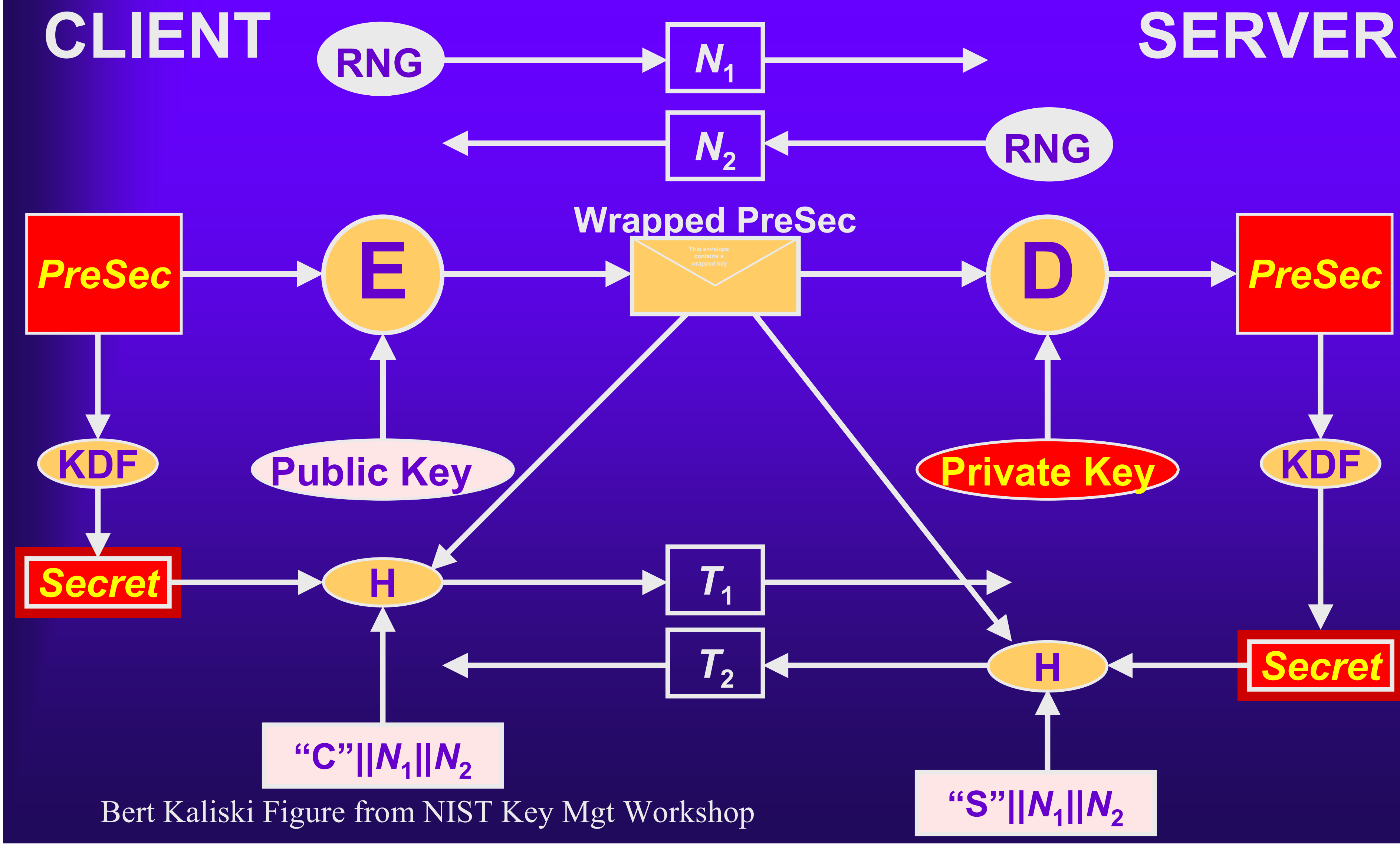
Transport Layer Security (TLS)

Bill Burr
william.burr@nist.gov
June 17, 2002

Transport Layer Security (TLS)

- Version 3.1 of Secure Socket Layer (SSL)
  - "standardized" by IETF RFC2246
  - Several earlier versions
    - V2 not very secure
- End-to-end between a client and server
  - Sits on top of TCP
  - Requires reliable connection
- Most important Internet crypto protocol?
  - Secure web pages
  - E-mail and LDAP access control

Generic Key Agreement Model



Bert Kaliski Figure from NIST Key Mgt Workshop

TLS Example: Client Authent.

Client

Server

Client Hello
supported ciphers
client_random
compression

Server Hello
TLS_RSA_WITH_3DES...
Server_random
compression
session_ID

Certificate
Server RSA Certificate

Certificate Request
certiricte_types
certificate_authorities

Hello Done

Optional messages for client authentication have gold background

TLS Example: Client Authent.

Client | **Server**

Client Certificate
client's PK certificate →

Client Key Exchange
E_{SK}(pre_master_secret) →

Certificate Verify
Signature (prev. messages) →

Change Cipher Spec →

Handshake Finished
verify_data →

← **Change Cipher Spec**

← **Handshake Finished**
verify_data

3DES Cipher Suites

- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_NULL_SHA

AES Cipher Suites

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_DH_DSS_WITH_AES_128_CBC_SHA
- TLS_DH_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DH_DSS_WITH_AES_256_CBC_SHA
- TLS_DH_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

Pseudorandom Function (PFR)

- Feeds a secret, a label, and a seed into an iterated HMAC to generate a pseudorandom stream
- Uses SHA1 & MD5
  - Intended to be secure if either is secure
    - The secret is split into halves, and one half is fed into the SHA1 HMAC and the other into the MD5 HMAC and the outputs are exclusive ORED. If one hash is bad enough, entropy would be lost.

Pseudorandom Function (PRF)

P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) || HMAC_hash(secret, A(2) || seed) || HMAC_hash(secret, A(3) || seed) || ...

Where || indicates concatenation.

A() is defined as:
A(0) = seed
A(i) = HMAC_hash(secret, A(i-1))

P_hash is iterated to produce the required quantity of data.

TLS's PRF is created by splitting the secret into two halves and using one half to generate data with P_MD5 and the other half to generate data with P_SHA-1, then exclusive-or'ing the outputs of these two expansion functions together.

PRF(secret, label, seed) = P_MD5(S1, label || seed) XOR P_SHA-1(S2, label || seed);

Pseudorandom Function (PRF)

- Used with pre_master_secret, client-random server_random & label “master_secret” to generate 48-byte master-secret
- Used with master_secret, server_random, client_random & label “key expansion” to generate key block
- Also used to generate the verify_data key confirmation parameter of the Handshake Finished message

Proposed Client Guidance

- Only do TLS (version 3.1)
  - But client is normally expected to be able to do highest version specified in Client Hello, plus every previous version!
- Server can choose any suite the client includes in Client Hello message, so
- Offer only 3DES or AES Cipher suites
  - Never include 40 or 56-bit suites
  - Encryption not required but anonymous cipher suites not allowed
- 1024-bit RSA/DSA client certificates are OK until 2015

Proposed Server Guidance

- Implement only TLS, not SSL
  - Clients that can't do TLS are out of luck
- Use only 3DES or AES Cipher Suites
- Server key management certificate subject key size needs to match confidentiality requirements
  - If data must be kept secret after 2015, then RSA/DH encryption keys larger than 1024 are needed.

For Maximum Security

- RSA or DSA authentication with ephemeral Diffie-Hellman key exchange
  - E.g., TLS_DHE_RSA_WITH_AES_128_CBC_SHA
  - Perfect forward secrecy
  - Potentially large DH keys for long term confidentiality, with whatever authentication key size is currently needed
- But,
  - How many products do ephemeral Diffie-Hellman?
  - Performance price?

Issues

- Do implementations check server identity?
- Use of same RSA server key for authentication and key management
- Is payload SHA-1 HMAC strong enough for 112, 128 & 256-bit encryption?
- Is 96-bit verify-data key confirmation strong enough for 112, 128 & 256-bit encryption?
- Is the PRF strong enough for 112, 128 & 256-bit encryption?
  - Does MD5 HMAC poison the well?
- “Non-standard” signature formats
- Diffie-Hellman doesn’t follow a NIST scheme

Issues

- Performance bottleneck is the server
  - Server gets to pick cipher suite
  - Weaker crypto (e.g. 512-bit RSA and RC4) is faster than 1024-bit RSA & 3DES or AES
    - do servers normally select weaker alternatives
    - do some servers not do 3DES at all?
  - Some old clients still do only 40-bit RC4
    - do servers select 40-bit when they could use 128?
- How do you configure clients and servers to select the stronger alternatives?

Questions and Discussion