Università degli Studi di Milano
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Informatica e Comunicazione

Vulnerabilità dei protocolli SSL/TLS

Andrea Visconti

Overview

- Introduction to SSL/TLS
- Security provided by SSL/TLS
- Browser Exploit Against SSL/TLS (2011)
- Null Prefix Attack (2009)
- Renegotiation Attack (2009)

Introduction

SSL and TLS were meant to provide a secure channel over untrusted networks;

1994: **Secure Sockets Layer** (SSL) protocol, created by Netscape;

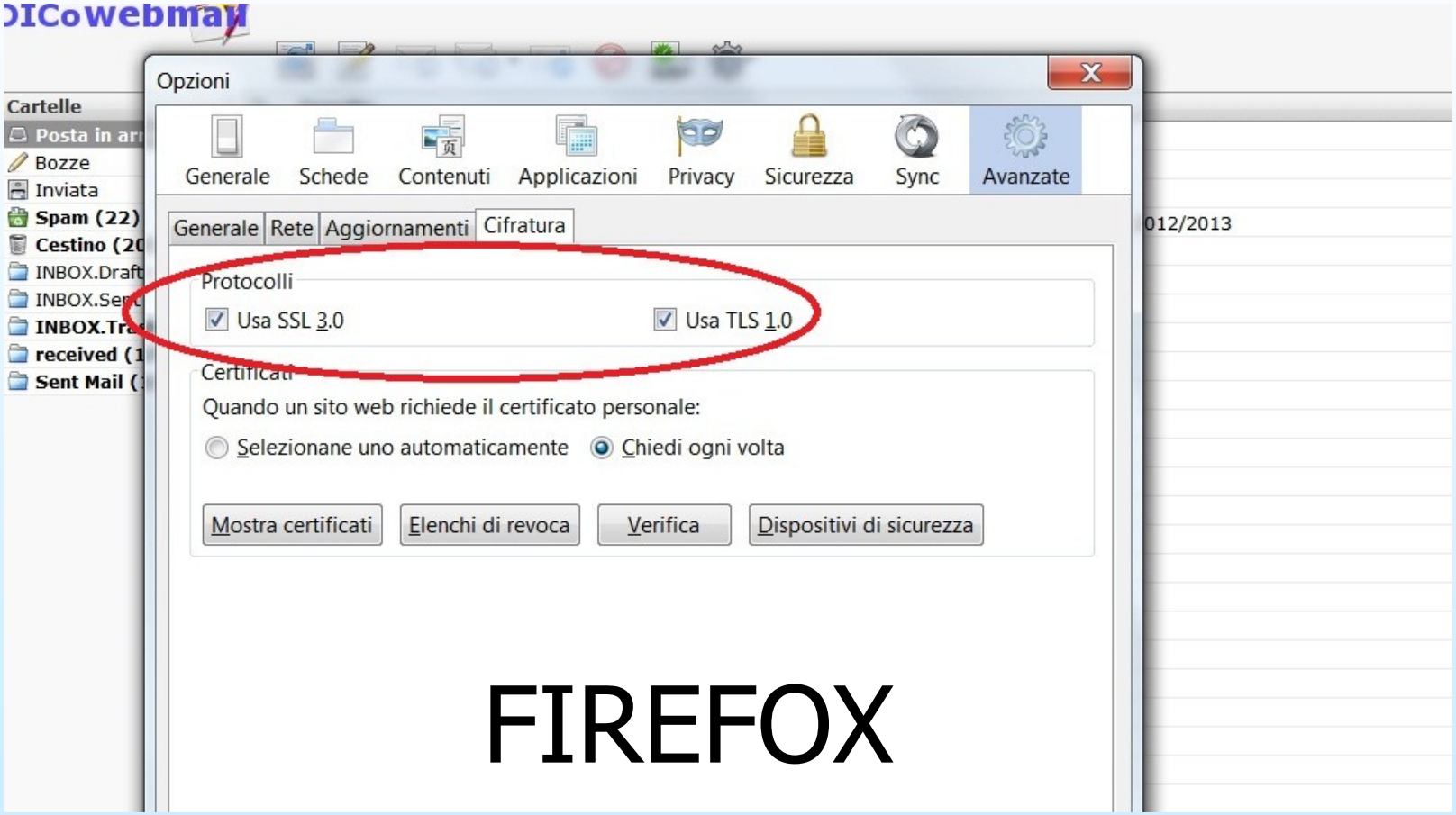
1996: **Transport Layer Security** (TLS), developed by the Internet Engineering Task Force (IETF);

SSL: SSL2 (Feb 95), **SSL3 (Mar 96)**;
TLS: **TLS1.0 (Jan 99)**, TLS1.1 (Apr 06), TLS1.2 (Ago 08);

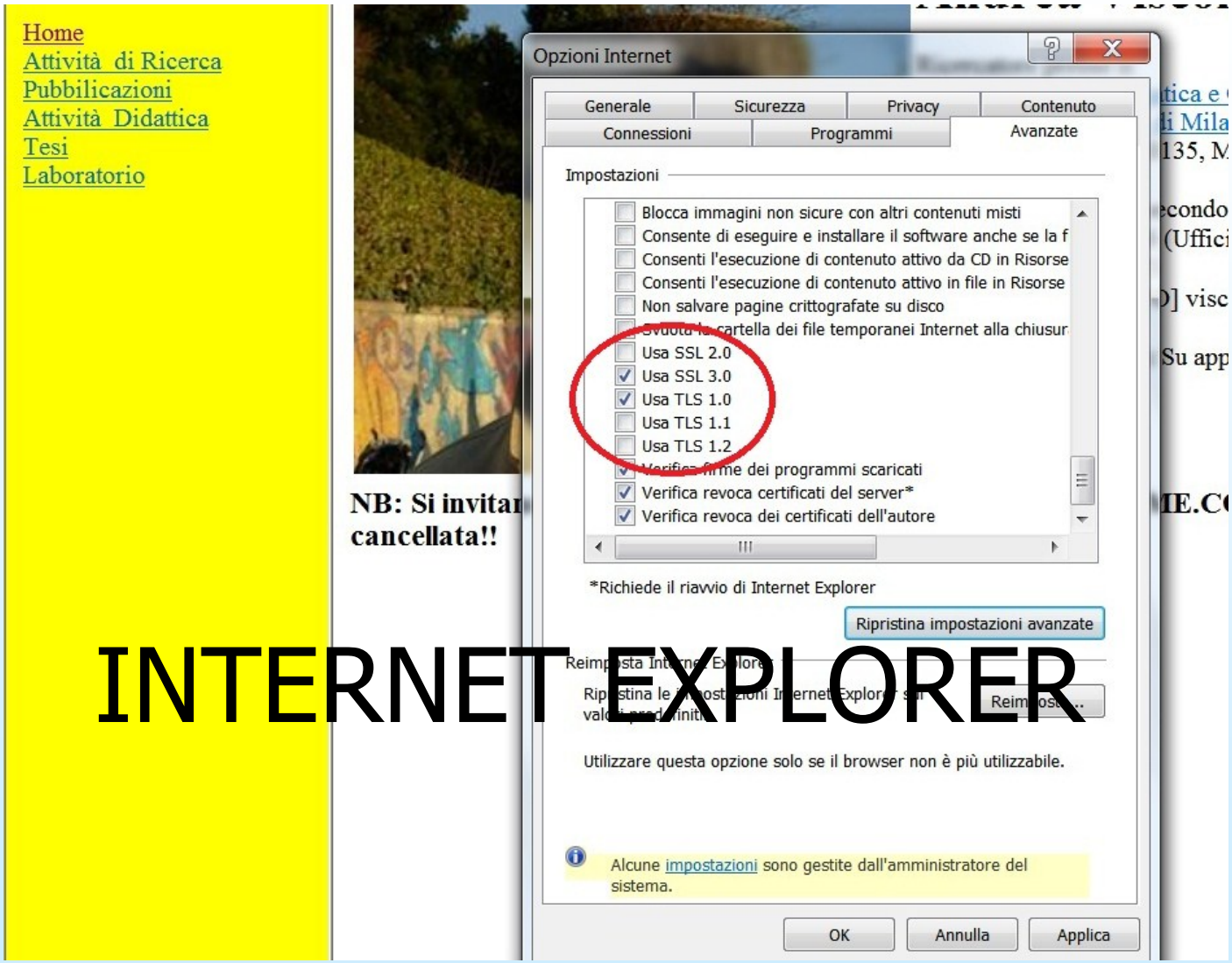
How do you choose which one to use?

Introduction

Let your browser choose for you ...



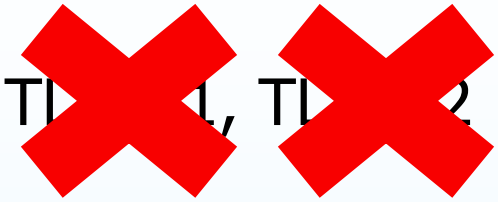
Introduction

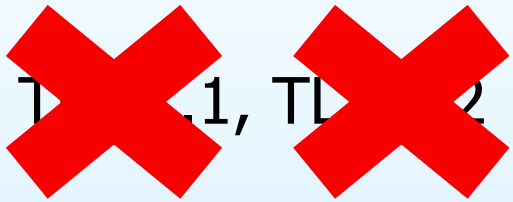


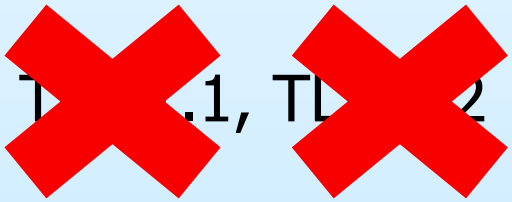
INTERNET EXPLORER

Introduction

- Firefox: TLS 1.0, TL[illegible]1, TL[illegible]2
- IE: TLS 1.0, TLS 1.1, TLS 1.2
- Chrome: TLS 1.0, T[illegible].1, TL[illegible]2
- Opera: TLS 1.0, TLS 1.1, TLS 1.2
- Safari: TLS 1.0, T[illegible].1, TL[illegible]2







Only Opera enables TLS 1.1 and TLS 1.2 by default.

Security

Security goals:

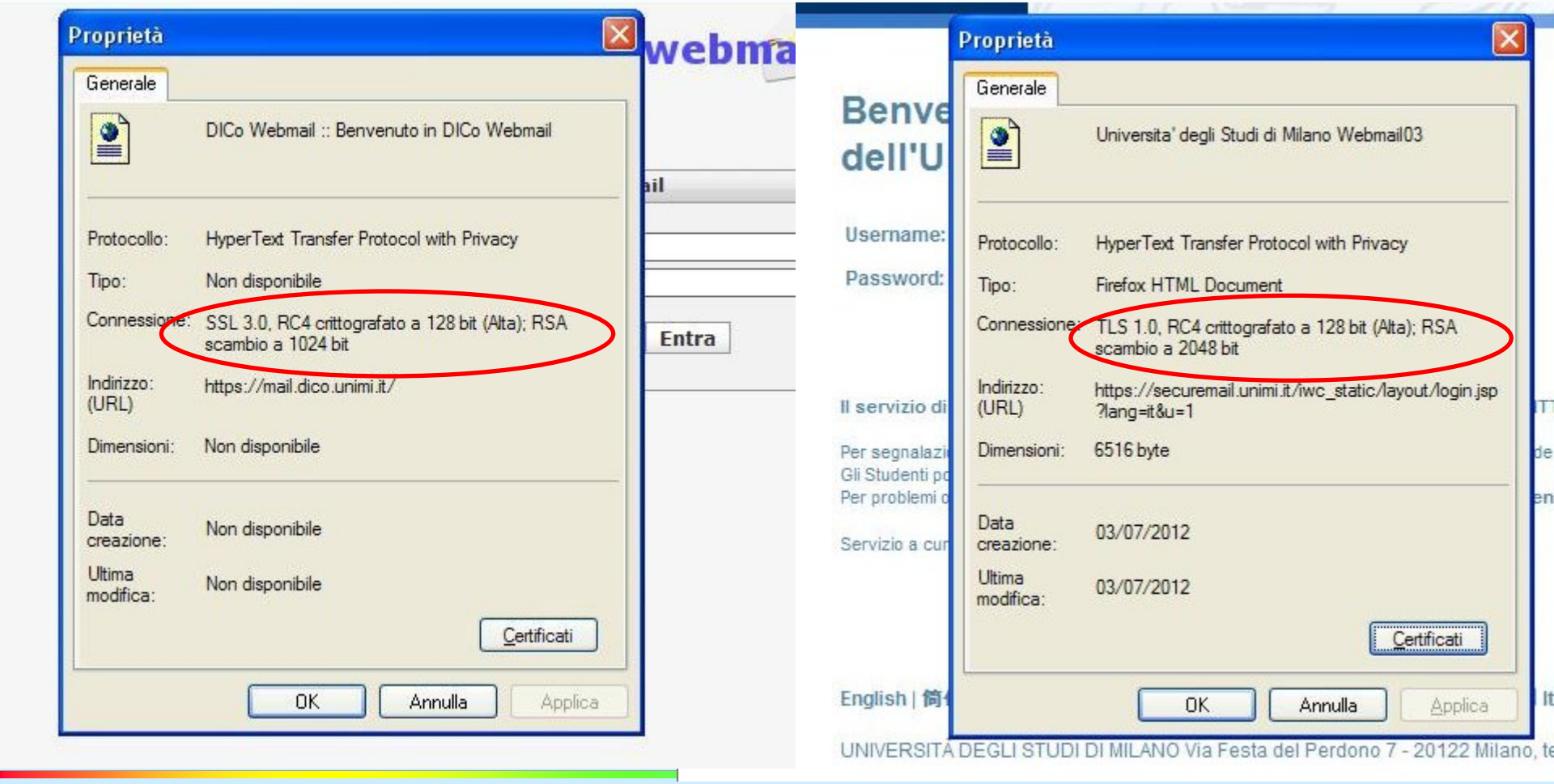
- Authentication (Certificates, Digital Signature);
- Integrity (Hash);
- Privacy (Encryption);

SSL and TLS should be able to prevent:

- Message Forgery;
- Tampering;
- Eavesdropping;

Security

The cipher suite includes algorithms for encrypting data, computing the MAC, and exchanging keys.



Browser Exploit Against SSL/TLS

Browser Exploit Against SSL/TLS

A security flaw: G.V. Bard (2004), (2006);

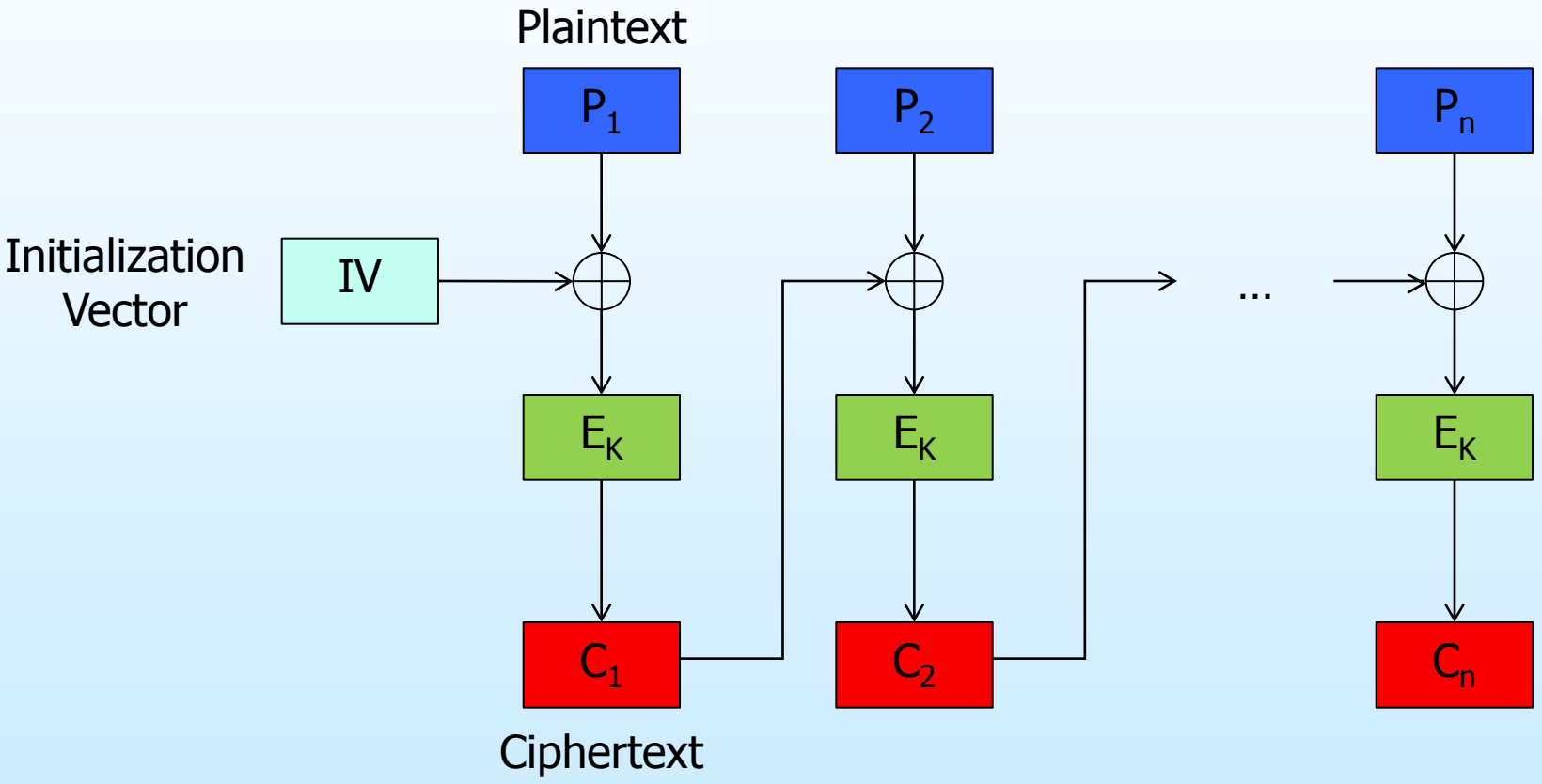
Possible solutions: fix the bug, upgrading to TLS 1.1 or later, the ostrich solution, etc.

Browser exploit against SSL/TLS: T. Duong, J. Rizzo (2011);

Unfortunately, the ostrich solution never works when it comes to security flaws.

Browser Exploit Against SSL/TLS

Cipher Block Chaining (CBC) mode encryption:



Browser Exploit Against SSL/TLS

- An attacker (Eva) can intercept network traffic;
- She will know C_0=IV, C_1, C_2, …, C_n;
- CBC mode encryption with chained initialization vectors;
- Initialization Vector (IV) is predictable;

An example:

Plaintext P=VISCONTIANDREA

P_1 = VISCONTI P_2 = ANDREA%%

Browser Exploit Against SSL/TLS

1. Block size B bytes (e.g. 8 bytes);

$$P_1 = V||I||S||C||O||N||T||I = 8 \text{ bytes};$$

2. Eva chooses a random string R (B – 1 bytes);

$$R = A||A||A||A||A||A||A = 7 \text{ bytes};$$

3. She prepends R to P:

$$P_1^* = A||A||A||A||A||A||A||V = 8 \text{ bytes};$$

4. She tries to guess P_1^*:

$$P_1' = \text{Random String R} \,||\, \text{Random character}$$

$$= A||A||...||A||? = 8 \text{ bytes};$$

Browser Exploit Against SSL/TLS

5. Eva chooses a random string R (B – 2 bytes);

   R = A||A||A||A||A||A = 6 bytes;

6. She prepends R to P_1:

   X_1^* = A||A||A||A||A||A||V||I = 8 bytes;

7. She tries to guess X_1^*:

   X_1' = Random String R || Random character =

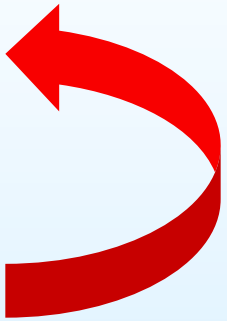
   = A||A||...||A||V||? = 8 bytes;

Browser Exploit Against SSL/TLS

CBC mode encryption:

$$C_0 = IV;$$

$$C_i = E_K(P_i \oplus C_{i-1})$$



1. Eva tries to guess P_1;
2. She knows that $C_1 = E_K(P_1 \oplus IV)$;
3. R is prepended to plaintext P, P^*=AAAAAAA||VISCONTIAN…
4. P^* is divided in blocks of B byte $P^* = P_1{}^*P_2{}^*P_3{}^*\ldots$;
5. P^* is encrypted, $C^* = C_1{}^*C_2{}^*C_3{}^*\ldots$;
6. C^* is transmitted over the channel;

Browser Exploit Against SSL/TLS

7. Eva tries to guess P_1^*=AAAAAAAV;
8. She knows that $C_1^* = E_K(P_1^* \oplus IV) = E_K(P_1^* \oplus C_0^*)$
9. She defines $G_1' = IV \oplus C_0^* \oplus P_1'$
10. She sends G_1' to the client;
11. If $P_1' = P_1^*$ then

    $C_1' = E_K(G_1' \oplus IV)$

    $= E_K(IV \oplus C_0^* \oplus P_1' \oplus IV)$

    $= E_K(C_0^* \oplus P_1')$

    $= E_K(C_0^* \oplus P_1^*) = C_1^*$
12. If $P_1' \neq P_1^*$ then $C_1' \neq C_1^*$

Browser Exploit Against SSL/TLS

- Deterministic algorithm;
- An attacker tries to guess the encoding of a byte instead of a block;
- 256 iterations (worst case);
- 128 iterations (average case);

Null Prefix Attack

Null Prefix Attack

The problem is related to how browsers handle certificate fields with **null value character** (**\0**).

- **String format**: PASCAL VS. C;
- **Common name**: Main field checked for authentication;
- **Authentication**: Domain validation certificates rely on email checking;

Null Prefix Attack

- Attackers generate and submit a fake certificate request to Certification Authorities;

  www.my_email.com\0I_am_cheating_you.com

- During validation, Certification Authorities do not check request content fully, ignoring the subdomains placed before the null value character;

  www.my_email.com\0I_am_cheating_you.com

Null Prefix Attack

- Domain validation certificates rely on email checking;

  www.I_am_cheating_you.com

- Browsers interpret "\0" character as a terminating point

  www.my_email.com**\0**I_am_cheating_you.com

  hence

  www.my_email.com

TLS Renegotiation Attack

RFC 5746: TLS Renegotiation Indication Extension

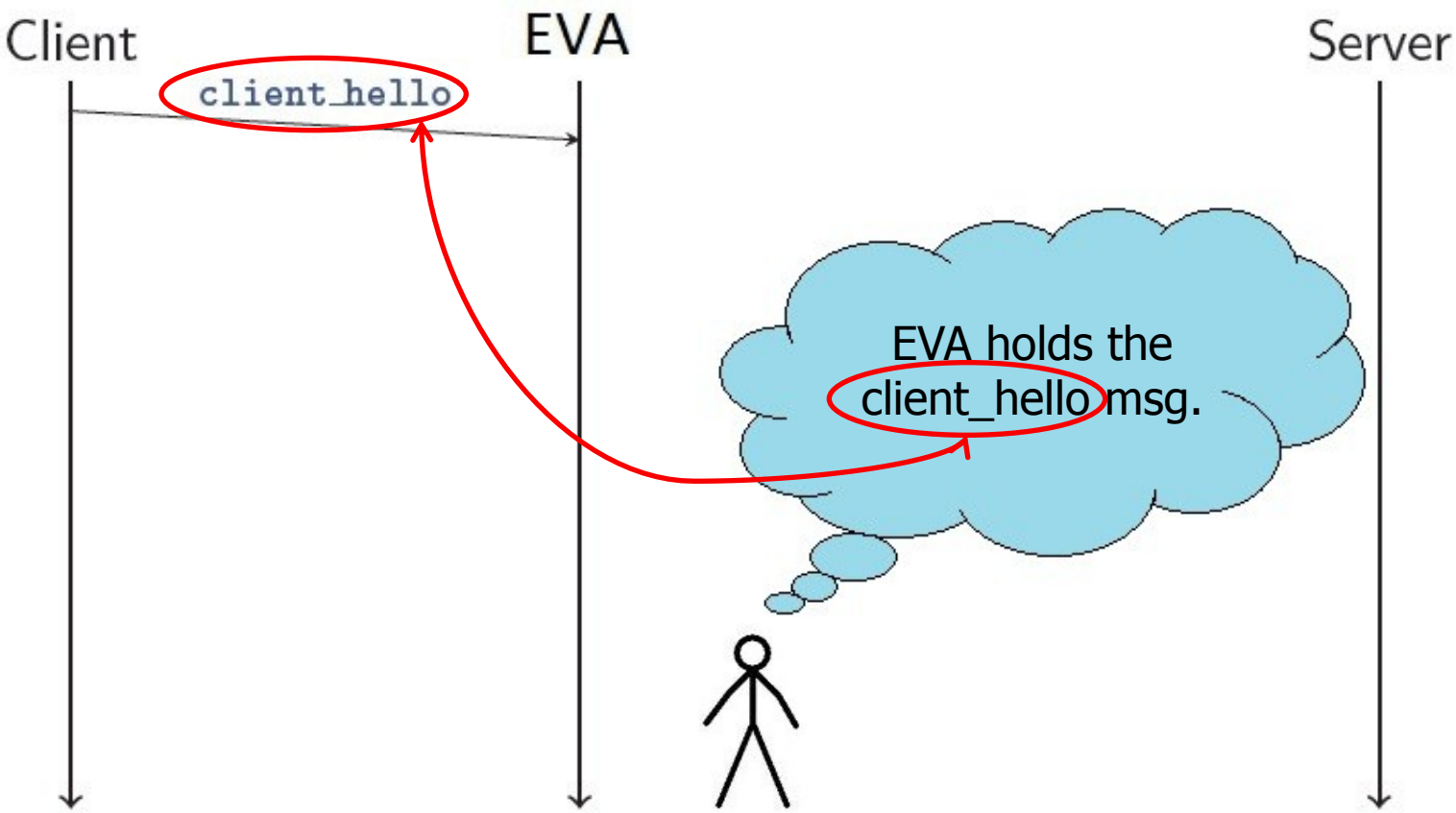
Feb 2010 – RFC 5746 – **Abstract**: Secure Socket Layer (**SSL**) and Transport Layer Security (**TLS**) **renegotiation are vulnerable to an attack** in which the attacker forms a TLS connection with the target server, injects content of his choice, and then splices in a new TLS connection from a client. The server treats the client's initial TLS handshake as a renegotiation and thus believes that the initial data transmitted by the attacker is from the same entity as the subsequent client data. **This specification defines a TLS extension** to cryptographically tie renegotiations to the TLS connections they are being performed over, thus **preventing this attack**.

RFC 5746: TLS Renegotiation Indication Extension

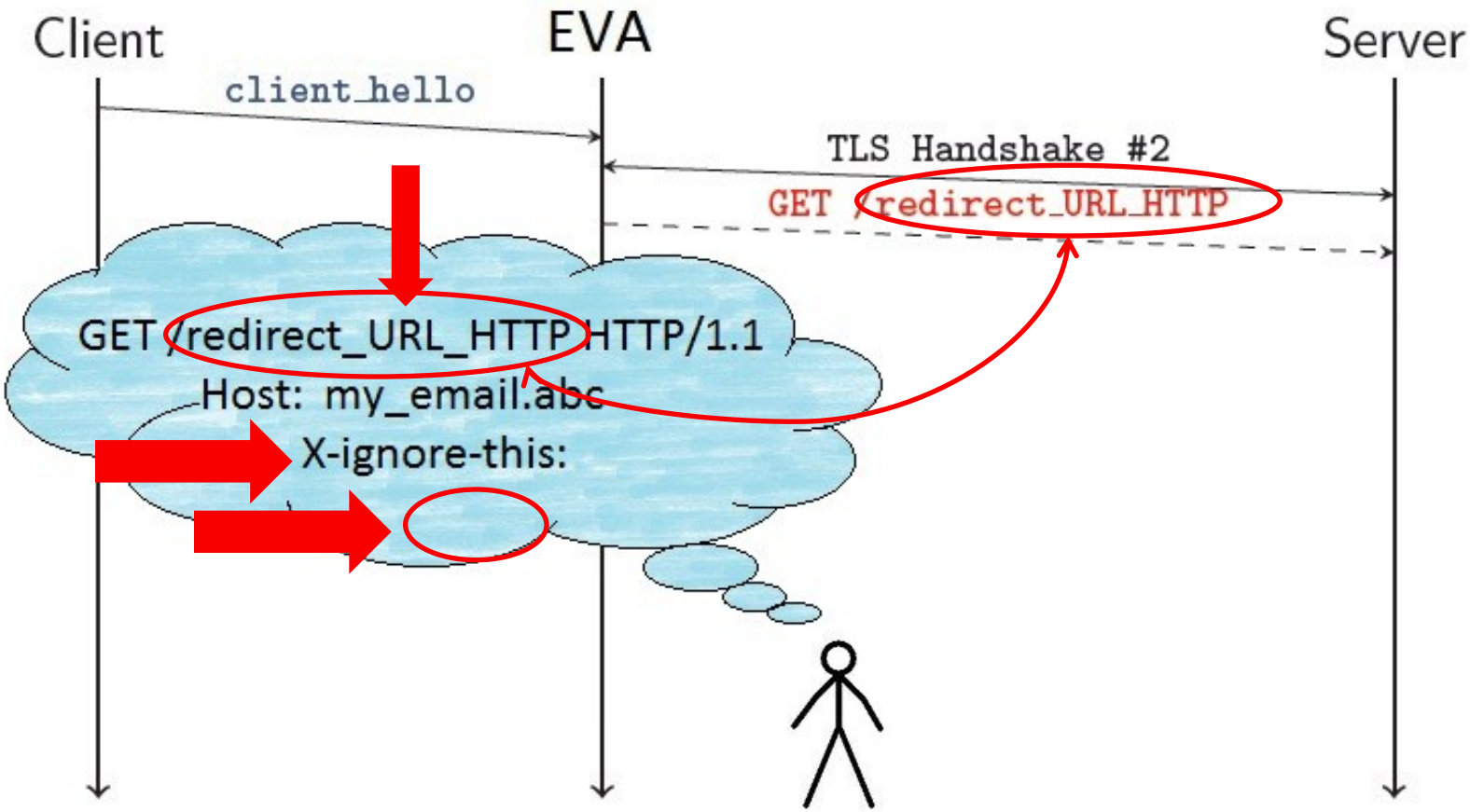
Feb 2010 – RFC 5746 – **Introduction**: … In some protocols **(notably HTTPS), no distinction is made between pre- and post-authentication stages** and the bytes are handled uniformly, **resulting in the server believing that the initial traffic corresponds to the authenticated client identity.** Even without certificate-based authentication, **a variety of attacks may be possible** in which the attacker convinces the server to accept data from it as data from the client.

For instance, **if HTTPS is in use with HTTP cookies, the attacker may be able to generate a request of his choice validated by the client's cookie.**

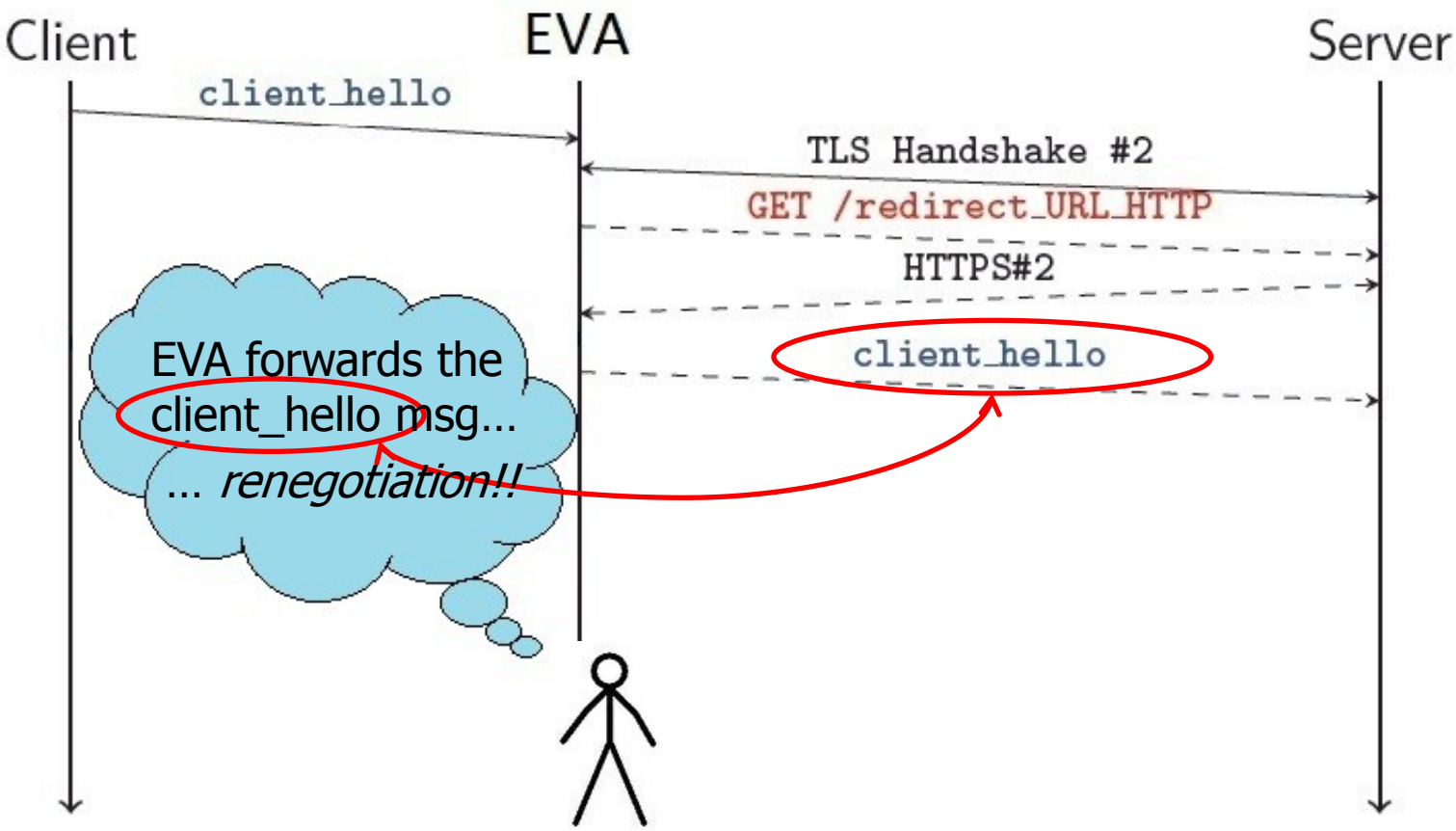
TLS Renegotiation Attack



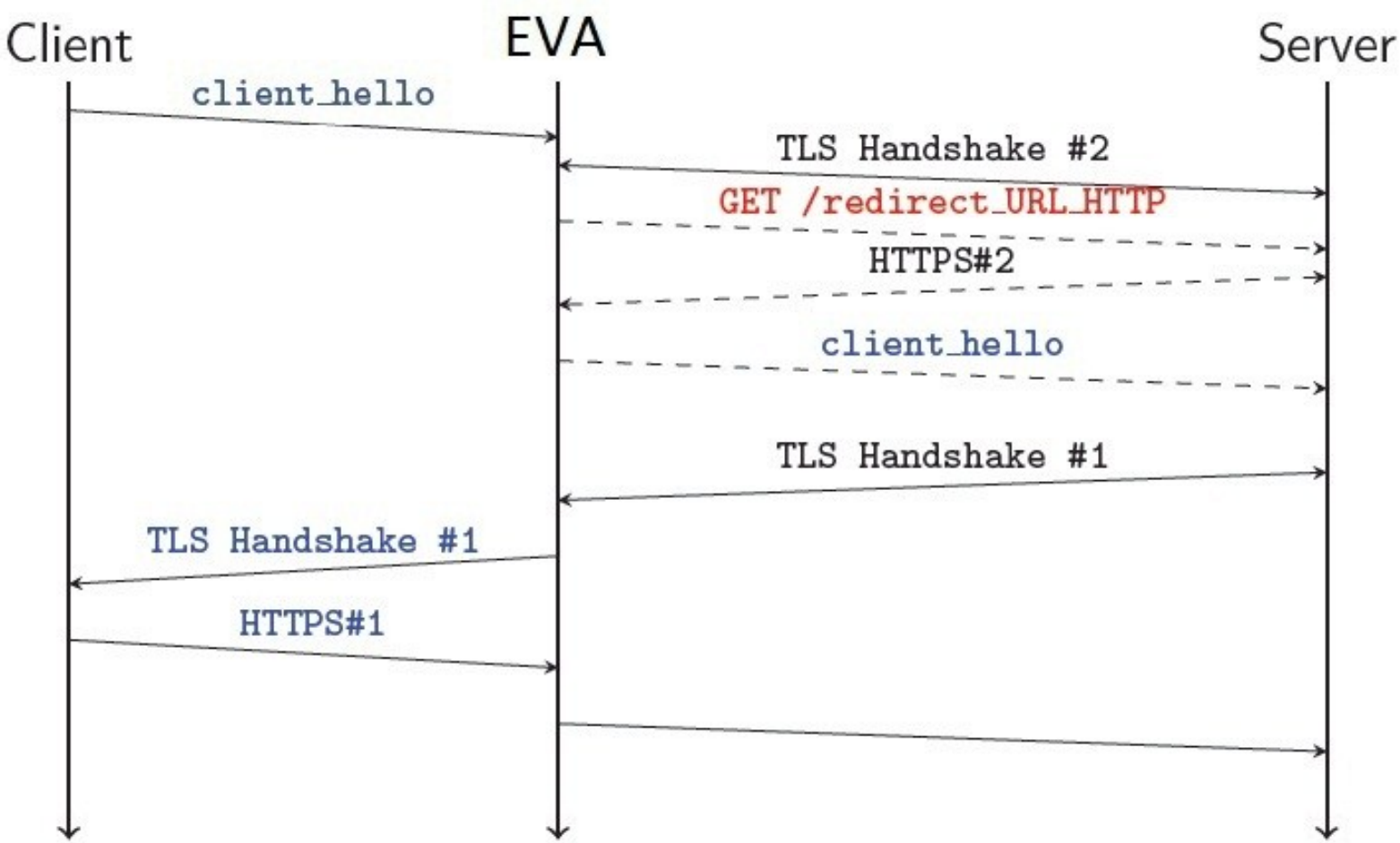
TLS Renegotiation Attack



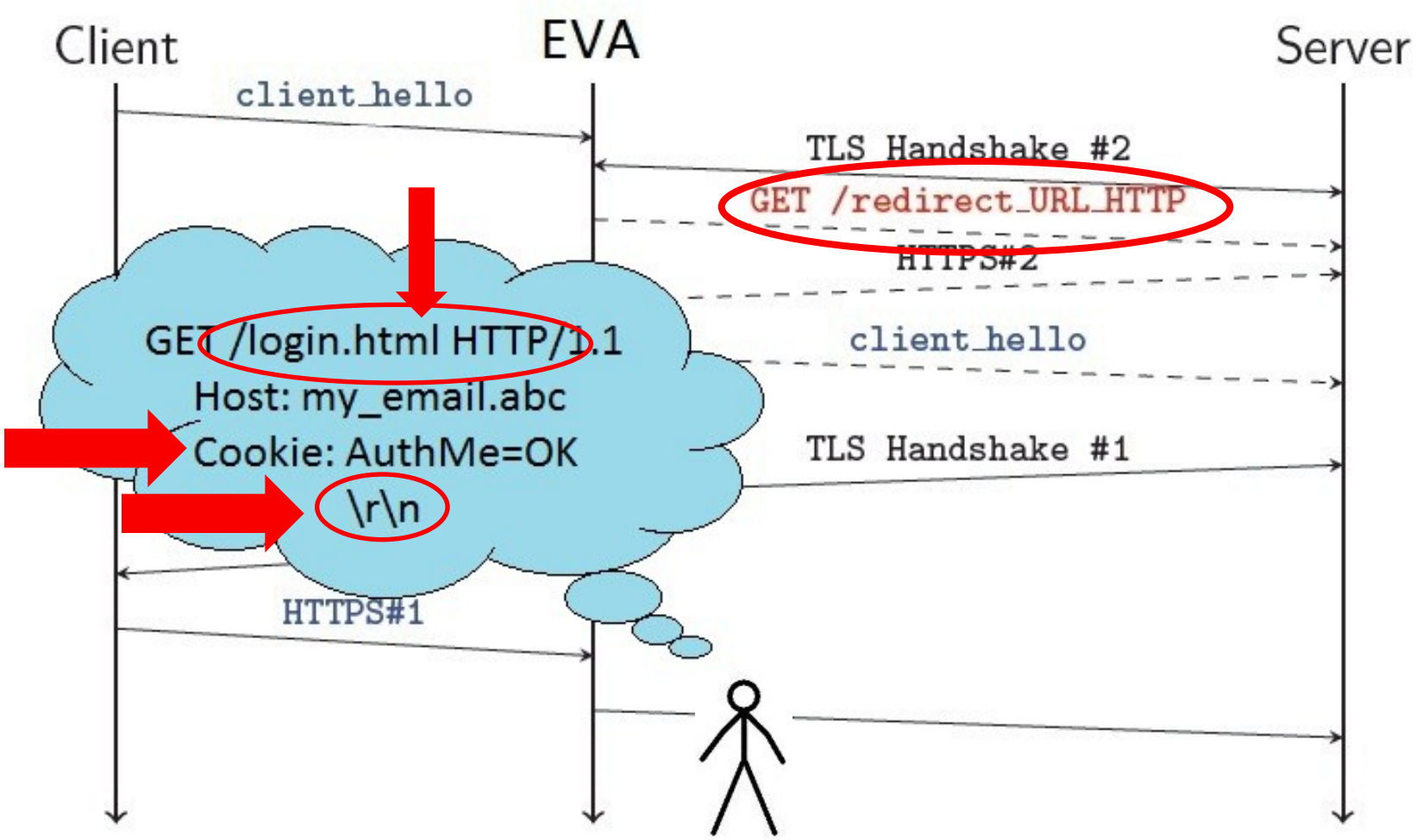
TLS Renegotiation Attack



TLS Renegotiation Attack



TLS Renegotiation Attack



TLS Renegotiation Attack