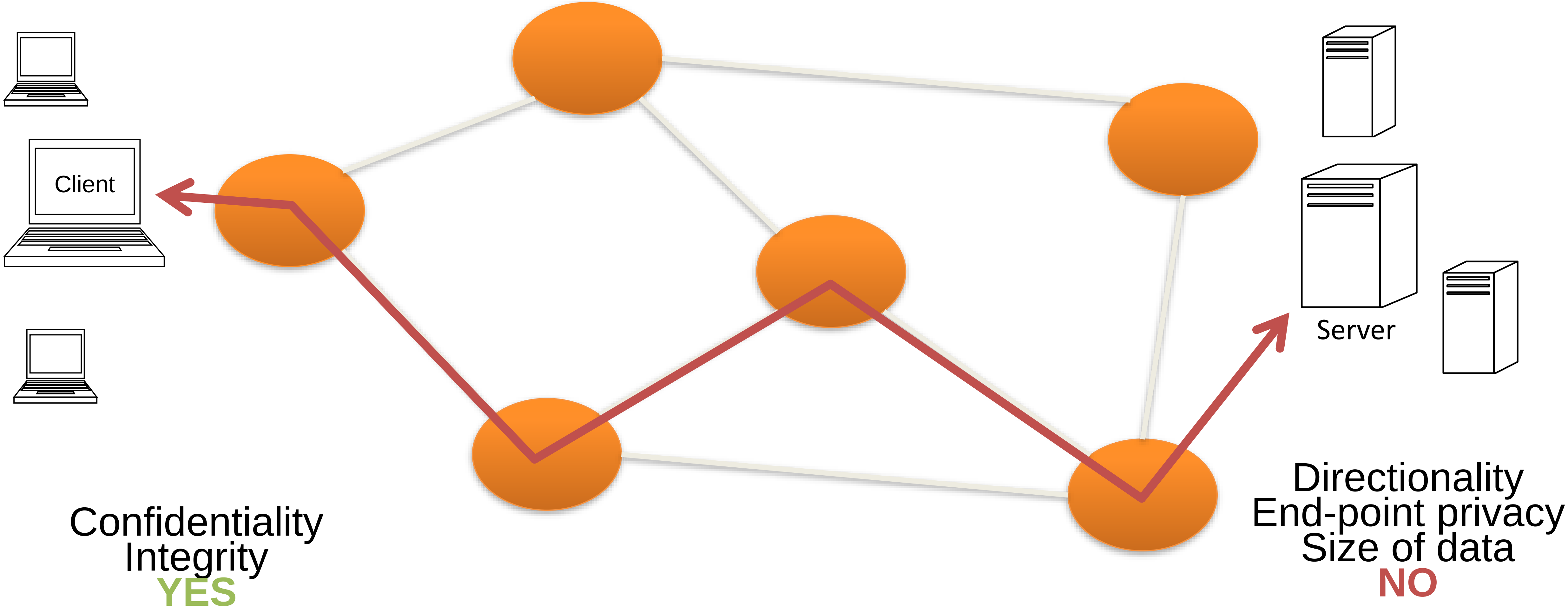
SSL/TLS

COMPUTER SECURITY

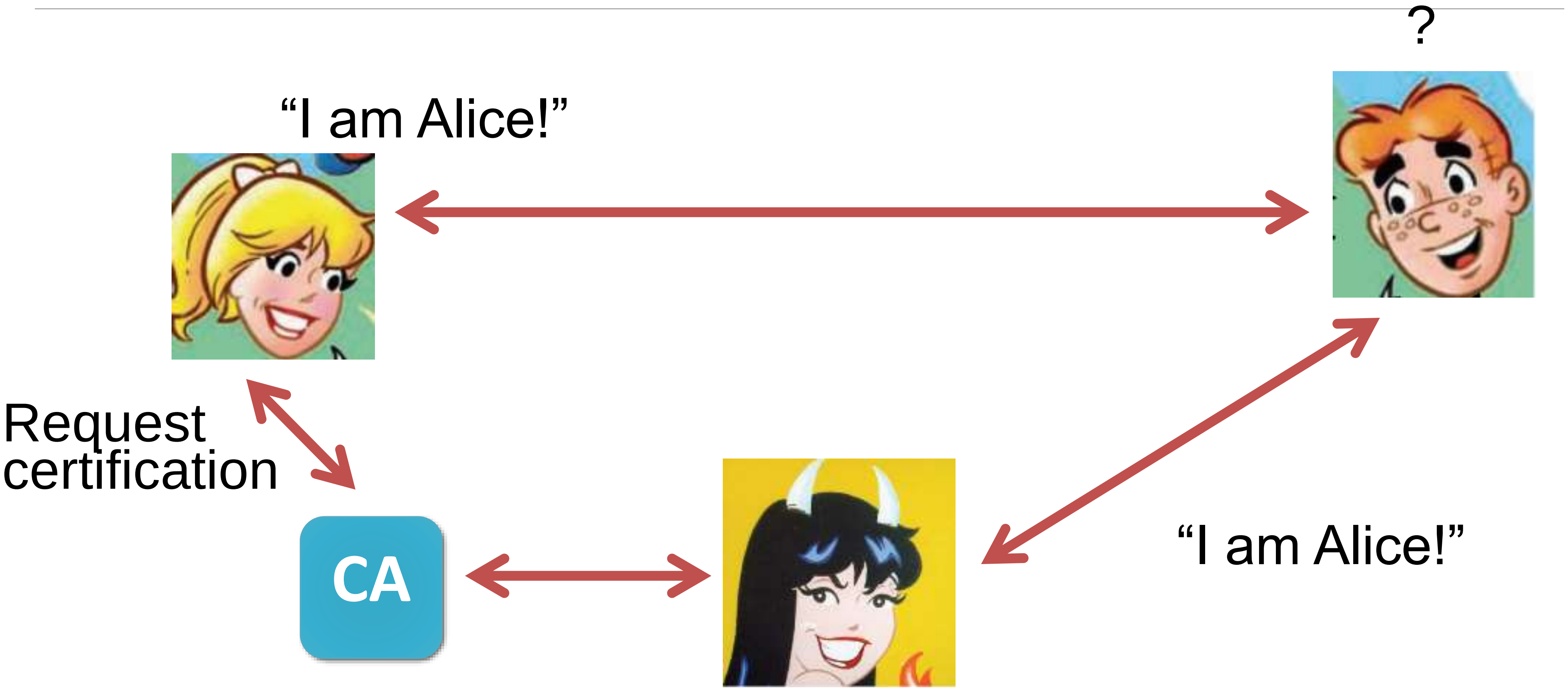
MARKULF KOHLWEISS

Some slides adapted from those by Myrto Arapinis, Kami Vaniea, Aggelos Kiayias, and Roberto Tamassia

Objective: point-to-point secure channel



Identification Problem



SSL/TLS

- Secure Sockets Layer: Developed by Netscape.
- SSL Version 3 released in 1996.
- Substituted by TLS 1.0 in 1999: (Transport Layer Security). Standardized by IETF. Published as RFC 2246
- TLS 1.3. Published as RFC 8445 in 2018.
- On top of TCP/IP, below application protocols.

Taher Elgamal

Source Alexander Klink via Wikipedia

An old story …

… with new twists

1976-1978
- New Directions in Cryptography
- Data Encryption Standard (DES)
- RSA

1994
- Netscape SSL
- SSL2

1995
- SSL3
- **Predictable IV (Rogaway)**

1996
- **MD5 (Dobbertin)**

1998
- **PKCS1 (Bleichenbacher)**

1999
- TLS 1.0

20

Comments P. Rogaway
draft-rogaway-ipsec-comments-00.txt UC Davis

STATU

ABSTR

Chosen
Based

Security Flaws Induced by CBC Padding

Lucky Thirteen: Breaking the TLS and DTLS Record Protocols

On the Security of RC4 in TLS[1]

Nadhem J. AlFardan
Information Security Group,

Daniel J. Bernstein
University of Illinois at Chicago and

Kenneth G. Paterson
Information Security Group,

Verifying TLS implementations

1999
- TLS 1.0

2002-2003
- CBC Padding (Vaudenay)

2006
- TLS 1.1

2008
- TLS 1.2
- Symbolic Model

2009
- Renegotiation

2011
- BEAST

2012-2013
- CRIME
- Lucky 13
- RC4
- Most Dangerous Code
- Implementation Attacks
- Cross Protocol Attacks

Renegotiating TLS

Marsh Ray
Steve Dispensa
PhoneFactor, Inc.

v1.1 November 4, 2009

Summary

Transport Layer Security (TL
is subject to a number of ser
renegotiation. In general, the
chosen plaintext into the beg

A Cross-Protocol Attack on the TLS Protocol

Nikos Mavrogiannopoulos
KU Leuven
ESAT/SCD/COSIC – IBBT
Leuven, Belgium
nikos@esat.kuleuven.be

Frederik Vercauteren
KU Leuven
ESAT/SCD/COSIC – IBBT
Leuven, Belgium
fvercaut@esat.kuleuven.be

Vesselin Velichkov
University of Luxembourg
Luxembourg
vesselin.velichkov@uni.lu

Bart Preneel
KU Leuven
ESAT/SCD/COSIC – IBBT
Leuven, Belgium
preneel@esat.kuleuven.be

Verifying TLS implementations

1999
- TLS 1.0

2002-2003
- **CBC Padding (Vaudenay)**

2006
- TLS 1.1

2008
- TLS 1.2
- **Symbolic Model**

2009
- **Renegotiation**

2011
- **BEAST**

2012-2013
- **CRIME**
- **Lucky 13**
- **RC4**
- **Most Dangerous Code**
- **Implementation Attacks**
- **Cross Protocol Attacks**

2013
- Implementing Verified Cryp Security Bhargavan, F **Kohlweiss**, F
- **Snowden**

Analyses the *entirety* of TLS using machine-assisted proof techniques.

- **ambitious** but **only real way.**
- **must not simplify** the underlying cryptography.

From a few thoughts on cryptographic engineering (2012): On the (provable) security of TLS

Toward TLS 1.3

2008
- TLS 1.2
- **Symbolic Model**

2009
- **Renegotiation**

2011
- **BEAST**

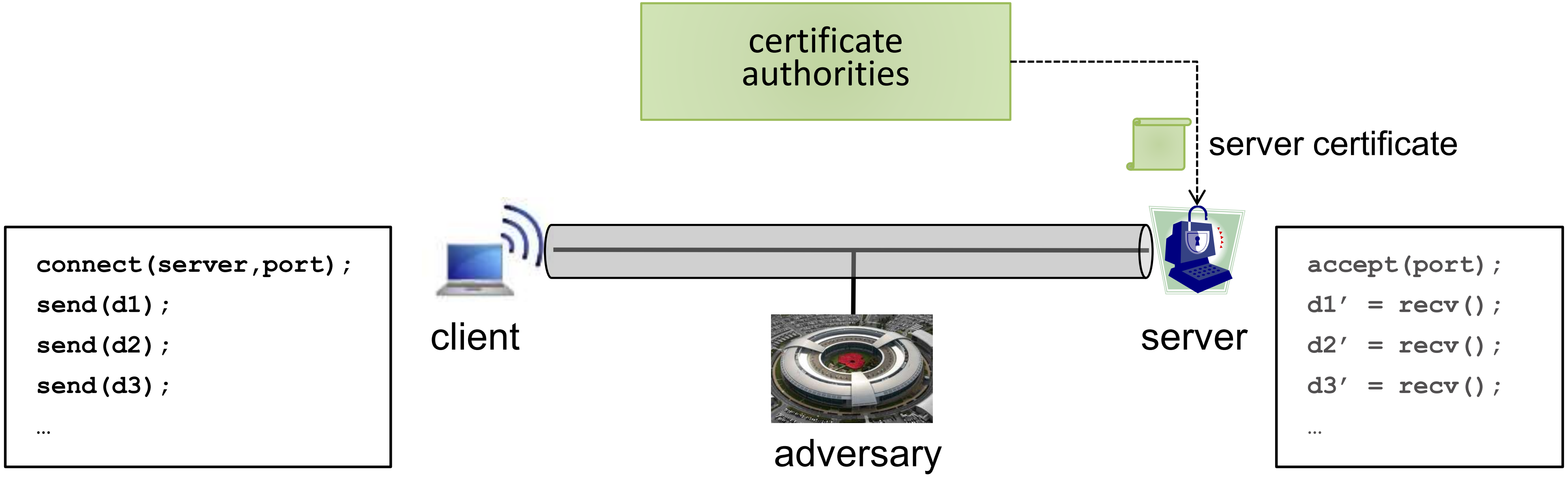
2012-2013
- **CRIME**
- **Lucky 13**
- **RC4**
- **Most Dangerous Code**
- **Implementation Attacks**
- **Cross Protocol Attacks**

2013
- Implementing TLS with Verified Cryptographic Security
  Bhargavan, Fournet, **Kohlweiss**, Pironti, Strub
- **Snowden**

2014
- Proving the TLS Handshake Secure (as it is)
  Bhargavan, Fournet, **Kohlweiss**, Pironti, Strub, Zanella-Beguelin
- **Triple Handshake attack**
- **New Bleichenbacher attacks**

```
Internet Engineering Task Force (IETF)                       E. Rescorla
Request for Comments: 8446                                       Mozilla
Obsoletes: 5077, 5246, 6961                                  August 2018
Updates: 5705, 6066
Category: Standards Track
ISSN: 2070-1721

        The Transport Layer Security (TLS) Protocol Version 1.3

Abstract

   This document specifies version 1.3 of the Transport Layer Security
   (TLS) protocol.  TLS allows client/server applications to communicate
   over the Internet in a way that is designed to prevent eavesdropping,
   tampering, and message forgery.

   This document updates RFCs 5705 and 6066, and obsoletes RFCs 5077,
   5246, and 6961.  This document also specifies new requirements for
   TLS 1.2 implementations.
```

Secure channels for the Web



Security Goal: As long as the client is honest and the adversary does not know the server's private key, it cannot

- Inject forged data into the data stream (integrity)
- Distinguish the data stream from random bytes (confidentiality)

Goals of SSL/TLS

- End-to End Confidentiality
  - Encrypt communication between client and server applications
- End-to-End Integrity
  - Detect corruption of communication between client and server applications
- Required server authentication
  - Identity of server always proved to client
- Optional client authentication
  - Identity of client optionally proved to server
- Modular deployment
  - Intermediate layer between application and transport layers
  - Handles encryption, integrity, and authentication on behalf of client and server applications

Certificates

- Public key certificate
  - Assurance by a third party that a public key is associated with an identity
  - E.g., QuoVadis certifies that the public key below is associated with University of Edinburgh web server
  - d6 23 7e f5 e7 56 2c e3 50 d7 e1 4e 98 f4 cc 97 61 b2 ae 07 b8 b8 3d 6e
    02 f4 9b c1 32 e5 56 bb 78 ea c0 2e 62 84 33 27 e4 2a 83 64 9f 53 cf e7
    04 92 3e 4b 6d f8 55 68 a3 40 21 ff 70 66 a6 a4 50 a8 d9 87 54 97 fe ee
    5a a4 b7 99 22 57 d2 df 84 35 5b 26 8c 09 2d 98 a9 74 0f e0 d9 d3 97 1d
    fd 80 8f 9c 5a e8 cc 78 0f 7b f7 95 2f 4f b4 07 cc 05 6d d3 0d 9a a4 37 fb
    ef 0d a8 b9 00 6f 4b d6 3f 80 04 38 09 9a e8 6e 27 aa a4 f1 20 7e 13 57
    f4 9b ca cb 7f 3c 93 4d 1f e6 55 a1 4e 7c e2 06 a5 4b 8e 20 25 5d 07 e8
    98 68 1a ea 0b cc fd 53 0a 66 93 be 31 b9 75 bb aa 04 b4 8f ac 56 8b 05
    4d e1 68 2e 65 04 b6 da 93 49 60 2c c7 d2 74 fa 34 da 70 8c 7d bb f2 e6
    b6 df 2a 8e 48 8f 15 ae 2f a0 ad 86 07 9c 9d c9 c0 1d 8b 06 b8 b9 ba

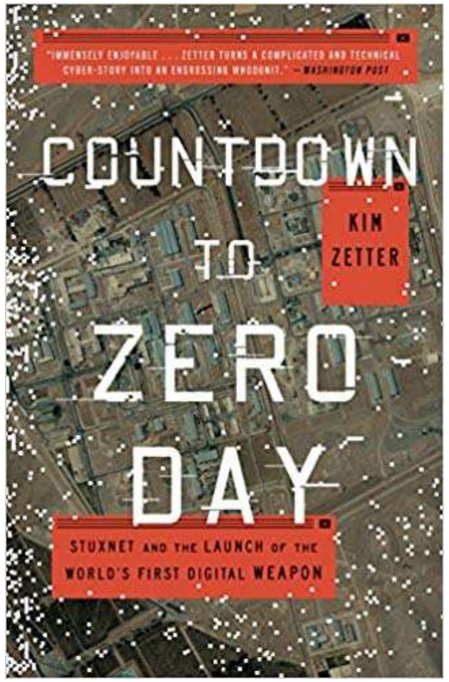
- Certificate fields
  - Issuer aka certificate authority
  - (QuoVadis)
  - Subject (University of Edinburgh; www.ed.ac.uk)
  - Subject's public key parameters (RSA2048)
  - Subject's public key
  - Validity period (29/11/16-29/11/19)
  - Signature parameters (SHA256; PKCS #1, RSA2048)
  - Signature

Chain of Trust and Revocation

- Transitive trust
  - Trust of (public key of) issuer implies trust of (public key of) subject
  - Issuer can be subject in another certificate
  - Chain of certificates
  - Root of trust?
  - Root certificates preconfigured in operating system and browser
- Certificate revocation
  - Mechanism to invalidate a previously issued certificate
  - E.g., when private key of the subject is compromised
- Revocation methods
  - List of revoked certificates posted on CA's website
  - Online verification service provided by CA (OCSP stapling)

Rogue Certificates: DigiNotar hacked in 2011

- Google noted a DigiNotar issued certificate for google.com that wasn’t on Google’s own list of Google certificates
- Used to impersonate Google mail web site and collect usernames and passwords
- Serious impact on Dutch government IT services
  - http://www.onderzoeksraad.nl/uploads/items-docs/1833/Rapport_Diginotar_EN_summary.pdf
- DigiNotar went bankrupt
- Do you know the CAs you are trusting?

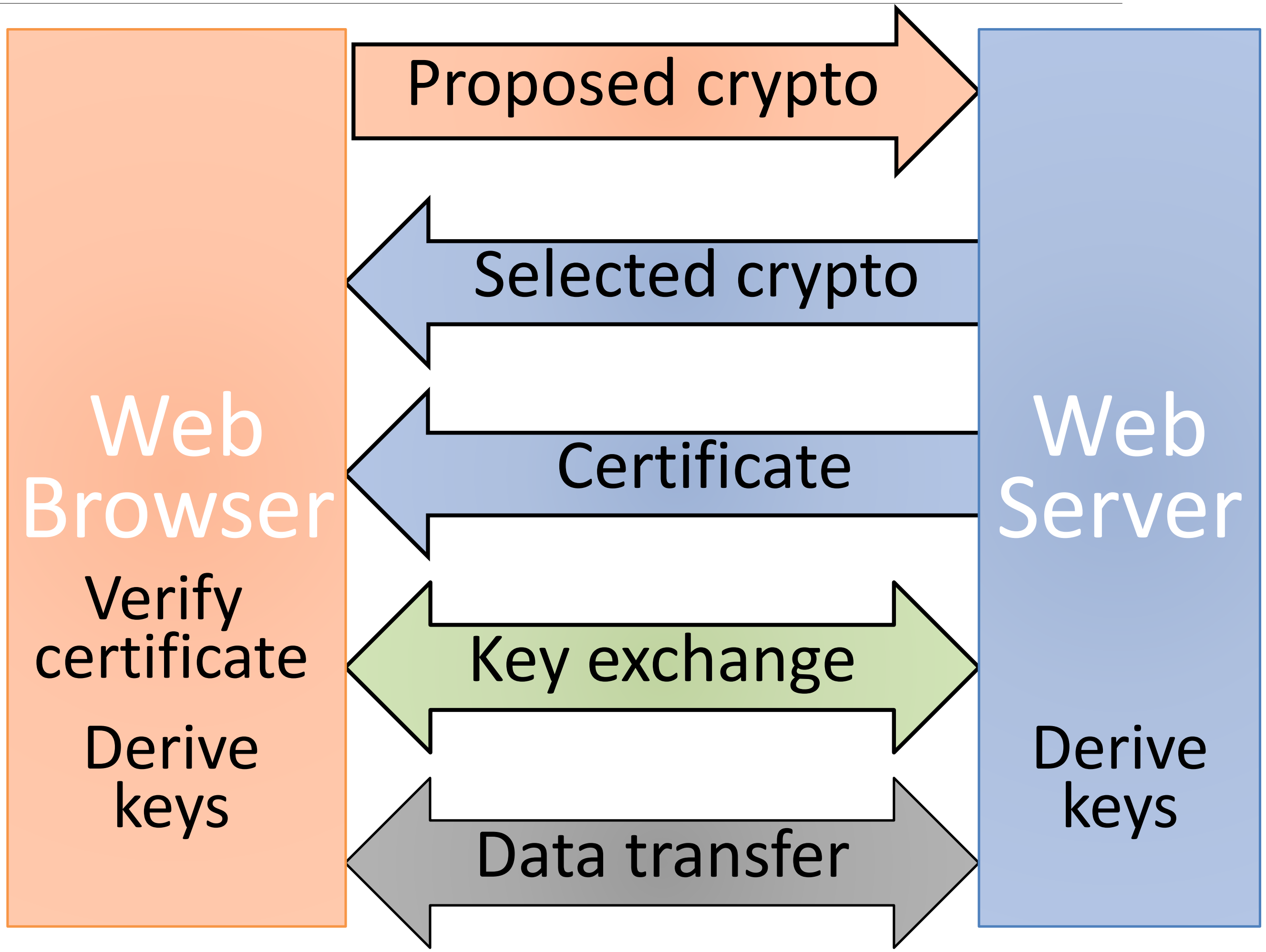


TLS Building Blocks

	Confidentiality	Integrity	Authentication
Setup	Public-key based key-exchange (RSA and DH)	Public-key digital signature (e.g., RSA)	Public-key digital signature (e.g., RSA)
Data transmission	Symmetric encryption (e.g., AES in CBC mode)	Hash-based MACs (e.g., HMAC using SHA256)	

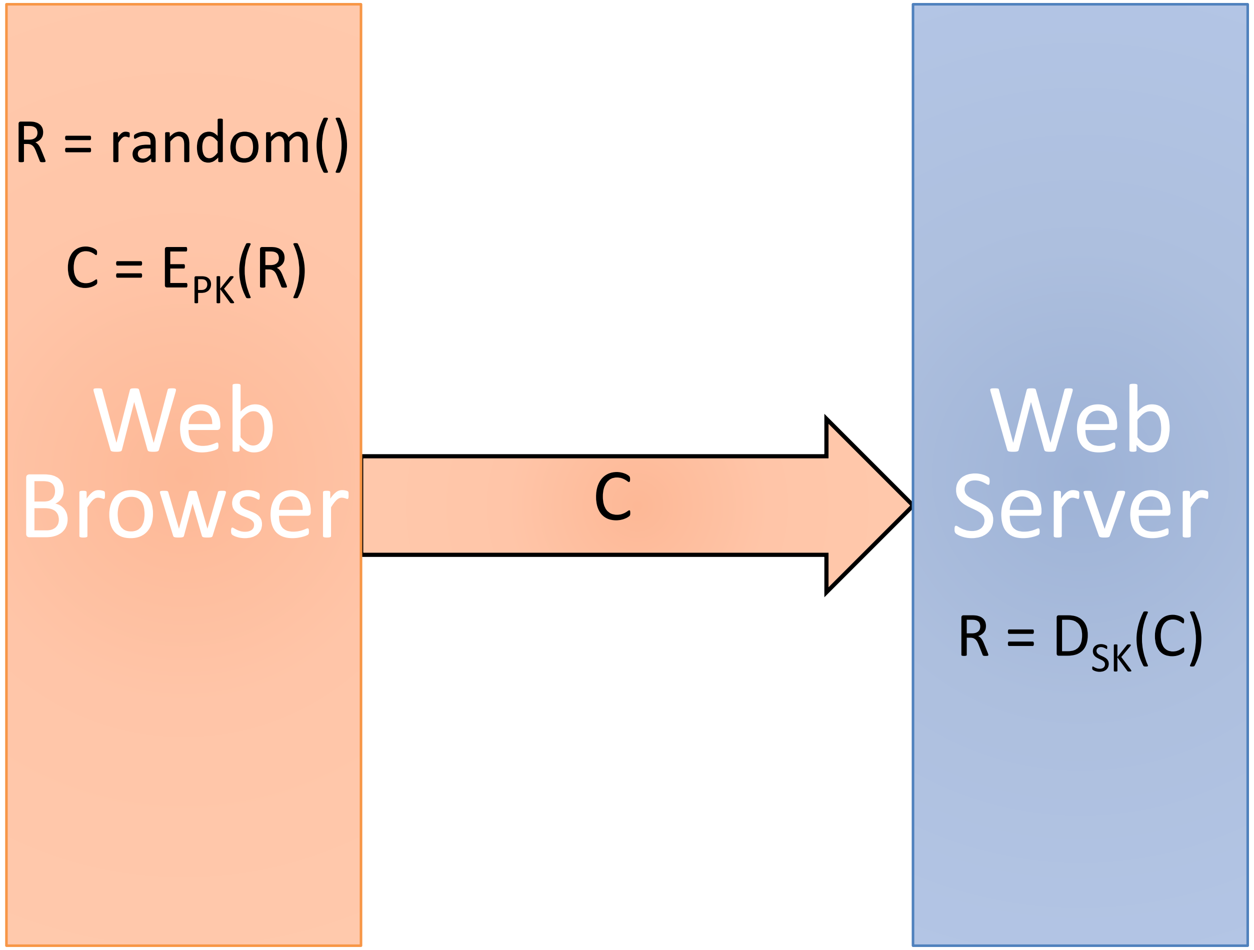
TLS Overview (Simplified)

- Browser sends supported crypto algorithms
- Server picks strongest algorithms it supports
- Server sends certificate (chain)
- Client verifies certificate (chain)
- Client and server agree on secret value R by exchanging messages
- Secret value R is used to derive keys for symmetric encryption and hash-based authentication of subsequent data transfer



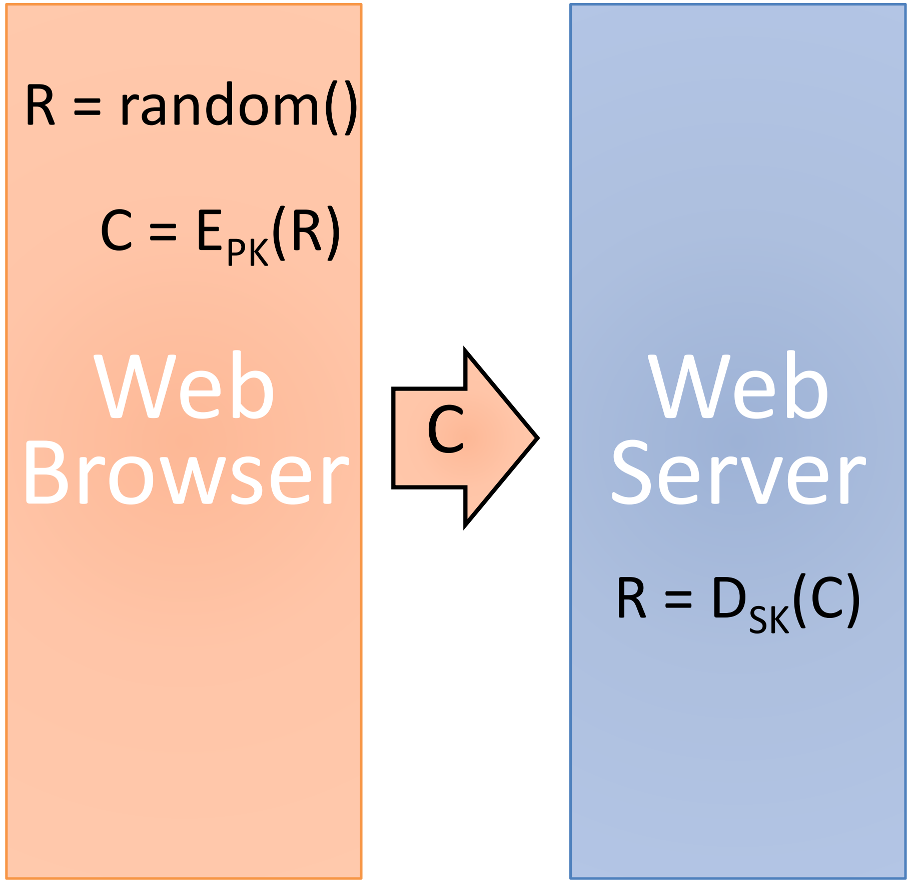
Basic Key Exchange

- Called RSA key exchange for historical reasons
- Client generates random secret value R
- Client encrypts R with public key, PK, of server $C = E_{PK}(R)$
- Client sends C to server
- Server decrypts C with private key, SK, of server
  $R = D_{SK}(C)$



Forward Secrecy

- Compromise of public-key pairs private keys does not break confidentiality of past messages
- TLS with basic key exchange does not provide forward secrecy
  - Attacker eavesdrop and stores communication
  - If server’s private key is compromised, attacker finds secret value R in key exchange and derives encryption keys

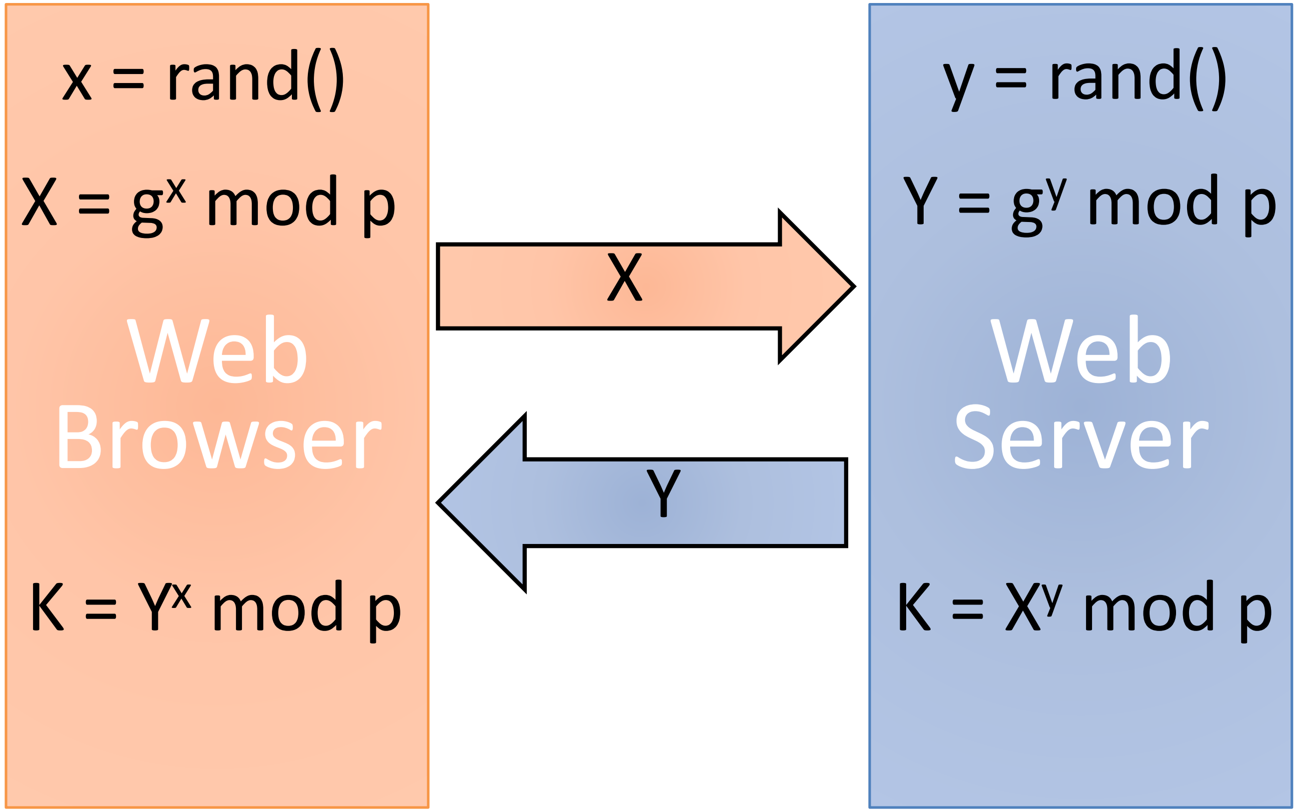


Source ACM

Achieves forward secrecy
Diffie Hellman Key Exchange

Source ACM

- Public parameters: prime p and generator g of Z_p
- Client generates random x and computes $X = g^x \bmod p$
- Server generates random y and computes $Y = g^y \bmod p$
- Client sends X to server
- Server sends Y to client
- Client and server compute $K = g^{xy} \bmod p$



Attacker in the Middle

Web Browser

$X = g^x$

$S = g^s$

$T = g^t$

$Y = g^y$

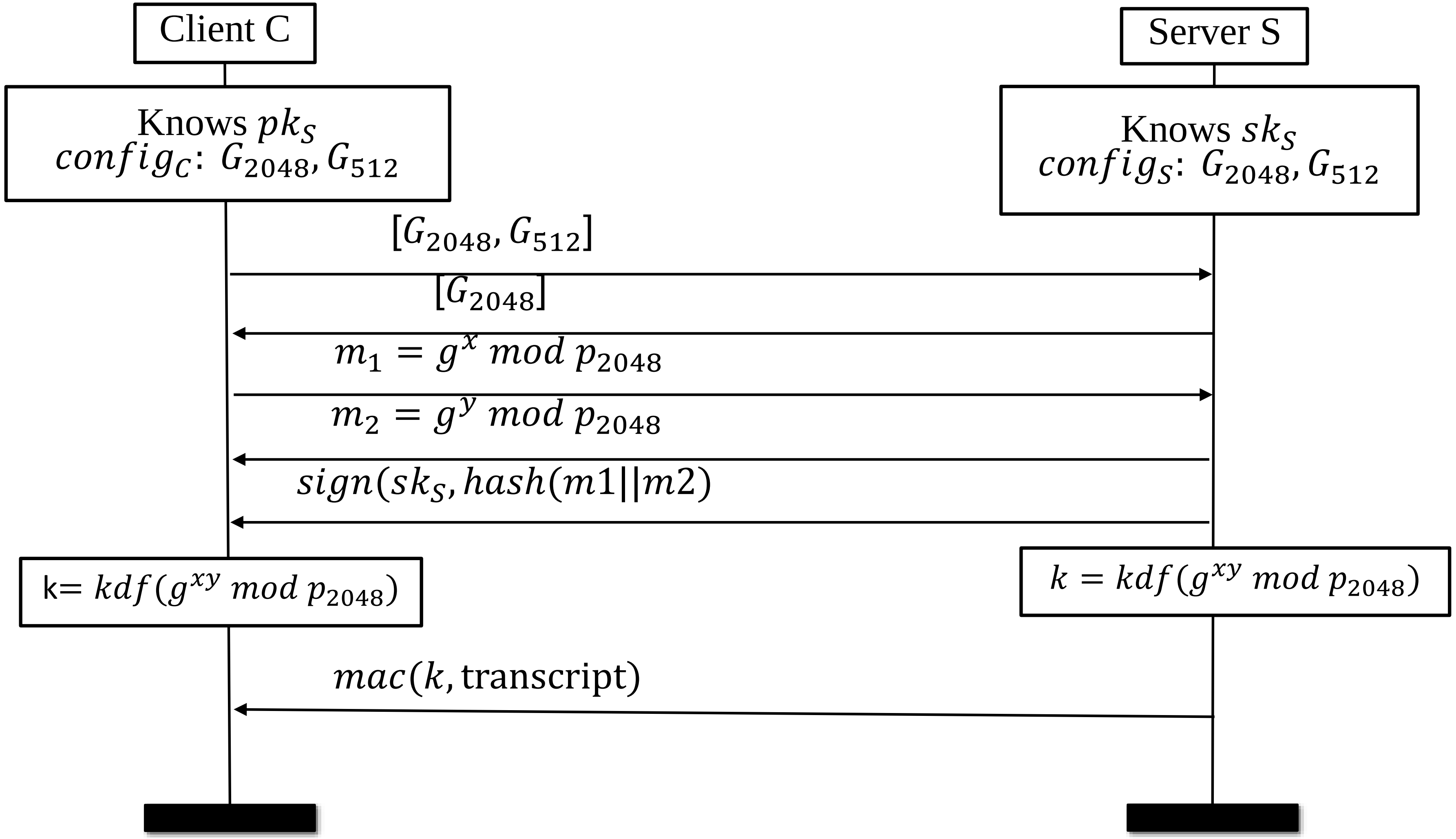
Web Server

$K_1 = g^{xs}$ $K_1 = g^{xs}$ $K_2 = g^{yt}$ $K_2 = g^{yt}$

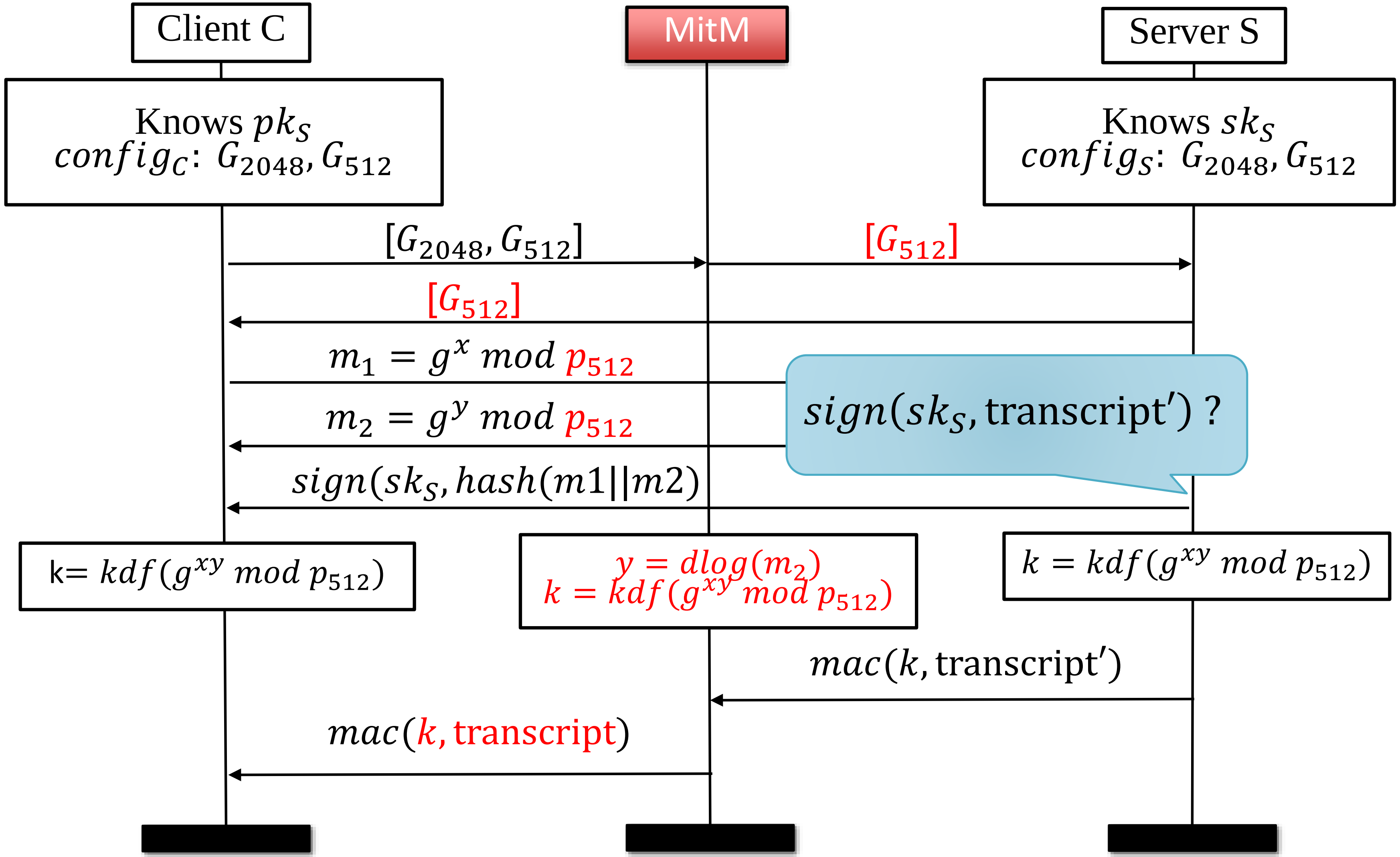
Solution

- Browser and server send signed X and Y
- Requires each to know the public key of the other
- One sides authentication if only server signs

Signed DH key exchange



LOGJAM



Client C
MitM
Server S
Knows pk_S
$config_C$: G_{2048}, G_{512}
Knows sk_S
$config_S$: G_{2048}, G_{512}
$[G_{2048}, G_{512}]$
$[G_{512}]$
$[G_{512}]$
$m_1 = g^x \ mod \ p_{512}$
$m_2 = g^y \ mod \ p_{512}$
$sign(sk_S, transcript')$?
$sign(sk_S, hash(m1||m2))$
k= $kdf(g^{xy} \ mod \ p_{512})$
$y = dlog(m_2)$
$k = kdf(g^{xy} \ mod \ p_{512})$
$k = kdf(g^{xy} \ mod \ p_{512})$
$mac(k, transcript')$
$mac(k, transcript)$

Back to the roots $\sqrt[e]{c}$ [RSA78]

There was RSA

$$E(m) = m^e \bmod n$$

$n = pq$ – product of primes

RSA is homomorphic

$$(m^e \cdot s^e)^d = (m \cdot s)^{ed} = m \cdot s$$

Padding Attacks [B98]

Bleichenbacher Attack on PKCS#1:

Pick s_i adaptively. For accepted $c \times s_i^e$ attacker knows that

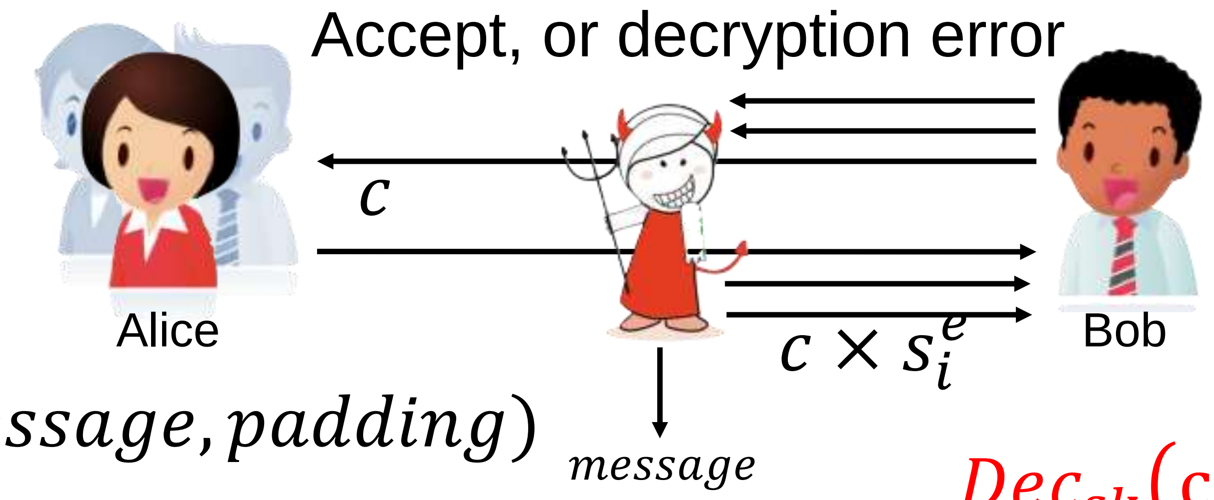
$$M \times s_i = (01\ 02\ \ *******\ 00\ \ *******).$$

Build system of inequalities:

$$(01\ 02\ 00 \ldots 00) \le M \times s_i \ \le (01\ 02\ \mathrm{ff} \ldots \mathrm{ff})$$

PKCS#1 standard for RSA:

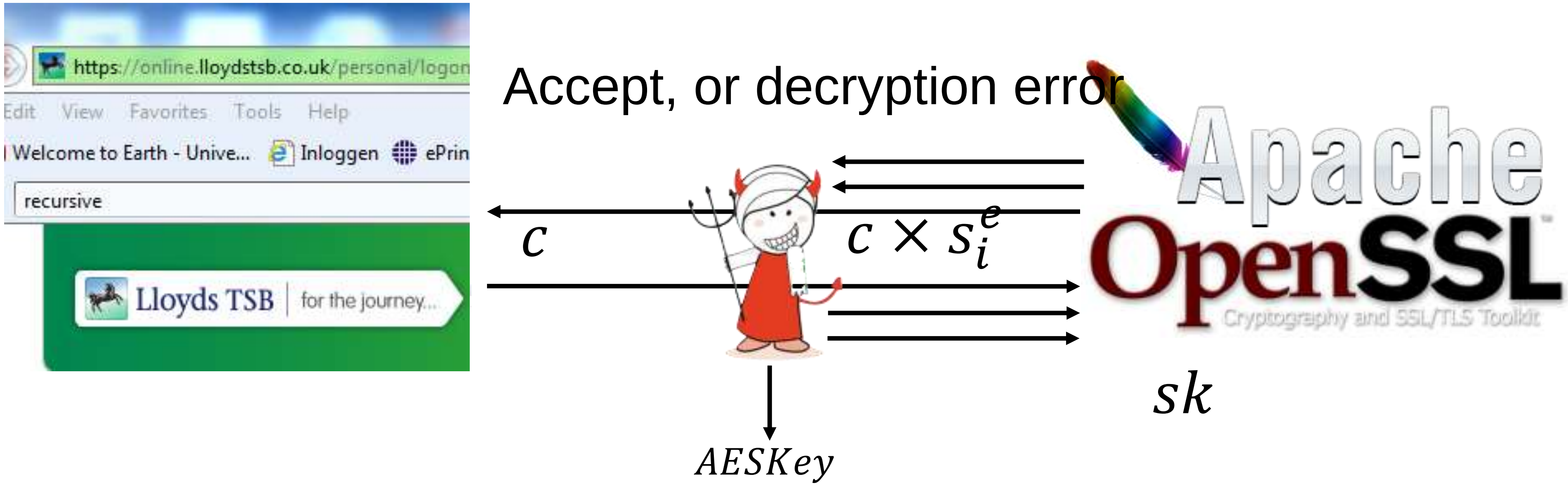
$Enc_{(n,e)}(\text{message}, \text{padding}) = M^e$ where $M = (01\ 02\ \text{padding}\ 00\ \text{message})$



$$padding \leftarrow P$$
$$c = Enc_{pk}(message, padding)$$

$$Dec_{sk}(\mathrm{c} \times s_i^e) =?$$

Padding Attacks [B98]



What We Have Learned

- Goals and history of the SSL/TLS protocol
- TLS Certificates, chain of trust, and revocation
- Overview of the TLS protocol
- Diffie-Hellman key exchange and forward secrecy
- Logjam and Bleichenbacher attack

References

- RFC 8445 - The Transport Layer Security (TLS) Protocol Version 1.3 (2018)
- Logjam attack (2015)