IPsec: Key Management (IPSEC: IKE)

The Internet Key Exchange (IKE) protocol

- IKE establishes not just keys, but Security Associations:
  - the protocol format used (many),
  - the cryptographic and hashing algorithm used, and, of course, keys...
- IKE is very flexible.
  - E.g., supports authentication based on a variety of pre-shared secrets (master keys).
- But also very complex.
  - Many options, alternative subprotocols, ...

IKE — a bit of history

- IKE evolved from a number of different protocols, including:
  - **ISAKMP (Internet Security Association and Key Management Protocol)**:
    - provides a framework and a generic negotiation *protocol for establishing SAs and cryptographic keys*,
      - but does not prescribe any particular authentication mechanism.
  - **Oakley**: a suite of *key agreement protocols* in which two parties generate a key jointly.
- Roughly speaking, IKE combines packet formats of ISAKMP and exchanges of OAKLEY, which are both based on *Diffie-Hellman*.
  - Bottom line: 2x Diffie-Hellman (+ extensions).

Perfect Forward Secrecy (PFS)

- Diffie-Hellman protocol: if you already share keys, why bother with Diffie-Hellman?
- Answer:

Perfect forward Secrecy (PFS)

A property of key-agreement protocols ensuring that a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in the future.

- The key used to protect transmission of data must not be used to derive any additional keys, and if the key used to protect transmission of data is derived from some other keying material, then that material must not be used to derive any more keys.
- In this way, compromise of a single key permits access only to data protected by that single key.

Perfect Forward Secrecy (PFS)

- The trick to achieving PFS is to generate a temporary session key, not derivable from information stored at the node after the session concludes, and then forget it after the session concludes.
- Example without PFS: SSL.
  - If server's private key is compromised, then past sessions can be decrypted.
- Moreover, periodic generation of new keys and keying material complicates cryptanalysis.

The IKE protocol — some details

IKE is a protocol suite for establishing and maintaining SAs.

- Typical key establishment protocol: one phase, in which two parties use master keys to establish shared keying material.
  - **Master keys**: pre-shared secret (symmetric) keys, public encryption key (used only for en/decryption), or public signature key (restricted to signing and signature verification).
- IKE proceeds in two such phases:
  - **Phase 1**: two parties negotiate an SA.
    - They agree on keying material and mechanisms (e.g., crypto-algorithms, hash functions, ...) to use in phase 2.
  - **Phase 2**: the SA of phase 1 is used to create "child SAs" to encrypt and authenticate further communications.

We consider just Phase 1 here.

IKE phase 1

Establish a secure channel so that phase 2 negotiations can occur privately.

- Negotiate a “ISAKMP” SA using Diffie-Hellman (DH) exchange with:
  - **Cookies**: to protect against DOS attacks.
  - **Identifiers**: to name partners.
  - **Authenticators**: data computed by a MAC, *h*(*key* , *data*).
- Typically (= main mode), exchange of authenticated IDs occurs after DH exchange, so that DH key material can be used to encrypt the messages in the authentication phase.

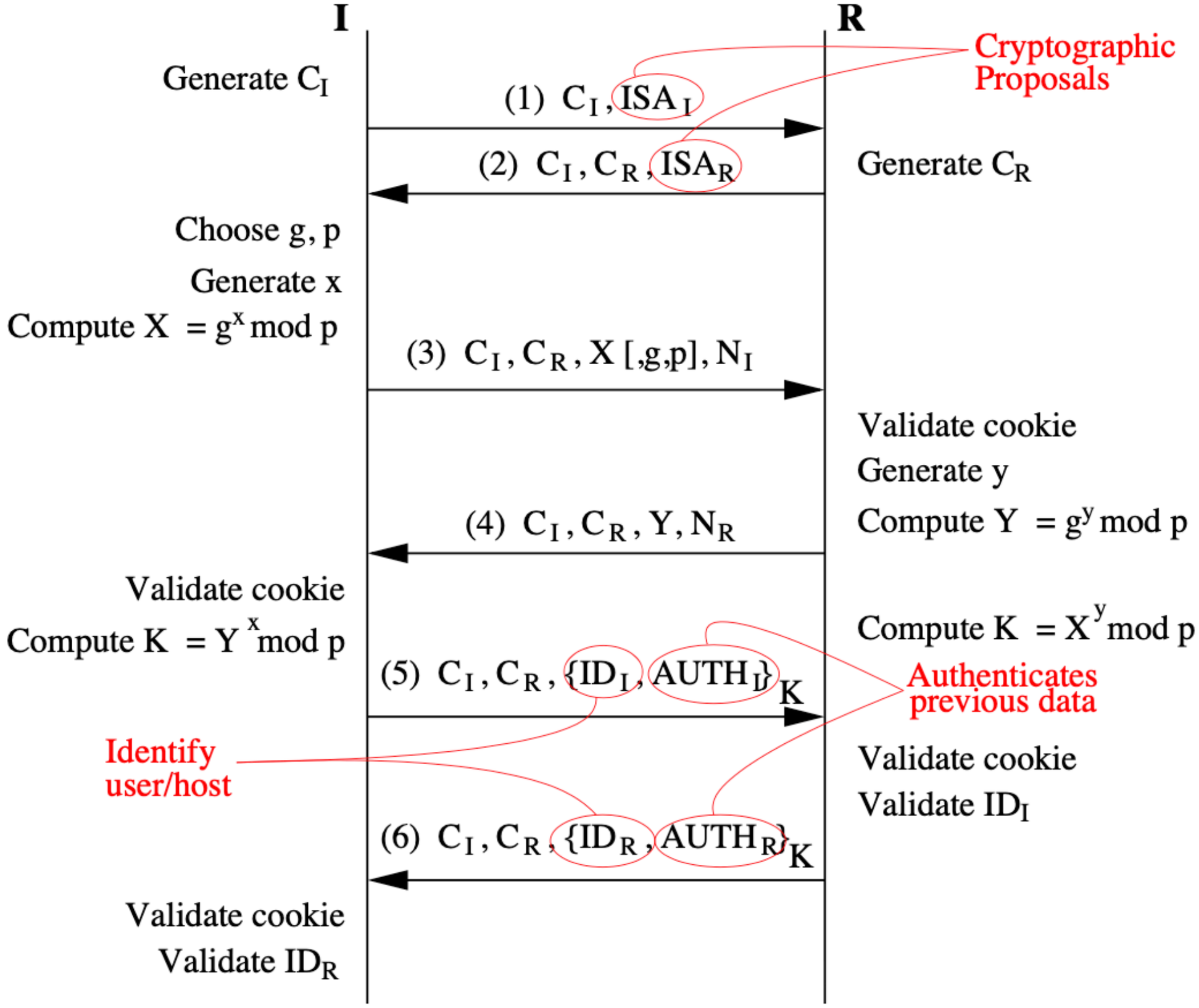
IKE phase 1 (cont.)

Phase 1 offers two modes (both resulting in the desired SA):

- **Main mode**:
  - 6 messages exchanged between I and R.
  - Offers identity protection and considerable flexibility in terms of the parameters and configurations that can be negotiated.
- **Aggressive mode**: is faster but without protection.
  - 3 messages exchanged between I and R.
  - This exchange occurs without identity protection, except when public-key encryption is used for authentication.

- Each mode has 4 variants, based on the authentication method (pre-shared key, digital signatures, and 2 public-key variants). The derivation of keying material and message contents depend on the authentication scheme used.

IKE phase 1, main mode (simplified & schematic)

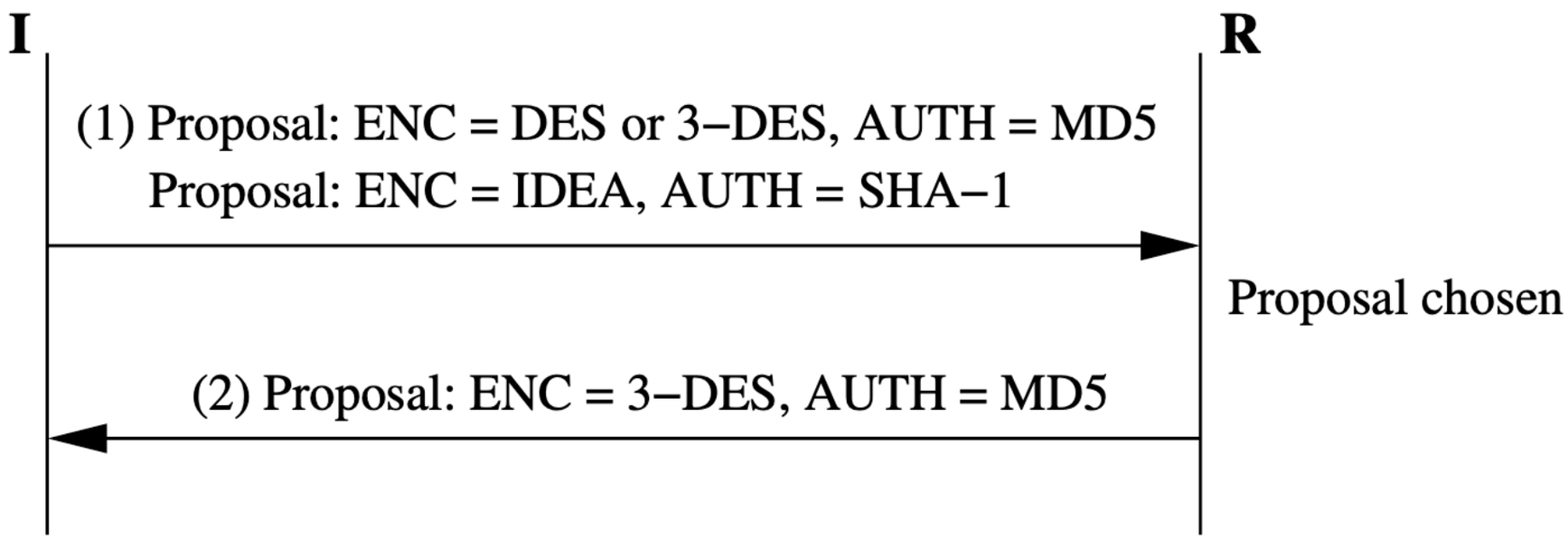


IKE phase 1, main mode (cont.)

$$(1) \quad I \rightarrow R: \quad C_I, ISA_I$$
$$(2) \quad R \rightarrow I: \quad C_I, C_R, ISA_R$$

1. ISA_I (ISAKMP SA data from I) includes a series of proposals and a list of algorithms for each proposal.
2. ISA_R (ISAKMP SA data from R) consists of the proposals that were accepted and the algorithm chosen for each.

Example negotiation (fails if R does not support any of I's proposals):



IKE phase 1, main mode (cont.)

$$
\begin{array}{lll}
(1) & \mathrm{I} \rightarrow \mathrm{R}: & \mathrm{C_I},\ \mathrm{ISA_I} \\
(2) & \mathrm{R} \rightarrow \mathrm{I}: & \mathrm{C_I},\ \mathrm{C_R},\ \mathrm{ISA_R} \\
(3) & \mathrm{I} \rightarrow \mathrm{R}: & \mathrm{C_I},\ \mathrm{C_R},\ \mathrm{X}[,\mathrm{g},\mathrm{p}],\ \mathrm{N_I} \\
(4) & \mathrm{R} \rightarrow \mathrm{I}: & \mathrm{C_I},\ \mathrm{C_R},\ \mathrm{Y},\ \mathrm{N_R} \\
(5) & \mathrm{I} \rightarrow \mathrm{R}: & \mathrm{C_I},\ \mathrm{C_R},\ \{\mathrm{ID_I},\ \mathrm{AUTH_I}\}_\mathrm{K} \\
(6) & \mathrm{R} \rightarrow \mathrm{I}: & \mathrm{C_I},\ \mathrm{C_R},\ \{\mathrm{ID_R},\ \mathrm{AUTH_R}\}_\mathrm{K}
\end{array}
$$

- Messages (3) and (4) contain the half-keys and nonces of I and R.
- Messages (5) and (6) are encrypted with the algorithm negotiated by the ISAKMP SA of (1) and (2)
  - Key K determined from Diffie Hellman *k*.
  - ID_I and ID_R each identify a user or host (e.g., an IP address, fully-qualified domain name, certificate, or email address).
  - $AUTH_I$ and $AUTH_R$ is authentication data, depending on variant used, e.g., MAC or signature.

IKE phase 1, main mode (cont.)

Concrete example of authentication variant using digital signatures:

$$
\begin{array}{lll}
(1) & \mathrm{I} \rightarrow \mathrm{R}: & \mathrm{C_I},\ \mathrm{ISA_I} \\
(2) & \mathrm{R} \rightarrow \mathrm{I}: & \mathrm{C_I},\ \mathrm{C_R},\ \mathrm{ISA_R} \\
(3) & \mathrm{I} \rightarrow \mathrm{R}: & \mathrm{C_I},\ \mathrm{C_R},\ g^x,\ \mathrm{N_I} \\
(4) & \mathrm{R} \rightarrow \mathrm{I}: & \mathrm{C_I},\ \mathrm{C_R},\ g^y,\ \mathrm{N_R} \\
(5) & \mathrm{I} \rightarrow \mathrm{R}: & \mathrm{C_I},\ \mathrm{C_R},\ \{\mathrm{ID_I},\ \mathrm{SIG_I}\}_{\mathrm{SKEYID_e}} \\
(6) & \mathrm{R} \rightarrow \mathrm{I}: & \mathrm{C_I},\ \mathrm{C_R},\ \{\mathrm{ID_R},\ \mathrm{SIG_R}\}_{\mathrm{SKEYID_e}}
\end{array}
$$

$SKEYID = h(\{N_I, N_R\}, g^{xy})$ (h is keyed hash)

$SKEYID_d = h(SKEYID, \{g^{xy}, C_I, C_R, 0\})$ deriving key

$SKEYID_a = h(SKEYID, \{SKEYID_d, g^{xy}, C_I, C_R, 1\})$ authentication key

$SKEYID_e = h(SKEYID, \{SKEYID_a, g^{xy}, C_I, C_R, 2\})$ encryption key

$HASH_I = h(SKEYID_a, \{g^x, g^y, C_I, C_R, ISA_I, ID_I\})$

$HASH_R = h(SKEYID_a, \{g^y, g^x, C_R, C_I, ISA_R, ID_R\})$

$SIG_I = \{HASH_I\}_{\{K_I\}^{-1}}$

$SIG_R = \{HASH_R\}_{\{K_R\}^{-1}}$

IKE phase 2

Protected by the SA established in phase 1, establish new SAs that can be used to protect “real” communication (i.e., the IPsec SA).

- **Quick mode**:
  - Establish new SAs (e.g., for ESP or AH).
- **New Group mode**:
  - Negotiate new parameters of the SAs (e.g., change cryptographic group).
  - This mode does not establish new SAs (and it does not replace or preempt quick mode).

IKE phase 2, quick mode (simplified)

- If perfect forward secrecy is required, quick mode supports a new DH exchange (optional message parts).

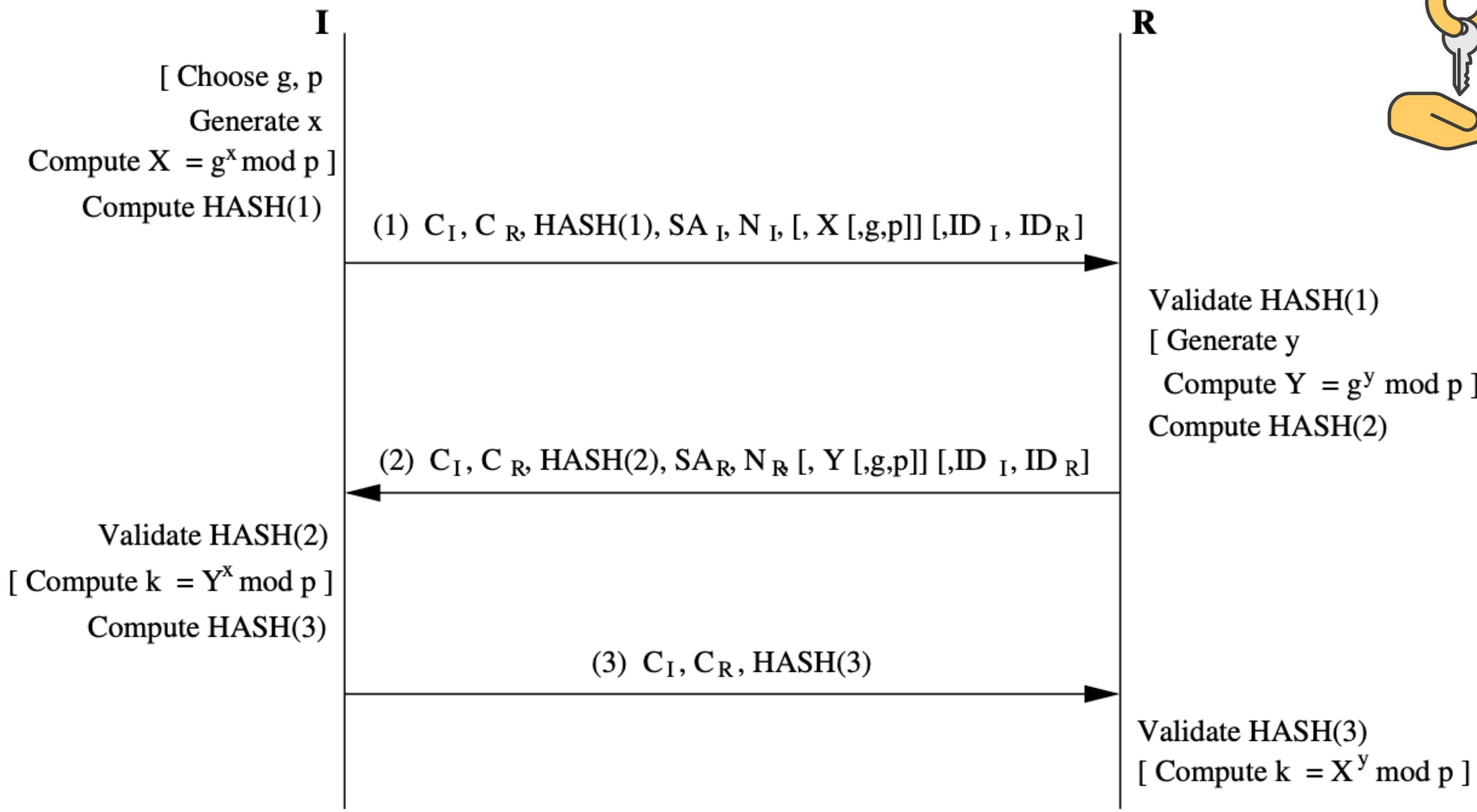
- Since multiple quick mode negotiations may take place simultaneously between two participants, and since each uses the cookie pair of phase 1, each is assigned a unique identifier.

- Encryption uses the data of phase 1.
  The new key for use with the newly established SA is

$$NEWKEY = h(SKEYID_d, \{k,protocol,SPI,N_I,N_R\})$$

where *protocol* and *SPI* are the protocol and SPI fields of the ISAKMP proposal payload, and *k* is a key produced by the DH exchange for perfect forward secrecy.

IKE phase 2, quick mode (simplified, cont.)



$$HASH(1) = h(SKEYID_a,\{SA_I,N_I,X,ID_I,ID_R\})$$

$$HASH(2) = h(SKEYID_a,\{SA_R,N_I,N_R,Y,ID_I,ID_R\})$$

$$HASH(3) = h(SKEYID_a,\{0,N_I,N_R\})$$

SSL

SSL/TLS

The SSL protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that prevents eavesdropping, tampering, or message forgery.

RFC 2246

- Goals: secrecy, integrity, (optionally mutual) authentication. Works by setting up a one (or two) way authentic channel for secret communication using public-key certificates.

- Original protocol (SSL 2.0) developed by Netscape (1995) to allow credit cards to be exchanged over the Internet. Current version TLS, based on SSL 3.1.

- Used for many applications, e.g., HTTP(S), SMTP, IMAP, ...

Protocol itself

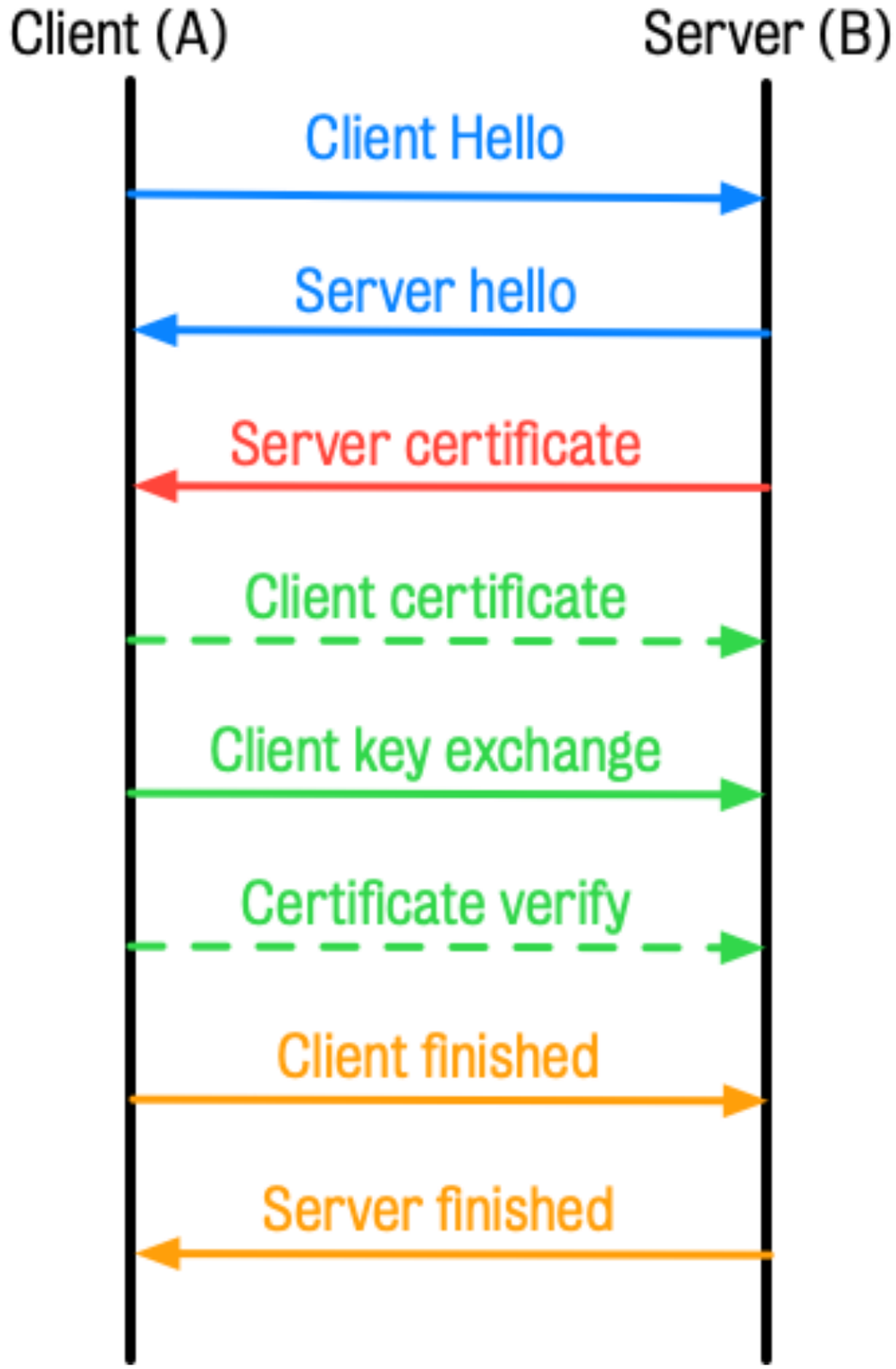
- Consists of various subprotocols, which run on top of TCP.
  - **SSL Handshake**: initiates (or reinitialises existing) connection. Establishes channel with desired channel properties.
  - **SSL Record**: protocol for sending application data. Describes way in which application data is compressed, authenticated with a MAC, and resulting payload is encrypted.
  - **Other protocols**: for renegotiating ciphers and error recovery.
- Although protocol is complex, underlying concepts fairly simple.
  - An **SSL session** is an association between the client and server with an associated state that specifies encryption method, client and server MAC secrets, encryption keys, etc.
  - An **SSL connection** is basically a secure stream within a session.

SSL Handshake

Presentation
(quite simplified):

1. Establish security capabilities.
   Negotiate ciphers (DH, RSA, ...).
2. Exchange server certificate.
3. Client key exchange. Optional authentication of client using client certificate.
4. Finish, establishing session.

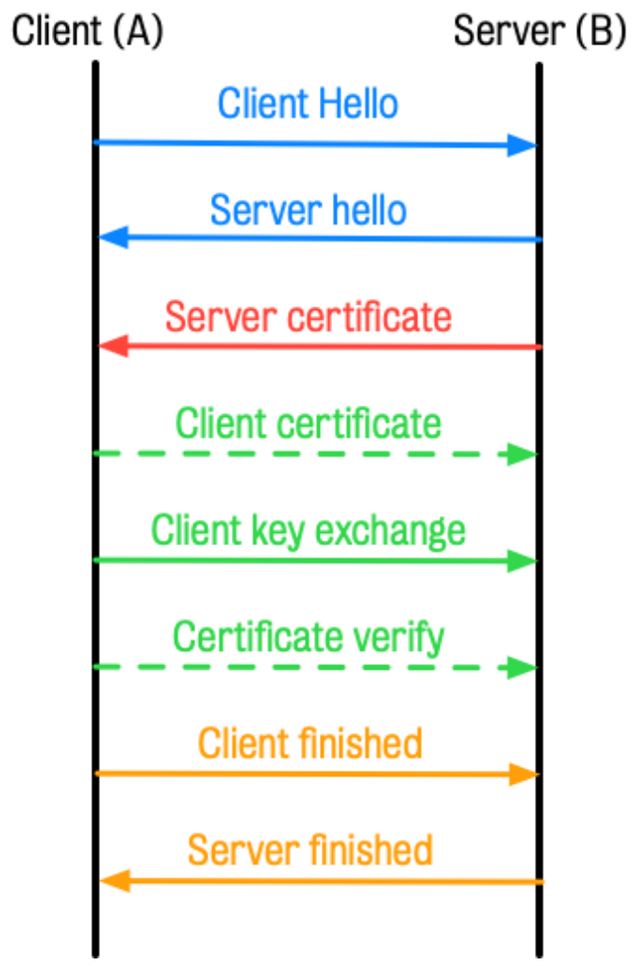
Let us look at these steps in more detail.



Handshake — hello

client hello A → B: A, Na, Sid, Pa

server hello B → A: Nb, Sid , Pb



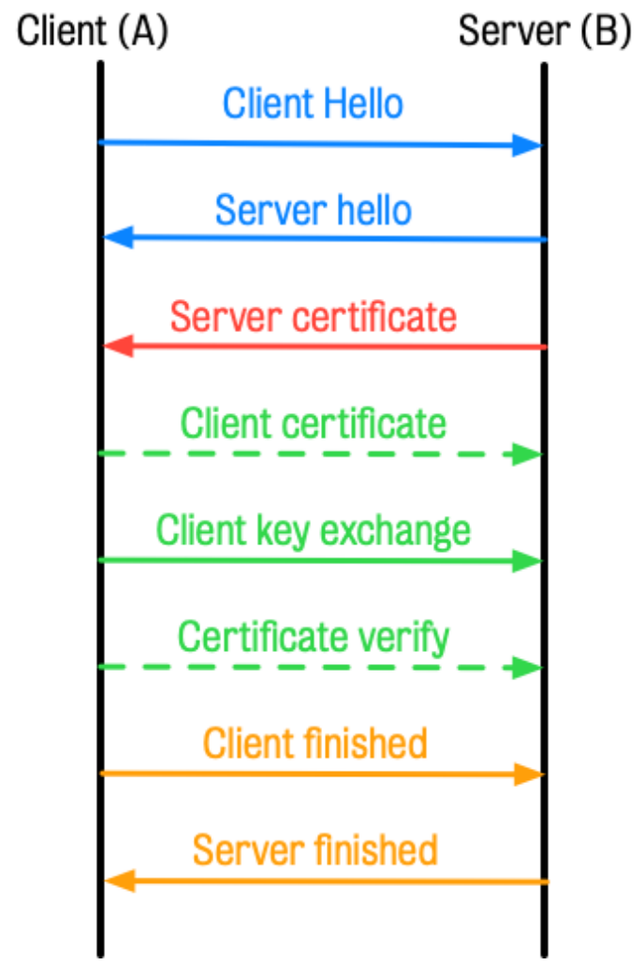
- *A* identifies client.
  - Actual standard doesn't include this.
  - In practice, IP address used from TCP.

- *Sid* is session identifier.
- *Pa* is list of *A*'s preferences for encryption and compression.
  - E.g., Diffie-Hellmann with signatures, RSA key exchange, ...
  - Such negotiation common due to (past) US export controls.

- *Pb* is chosen from *Pa*.
- Items (e.g., *Pa* and *Pb*) unprotected. Authenticated later.

Handshake — server certificate

server certificate $B \rightarrow A$: $certificate(B, K_B)$



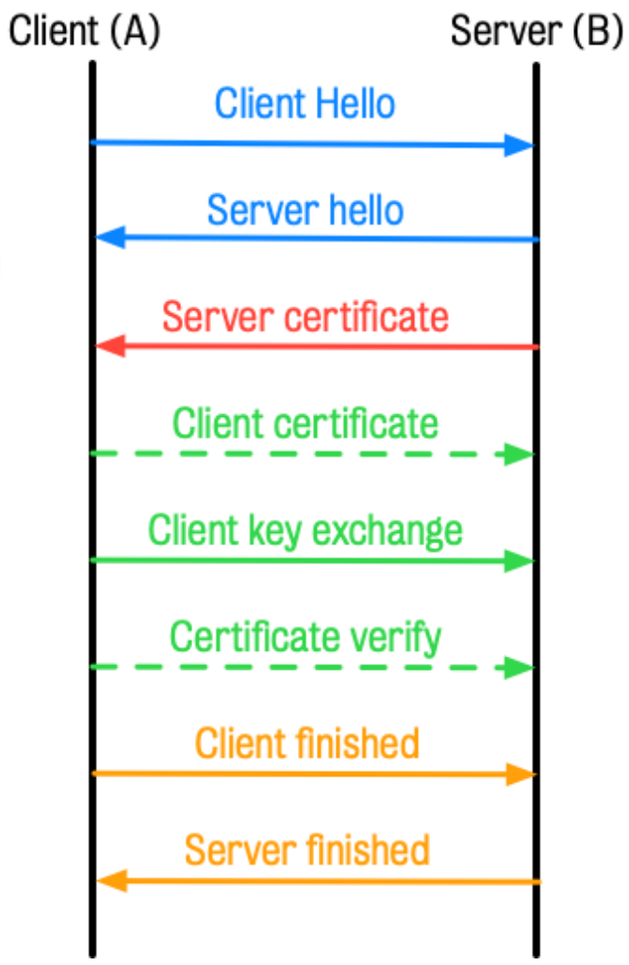
- Certificate: X.509v3, signed by trusted 3rd party. Here details other than the name and public key are omitted.
- In what follows, we will make additional, slightly simplifying assumptions for initial key exchange (next slide).
  - Actual protocol allows additional key-exchange message, needed if negotiating, e.g., a DH key.
  - In full protocol (here omitted), server may also request a certificate from client too. Rarely used in practice as it requires clients have personal public key certificates.

Handshake — client exchange

client certificate (opt) $A \rightarrow B$: certificate(A, K_A) {PM

client key exchange $A \rightarrow B$: $\{PMS\}_K$

certificate verify (opt) $A \rightarrow B$: $\{hash(\ldots)\}_{\{K_A\}^{-1}}$

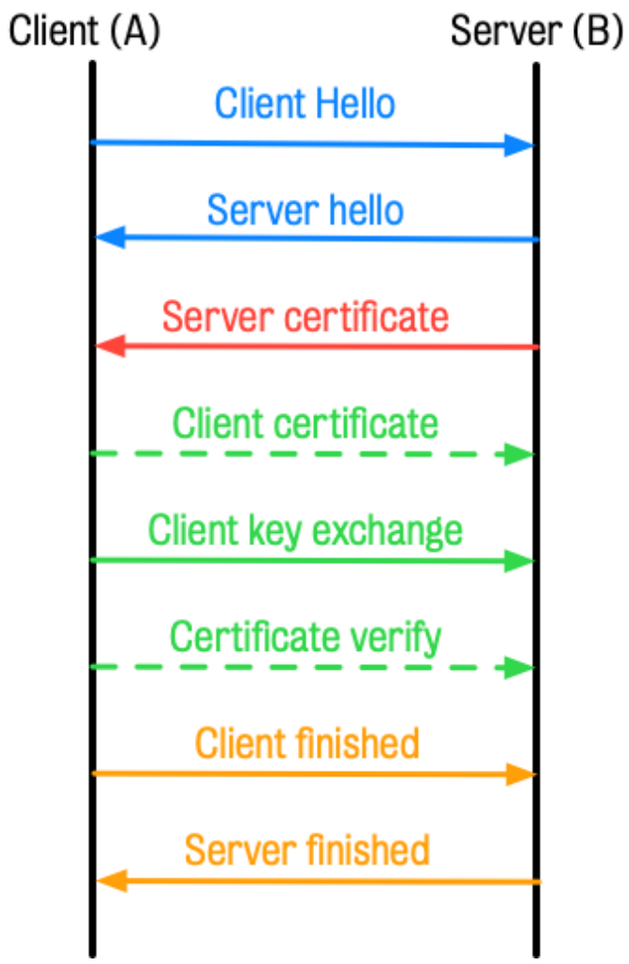


- *PMS* is a *pre-master secret* used to compute a master secret *M*, which is in turn used to generate various keys.
  E.g., *M* = *PRF* (*PMS*, *Na*, *Nb*), where *PRF* is a pseudo-random function, e.g., constructed from a MAC.
- The first and third messages authenticate the client. Client sends certificate and *hash*(. . .), computed using all previous messages exchanged

Handshake — finish

client finished A → B: {Finished}$_{clientK}$

server finished B → A: {Finished}$_{serverK}$

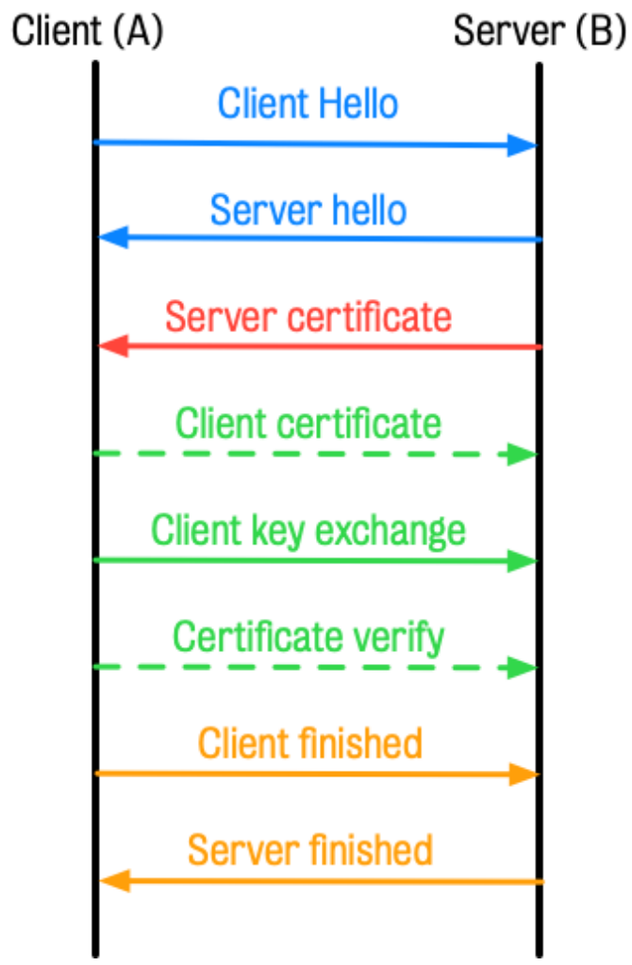


- *Finished* is hash of previous messages: *M* , *Na*, . . . Guarantees integrity of previous items. E.g., prevents downgrading attack on chosen ciphers or man-in-the-middle attack.
- *clientK* and *serverK* are symmetric keys for client and server encryption/decryption.
  - Each generated from *Na*, *Nb*, and *M*.
  - 4 other keys are also generated: 2 for MACS and 2 initialization vectors for stream cipher (for encrypting data, including MAC).
- Subsequent messages authenticated/encrypted using these key

Taking stock — an interpretation

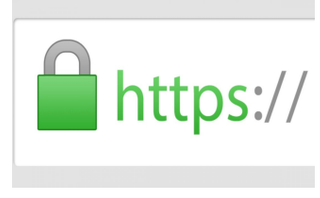
- Heart of protocol are exchanges
  1. server certificate $B \rightarrow A$: certificate(B, K_B)
  2. client key exchange $A \rightarrow B$: $\{PMS\}_{KB}$

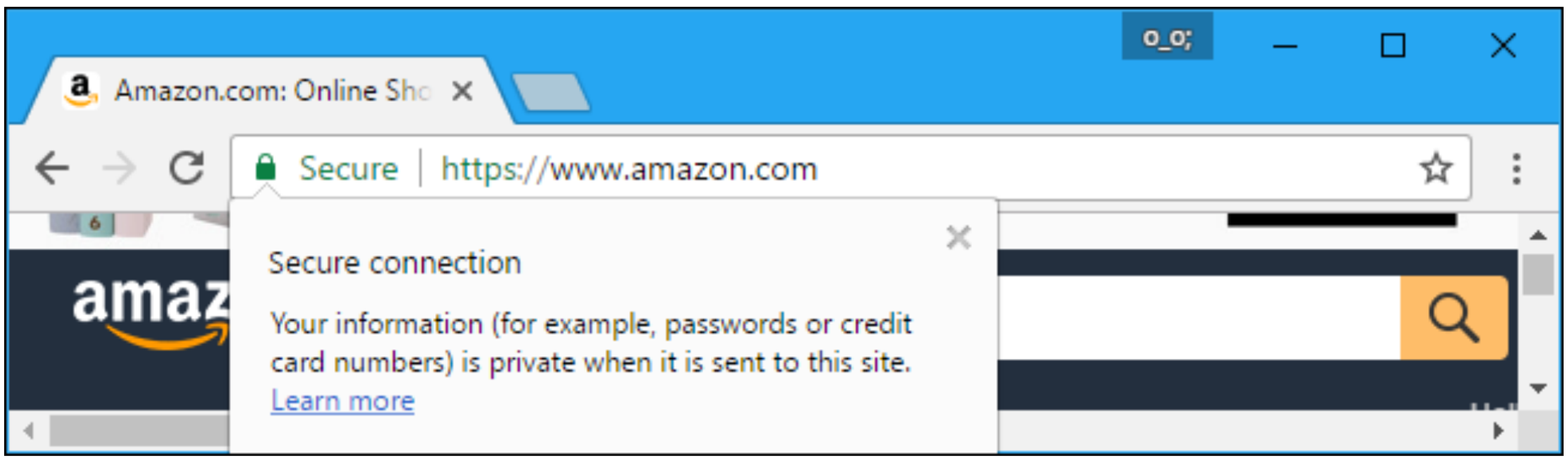
- If *A* shares one-sided key with a trusted *CA* and has a communication channel with *B* then (1) promotes this to an authentic channel.
- In (2), a non-authenticated agent *A* sends keying material to *B*. Effect of encryption/MAC-ing with this can be understood as follows:
  - When *A* subsequently sends: we have *sender invariant* channel.
  - When *B* subsequently responds: channel is *receiver invariant*.
  - Both yield essentially a *secure channel with a pseudonym*.
- Secure channel with a pseudonym cannot be promoted to a secure channel, where sender's identity is authenticated. Hence SSL often followed by additional client authentication.



SSL for web servers

- Recall: Netscape's motivation: Support secure communication of credit card numbers and customer data with web servers.
- Requires that both browser and server are SSL-enabled. Various libraries available: SSLRef, OpenSSL, ...
- Objective is user-friendly security for the masses.
  - Does a closed lock in browser actually provide this?
  - What are some of the vulnerabilities in practice?

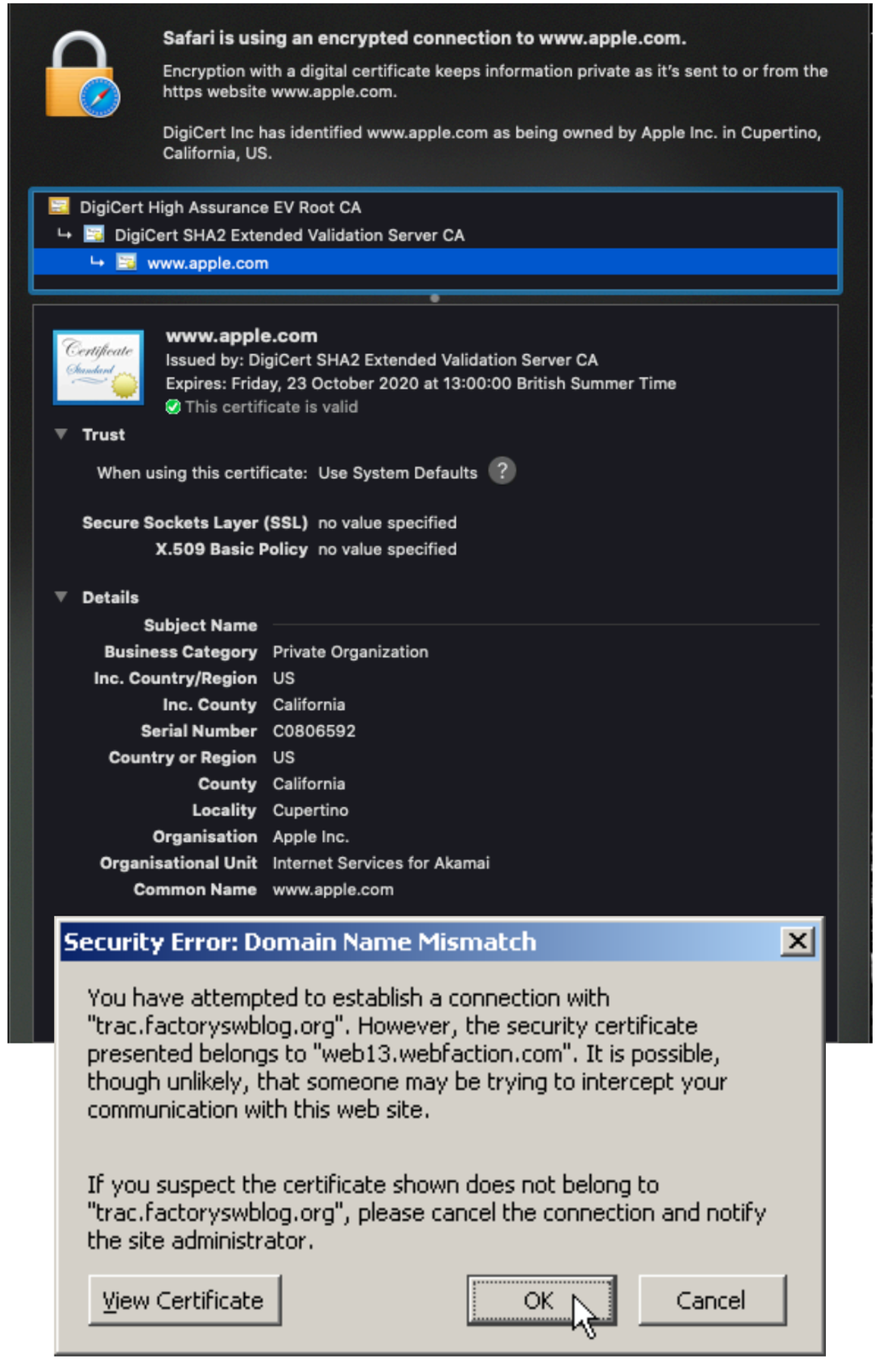




User interface vulnerabilities

- Users don’t understand lock=SSL.
- Users never click on the lock!
- Users don’t understand certificates.
- Confusion over warning messages.

- Is the organisation name right?
  - “UBS AG” versus “UBS”.
  - Microsoft versus Micr0soft.

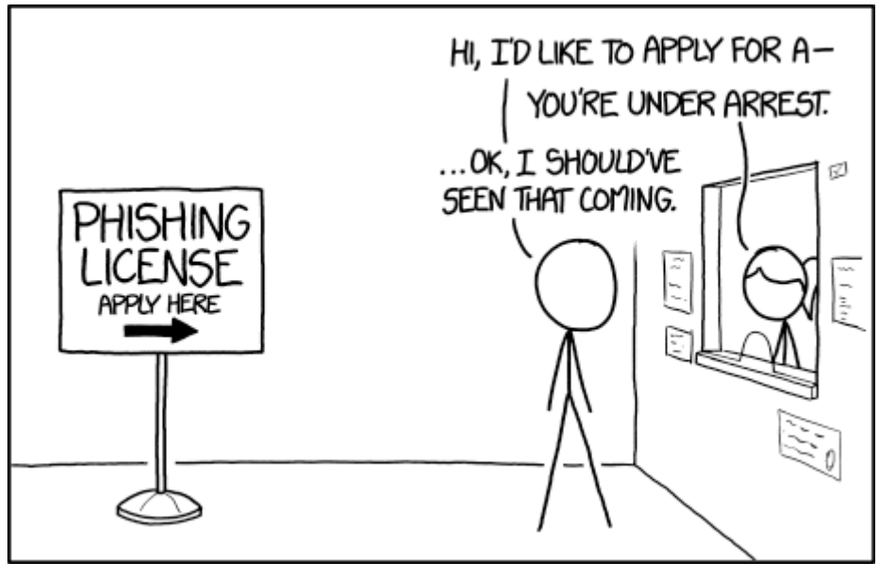


Example 1: Bad uses of SSL in Internet

PKIs are supposed to provide authentication, but they don't even do that. One company F-Secure sells software from its web site at www.datafellows.com. If you click to buy software, you are redirected to the Web site www.netsales.net, which makes an SSL connection with you. The SSL certificate was issued to "NetSales, Inc., Software Review LLC" in Kansas. F-Secure is headquartered in Helsinki and San Jose. By any PKI rules, no one should do business with this site. The certificate received is not from the same company that sells the software. This is exactly what a man-in-the-middle attack looks like, and exactly what PKI is supposed to prevent.

Bruce Schneier, Beyond Fear, 2003

Example 2: Phishing



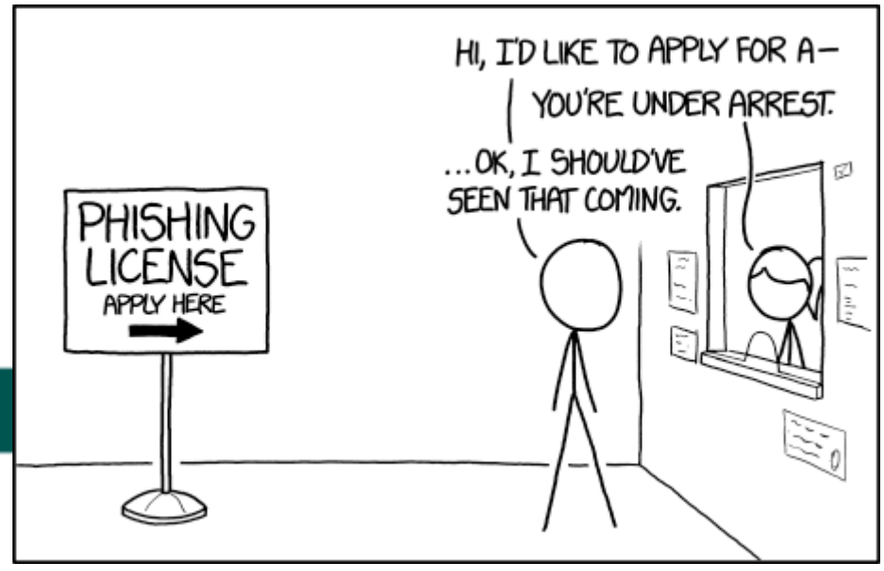
- Social engineering + technological trickery.

From account@LaSSale.com
Wed Jun 8 20:49:11 2005
We are glad to inform you, that our bank has a new security system. The new updated technology will ensure the security of your payments through our bank. As we are doing this for your own safety, we suggest you to test your account validation, this test will maintain the safety of your account. All you have to do is to complete our online secured form. Thank You.

- Users directed to spoofed sites (via email-SPAM, DNS hijacking, etc.) and tricked into revealing personal data.

Example: credit card number+CVV (Credit-card Verification Value, designed to ensure card possession).

Phishing — digital identity theft



Test Your Account Validation

Please enter the following information to begin the verification process.

User ID:

Password:

Your LaSalle ATM/Debit Card Number:
(Please enter card number without dashes.)

Your ATM/Debit Card Expiration Date: month year

Your ATM/Debit Card CVV2:

Your ATM/Debit Card PIN:

Your ATM or Debit Card Number, Cvv2 and PIN are being used for verification.

Clear Cancel Continue

Phishing — solutions?

- Problem is enormous, despite SSL!
- Industry trend is towards multifactor authentication of client.
  But phishing is a server-side authentication problem!
- A possible solution: multiple communication channels.
  - SSL combined with GSM (mobile telephones).
  - Server sends user a TAN by SMS, required for authentication.

  Succeeds as additional transaction key:
  "transfer $50 to UBS account 23171: confirm with JX3A17" Here attack requires man-in-the-middle on both channels.
  - Drawback: requires existing relationship with server.
    I.e., client's cell-phone number is registered with server.

Recent attacks on TLS: Lucky 13

- A cryptographic timing attack against implementations of TLS, first reported in February 2013 by its developers AlFardan and Paterson of the Information Security Group at Royal Holloway.

- Uses a timing side-channel attack against the message authentication code (MAC) check stage in the TLS algorithm to break the algorithm.

- OpenSSL and GnuTLS were both vulnerable.

- All tested Free Software implementations of TLS found to be potentially vulnerable.

Recent attacks on TLS: Heartbleed (CVE-2014-0160)

- A catastrophic vulnerability disclosed in the OpenSSL cryptography library (widely used TLS implementation) in April 2014.
  - May be exploited regardless of whether the principal using a vulnerable OpenSSL instance for TLS is a server or a client.
  - Results from improper input validation (due to a missing bounds check) in the implementation of the **TLS heartbeat extension** (RFC 6520), which provides a way to keep alive secure communication links without the need to renegotiate the connection each time.
    - Vulnerability classified as a **buffer over-read**, a situation where more data can be read than should be allowed.
- At time of disclosure, some 17% (around half a million) of the Internet's secure web servers certified by trusted authorities were believed to be vulnerable to the attack, allowing theft of the servers' private keys and users' session cookies and passwords.

Recent attacks on TLS: Heartbleed (CVE-2014-0160)

- A British Cabinet spokesman recommended that

  People should take advice on changing passwords from the websites they use... Most websites have corrected the bug and are best placed to advise what action, if any, people need to take.

- On the day of disclosure, the Tor Project advised anyone seeking "strong anonymity or privacy on the Internet" to "stay away from the Internet entirely for the next few days while things settle".
- 20 May 2014: 1.5% of the 800,000 most popular TLS-enabled websites were still vulnerable to Heartbleed.
- **Not a problem of the Internet protocol itself!**
  - Flaw existed in the OpenSSL's implementation of TLS.
  - TLS implementations other than OpenSSL (e.g., GnuTLS, Mozilla's Network Security Services, the TLS stack used by Microsoft products) were not affected.

Conclusions (I)

- "Real world" security protocols are complex.
  - Complexity is in part due to standardisation process.
  - The problems themselves are complex in their full generality.
  - But security's worst enemy is complexity!
- Example: IPsec.

> IPsec was a great disappointment to us. Given the quality of the people that worked on it and the time that was spent on it, we expected a much better result.
>
> — Ferguson & Schneier

- Analysis is nontrivial.
  - Minor flaws in IKE known.
  - No (known) full comprehensive analysis made, even assuming perfect cryptography.

Conclusions (I)

> Until a few years ago, you could connect to the Internet and be in contact with hundreds of millions of other nodes, without giving even a thought to security. The Internet in the ’90’s was like sex in the ’60’s. It was great while it lasted, but it was inherently unhealthy and was destined to end badly. I’m just really glad I didn’t miss out again this time.
>
> — Kaufman, Perlman, Speciner, “Network Security: Private Communication in a Public World”