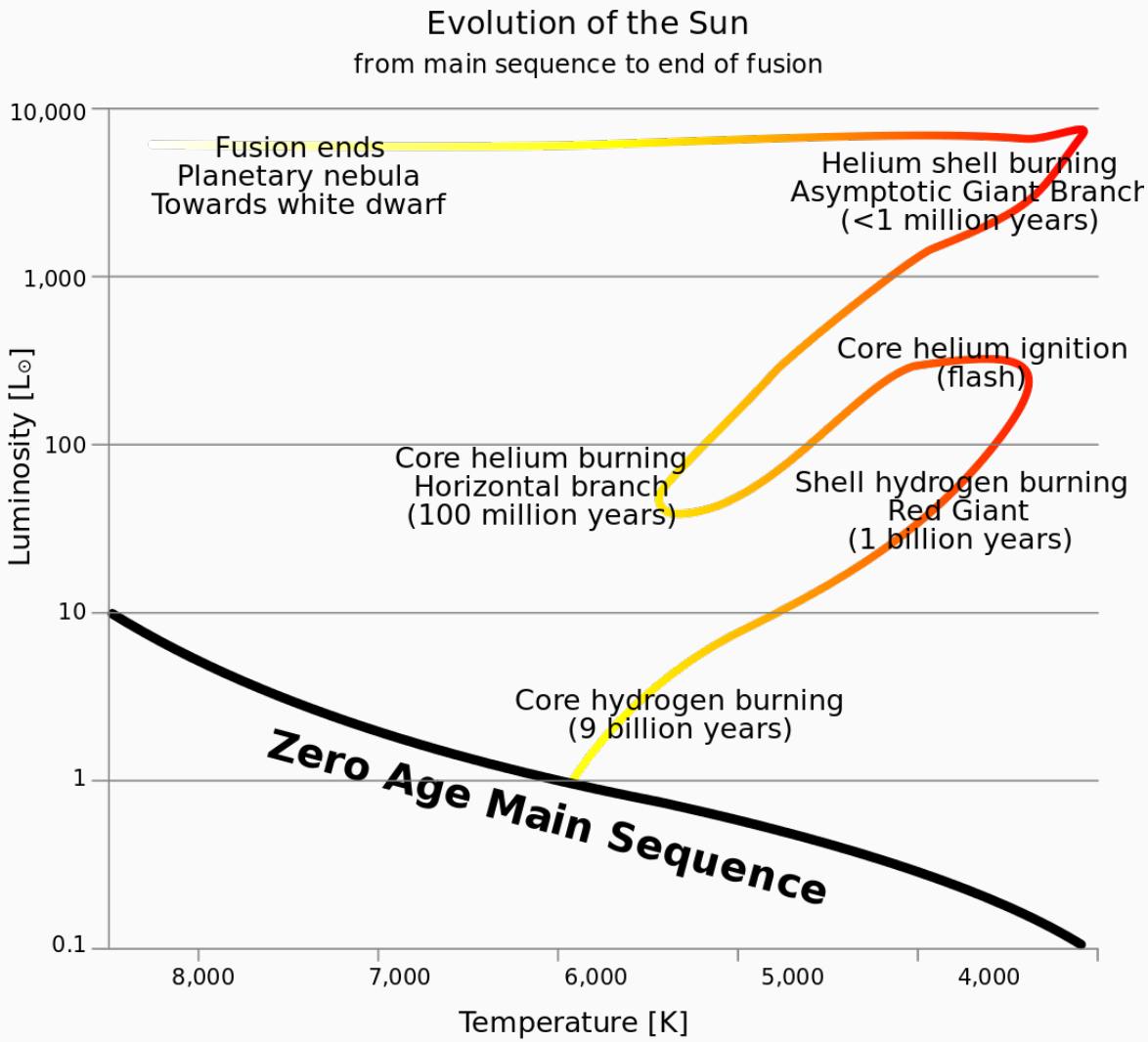
Bhargavan et al., `https://eprint.iacr.org/2016/1178`

TLS 1.3 Record Layer: Just wait

64 bit counter

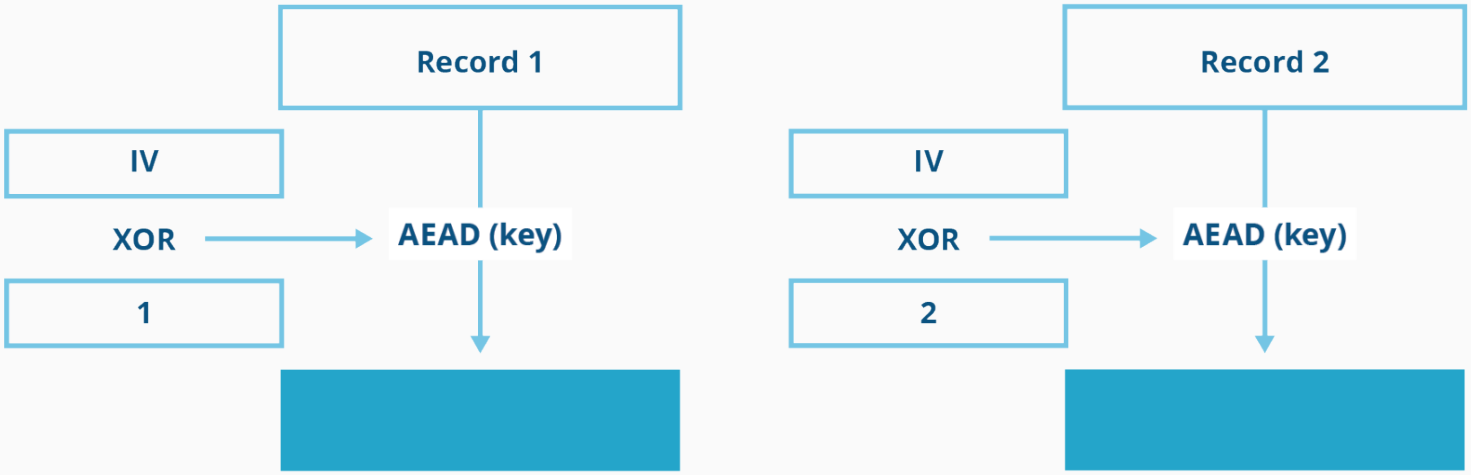
TLS 1.3 Record Layer: Just wait

64 bit counter



TLS 1.3 Record Layer: Reordering

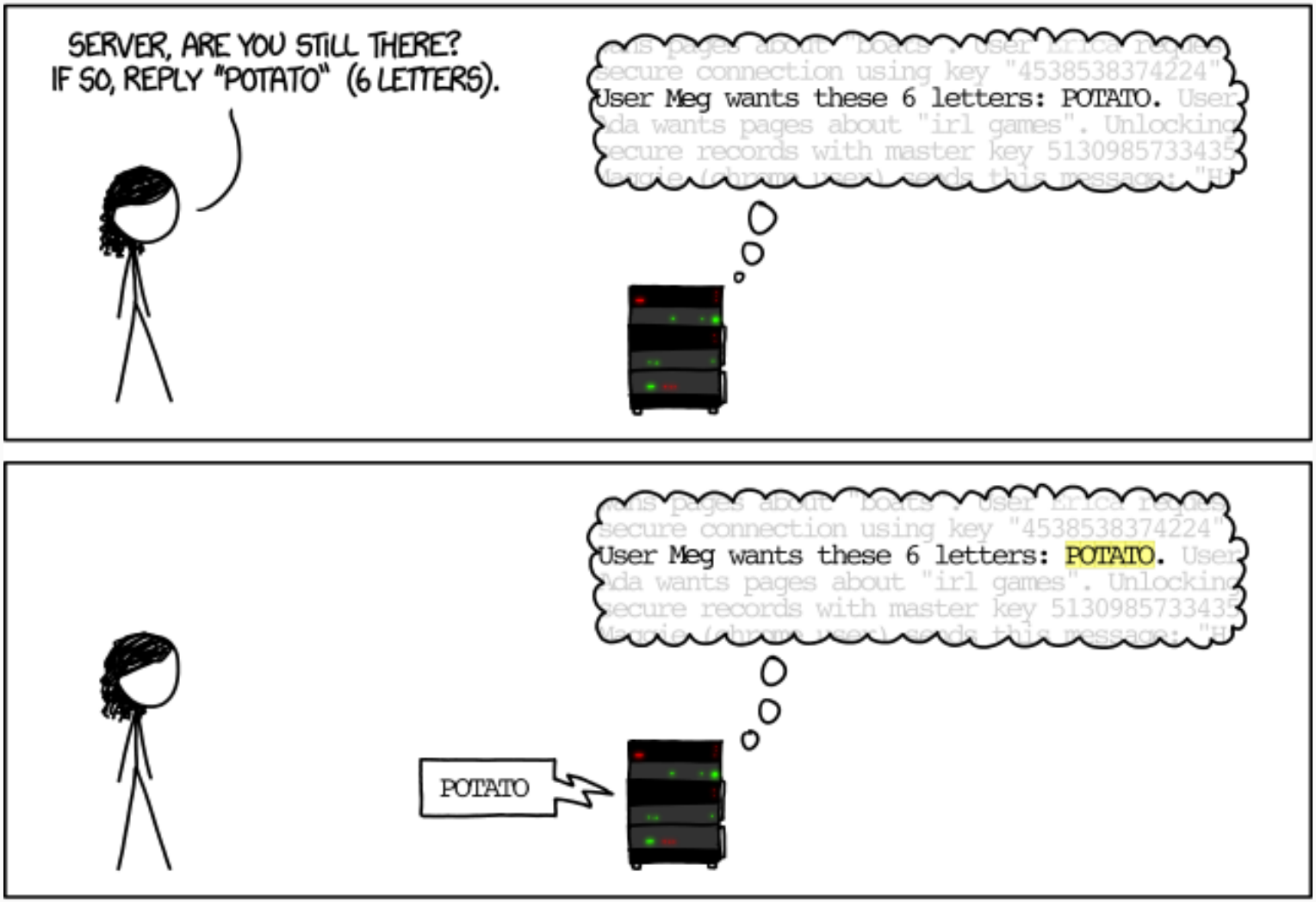
TLS 1.3 Record Layer: Reordering



CLOUDFLARE

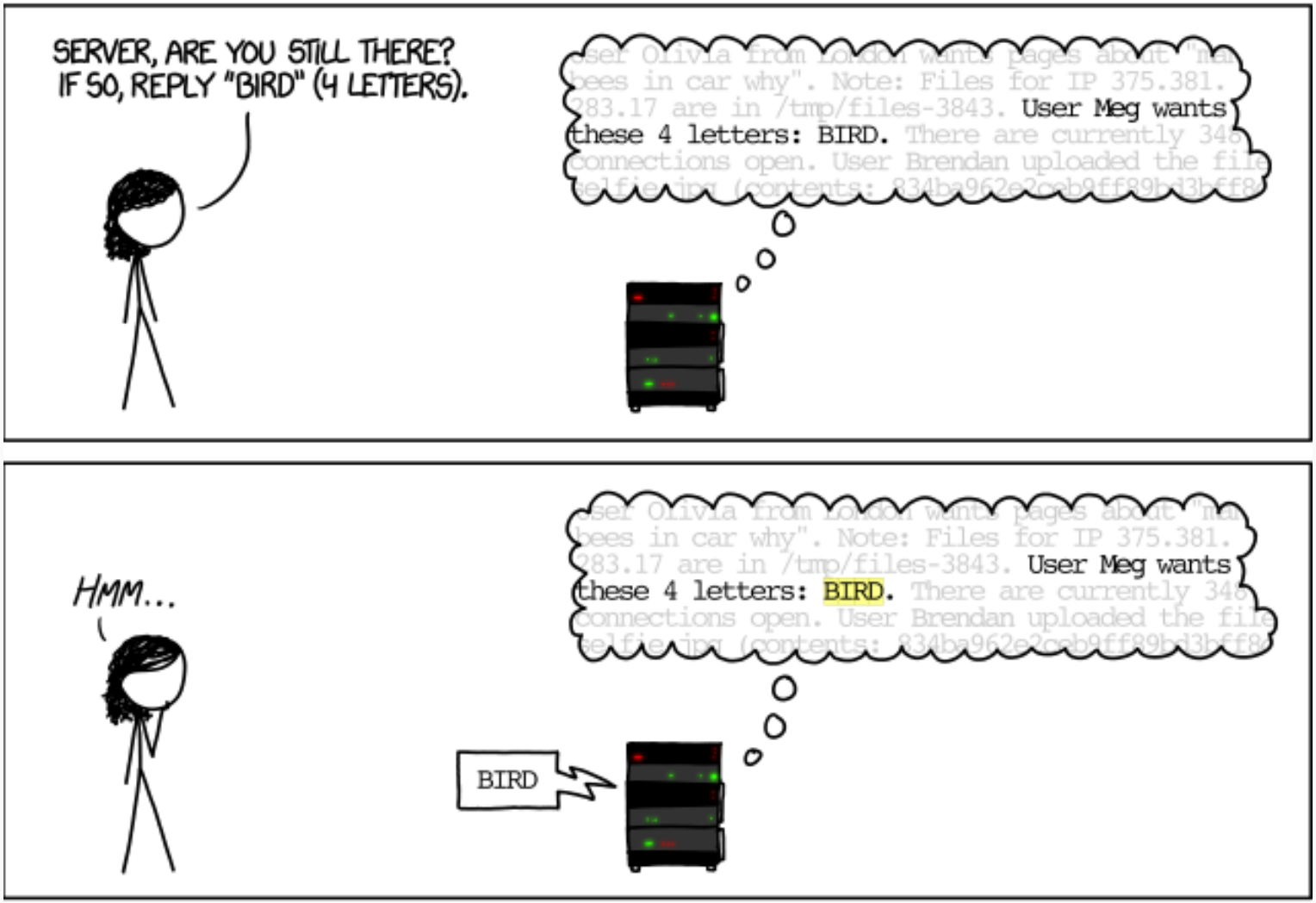
https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/

Heartbleed Explanation



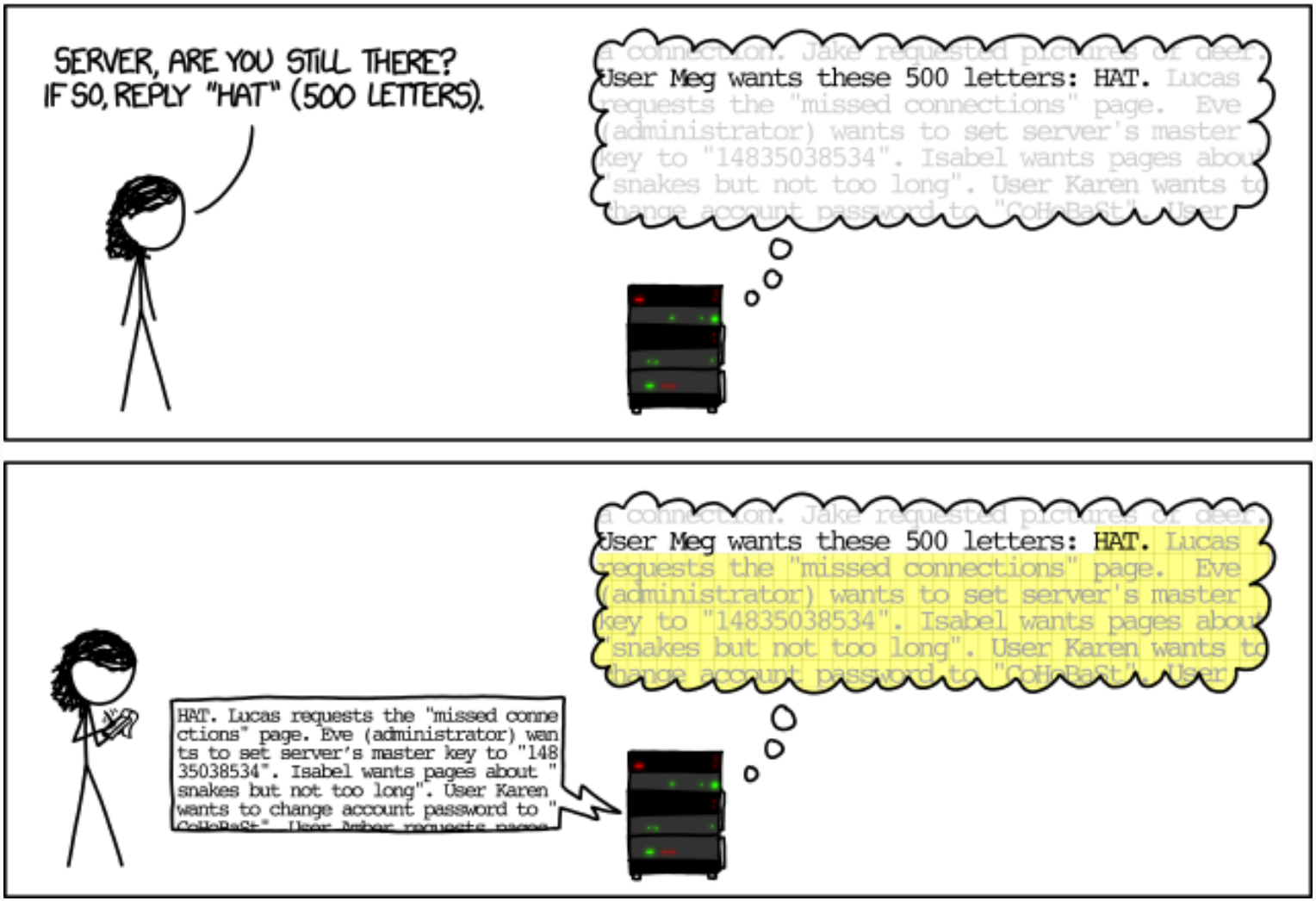
https://xkcd.com/1354/

Heartbleed Explanation



https://xkcd.com/1354/

Heartbleed Explanation



https://xkcd.com/1354/

Heartbleed

Need auth?

- Not really (TLS 1.2): heartbeat possible during handshake, pre-auth

Heartbleed

Need auth?

- Not really (TLS 1.2): heartbeat possible during handshake, pre-auth

Heartbleed

Need auth?

- Not really (TLS 1.2): heartbeat possible during handshake, pre-auth

Will PFS save us?

- Of course!

Heartbleed

Need auth?

- Not really (TLS 1.2): heartbeat possible during handshake, pre-auth

Will PFS save us?

- Of course!
- (but only for past sessions!)

Goto fail

```
static OSStatus SSLVerifySignedServerKeyExchange (SSLContext *ctx, bool isRsa,
        SSLBuffer signedParams, uint8_t *signature, UInt16 signatureLen) {

    OSStatus err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
            goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
            goto fail;
            goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
            goto fail;
    ...
    err = sslRawVerify(...);

fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
}
```

GNUTLS fail

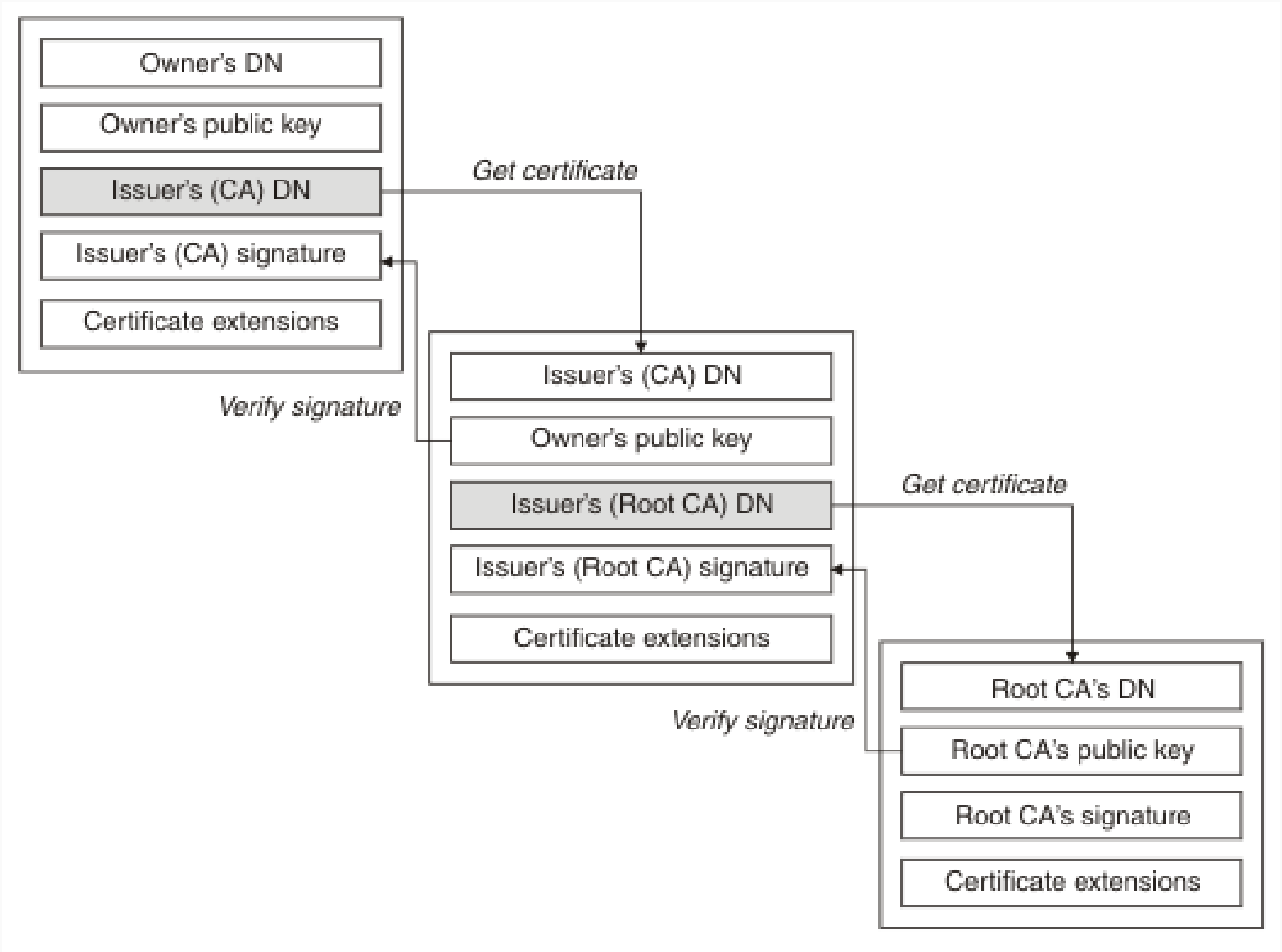
```
static OSStatus SSLVerifySignedServerKeyExchange (SSLContext *ctx, bool isRsa,
        SSLBuffer signedParams, uint8_t *signature, UInt16 signatureLen) {

    OSStatus err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
            goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
            goto fail;
            goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
            goto fail;
    ...
    err = sslRawVerify(...);

fail:
        SSLFreeBuffer(&signedHashes);
        SSLFreeBuffer(&hashCtx);
        return err;
}
```

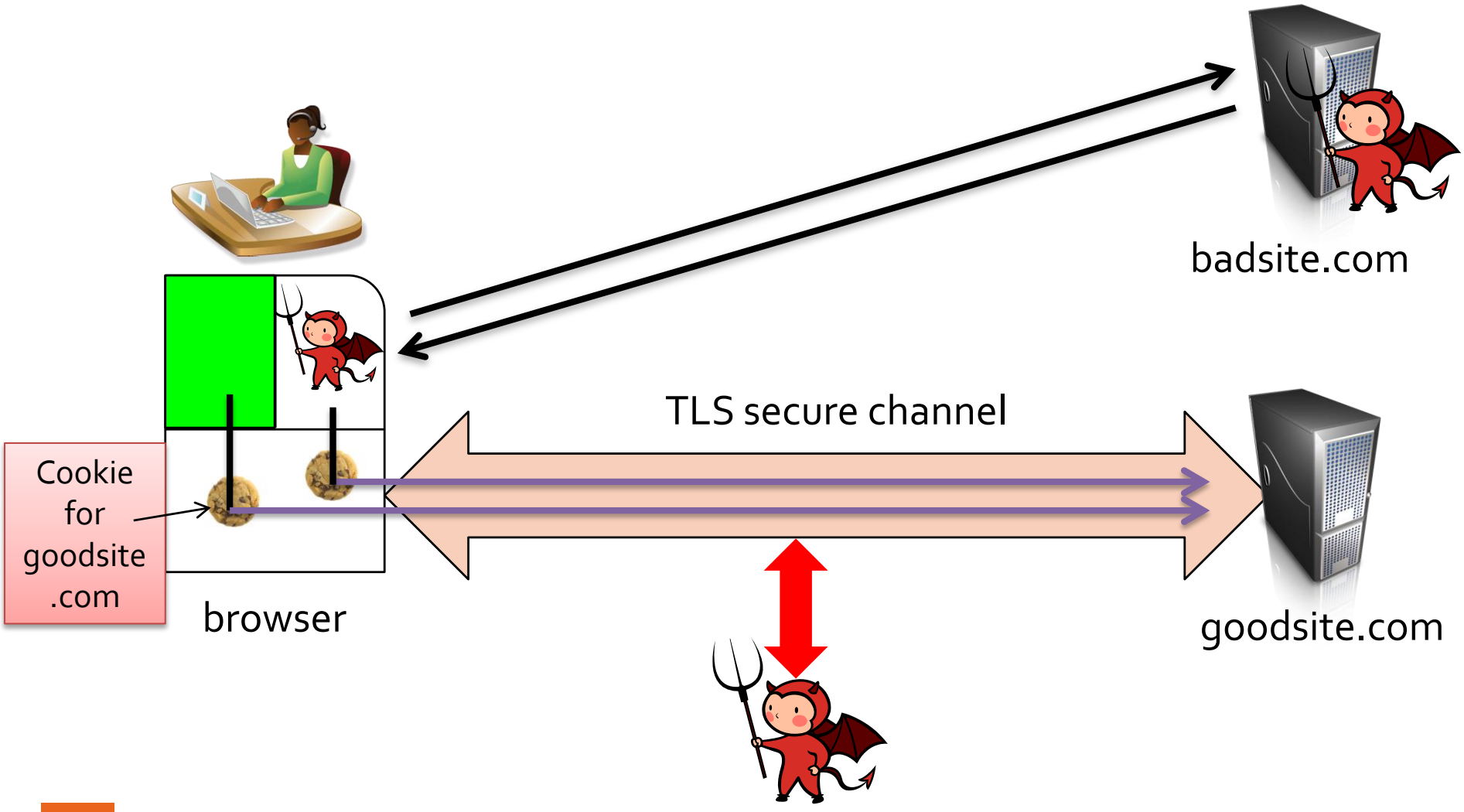
Certificate Chain



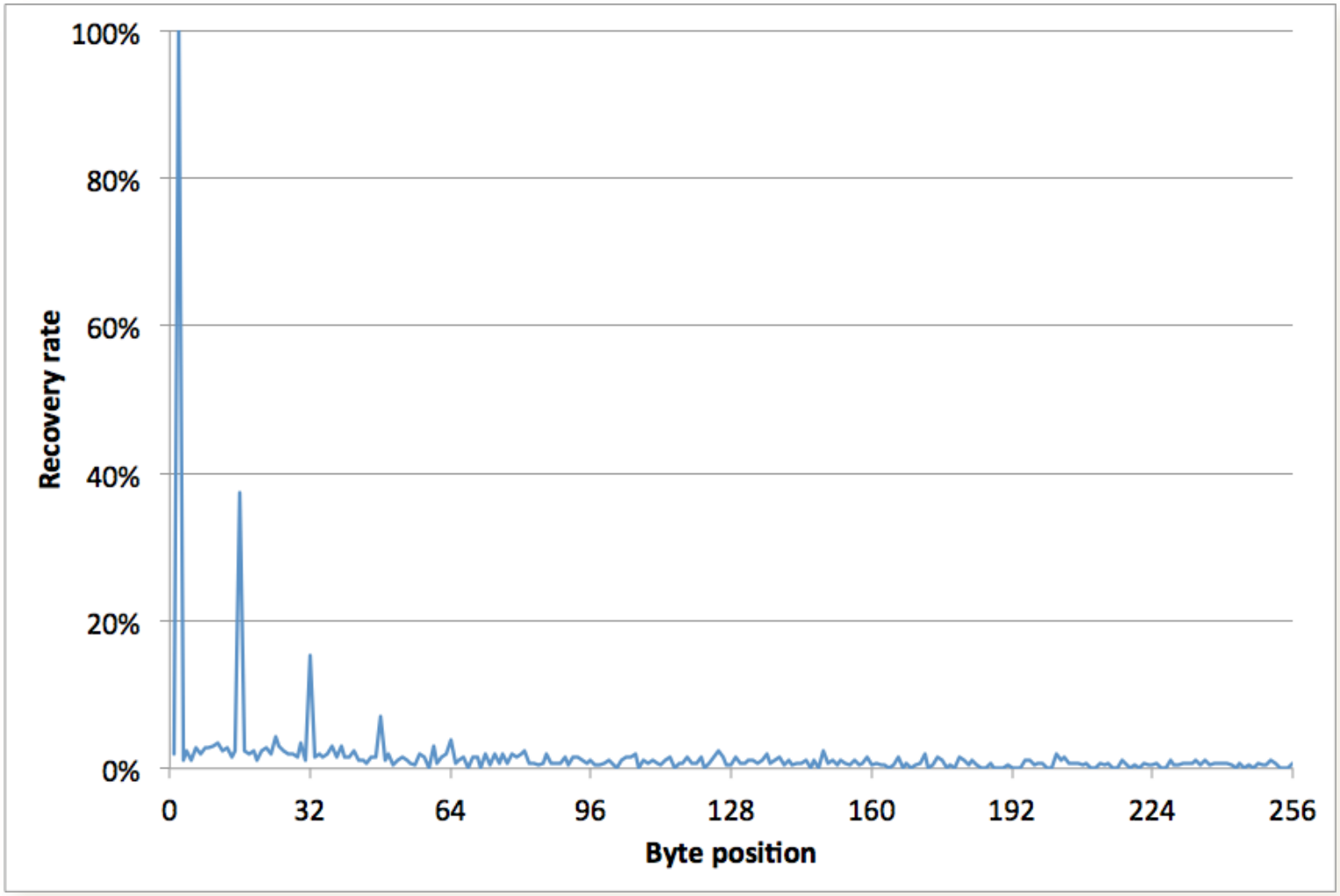
https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10600_.htm

Slides from Kenny Paterson

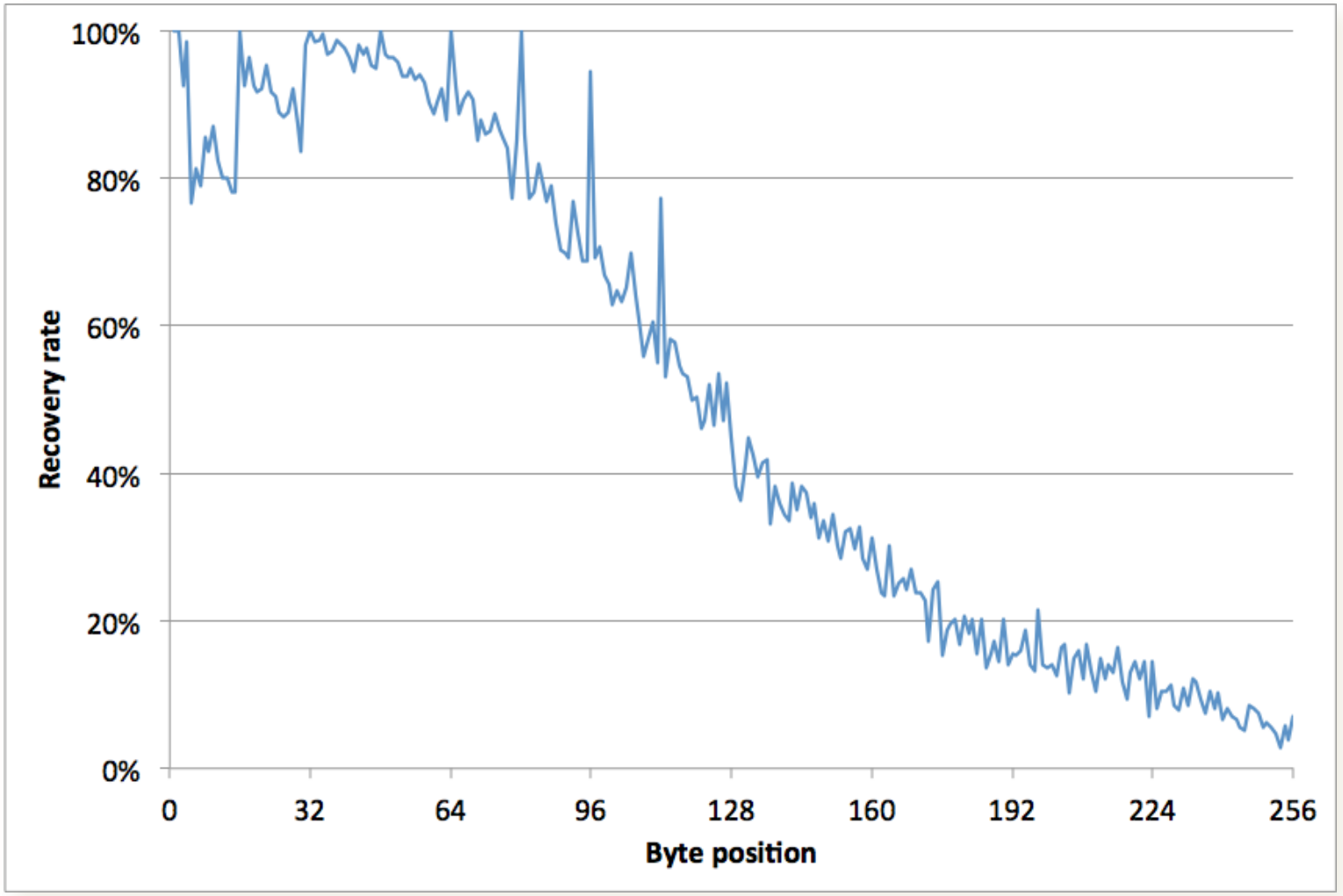
How the Web Works



Success Probability 2^{20} Connections



Success Probability 2^{27} Connections



References

- On the Security of RC4 in TLS, USENIX '13, ABPPS
- A Messy State of the Union: Taming the Composite State Machines of TLS, BBDFKPSZ
- RFC8446, TLS-1.3 specification