Secure Communication

Secure Communication

Lecture 14

Wrap-Up

We saw...

- Symmetric-Key Components
  - SKE, MAC
- Public-Key Components
  - PKE, Digital Signatures
- Building blocks: Block-ciphers (AES), Hash-functions (SHA-3), Trapdoor PRG/OWP for PKE (e.g., DDH, RSA) and Random Oracle heuristics (in RSA-OAEP, RSA-PSS)
- Symmetric-Key primitives much faster than Public-Key ones
  - Hybrid Encryption gets best of both worlds

Secure Communication in Practice

Secure Communication in Practice

- Can do at application-level

Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server

Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server
- Or lower-level infrastructure to allow use by more applications

Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server
- Or lower-level infrastructure to allow use by more applications
  - e.g. between OS kernels, or between network gateways

Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server
- Or lower-level infrastructure to allow use by more applications
  - e.g. between OS kernels, or between network gateways
- Standards in either case

Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server
- Or lower-level infrastructure to allow use by more applications
  - e.g. between OS kernels, or between network gateways
- Standards in either case
  - To be interoperable

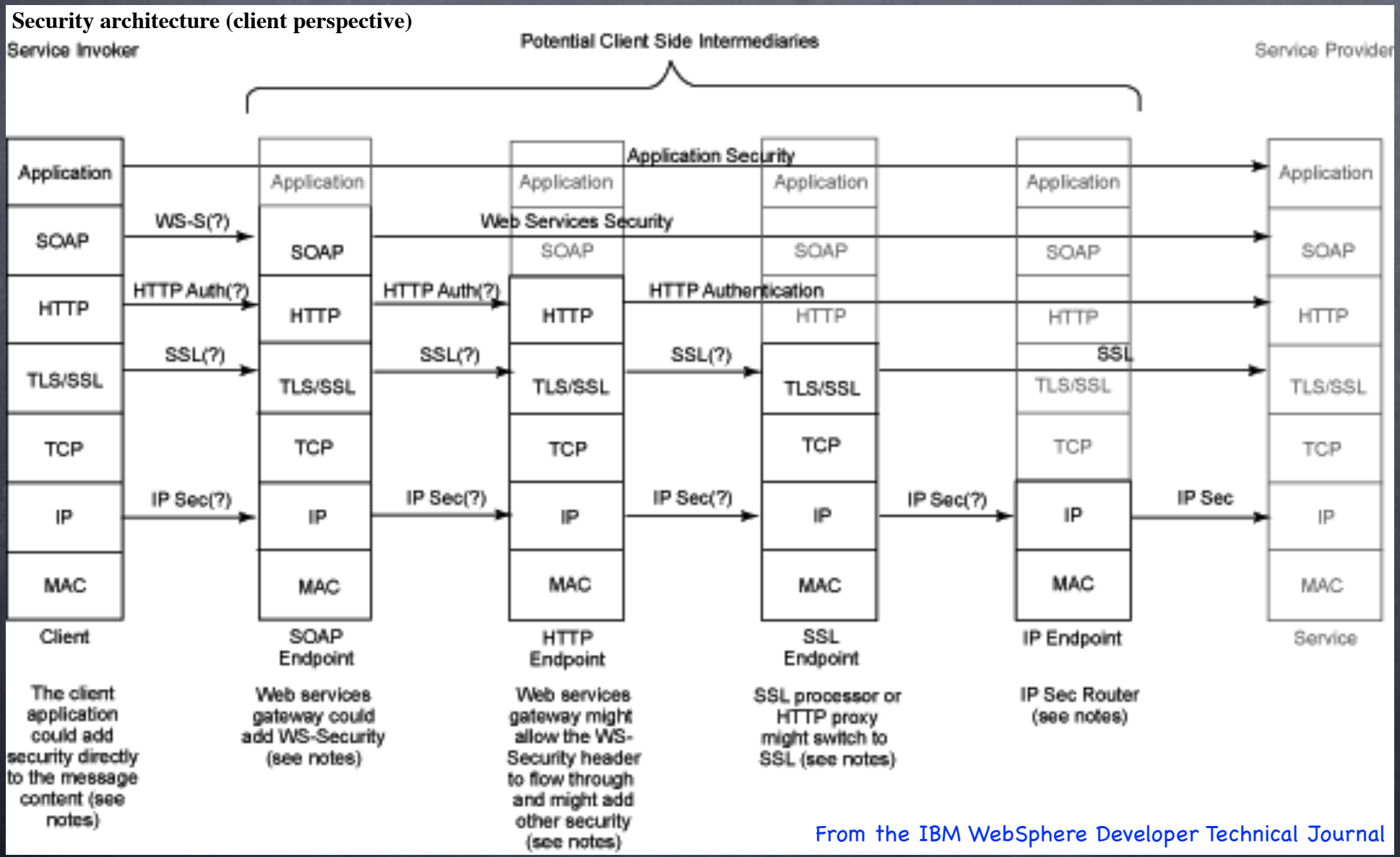
Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server
- Or lower-level infrastructure to allow use by more applications
  - e.g. between OS kernels, or between network gateways
- Standards in either case
  - To be interoperable
  - To not insert bugs by doing crypto engineering oneself

Secure Communication in Practice

- Can do at application-level
  - e.g. between web-browser and web-server
- Or lower-level infrastructure to allow use by more applications
  - e.g. between OS kernels, or between network gateways
- Standards in either case
  - To be interoperable
  - To not insert bugs by doing crypto engineering oneself
  - e.g.: SSL/TLS (used in https), IPSec (in the "network layer")

Security Architectures
(An example)



From the IBM WebSphere Developer Technical Journal

Secure Communication Infrastructure

Secure Communication Infrastructure

- Goal: a way for Alice and Bob to get a private and authenticated communication channel (can give a detailed SIM-definition)

Secure Communication Infrastructure

- Goal: a way for Alice and Bob to get a private and authenticated communication channel (can give a detailed SIM-definition)
- Simplest idea: Use a (SIM-CCA secure) public-key encryption (possibly a hybrid encryption) to send signed (using an existentially unforgeable signature scheme) messages (with sequence numbers and channel id)

Secure Communication Infrastructure

- Goal: a way for Alice and Bob to get a private and authenticated communication channel (can give a detailed SIM-definition)
- Simplest idea: Use a (SIM-CCA secure) public-key encryption (possibly a hybrid encryption) to send signed (using an existentially unforgeable signature scheme) messages (with sequence numbers and channel id)
  - Limitation: Alice, Bob need to know each other's public-keys

Secure Communication Infrastructure

- Goal: a way for Alice and Bob to get a private and authenticated communication channel (can give a detailed SIM-definition)
- Simplest idea: Use a (SIM-CCA secure) public-key encryption (possibly a hybrid encryption) to send signed (using an existentially unforgeable signature scheme) messages (with sequence numbers and channel id)
  - Limitation: Alice, Bob need to know each other's public-keys
- But typically Alice and Bob engage in "transactions," exchanging multiple messages, maintaining state throughout the transaction

Secure Communication Infrastructure

- Goal: a way for Alice and Bob to get a private and authenticated communication channel (can give a detailed SIM-definition)
- Simplest idea: Use a (SIM-CCA secure) public-key encryption (possibly a hybrid encryption) to send signed (using an existentially unforgeable signature scheme) messages (with sequence numbers and channel id)
  - Limitation: Alice, Bob need to know each other's public-keys
- But typically Alice and Bob engage in "transactions," exchanging multiple messages, maintaining state throughout the transaction
  - Makes several efficiency improvements possible

Secure Communication Infrastructure

Secure Communication Infrastructure

- Secure Communication Sessions

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys

(Authenticated) Key-Exchange

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys
  - Communication phase: use efficient shared-key schemes

(Authenticated) Key-Exchange

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys
  - Communication phase: use efficient shared-key schemes
- Server-to-server communication: Both parties have (certified) public-keys

(Authenticated) Key-Exchange

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys
  - Communication phase: use efficient shared-key schemes
- Server-to-server communication: Both parties have (certified) public-keys
- Client-server communication: server has (certified) public-keys

(Authenticated) Key-Exchange

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys
  - Communication phase: use efficient shared-key schemes
- Server-to-server communication: Both parties have (certified) public-keys
- Client-server communication: server has (certified) public-keys
  - Client "knows" server. Server willing to talk to all clients

(Authenticated) Key-Exchange

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys
  - Communication phase: use efficient shared-key schemes
- Server-to-server communication: Both parties have (certified) public-keys
- Client-server communication: server has (certified) public-keys
  - Client "knows" server. Server willing to talk to all clients

(Authenticated) Key-Exchange

Server may "know" (some) clients too, using passwords, pre-shared keys, or if they have (certified) public-keys. Often implemented in application-layer

Secure Communication Infrastructure

- Secure Communication Sessions
  - Handshake phase: establish private shared keys
  - Communication phase: use efficient shared-key schemes
- Server-to-server communication: Both parties have (certified) public-keys
- Client-server communication: server has (certified) public-keys
  - Client "knows" server. Server willing to talk to all clients
- Client-Client communication (e.g., email) Clients share public-keys in ad hoc ways

(Authenticated) Key-Exchange

Server may "know" (some) clients too, using passwords, pre-shared keys, or if they have (certified) public-keys. Often implemented in application-layer

Certificate Authorities

- How does a client know a server's public-key?
  - Based on what is received during a first session? (e.g., first ssh connection to a server)
- Better idea: Chain of trust
  - Client knows a certifying authority's public key (for signature)
    - Bundled with the software/hardware
  - Certifying Authority signs the signature PK of the server
    - CA is assumed to have verified that the PK was generated by the "correct" server before signing
    - Validation standards: Domain/Extended validation

Forward Secrecy

Forward Secrecy

- Servers have long term public keys that are certified

Forward Secrecy

- Servers have long term public keys that are certified
  - Would be enough to have long term signature keys, but in practice long term encryption keys too

Forward Secrecy

- Servers have long term public keys that are certified
  - Would be enough to have long term signature keys, but in practice long term encryption keys too
  - Problem: if the long term key is leaked, old communications are also revealed

Forward Secrecy

- Servers have long term public keys that are certified
  - Would be enough to have long term signature keys, but in practice long term encryption keys too
  - Problem: if the long term key is leaked, old communications are also revealed
    - Adversary may have already stored, or even actively participated in old sessions

Forward Secrecy

- Servers have long term public keys that are certified
  - Would be enough to have long term signature keys, but in practice long term encryption keys too
  - Problem: if the long term key is leaked, old communications are also revealed
    - Adversary may have already stored, or even actively participated in old sessions
  - Solution: Use fresh public-keys/do a fresh key-exchange for each session (authenticated using signatures)

A Simple Secure Communication Scheme

A Simple Secure Communication Scheme

- Handshake

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

Recall "inefficient" domain-extension of MAC: Add a session-specific nonce and a sequence number to each message before MAC'ing

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

Recall "inefficient" domain-extension of MAC: Add a session-specific nonce and a sequence number to each message before MAC'ing

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

Recall "inefficient" domain-extension of MAC: Add a session-specific nonce and a sequence number to each message before MAC'ing

A Simple Secure Communication Scheme

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

Server's PK either trusted (from a previous session for e.g) or certified by a trusted CA, using a Digital Signature scheme

Need to avoid replay attacks (infeasible for server to explicitly check for replayed ciphertexts)

Recall "inefficient" domain-extension of MAC: Add a session-specific nonce and a sequence number to each message before MAC'ing

Authentication for free: MAC serves dual purposes!

TLS (SSL)

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

TLS (SSL)

Negotiations on protocol version etc. and "cipher suites" (i.e., which PKE/ key-exchange, SKE, MAC (and CRHF)).

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

TLS (SSL)

Negotiations on protocol version etc. and "cipher suites" (i.e., which PKE/ key-exchange, SKE, MAC (and CRHF)).

e.g. cipher-suite: RSA-OAEP for key-exchange, AES for SKE, HMAC-SHA256 for MAC

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

TLS (SSL)

Negotiations on protocol version etc. and "cipher suites" (i.e., which PKE/ key-exchange, SKE, MAC (and CRHF)).

e.g. cipher-suite: RSA-OAEP for key-exchange, AES for SKE, HMAC-SHA256 for MAC

Server sends a certificate of its PKE public-key, which the client verifies

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

TLS (SSL)

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

Negotiations on protocol version etc. and "cipher suites" (i.e., which PKE/ key-exchange, SKE, MAC (and CRHF)).

e.g. cipher-suite: RSA-OAEP for key-exchange, AES for SKE, HMAC-SHA256 for MAC

Server sends a certificate of its PKE public-key, which the client verifies

Server also "contributes" to key-generation (to avoid replay attack issues): Roughly, client sends a key K for a PRF; a master key generated as $PRF_K(x,y)$ where x from client and y from server. SKE and MAC keys derived from master key

TLS (SSL)

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

Negotiations on protocol version etc. and "cipher suites" (i.e., which PKE/key-exchange, SKE, MAC (and CRHF)).

e.g. cipher-suite: RSA-OAEP for key-exchange, AES for SKE, HMAC-SHA256 for MAC

Server sends a certificate of its PKE public-key, which the client verifies

Server also "contributes" to key-generation (to avoid replay attack issues): Roughly, client sends a key K for a PRF; a master key generated as $PRF_K(x,y)$ where x from client and y from server. SKE and MAC keys derived from master key

Uses MAC-then-encrypt! Not CCA secure in general, but secure with stream-cipher (and with some other modes of block-ciphers, like CBC)

TLS (SSL)

- Handshake
  - Client sends session keys for MAC and SKE to the server using SIM-CCA secure PKE, with server's PK (i.e. over an unauthenticated, but private channel)
- For authentication only: use MAC
  - In fact, a "stream-MAC": To send more than one message, but without allowing reordering
- For authentication + (CCA secure) encryption: encrypt-then-MAC
  - stream-cipher, and "stream-MAC"

Negotiations on protocol version etc. and "cipher suites" (i.e., which PKE/key-exchange, SKE, MAC (and CRHF)).

e.g. cipher-suite: RSA-OAEP for key-exchange, AES for SKE, HMAC-SHA256 for MAC

Server sends a certificate of its PKE public-key, which the client verifies

Server also "contributes" to key-generation (to avoid replay attack issues): Roughly, client sends a key K for a PRF; a master key generated as $PRF_K(x,y)$ where x from client and y from server. SKE and MAC keys derived from master key

Uses MAC-then-encrypt! Not CCA secure in general, but secure with stream-cipher (and with some other modes of block-ciphers, like CBC)

Several details on closing sessions, session caching, resuming sessions ...

Vulnerabilities

Vulnerabilities

- Numerous vulnerabilities keep surfacing

  FREAK, DROWN, POODLE, Hearbleed, Logjam, ...

  And numerous unnamed ones: www.openssl.org/news/vulnerabilities.html

  Listed as part of Common Vulnerabilities and Exposures (CVE) list: cve.mitre.org/

Vulnerabilities

- Numerous vulnerabilities keep surfacing

  FREAK, DROWN, POODLE, Hearbleed, Logjam, ...

  And numerous unnamed ones: www.openssl.org/news/vulnerabilities.html

  Listed as part of Common Vulnerabilities and Exposures (CVE) list: cve.mitre.org/

- Bugs in protocols
  - Often in complex mechanisms created for efficiency
  - Often facilitated by the existence of weakened "export grade" encryption and improved computational resources
  - Also because of weaker legacy encryption schemes (e.g. Encryption from RSA PKCS#1 v1.5 — known to be not CCA secure and replaced in 1998 — is still used in TLS)

Vulnerabilities

- Numerous vulnerabilities keep surfacing

  FREAK, DROWN, POODLE, Hearbleed, Logjam, ...

  And numerous unnamed ones: www.openssl.org/news/vulnerabilities.html

  Listed as part of Common Vulnerabilities and Exposures (CVE) list: cve.mitre.org/

- Bugs in protocols
  - Often in complex mechanisms created for efficiency
  - Often facilitated by the existence of weakened "export grade" encryption and improved computational resources
  - Also because of weaker legacy encryption schemes (e.g. Encryption from RSA PKCS#1 v1.5 — known to be not CCA secure and replaced in 1998 — is still used in TLS)
- Bugs in implementations

Vulnerabilities

- Numerous vulnerabilities keep surfacing

  FREAK, DROWN, POODLE, Hearbleed, Logjam, ...

  And numerous unnamed ones: www.openssl.org/news/vulnerabilities.html

  Listed as part of Common Vulnerabilities and Exposures (CVE) list: cve.mitre.org/

- Bugs in protocols
  - Often in complex mechanisms created for efficiency
  - Often facilitated by the existence of weakened "export grade" encryption and improved computational resources
  - Also because of weaker legacy encryption schemes (e.g. Encryption from RSA PKCS#1 v1.5 — known to be not CCA secure and replaced in 1998 — is still used in TLS)
- Bugs in implementations
- Side-channels originally not considered

Vulnerabilities

- Numerous vulnerabilities keep surfacing
  FREAK, DROWN, POODLE, Hearbleed, Logjam, ...
  And numerous unnamed ones: www.openssl.org/news/vulnerabilities.html
  Listed as part of Common Vulnerabilities and Exposures (CVE) list: cve.mitre.org/
- Bugs in protocols
  - Often in complex mechanisms created for efficiency
  - Often facilitated by the existence of weakened "export grade" encryption and improved computational resources
  - Also because of weaker legacy encryption schemes (e.g. Encryption from RSA PKCS#1 v1.5 — known to be not CCA secure and replaced in 1998 — is still used in TLS)
- Bugs in implementations
- Side-channels originally not considered
- Back-Doors (?) in the primitives used in the standards

Beyond Communication

- Encryption/Authentication used for data at rest
  - e.g., disk encryption, storing encrypted data on a cloud server, ...
- Security definitions like SIM-CCA do not directly extend to all these settings
  - New concerns that do not arise in setting up communication channels
  - e.g., circular (in)security: encrypting the SK using its own PK

Coming Up

Coming Up

- Beyond communication

Coming Up

- Beyond communication
  - Secure Multi-party computation

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval
  - Searchable Encryption

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval
  - Searchable Encryption
  - Attribute-Based cryptography

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval
  - Searchable Encryption
  - Attribute-Based cryptography
  - Anonymous Credentials & Digital Cash

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval
  - Searchable Encryption
  - Attribute-Based cryptography
  - Anonymous Credentials & Digital Cash
  - Oblivious RAM, ...

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval
  - Searchable Encryption
  - Attribute-Based cryptography
  - Anonymous Credentials & Digital Cash
  - Oblivious RAM, ...
- Tools: Secret sharing, homomorphic encryption, bilinear-pairings, lattices...

Coming Up

- Beyond communication
  - Secure Multi-party computation
  - Electronic Voting
  - Private Information Retrieval
  - Searchable Encryption
  - Attribute-Based cryptography
  - Anonymous Credentials & Digital Cash
  - Oblivious RAM, ...
- Tools: Secret sharing, homomorphic encryption, bilinear-pairings, lattices...
- Quantum cryptography (secure communication)