2020BTECS00203

Cryptography and Network Security Lab

Assignment No 16

Batch – B4

Title:

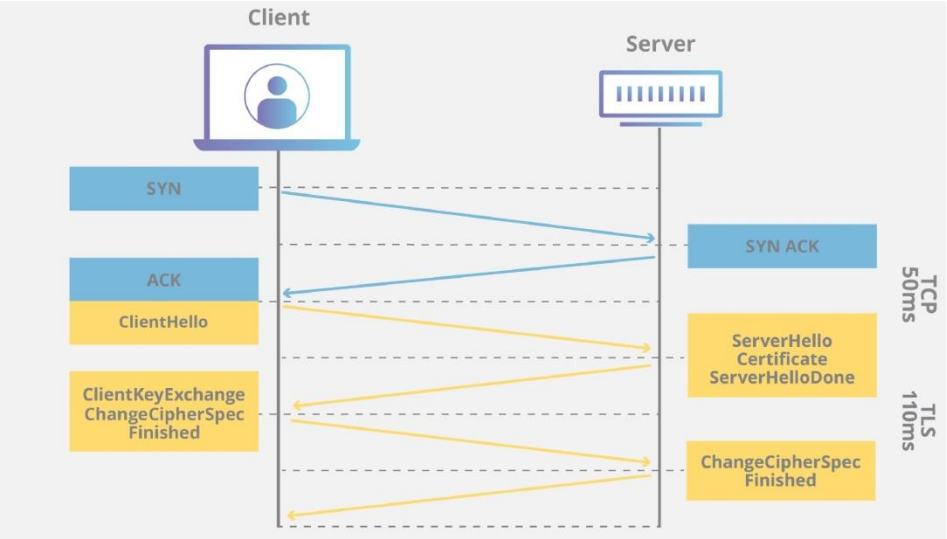
Implementation of SSL_TSL

Theory:

SSL

SSL (Secure Sockets Layer) and its successor, TLS (Transport Layer Security), are protocols for establishing authenticated and encrypted links between networked computers. Although the SSL protocol was deprecated with the release of TLS 1.0 in 1999, it is still common to refer to these related technologies as “SSL” or “SSL/TLS.” The most current version is TLS 1.3, defined in RFC 8446

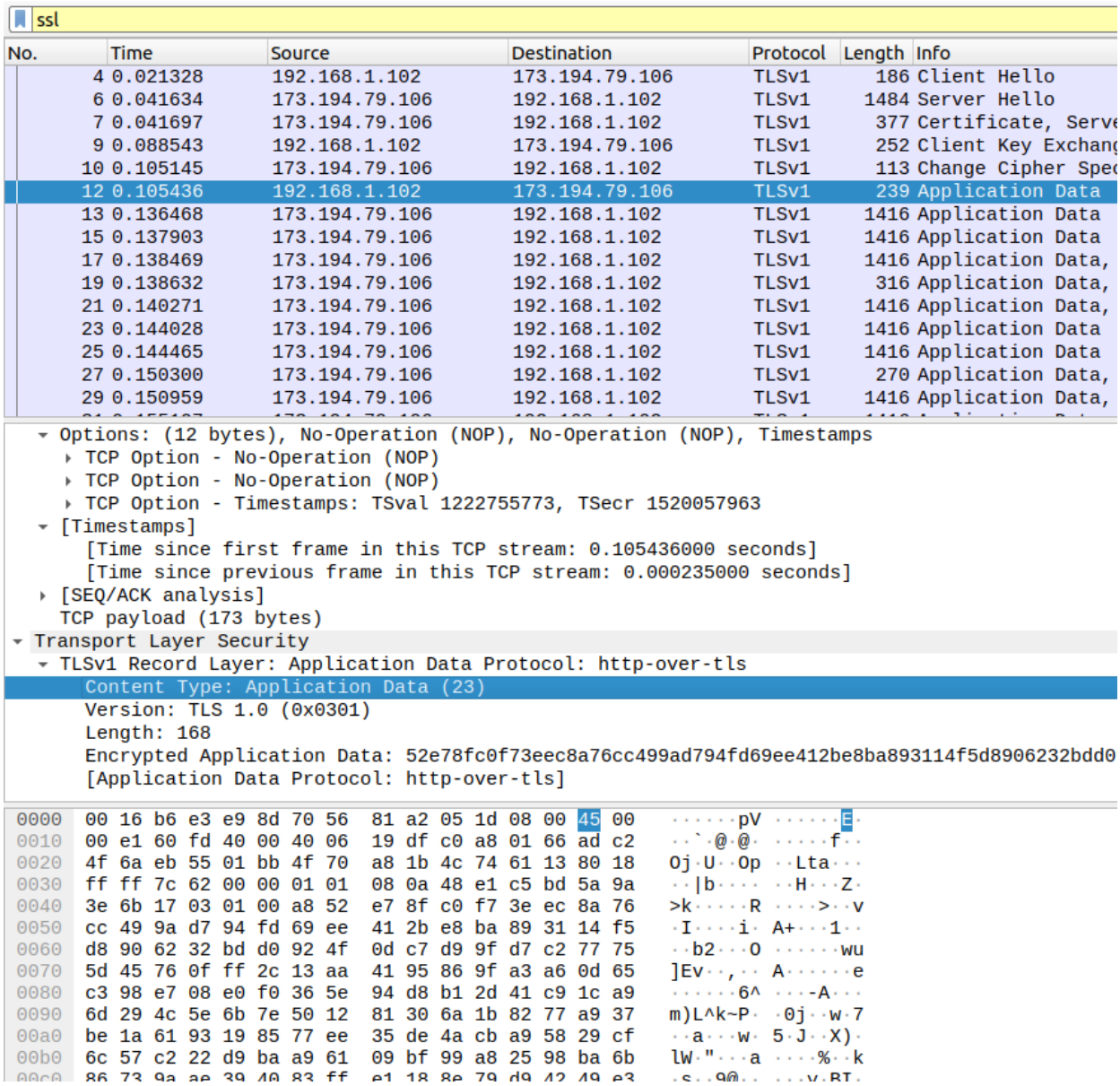
TLS (Transport Layer Security), released in 1999, is the successor to the SSL (Secure Sockets Layer) protocol for authentication and encryption. TLS 1.3 is defined in in RFC 8446 (August 2018).



Questions:

1. What is the Content-Type for a record containing Application Data?

Content-Type is: Application Data (23)

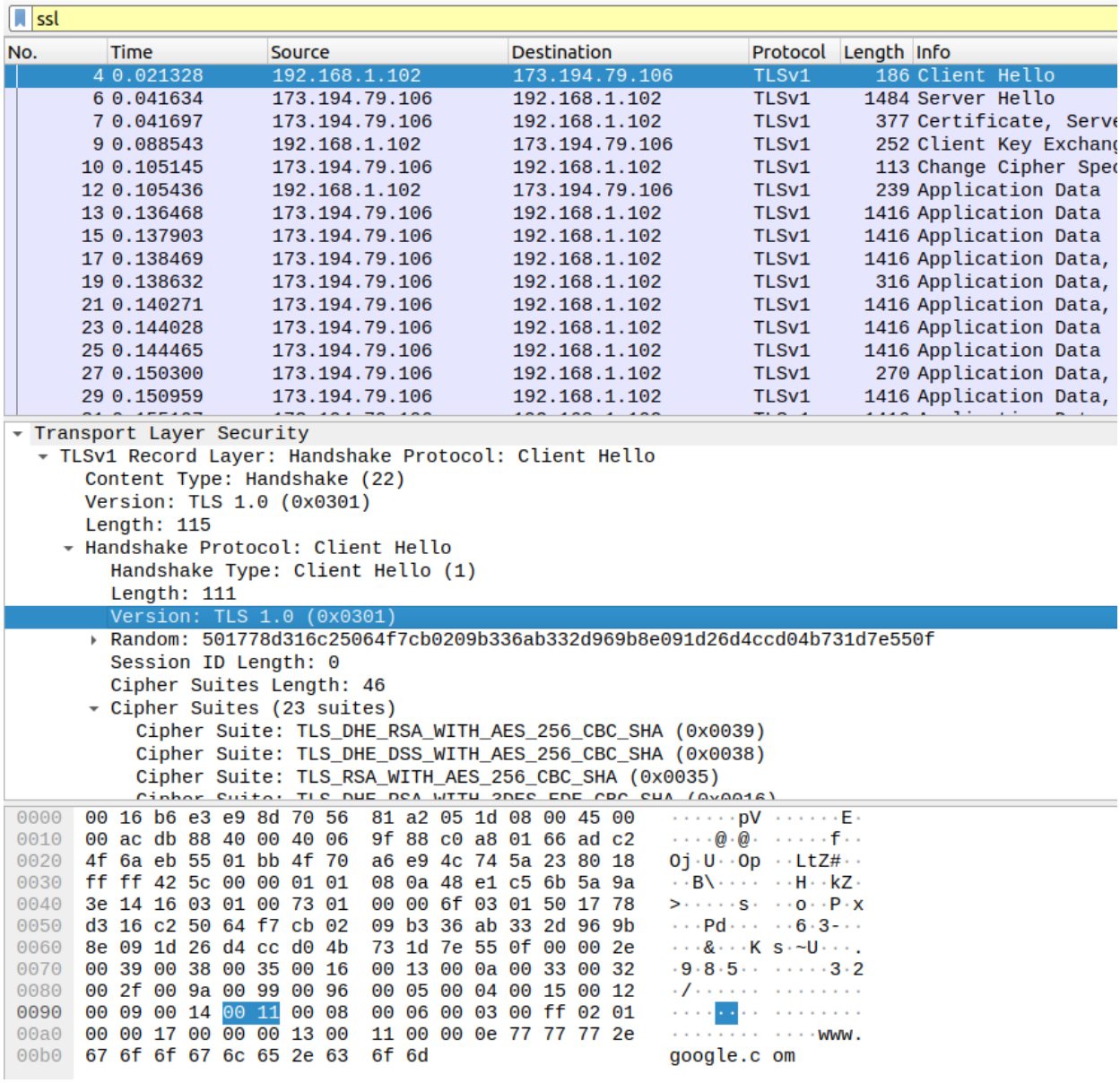


2. What version constant is used in your trace, and which version of TLS does it represent?

Ans:

TLS version : 1.0

TLS version constant: 0x0301

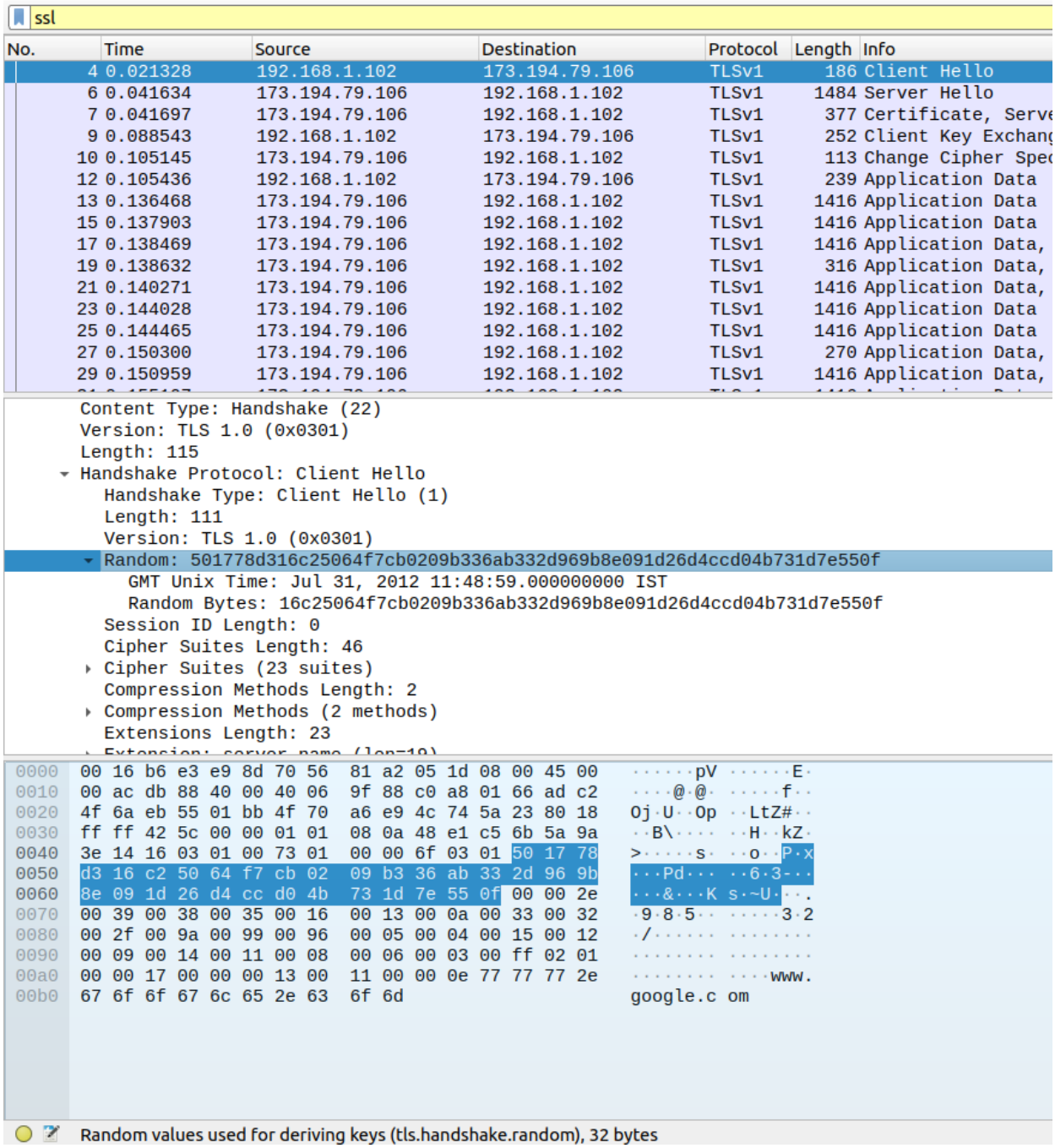


4.1 Hello Message

1. How long in bytes is the random data in the Hellos? Both the Client andServer include this random data (a nonce) to allow the establishment of session keys.

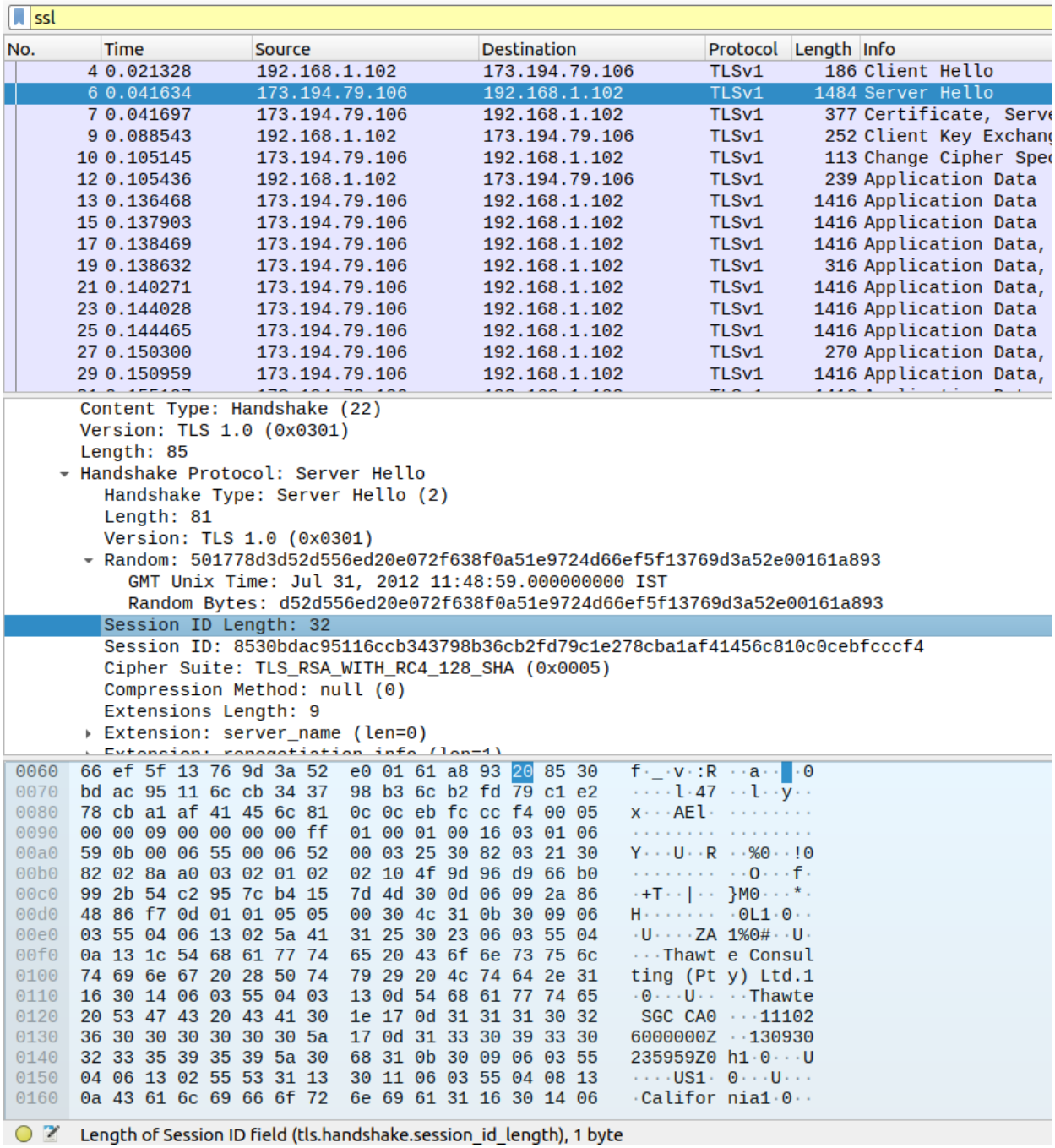
Ans:

Length of Random data filed: 32 bytes



2. How long in bytes is the session identifier sent by the server?This identifier allows later resumption of the session with an abbreviated handshake when both the client and server indicate the same value. In our case, the client likely sent no session ID as there was nothing to resume.

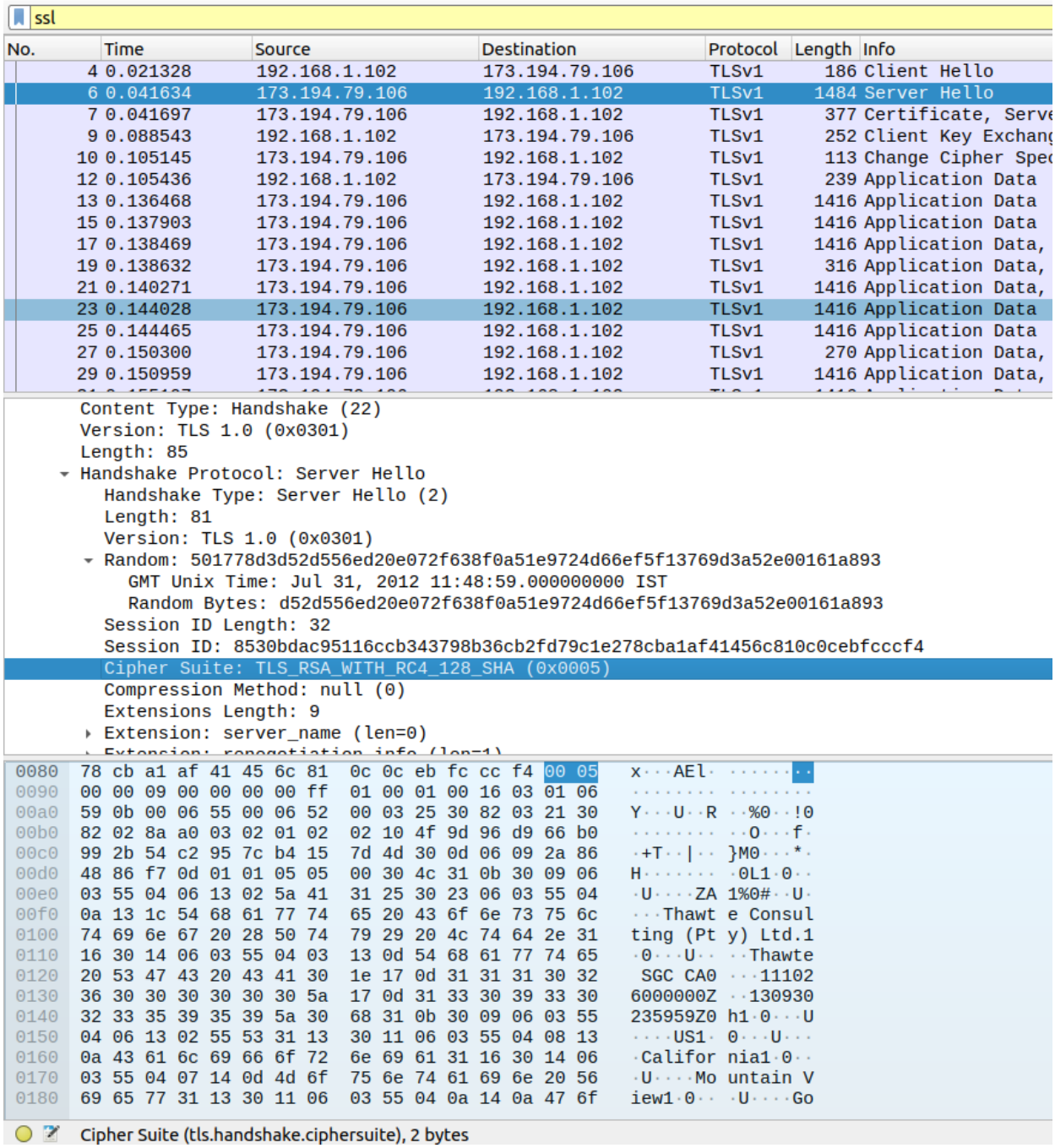
Ans: Session ID length : 32



3. What Cipher suite is chosen by the Server? Give its name and value. The Client will list the different cipher methods it supports, and the Server will pick one of these methods to use.

Ans:

Cipher Suit name: TLS_RSA_WITH_RC4_128_SHA (0x0005)

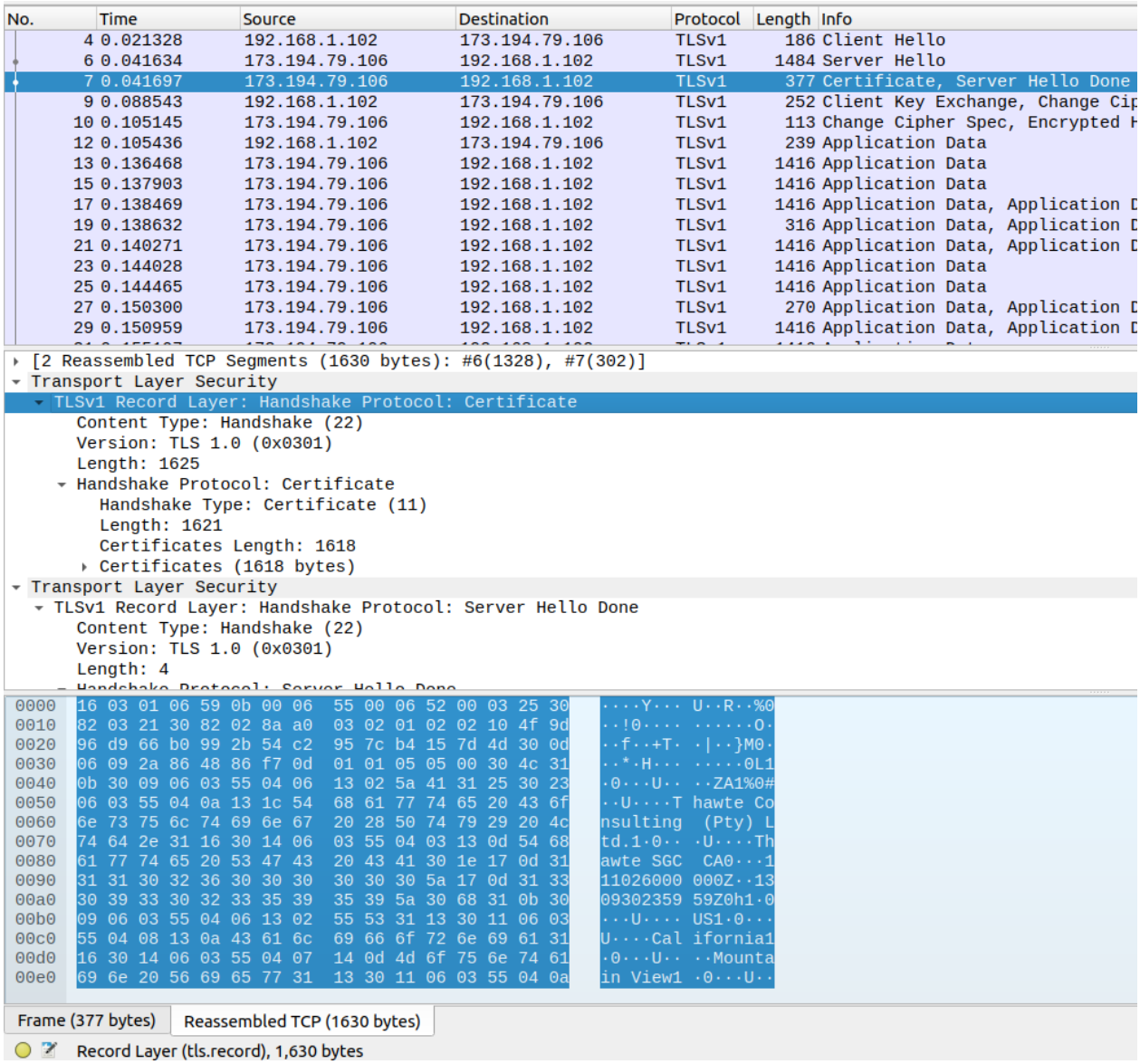


4.2 Certificate Messages

1. Who sends the Certificate, the client, the server, or both?A certificate is sent by one party to let the other party authenticate that it is who it claims to be. Based on this usage, you should be able to guess who sends the certificate and check the messages in your trace.

Ans:

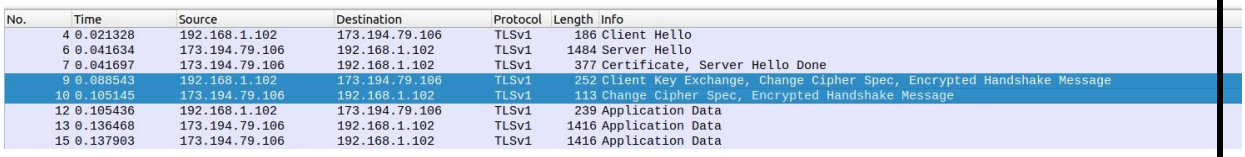
Server sends the certificate



4.3 Client Key Exchange and Change Cipher Messages

1. Who sends the Change Cipher Spec message, the client, the server, or both?

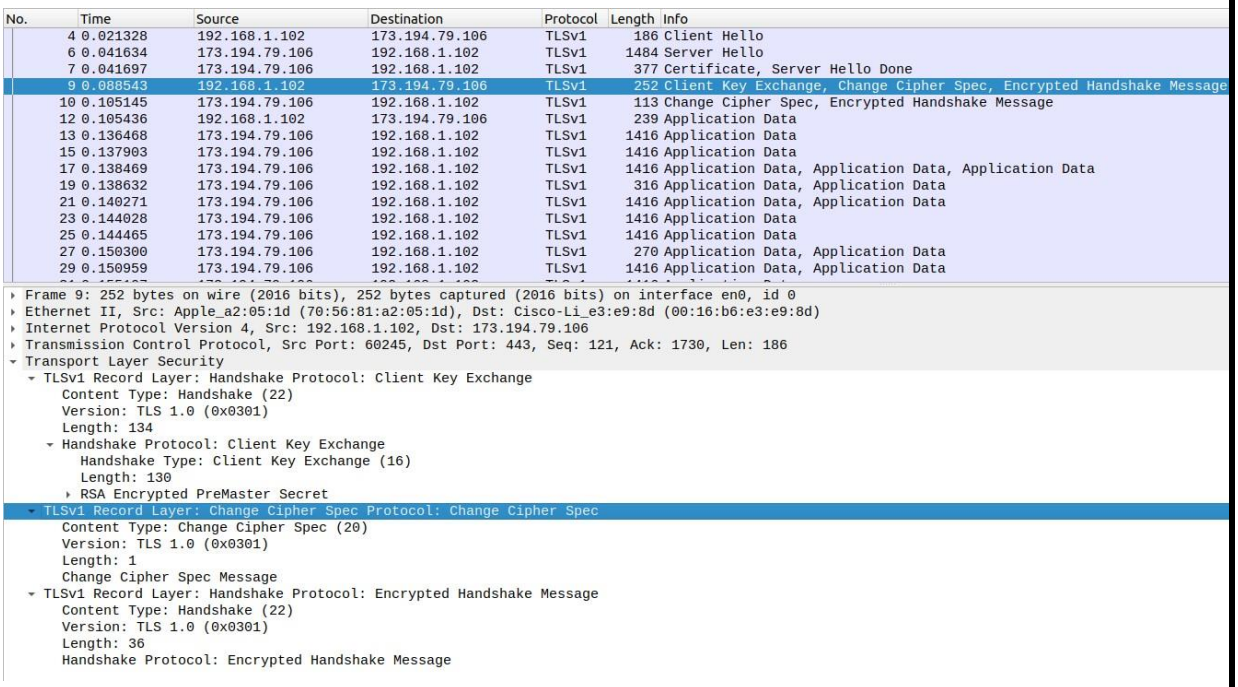
Ans: Both server and client sends the Change Cipher Spec message



No.	Time	Source	Destination	Protocol	Length	Info
4	0.021328	192.168.1.102	173.194.79.106	TLSv1	186	Client Hello
6	0.041634	173.194.79.106	192.168.1.102	TLSv1	1484	Server Hello
7	0.041697	173.194.79.106	192.168.1.102	TLSv1	377	Certificate, Server Hello Done
9	0.088543	192.168.1.102	173.194.79.106	TLSv1	252	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.105145	173.194.79.106	192.168.1.102	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12	0.105436	192.168.1.102	173.194.79.106	TLSv1	239	Application Data
13	0.136468	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data
15	0.137903	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data

2. What are the contents carried inside the Change Cipher Spec message? Lookpast the Content Type and other headers to see the message itself

Ans: Change Cipher Spec message contains:
Content Type, Version, Length and Change Cipher Spec Message



No.	Time	Source	Destination	Protocol	Length	Info
4	0.021328	192.168.1.102	173.194.79.106	TLSv1	186	Client Hello
6	0.041634	173.194.79.106	192.168.1.102	TLSv1	1484	Server Hello
7	0.041697	173.194.79.106	192.168.1.102	TLSv1	377	Certificate, Server Hello Done
9	0.088543	192.168.1.102	173.194.79.106	TLSv1	252	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.105145	173.194.79.106	192.168.1.102	TLSv1	113	Change Cipher Spec, Encrypted Handshake Message
12	0.105436	192.168.1.102	173.194.79.106	TLSv1	239	Application Data
13	0.136468	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data
15	0.137903	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data
17	0.138469	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data, Application Data, Application Data
19	0.138632	173.194.79.106	192.168.1.102	TLSv1	316	Application Data, Application Data
21	0.140271	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data, Application Data
23	0.144028	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data
25	0.144465	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data
27	0.150300	173.194.79.106	192.168.1.102	TLSv1	270	Application Data, Application Data
29	0.150959	173.194.79.106	192.168.1.102	TLSv1	1416	Application Data, Application Data

```
▸ Frame 9: 252 bytes on wire (2016 bits), 252 bytes captured (2016 bits) on interface en0, id 0
▸ Ethernet II, Src: Apple_a2:05:1d (70:56:81:a2:05:1d), Dst: Cisco-Li_e3:e9:8d (00:16:b6:e3:e9:8d)
▸ Internet Protocol Version 4, Src: 192.168.1.102, Dst: 173.194.79.106
▸ Transmission Control Protocol, Src Port: 60245, Dst Port: 443, Seq: 121, Ack: 1730, Len: 186
▾ Transport Layer Security
  ▾ TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 134
    ▾ Handshake Protocol: Client Key Exchange
        Handshake Type: Client Key Exchange (16)
        Length: 130
      ▸ RSA Encrypted PreMaster Secret
  ▾ TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.0 (0x0301)
      Length: 1
      Change Cipher Spec Message
  ▾ TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 36
      Handshake Protocol: Encrypted Handshake Message
```

CODE:

```cpp
#include<bits/stdc++.h>

using namespace std;

// returns x where (a * x) % b == 1
int mul_inv(int a, int b)
{
    int b0 = b, t, q;
    int x0 = 0, x1 = 1;
    if (b == 1) return 1;
    while (a > 1) {
        q = a / b;
```

```
        t = b, b = a % b, a = t;
        t = x0, x0 = x1 - q * x0, x1 = t;
    }
    if (x1 < 0) x1 += b0;
    return x1;
}

int chinese_remainder(int *n, int *a, int len)
{
    int p, i, prod = 1, sum = 0;

    for (i = 0; i < len; i++)
        prod *= n[i];

    cout<<"The Product of Divisors is: "<<prod<<endl;

    for (i = 0; i < len; i++) {
        p = prod / n[i];
        sum += a[i] * mul_inv(p, n[i]) * p;
    }

    return sum % prod;
}

int main(void)
{
    int n[] = { 5, 7, 9 };
    int r[] = { 2, 3, 2 };

    cout<<"The Divisors are: ";

    for(int i = 0;i < 3;i++)
        cout<<n[i]<<" ";

    cout<<"and their respective remainder are: ";

    for(int i = 0;i < 3;i++)
        cout<<r[i]<<" ";

    cout<<endl;

    int ans = chinese_remainder(n, r, sizeof(n)/sizeof(n[0]));

    cout<<"Output: "<<ans<<endl;
    return 0;
}
```

OUTPUT:

```
PS C:\Users\Admin\Desktop\WCE VII\CNS LAB\Assignment 14> cd "c:\Users\Admin\Desktop\WCE VII\CNS LAB\Assignment 14
\" ; if ($?) { g++ Chinese.cpp -o Chinese } ; if ($?) { .\Chinese }
The Divisors are: 5 7 9 and their respective remainder are: 2 3 2
The Product of Divisors is: 315
Output: 227
PS C:\Users\Admin\Desktop\WCE VII\CNS LAB\Assignment 14>
```