

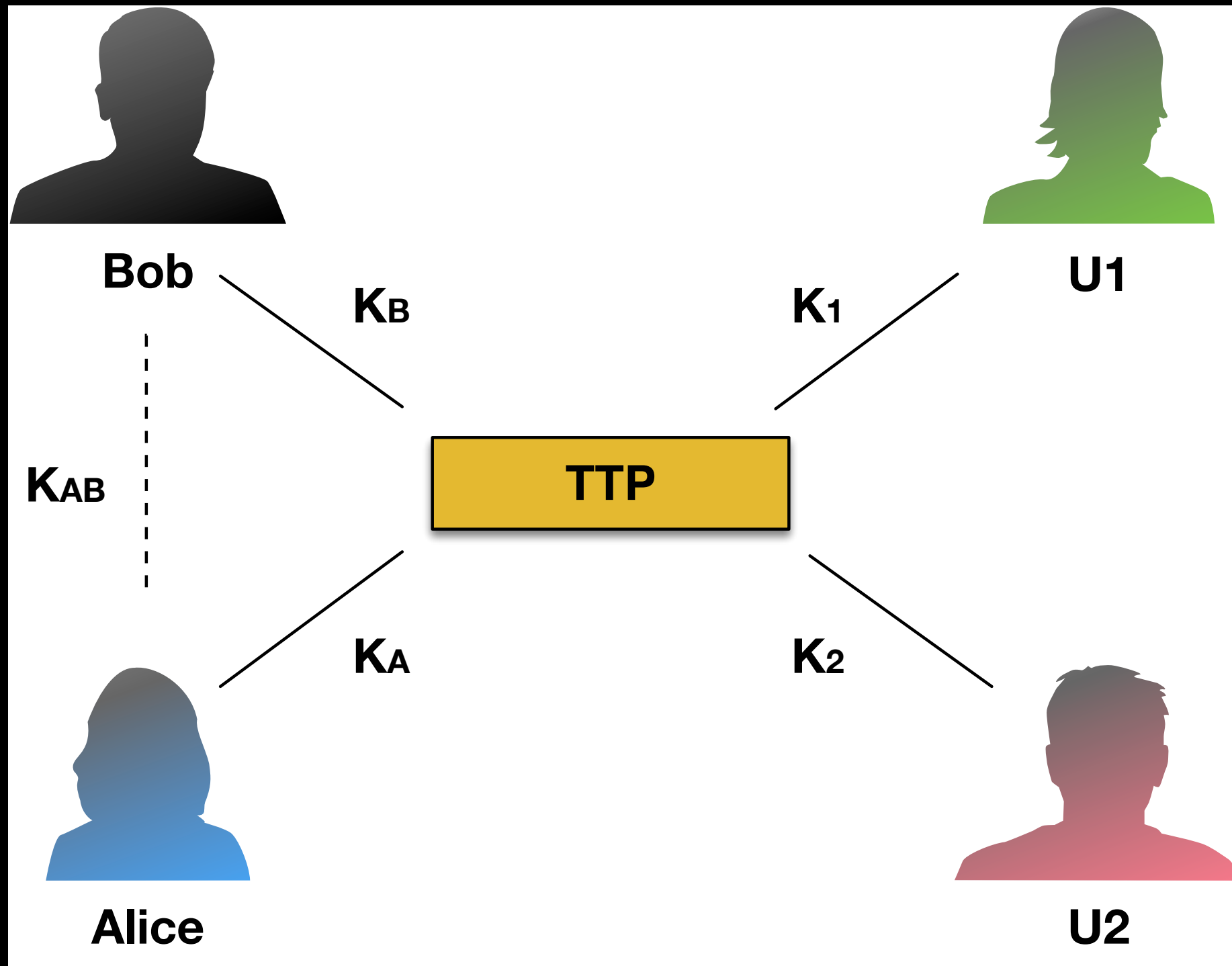
TLS and HTTPS

Dimitris Mitropoulos
dimitro@di.uoa.gr

Primitives

- Συμμετρική Κρυπτογραφία:
$$\mathbf{C = E(K, P), P = D(K, C)}$$
- Κρυπτογραφία Δημοσίου Κλειδιού:
$$\mathbf{C = E(PK, P), P = D(SK, C)}$$
- Ψηφιακές Υπογραφές:
$$\mathbf{S = Sign(SK, M), Verify(PK, M, S) = 'yes' ?}$$

Θυμηθείτε (TTP)



TTP

Δημιουργία Κλειδιού

Bob (K_B)

Alice (K_A)

TTP

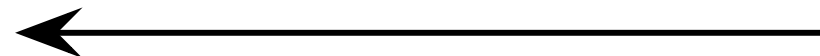
“Η Alice θέλει να
επικοινωνήσει με
τον Bob”

επιλογή
ενός K_{AB}

$E(K_A, \text{“A, B”} \parallel K_{AB})$

ticket = $E(K_B, \text{“A, B”} \parallel K_{AB})$

ticket



θα μπορούσε αυτή η
προσέγγιση να
εφαρμοστεί στο διαδίκτυο;

Προβλήματα που θα υπήρχαν:

1. Θα πρέπει να εμπιστεύονται όλοι το TTP.
2. Δύσκολη διαχείριση των κλειδιών.
3. Όλοι πρέπει να έχουν κλειδί και να υπάρχουν στην βάση του TTP.
4. Κακή κλιμάκωση.
5. Θα πρέπει πάντα να είναι online.

Λύση (;)

Χρησιμοποίηση κρυπτογράφησης δημοσίου κλειδιού
(λ.χ. ξέρω όλα τα public keys των file server του DI);

Alice

Bob

$S \leftarrow \text{rand}()$

$E(PK_B, S)$

$E(S, m)$

Τι πιθανά προβλήματα μπορεί να υπάρχουν εδώ;

Replay Attack (;)

Ο Bob όμως μπορεί να δημιουργήσει μια τυχαία τιμή (nonce). Έτσι θα έχουμε:

$$S' \leftarrow H(S \parallel \text{nonce})$$

Ταυτοποίηση (;)

Η Alice γνωρίζει τον Bob, αλλά το αντίθετο δεν ισχύει.

Λύση

Alice

Bob

$S \leftarrow \text{rand}()$

$\text{Sign}(\text{SK}_A, \text{E}(\text{PK}_B, S))$



που βρίσκονται τα
public keys;

Certificate Authority

(CA - Αρχή Πιστοποίησης)

| Principle | Public Key |
|-----------|------------|
| name_a | key_1ze34 |
| ... | ... |

έχουμε πολλά “TTPs”
πλεον που μπορούμε να
εμπιστευθούμε

CA

Χρειάζεται να είναι πάντα online;

Πιστοποιητικό (Certificate)

Sign (SK_{CA}, { name_1, key_1ze34 })

Certificate

Κλιμάκωση

Χρειάζεται ο client (browser) να έχει certificate;

Chicken and Egg Problem

Που βρίσκονται τα public keys των CAs;

Hardcoded μέσα σε εφαρμογές (λ.χ. browsers).

Certificate

Προβλήματα

- Τι θα συμβεί εάν μια CA εκδόσει ένα certificate για ένα λάθος όνομα;
- Τι θα γίνει εάν χάσει ή δημοσιοποιήσει κατα λάθος το SK του ένα website που έχει certificate από μια αρχή;
- Τι θα συμβεί αν το SK μιας CA βρεθεί στα χέρια ενός αντιπάλου;

Certificate

Ανάκληση

Δυο (+1) εναλλακτικές:

1. Certificate Revocation List (CRL): Λίστα με λάθη.
 - Περιοδικά, οι εφαρμογές θα πρέπει να ελέγχουν τις λίστες αυτές.
 - Οι περισσότερες CA δίνουν άδειες λίστες (!).
2. Online Certificate Status Protocol (OCSP): Μας πηγαίνει σε μια λύση κοντά στην αρχική ιδέα **ενός** TTP.

3: (πάλι) hardcoded μέσα σε εφαρμογές (λ.χ. browsers).

Στον browser

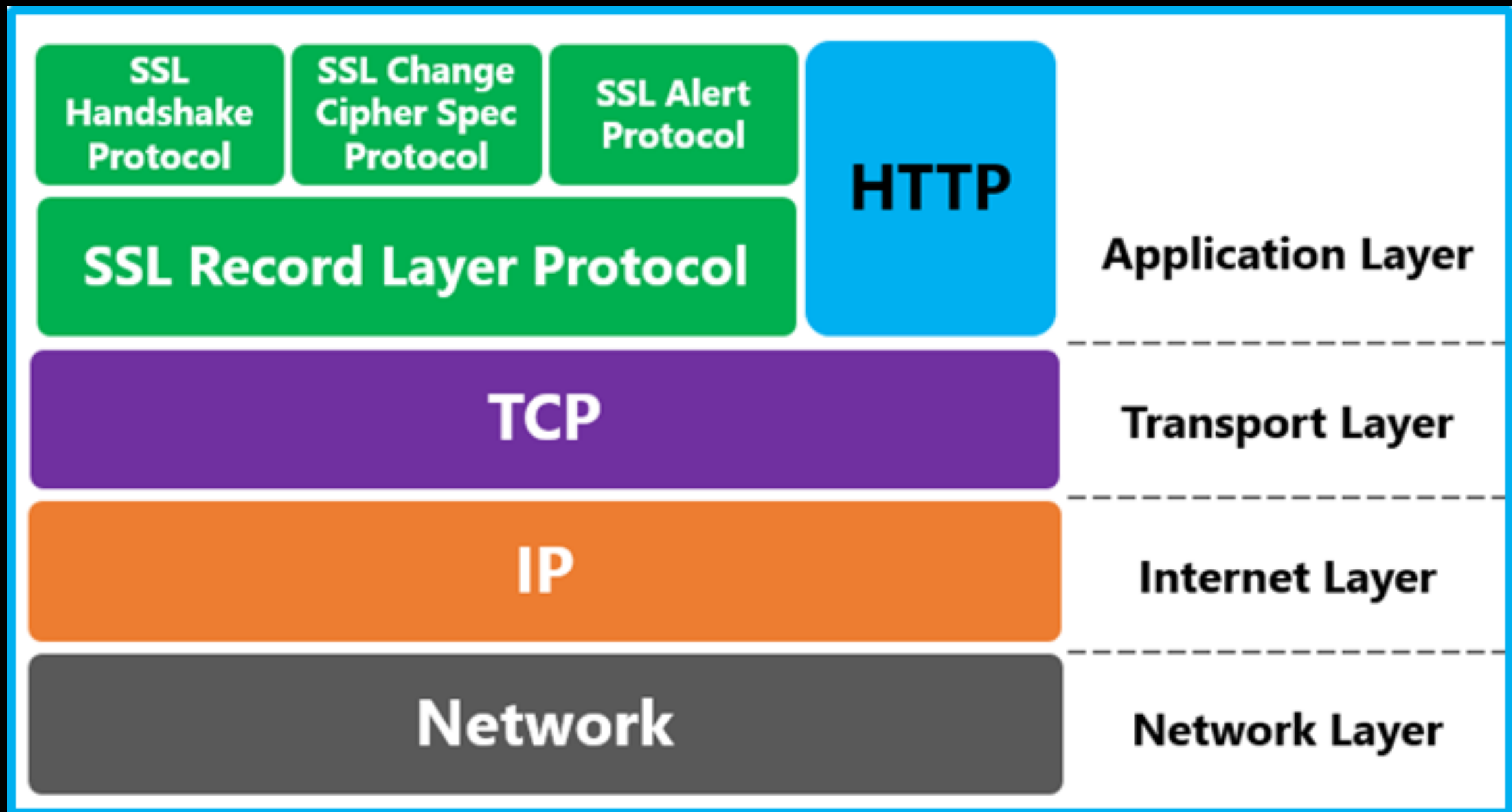
Τι θα πρέπει να προσέξουμε:

1. Δεδομένα που μεταφέρονται μέσω δικτύου (**TLS**).
2. Κώδικας, δεδομένα στον browser (**HTTPS**).
3. Το user interface (...).

Secure Sockets Layer

- Ο προκάτοχός του TLS, το **SSL**, αναπτύχθηκε από την Netscape.
- Το 1996 κυκλοφόρησε η 3η και τελευταία έκδοση του SSL.
- Αντικαταστάθηκε από το TLS το 1999.

SSL (layering)



Transport Layer Security (TLS)

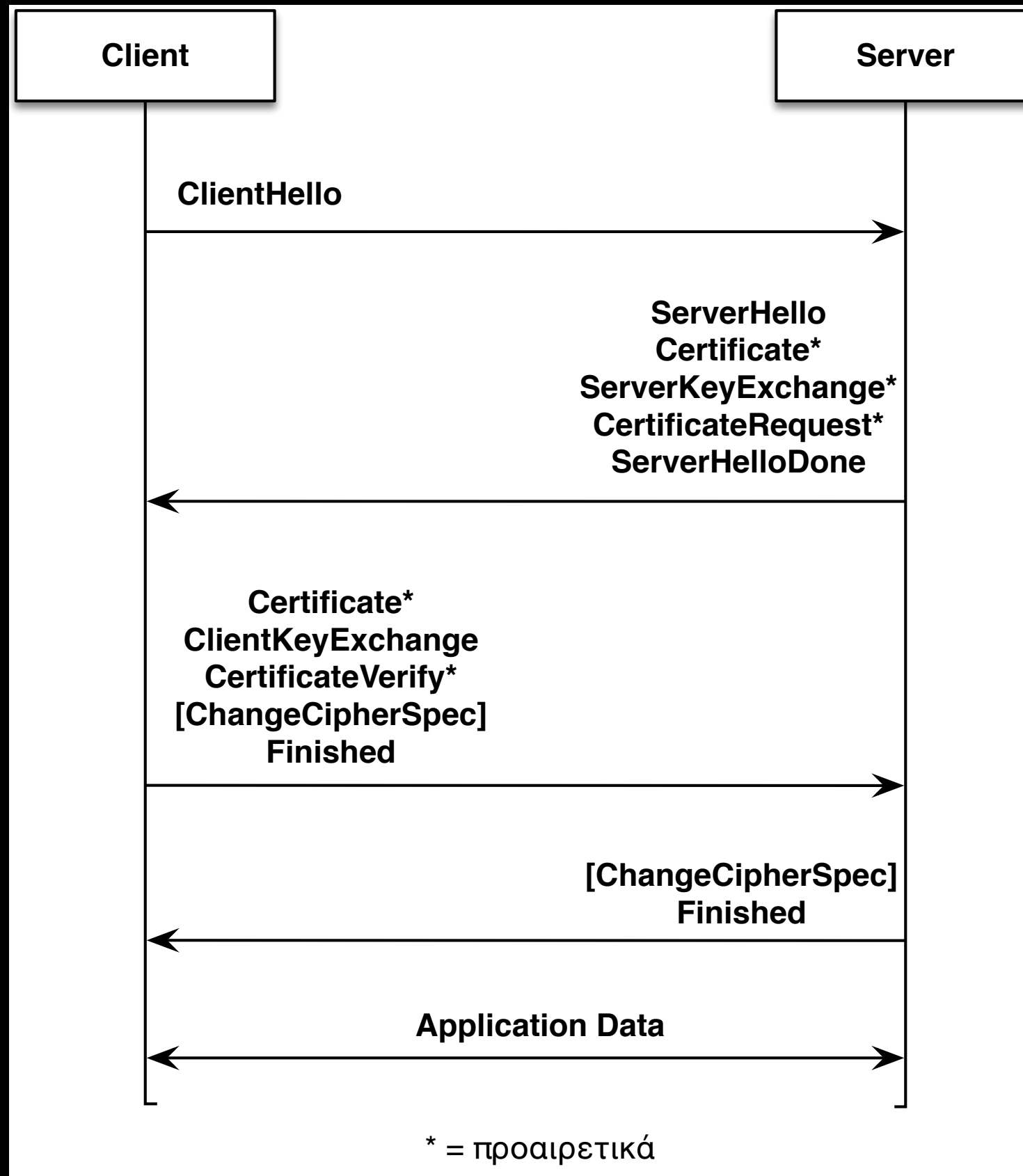
- **Handshake Protocol:** Ταυτοποίηση και εγκαθίδρυση του κλειδιού της συνόδου.
- **Record Protocol:** Ακεραιότητα των δεδομένων.

TLS

Handshake Protocol

- Ταυτοποίηση οντοτήτων (προαιρετική). Συνήθως ο server θα **πρέπει** να ταυτοποιηθεί.
- Το **συμμετρικό** κλειδί της συνόδου (session key) δημιουργείται χρησιμοποιώντας κρυπτογράφηση δημοσίου κλειδιού.

Handshake Protocol



ClientHello

```
struct {  
    ProtocolVersion client_version;  
    Random random;  
    SessionID session_id;  
    CipherSuite cipher_suites<2..2^16-1>;  
    CompressionMethod compression_methods<1..2^8-1>;  
    select (extensions_present) {  
        case false:  
            struct {};  
        case true:  
            Extension extensions<0..2^16-1>;  
    };  
} ClientHello;
```

γιατί τα extensions είναι bold;

ClientHello

παράδειγμα

ClientVersion 3,1

ClientRandom [32]

SessionID: None (new session)

Suggested Cipher Suites:

TLS_RSA_WITH_3DES_EDE_CBC_SHA

TLS_RSA_WITH_DES_CBC_SHA

Suggested Compression Algorithm: NONE

Extensions: NONE

ServerHello

```
struct {  
    ProtocolVersion client_version;  
    Random random;  
    SessionID session_id;  
    CipherSuite cipher_suite;  
    CompressionMethod compression_method;  
    select (extensions_present) {  
        case false:  
            struct {};  
        case true:  
            Extension extensions<0..2^16-1>;  
    };  
} ServerHello;
```

ServerHello

παράδειγμα

Version 3,1

ClientRandom [32]

SessionID: bd608869f0c629767ea7e3e

Suggested Cipher Suites:

TLS_RSA_WITH_3DES_EDE_CBC_SHA

Suggested Compression Algorithm: NONE

Extensions: NONE

(λαμβάνει υπόψη το ClientHello)

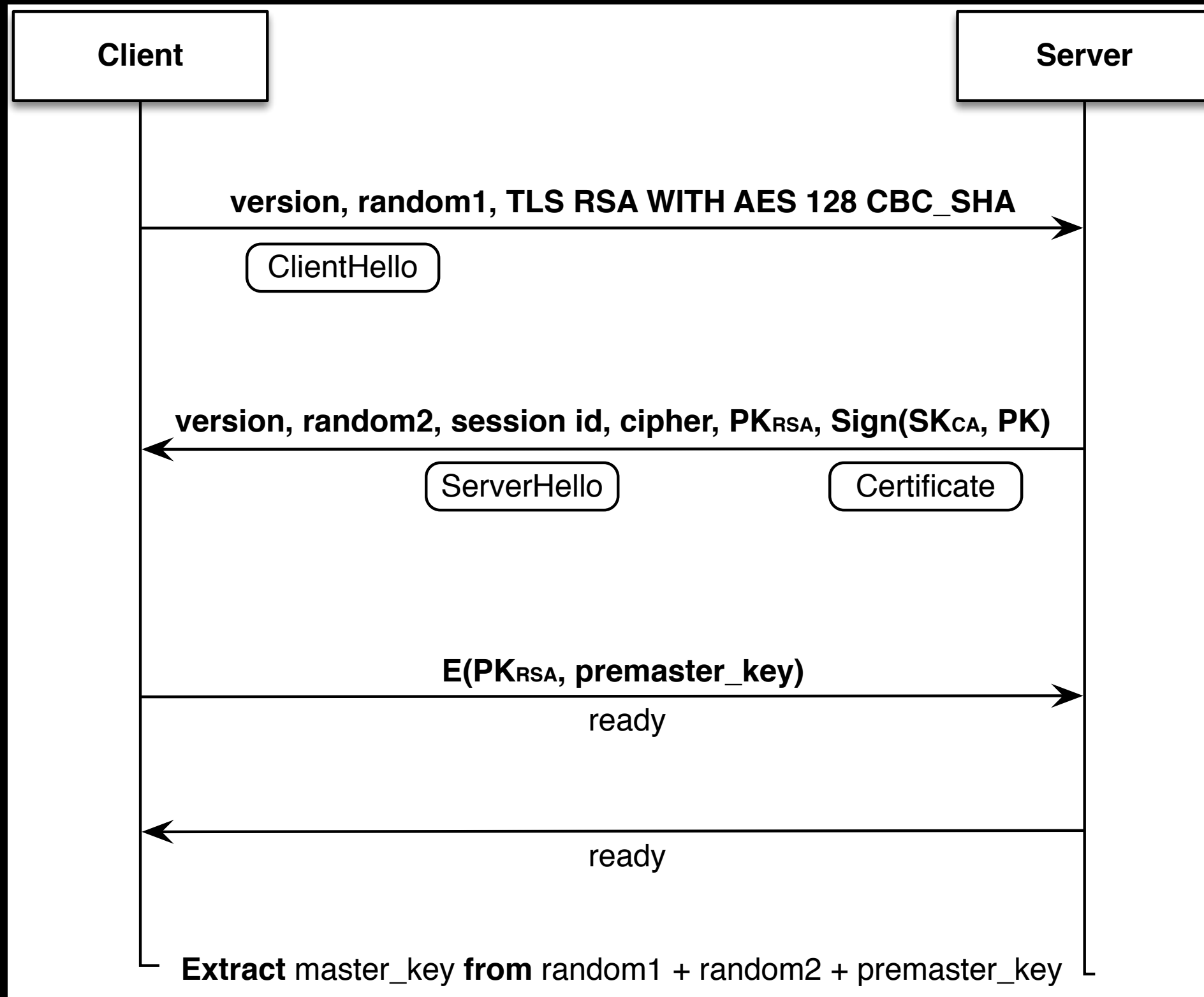
Handshake

Σημειώσεις

- Το μήνυμα **ServerKeyExchange** θα σταλεί εάν δεν έχει ταυτοποιηθεί πλήρως ο server (λ.χ. το certificate του είναι μόνο για signing).
- Το **CertificateRequest** θα σταλεί εάν ο server θέλει να ταυτοποιήσει τον client.
- Εάν ο client διαθέτει ένα certificate που έχει signing capability, το μήνυμα **CertificateVerify** θα περιέχει όλα τα μηνύματα που έχουν σταλεί μέχρι τότε υπογεγραμμένα από τον client.
- Το μήνυμα **ClientKeyExchange** μεταφέρει ένα premaster key κρυπτογραφημένο με το PK του server.
- Το μήνυμα **Finished** περιέχει το master key μαζί με ένα digest όλων των μηνυμάτων που έχουν σταλεί μέχρι τότε.

Handshake

(παράδειγμα)

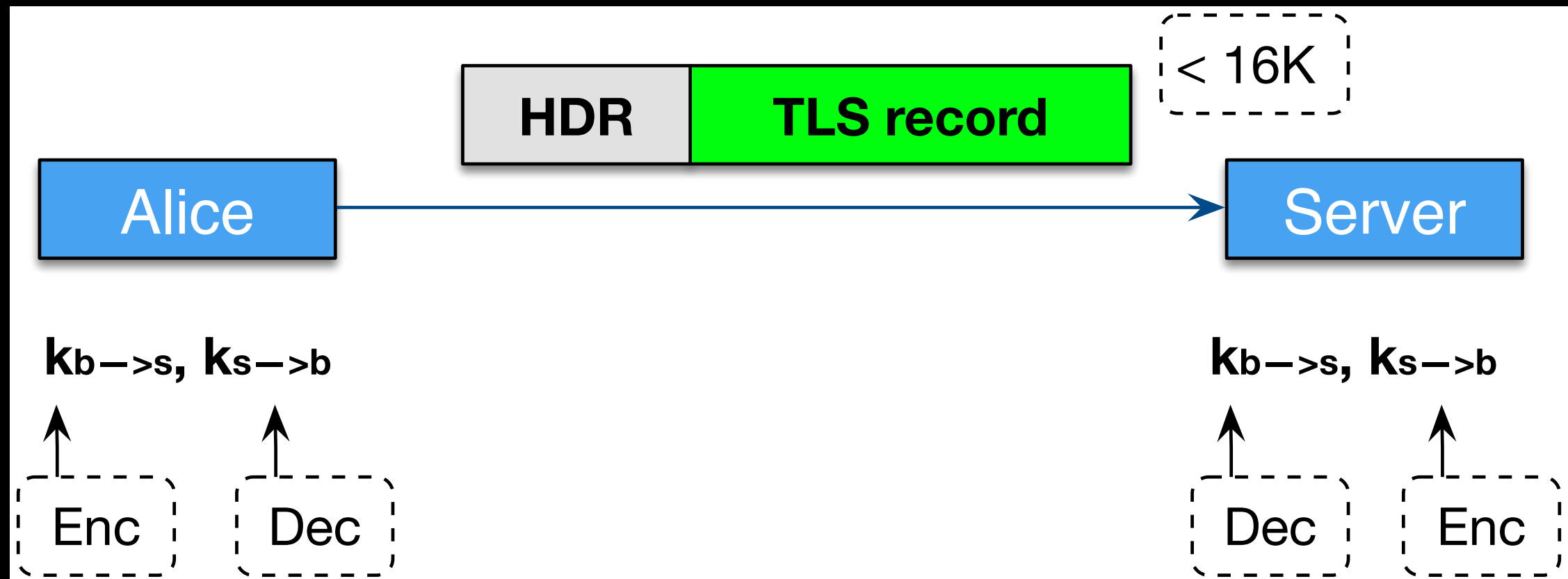


TLS

Record Protocol

- Χωρίζει τα μηνύματα σε διαχειρίσιμα **blocks** και μετά την αποστολή τους τα ξαναενώνει.
- Χρησιμοποιεί **MAC** για να διασφαλίσει την ακεραιότητα των δεδομένων.
- Κρυπτογραφεί και αποκρυπτογραφεί μηνύματα (μετά τα μηνύματα περνούν στο TCP layer).
- Προαιρετικό compression των δεδομένων.

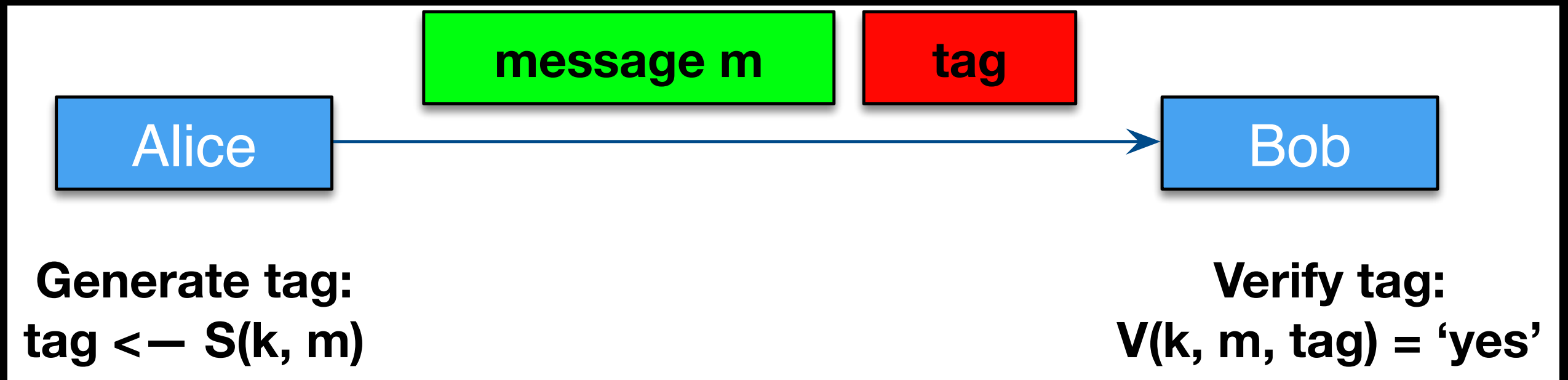
Record Protocol



- Unidirectional Keys
- Stateful Encryption
 - Κάθε πλευρά διατηρεί δυο counters: $\mathbf{ctr}_{b \rightarrow s}$, $\mathbf{ctr}_{s \rightarrow b}$
 - Οι counters μηδενίζονται όταν ξεκινάει το session ($ctr++$ για κάθε record).

MAC

(Επανάληψη)



MAC: ένα ζεύγος αλγορίθμων (S, V) που ορίζονται από 3 σύνολα (K, M, T) :

1. $S(k, m)$ παράγει ένα T
2. $V(k, m, t)$ δίνει ως έξοδο 'yes' ή 'no'.

Record Protocol

Encryption



$$k_{b \rightarrow s} = (k_{\text{mac}}, k_{\text{enc}})$$

Browser Side **$E(k_{b \rightarrow s}, \text{data}, \text{ctr}_{b \rightarrow s})$** :

- $\text{tag} \leftarrow S(k_{\text{mac}}, [++\text{ctr}_{b \rightarrow s} || \text{header} || \text{data}])$
- Κρυπτογράφηση των δεδομένων και του tag με k_{enc} .
- Prepend header.

Record Protocol

Decryption

Server Side **$D(k_{b \rightarrow s}, \text{record}, \text{ctr}_{b \rightarrow s})$** :

- Αποκρυπτογράφηση με k_{enc} .
- Έλεγχος του tag: $[++\text{ctr}_{b \rightarrow s} \parallel \text{header} \parallel \text{data}]$.
- Σε περίπτωση που το validation του tag αποτύχει στέλνει πίσω : **bad_record_mac**.

X.509

Public Key Certificate Format

- Internet standard από το 1988.
- Ακολουθεί ένα ιεραρχικό μοντέλο.

X.509: Public Key Certificate Format

Δομή και Συνήθη Πεδία

| | |
|---|---|
| Serial Number | 10:e6:fc:62:b7[...] |
| Subject | wikipedia.org |
| Issuer | GlobalSign |
| Key Usage | Signature, Key Agreement |
| Period of Validity: not before date not after date | Period of Validity: Nov 21 08:00:00 2016 GMT Nov 22 07:59:59 2017 GMT |
| Public Key | 04:c9:22:69:31[...] |
| Signature Algorithm | sha256WithRSAEncryption |
| Signature | [...] |
| ... | ... |

```
</div>
<div class="glif-promo">
<h3>New look for sign-in coming soon</h3>
<p>We're making it faster & easier to sign in to your Google Account</p>
<a href="https://support.google.com/accounts?p=signin_newlook" target="_blank">Learn more</a>
</div>
```

HTTPS

- Ένα **νέο** URL scheme.
- **Συνδυασμός** HTTP και TLS.
- “Ακούει” στο Port **443**.
- Ότι μεταδίδεται με HTTPS είναι encrypted ενώ με HTTP όχι.
- Το hostname στο HTTPS URL θα πρέπει να είναι το ίδιο με αυτό του certificate.

HTTPS

Misconfigurations

- Οι διαχειριστές ξεχνούν να ανανεώσουν τα certificates.
- Δεν έχουν περιλάβει όλα τα ονόματα που μπορεί να αντιστοιχούν στο site μέσα στο certificate (λ.χ. στη wikipedia, ανάμεσα σε άλλα υπάρχουν τα:
DNS:*.m.wikiversity.org, DNS:*.m.wikivoyage.org,
DNS:*.m.wiktionary.org, DNS:*.mediawiki.org, [...])

σε τέτοιες περιπτώσεις ο
browser θα ρωτήσει τον χρήστη
εαν δέχεται ή όχι το certificate...

και εαν ο DNS είναι υπο τον έλεγχο ενός κακόβουλου χρηστη;

HTTPS

Τι συμβαίνει με την JavaScript;

```
<script src="http://jquery.com/...">
```

Σημείωση: πλέον τέτοιου τύπου “mixed, active content” δεν επιτρέπεται από τους browsers

HTTPS

Τι συμβαίνει με τα cookies;

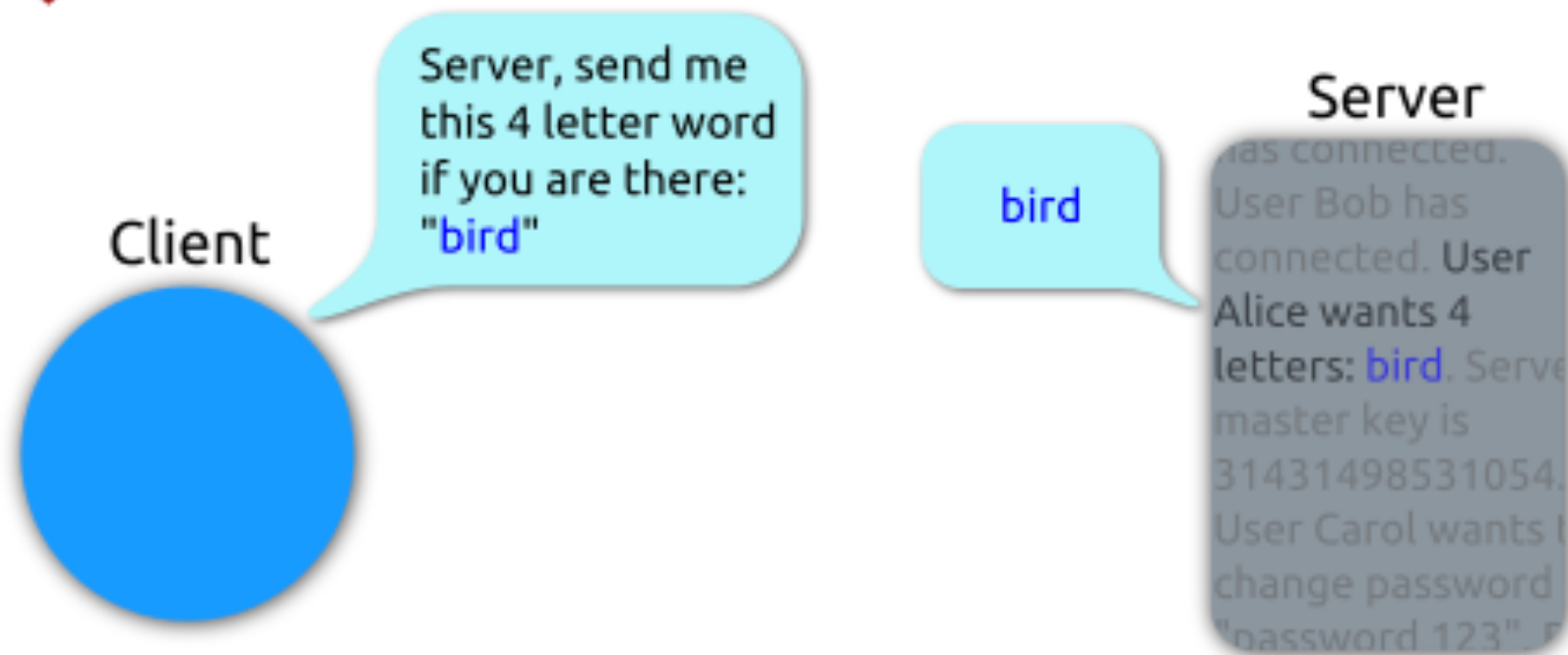
- Ένα cookie που έχει secure flag θα σταλεί μέσω HTTPS.
- Ένα cookie που **δεν** έχει secure flag μπορεί να σταλεί είτε μέσω HTTP είτε μέσω HTTPS.

Πόσο επικίνδυνο είναι αυτό;

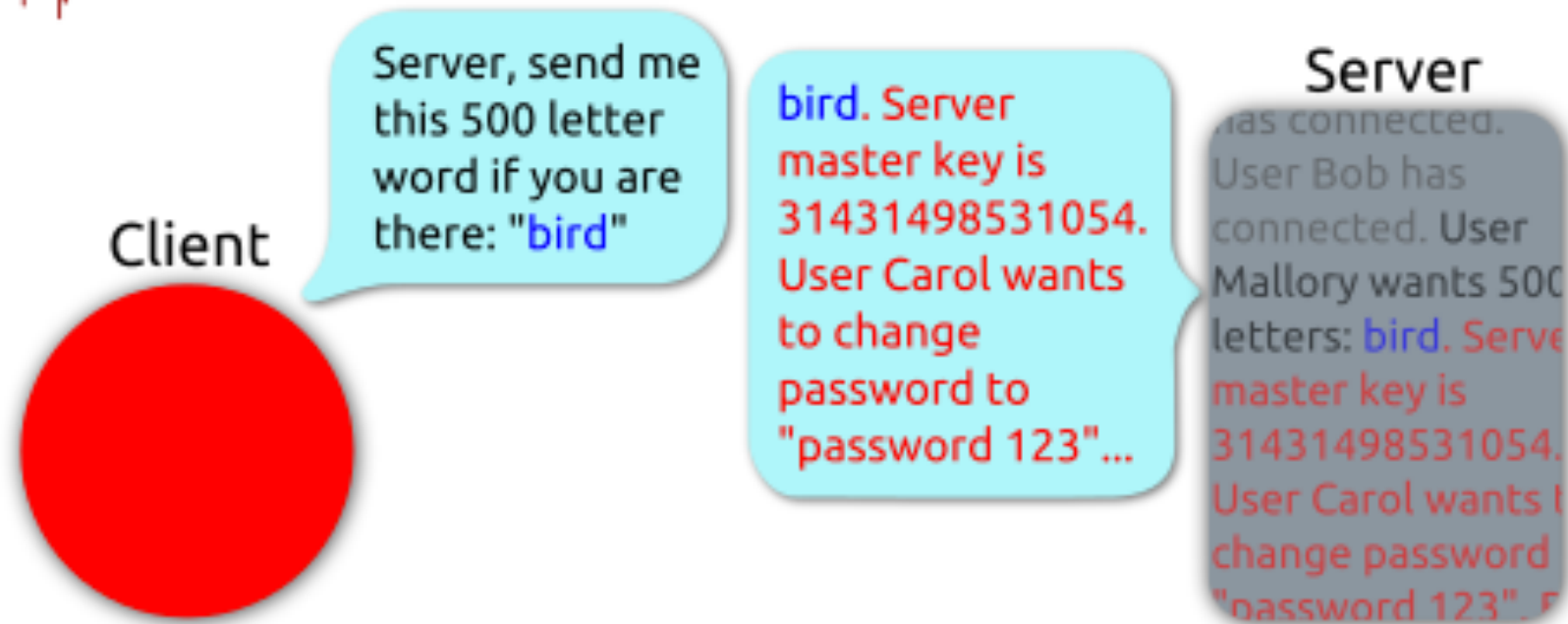




Heartbeat – Normal usage



Heartbeat – Malicious usage



ssl/d1_both.c

```
int dtls1_process_heartbeat (SSL *s) {  
    unsigned char *p = &s->s3->rrec.data[0], *pl;  
    unsigned short hbtype;  
    unsigned int payload;  
    unsigned int padding = 16; /* use minimum padding */  
    ...  
    ...  
}
```

TLS Record Struct

```
typedef struct ssl3_record_st {  
    int type; /* type of record */  
    unsigned int length; /* How many bytes available */  
    unsigned int off; /* read/write offset into 'buf' */  
    unsigned char *data; /* pointer to the record data */  
    unsigned char *input; /* where the decode bytes are */  
    unsigned char *comp; /* only used with decompression */  
    unsigned long epoch; /* epoch number, needed by DTLS1 */  
    unsigned char seq_num[8]; /* sequence number, needed by DTLS1 */  
} SSL3_RECORD;
```


The Bug (Part 1)

```
...  
/* Read type and payload length first */  
hbtype = *p++;  
n2s (p, payload);  
pl = p;  
...
```

The Bug (Part 2)

```
unsigned char *buffer, *bp;
```

```
int r;
```

```
/* Allocate memory for the response,  
 * size is 1 byte message type, plus 2 bytes payload length,  
 * plus payload, plus padding  
 */
```

```
buffer = OPENSSL_malloc (1 + 2 + payload + padding);
```

```
bp = buffer;
```

The Bug (Part 3)

```
...  
/* Enter response type, length and copy payload */  
*bp++ = TLS1_HB_RESPONSE;  
s2n (payload, bp);  
memcpy (bp, pl, payload);  
...
```

Βιβλιογραφία

The TLS Protocol Version 1.0. January 1999 [Online]. Available: <https://tools.ietf.org/html/rfc2246>.

Kaushal Kumar Panday. SSL Handshake and HTTPS Bindings on IIS. August 2013 [Online]. Available: <https://blogs.msdn.microsoft.com/kaushal/2013/08/02/ssl-handshake-and-https-bindings-on-iis/>.

Search Results Internet Engineering Task Force (IETF). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [Online]. Available: <https://www.ietf.org/rfc/rfc3280.txt>.

C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 114–129, 2014.

D. Kaminsky, M. L. Patterson, and L. Sassaman. PKI Layer Cake: New Collision Attacks Against the Global x.509 Infrastructure. In *Proceedings of the International Conference on Financial Cryptography and Data Security*, pages 289–303, 2010.

Zakir Durumeric, James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, and Vern Paxson. 2014. The Matter of Heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)*. ACM, New York, NY, USA, 475-488.

Mozilla Developer Network. Mixed content [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content.