University of Houston

Foundations of Security

COSC 6347

Final Exam Review

Author
K.M. Hourani

Based on Notes By
Dr. Aron Laszka

December 6, 2020

1 Advanced crypto

1.1 Advanced Cryptographic Primitives

- secure multiparty computation
- homomorphic encryption

1.2 Commitment Schemes

Commitment Problem

- Bob "calls" the coin flip (i.e., heads or tails)
- Alice flips the coin
- Bob wins if her call is correct, Alice wins otherwise

Can we prevent Alice from cheating even if the players are not in the same physical location?

Commitment Scheme

- Two phases
    1. commit: A chooses a value V, A sends a **commitment** of V to B
    2. reveal: A reveals the value of V
- Example: coin flipping
    1. commit: A flips a coin, A sends a commitment (i.e., coin is heads or tails) to B
    - B calls the coin flip (i.e., heads or tails)
    2. reveal: A reveals the value of the coin flip
- Requirements for commitment scheme
    - B cannot learn the value of V from the commitment
    - A can reveal only the originally chosen value for a commitment

Naive Attempt Using Hash Function

- H: cryptographic hash function
- If the set of possible values of V are small (e.g., "heads" or "tails"), B can learn V by simply trying all possible values

Secure Commitment Using Hash Function

- Collision-free hash function $\rightarrow$ A cannot cheat by finding V_1 and V_2 such that $H(r_1 \mid r_2 \mid V_1) = H(r_1 \mid r_2 \mid V_2)$
    - r_1 prevents pre-computation of colliding V_1 and V_2

1.3 Secret Sharing

- Problem: distribute a secret among N participants such that
    - any group of at least T participants can reconstruct the secret
    - no group of fewer than T participants can reconstruct any part of it
- Types
    - **unconditionally secure**: information-theoretically secure (unbounded attacker)
    - **conditionally secure**: typically more efficient

Special Case: $T = N$

- Unconditionally secure scheme:
    1. let the secret be a binary number S
    2. pick $N - 1$ random numbers R_1, R_2, $\ldots$, R_{N-1} of the same length
    3. give each participant i, $i < N$, the number R_i

4. give the last participant the result of $S \oplus R_1 \oplus R_2 \oplus \ldots \oplus R_{N-1}$

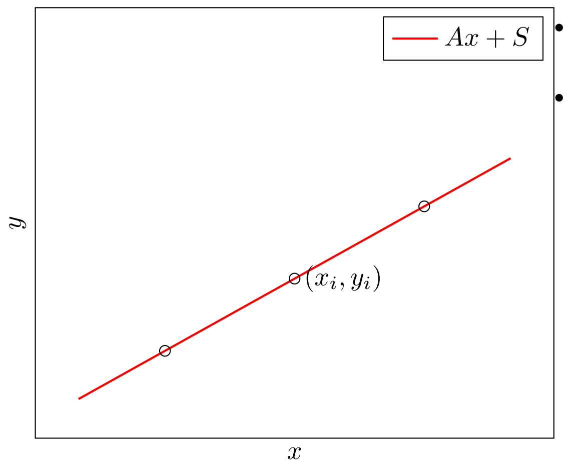
- N participants can reconstruct the secret by XORing their numbers:
  –

$$\begin{aligned} &R_1 \oplus R_2 \oplus \ldots \oplus R_{N-1} \oplus (S \oplus R_1 \oplus R_2 \oplus \ldots \oplus R_{N-1}) \\ &= (R_1 \oplus R_1) \oplus (R_2 \oplus R_2) \oplus \ldots \oplus (R_{N-1} \oplus R_{N-1}) \oplus S \\ &= S \end{aligned}$$

- $N-1$ participants can compute only $S \oplus R_i$, where i is the missing participant (or $R_1 \oplus R_2 \oplus \ldots \oplus R_{N-1}$ if the last participant is missing)

1.4 Shamir's Secret Sharing

- Proposed by Shamir in 1979
- Unconditionally secure
- Special case $T = 2$
  1. let the secret be a number S
  2. pick random number A
  3. let each participant's share be a random point on the line $Ax + S$



- $T = 2$ participants can reconstruct the secret since any two points define a line
- Single participant cannot learn the slope

General Case

- Arbitrary T:
  1. let the secret be a number S
  2. pick random numbers A_1, A_2, ..., A_{T-1}
  3. let each participant's share be a random point from the curve

$$y = S + A_1 x + A_2 x^2 + \cdots + A_{T-1} x^{T-1}$$

- At least T points are necessary to define a polynomial of degree $T - 1$
- Example $T = 3$
  – secret is a parabola (i.e., $A_2 x^2 + A1x + S$)
  – there an infinite number of parabolas fitting two points
  – but three point define one uniquely

1.5 Secure Multiparty Computation

- Problem: N participants with private data d_1, d_2, ..., d_N
  – participants would like to compute the value $F(d_1, d_2, \ldots, d_N)$ of a public function F over their private data

  - no participant i would like to reveal any information about its data d_i
- Requirements
  - **privacy**: no information is revealed about any private data (other than what is revealed by the public output)
  - **correctness**: public function is correctly computed
- Adversaries may be semi-honest (passive) or malicious (active)

2 WiFi security

2.1 Security Challenge

- Problem: no inherent physical protection
- **joining** a network does not require physical access
- radio transmissions are broadcast → anyone in range can **eavesdrop**
- **injecting** new messages or **replaying** old messages is possible
- **jamming** attacks against availability
- jamming and injecting messages can be combined into **tampering** attacks

2.2 Simple "Solutions" for Access Control

Hidden SSID

- Association request must contain the SSID of the network
  - by default, the AP broadcasts it periodically in the beacon
- AP may be configured to **stop announcing the SSID** → SSID may be be used as a "password"
- However,
  - SSID must be hard to guess
  - every authorized user must know the SSID
  - **SSID can be easily eavesdropped** whenever an authorized station connects to the network → does not provide any security
- Tools are available for eavesdropping (e.g. Aircrack-ng)

MAC Address Based Filtering

- AP may be configured to **allow only devices with certain MAC addresses** to connect
  - MAC addresses of all authorized devices must be registered in advance
- However,
  - **MAC address is sent in plaintext** in every packet
  - many WLAN devices allow their MAC addresses to be changed → attacker can easily impersonate an authorized user

2.3 802.11 Security Standards

WEP

- security is based on a 40 or 104-bit secret key
  - WiFi "password" shared by all users
- confidentiality: RC4 stream cipher
  - key is extended by a 24-bit IV, which is changed for each message → used as nonce to prevent key reuse problems
- integrity: encrypted CRC32 (Cyclic Redundancy Check) checksum
- access control: challenge-response between AP and station

WEP Design Flaws

- Authentication
  - **one-way authentication** (only for station) → AP can be impersonated
- Integrity protection
  - based on **error-detection code** (CRC32) instead of cryptographic hash → forging authentication tags is trivial
  - **no message replay protection**
- Key usage
  - **no session key**: long-term key used for all purposes (authentication, encryption, integrity protection)
  - **short nonce** (i.e., 24-bit IV) → danger of key reuse for stream cipher
    * busy network with 1000 packets per second reuses in less than 5 hours
  - vulnerable to Fluhrer-Mantin-Shamir Attack
    * In practice, WEP keys can be broken in a matter of minutes (or less) → WEP is **not secure**

2.4 WiFi Protected Access (WPA)

WPA

- Standard: 802.11i TKIP (Temporal Key Integrity Protocol
- Design goals: **fix the flaws of WEP** and be **compatible with legacy hardware**
- Overview
  - key usage: session key is established during a secure two-way authentication
  - confidentiality: RC4 encryption, but with **48-bit IV**, which is **mixed thoroughly** with the session key and source MAC address
    * prevents key reuse and the Fluhrer-Mantin-Shamir attack
  - integrity: 64-bit message integrity codes computed using Michael, which is **computationally very efficient** but provides only 20 bits of effective security
    * after wrong code, station is banned for a minute and needs to re-authenticate
  - Deprecated in later revisions of the standard
-

WPA-2

- Standard: **IEEE 802.11i**
- WPA 2 Devices can be certified by the Wi-Fi Alliance

Phases

1. Discovery
   - agree on what authentication method and ciphers to use
2. Authentication
   - may use an authentication server
   - create a master session key
3. Key management
   - derive keys for various purposes
4. Protected data transfer
5. Connection termination

Discovery Phase

- Goal: station and AP may support different sets of authentication methods and ciphers → they need to agree on which ones they will use
- **Authentication and key-management suite**: how to perform mutual authentication and derive fresh keys
  - IEEE 802.1X, pre-shared key (PSK), or vendor-specific
- **Cipher suite**: what ciphers to use for confidentiality and integrity
  - WEP, TKIP, CCMP, or vendor-specific
- Protocol
  1. AP can periodically **broadcast** its security capabilities using a **Beacon** (or station can ask for it using a Probe Request message)
  2. Station **specifies** an authentication and cipher suite in an **Association Request**
  3. if the AP **accepts** the specified suites, it sends an **Association Response**

Authentication Phase

- Goals:
  - mutual authentication:
    1) only authorized stations can use the network
    2) station is assured that it communicates with a legitimate network
  - generate **pairwise master key** (PMK)
- Approaches
  - Pre-shared key (PSK)
    * password is deployed on each station and the AP manually
    * PMK = PSK = generated from the password using a hash function
    * ideal for home and small office networks
  - IEEE 802.1X

Key-Management Phase

- Goals:
  - derive **pairwise transient keys** from the PMK
  - distribute **group keys**
- Pairwise transient key (PTK)
  - protecting data between station and AP
  - generated from PMK and the AP's and station's MAC addresses and nonces
- Group temporal key (GTK)
  - protecting multicast communication
  - group master key (GMK): generated randomly by the AP
  - distributed using the PTK

Protected Data Transfer Phase

- Standard defines two schemes: TKIP and CCMP
- TKIP: see WPA
- CCMP (Counter mode CBC-MAC Protocol)
  - based on the CCM (Counter with CBC-MAC) authenticated encryption mode
  - integrity: CBC-MAC based on AES encryption
  - confidentiality: AES encryption in counter (CTR) mode
  - same 128-bit key for integrity and confidentiality (from PTK)
  - 48-bit packet number to prevent replay attacks

IEEE 802.1X

- Standard for port-based network access control
- Entities
  - supplicant = station
  - authenticator = access point
  - authentication server
- Port-based: supplicant can access only the authentication server until the authentication succeeds
- Authentication server does not have to be implemented on the access point → little overhead for the access point

EAP Authentication Methods

- Extensible framework, not a specific authentication mechanism
- Example methods
  - EAP-TLS: based on public-key certificates
  - EAP-GPSK (Generalized Pre-Shared Key): based on secret keys shared by the client and the server, uses symmetric-key cryptography

3 IPSec

- Collection of protocols and mechanisms, standardized by the Internet Engineering Task Force (IETF) in a series of publications
- Provides
  - data confidentiality and integrity
  - source authentication (prevent address spoofing, i.e., sending from fake address)
  - protection against packet replay
- Below the transport layer (TCP or UDP) → transparent to applications
- End-to-end security between two hosts, a host and a network, or between two networks
- Example Applications of IPSec
  - Secure remote access over the Internet
  - Secure virtual private network

3.1 Transport Mode and Tunnel Mode

- Transport mode
  - protects the payload of the IP packet
  - typically host-to-host communication
- Tunnel mode
  - protects the entire IP packet by encapsulating it in the payload of a new IP packet
  - typically host-to-network or network-to-network communication

	Protocol	
	Authentication Headher (AH)	Encapsulating Security Payloads (ESP)
Modes	both transport and tunnel	
Provides	integrity, replay protection	integrity, confidentiality, replay protection
Protects	payload and IP header	payload

Authentication Header

- Services
  - data and origin integrity
  - replay-prevention
- Message authentication

  - computed from immutable fields of the IP header, AH header (except ICV), and original payload
  - algorithms: HMAC-MD5, HMAC-SHA-1, HMAC-SHA-2, ...

Encapsulating Security Payload

- Services: confidentiality, integrity (optional), replay prevention
- Encryption: AES-CBC, 3DES-CBC, ...
- Message authentication: HMAC-SHA-1, AES-GMAC, ...
- Authenticated encryption: AES-GCM

Combining Modes and Protocols

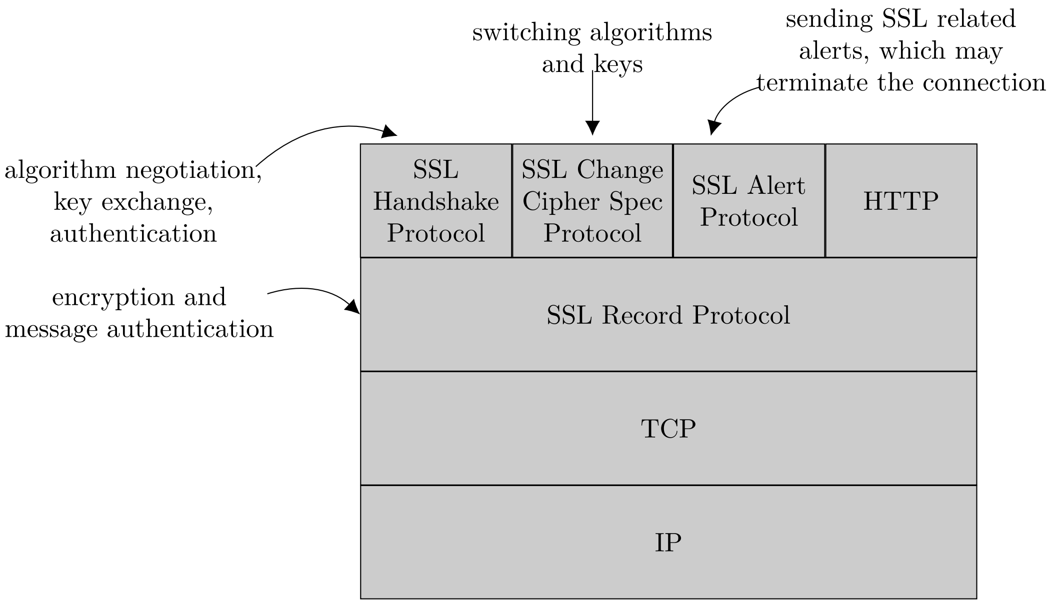
- Tunnel mode advantage: requires support only at the gateways
- Transport mode advantage: requires support only at the hosts
- AH advantage: authenticates some elements of the original header
- ESP advantage: protects both integrity and confidentiality
- Combining modes
  - IPSec tunnel can carry any IP packet → IPSec transport or tunnel packets can be sent through an IPSec tunnel
  - IPSec transport can protect any IP packet → IPSec transport or tunnel packets can be protected by outer IPSec transport
  - ...
  - can be nested to any depth
- Combination Examples
  1. AH in transport (for integrity) + ESP in transport (for confidentiality)
  2. IPSec packets over tunnel

4 SSL / TLS

Secure Socket Layer

- End-to-end security between two applications
- Endpoint applications can implement it without the help of the operating systems or any intermediate devices
- Developed by Netscape for securing HTTP → HTTPS = HTTP over SSL
- very widely used, not just for HTTP (e.g., FTP, POP3, IMAP)

Overview



SSL Record Protocol

- Security 528
  - confidentiality: symmetric-key encryption (AES GCM, Salsa20, . . .)
  - integrity: message authentication codes based on symmetric-key cryptography (HMAC-SHA256, . . .)
- Additional Services
  - fragmentation: fragment application data into records of at most 16 KiB
  - lossless compression: optional (default is no compression)

SSL Handshake Protocol

- Phase 1: establish security capabilities
  - client_hello:
    * highest SSL version supported by the client
    * nonce (timestamp + random value)
    * cipher suite: list of key-exchange methods, as well as encryption and MAC algorithms
    * compression method: list of supported compression algorithms
  - server_hello:
    * highest SSL version supported by both the client and the server
    * nonce
    * chosen cipher suite and compression method
- Phase 2: server authentication and key exchange
  - certificate (optional): X.509 certificate (may be a chain)
  - server_key_exchange (optional):
    * parameters for Anonymous or Ephemeral Diffie-Hellman exchange
    * public-key for RSA exchange if the certificate contains only a signing key
    * signed by the server (together with the nonces)
  - certificate_request (optional): ask client for an X.509 certificate
  - server_hello_done: server is finished
- Phase 3: client authentication and key exchange
  - certificate (optional): X.509 certificate if the server asked for a client certificate
  - client_key_exchange:
    * pre-master secret encrypted using public RSA key of the server

        * parameters for D-H exchange
    - certificate_verify (optional): digital signature of all previous handshake messages

Key Exchange Methods

- Goal: exchange or agree on a pre-master secret (PMS)
- RSA
    - client generates pre-master secret (PMS)
    - sends PMS to the server encrypted using RSA public-key encryption
- Diffie-Hellman protocol
    - anonymous D-H: basic D-H with no authentication
    - fixed D-H: D-H parameters of the server (X_A and Y_A) are fixed and Y_A is contained in a digital certificate
    - ephemeral D-H: D-H with authentication

SSL Change Cipher Spec Protocol

- Same for client and server
    - change_cipher_spec: signals that the communication party is switching to the negotiated cryptographic algorithms and keys
    - finished: hash value computed from the master secret and all handshake messages using HMAC with SHA hash function

Session Resume

- Authentication and key exchange are complex → result needs to be reusable
- Session
    - association between a client and a server
    - cipher suite, compression method, and master secret
- Connection
    - within a session
    - keys and IVs for encryption and message authentication
- Session ID: identifies a session
    - sent in ClientHello → may specify an existing session to be resumed
    - sent in ServerHello → server can accept resume by sending the same ID
- Session resume skips all messages in the Handshake after the ClientHello and ServerHello messages
    - new keys and IVs are generated from the nonces in the Hello messages

Transport Layer Security

- only minor differences compared to SSL 3.0:
    - pseudorandom function for generating keys and MAC is based on HMAC
    - variable length padding (may prevent traffic analysis)
    - other minor changes

HTTPS

- HTTP over SSL/TLS
- Between the web browser and server:
    - HTTP client = SSL client
    - HTTP server = SSL server
- Conventions

  - URL: `https://` instead of `http://`
  - default TCP port is 443 instead of 80
- HTTP request may be sent after SSL Finished messages
- Protected information
  - URL, contents of the document, browser forms, cookies, headers
  - page served over HTTPS may include elements retrieved using HTTP

5 DNSSEC

Domain Name System (DNS)

- Millions of domain names → distributed database
  - dynamic data ← IP address for a host may change
  - decentralized authority ← each name has an owner
- Hierarchical name space
  - domain: node in the DNS tree
  - for each domain, there is an authoritative server
- Authoritative server
  - responds to queries about the domains for which it is responsible
  - may refer to other authoritative servers for a subdomain

DNS Queries and Responses

- Transport protocol
  - UDP port 53
  - TCP for long responses (and some tasks between nameservers)
- Messages: query and reply
  - 16-bit identification field: match queries with replies
- Caching
  - received responses are cached by servers
  - each record has a Time-to-Live field, after expiry it must be queried again

DNS Weaknesses

- DNS responses are not authenticated
  - responses can be sent over UDP transport protocol → anyone can respond from a spoofed (i.e., fake) IP address to a query
- DNS is a key infrastructure
  - resolvers trust responses, users trust resolvers
  - by tampering with responses, an attacker may direct users to malicious websites or direct e-mail to malicious servers
- DNS cache poisoning
  - attacker sends malicious response to a DNS server, which caches it → malicious response is served to all clients using the server
  - attacker does not have to be man-in-the-middle
- Race to Respond First
  - If the attacker responds before the authoritative server, the DNS cache is poisoned with the malicious IP address
    * fake response is a single UDP packet from the spoofed (i.e., forged) IP address of the authoritative server

Domain Name System Security Extensions (DNSSEC)

- Set of extensions defined by the IETF to the DNS protocol

- Guarantees origin authenticity and data integrity of DNS replies
- Backwards compatible: responses can be interpreted by DNS servers and clients that do not support DNSSEC
  - of course, no guarantees are provided for these servers and clients
- Does not provide confidentiality
  - responses are only authenticated but not encrypted
- Based on public-key cryptography
  - every response is digitally signed

DNSSEC Public Keys and Signatures

- Signature algorithms: RSA-SHA1, RSA-SHA256, ECDSA-SHA256, ...
- Trust anchor
  - known public key for an authoritative nameserver
  - typically included in the operating systems
- Authentication chain

$$\begin{tikzcd} \text{trust anchor} \arrow[dr, bend right] & & & \\ & \text{root} \arrow[r, "\text{signs}"] & \text{edu} \arrow[r, "\text{signs}"] & \text{uh.edu} \end{tikzcd}$$

- Responses may include
  - RRSIG: digital signature for the contents of the response
  - DNSKEY: public key for a zone ← if the response delegates

DNS over HTTPS and TLS

- DNS over HTTPS (DoH)
  - DNS queries and responses are encoded into HTTPS requests and responses
    - provides integrity and confidentiality
    - server may use HTTP/2 Push to send records in advance
  - controversies and criticism: can impede traffic analysis for cybersecurity, may provide false sense of privacy, can impede traffic filtering by ISPs, ...
- DNS over TLS (DoT)
  - provides integrity and confidentiality

6 SSH

Secure Shell (SSH) Protocol Stack

<table>
<tr><td>SSH User Authentication Protocol
Authenticates the client-side user to the server.</td><td>SSH Connection Protocol
Multiplexes the encrypted tunnel into several logical channels.</td></tr>
<tr><td colspan="2">SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity.
It may optionally also provide compression.</td></tr>
<tr><td colspan="2">TCP
Transmission control protocol provides reliable, connection-oriented end-to-end delivery.</td></tr>
<tr><td colspan="2">IP
Internet protocol provides datagram delivery across multiple networks.</td></tr>
</table>

SSH Transport Layer

- Packet Exchange
- Identification string exchange
  - both the client and the server send: `SSH-protocolVersion-softwareVersion`
- Algorithm negotiation (KEXINIT)
  - both the client and the server send the list of key exchange, encryption, MAC, and compression algorithms that they support
  - chosen one: first on the client's list that is also on the server's list
- End of key exchange (NEWKEYS)
  - start using the algorithms and keys

Server authentication

- servers signs a hash of all the earlier messages and the new symmetric key
- servers sends the signature and its public key to the client
- Client needs to verify the public-key sent by the server (i.e., verify that the public key belongs to the server host)
- Trust models
  - Certificate authority
    - client accepts public keys that are certified by a trusted CA
  - Local database
    - client has a list of known pairs of hosts and public-keys
    - typically, each user has a list stored in its home directory
      - default location: `~/.ssh/known_hosts`
- Known Hosts
  - First connection: verify and store host and public key
  - Subsequence connections: compare to stored key

SSH User Authentication Methods

- Passwords
  - client sends a username and a password
- Public key
  - client sends a public key and a signature based on the corresponding private key
  - server checks if the public key is acceptable and verifies the signature
  - typically, for every user account on the server host, there is a list of acceptable public keys stored at `~/.ssh/authorized_keys`
- Host based
  - assumes that the server trusts the client host → since the client host has already authenticated the user, the server only needs to verify the identity of the client host
  - client sends a signature based on the private key of the client host

7 E-mail security

8 Authentication and access control

9 Software vulnerabilities and countermeasures

10 Web vulnerabilities

11 Malware

12 Secure development

13 Detection

14 Isolation

15 Denial of Service attacks

16 Vulnerability scanners