Module 8: Access Control and Authentication

Disclaimer: large parts from Stefan Katzenbeisser, Günter Schäfer

Dresden, WS 18
Reprise from the last modules (30k ft)

Security goals and services describe and implement protection from threats

History has been an arms race between cryptography and cryptanalysis

Each success for the cryptanalysis community has helped make ciphers more secure
• We now try to prove security of ciphers
• The main vulnerabilities in the code, today…

Different flavors of ciphers with different properties aim at confidentiality

Secure pseudo-random numbers are essential for the security of ciphers

MACs and signatures aim at providing integrity

Keys can be agreed upon apriori, exchanged, or agreed upon online

Stream- and block ciphers are commonly symmetric and have different properties

Asymmetric crypto allows for public keys and has many different applications
Module Outline

Access Control
- How do we control access in the physical world
- The basic concept of reference monitors
- MAC/DAC and IBAC/RBAC/ABAC

Authentication
- …in the physical world
- Authentication factors
- Authentication over the network

The (f)utility of Blockchain
Access Control

Recall security goals of confidentiality and integrity

So far, using crypto:

- Conceal information in seemingly random noise
- Prove absence of tampering by signature

- *How does this solution relate to „real life“?*
Confidentiality and Integrity „in RL“
Physical access control

Objects vs. Subjects

Subjects have controlled access to objects
- Prevents information disclosure
- Prevents tampering

Requires some gatekeeper:
- Identification of subjects (authentication)
- Explicit instructions (authorization, policy, <policy descriptions>)
- Controlling (granting/denying) access
Some Terminology

Def: **Access control** comprises mechanisms to enforce mediation on subject *requests for access* to objects as defined in a security policy.

Def: A **subject** is an active entity that can initiate a request for resources and utilize these resources to complete some task.

Def: An **object** is a resource that is used to store, access, or process information.

Def: An **operation** (action) is an instance of access, commonly a utilization, retrieval, or manipulation event, of a subject on an object.

*Objects* historically had the notion of files, or repositories

*Subjects* commonly processes (local or remote)

*Operations* historically: “r,w,x”
Common Concept of Access Control

„Reference monitor“ is a concept to detail decision process:

RM not necessarily a physical/logical component in the system

AC/RM may be implemented on different levels:

- **Online application**: control access to functions/data
- **Databases**: control access to tables, columns
- **OS**: control access to resources (files, devices)
Security Objectives, Levels, and Categories

A **security objective** is a *statement of intent* to counter a given threat or enforce a given organisational security policy.

A **security level** is defined as a *hierarchical attribute* with entities of a system to denote their degree of sensitivity
—Examples:
  - Military: unclassified < confidential < secret < top secret
  - Commercial: public < sensitive < proprietary < restricted

A **security category** is defined as a nonhierarchical *grouping of entities* to help denote their degree of sensitivity
—Example (commercial): department A, department B, administration, etc.

--> Security categories facilitate the “Need-to-know” principle
Security Labels and Policies

A **security label** is defined as an *attribute* that is *associated with system entities* to denote their hierarchical sensitivity level *and* security categories.

Security labels that denote the security sensitivity of:
- Subjects are called *clearances*
- Objects are called *classifications*

The **security policy** of a system defines the *conditions* under which *subject accesses to objects* are granted and denied by the system reference monitor functionality:
- Derived from the organizational policy (IPRs, procedures)
- Compliance to be monitored (on introduction, regularly)
Classes of Security Models

Access control models

• Identity-based access control (IBAC)
• Role-based access control (RBAC)
• Attribute-based access control (ABAC)

Information flow models (e.g. Chinese Wall model)
• Multilevel security models (e.g. Bell-La Padula model)

Non-interference models

General types of access control:
• Discretionary
• Mandatory
Types of Access Control: DAC / MAC

**Discretionary Access Control**
- Owner is responsible for security of her objects
- Authorization per object
- No system-wide security properties
- Rights commonly to be granted: read, write, execute (*NIX, win)
  --> commonly challenged by lack of competence, overview

**Mandatory Access Control**
- System-wide (usually: rule-based) security policy configuration
- User may change authorization, but system policy dominates
  --> commonly challenged by lack of overview, BOfH
IBAC: Access Control Matrix

Task: Configuration of authorizations (rights of subjects on objects)

Define: Set of objects $O$, set of subjects $S$, set of rights $R$ (e.g. rwx…)
Access Control Matrix defines mapping $M : S \times O \rightarrow 2^R$ (e.g.: \{true,false\})

Advantages of ACM:
- Intuitive, flexible
- Easy to implement

Disadvantages of ACM:
- Huge, sparse
- static

<table>
<thead>
<tr>
<th></th>
<th>o1</th>
<th>o2</th>
<th>o3</th>
<th>o4</th>
<th>o5</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>{ read, write }</td>
<td></td>
<td>{ read, write }</td>
<td></td>
<td>{ send, receive }</td>
</tr>
<tr>
<td>s2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>{ send, receive}</td>
</tr>
<tr>
<td>s3</td>
<td></td>
<td></td>
<td>{ owner, execute }</td>
<td></td>
<td>{ signal }</td>
</tr>
</tbody>
</table>
Access Control Description Schemes

Access Control Lists (ACLs)

- Columns of the ACM: list of authorizations on an object
- $ACL(o1) = \{(s1,\{r,w\}), (s2,\{r\})\}; \quad ACL(o2) = \{(s3,\{r,w,x\})\};$
  
  (*NIX: subjects only identified as owner, group, others)

- Assessing authorizations to an object is simple
- Assessing authorizations granted to a subject is difficult

Capabilities

- Rows of ACM: list of objects and rights granted to a subject
- $CL(s3) = \{(o2,\{o,x\}), (o4,\{s,r\})\}; \quad ...$
- Advantages/disadvantages inverse to ACLs...
From IBAC to RBAC

Complexity of IBAC yields problems of overview and adaptation
Subjects usually act in „roles“ (specificly in organizations)
Introduce indirection of the role abstraction:

Dr. Brains
Dr. Bones
Nurse Kathy
Carer Tuck

physician

Read patient information
Write diagnosis
Read prescriptions
Write blood values

User↔Role - Relation  Role↔ Right - Relation
Role-based Access Control

Extend IBAC:
- Set of subjects $S$
- Set of roles $R$
- Set of objects $O$
- Set of permissions $P$

Define mappings $sr: S \rightarrow 2^R$; $rp: R \rightarrow 2^P$

*Sessions* are dynamic role assignments (a subject is active in a role)
Subject is assigned permissions from role for the session accordingly

*Role hierarchies and constraints extend RBAC*
Entity Authentication

„Who‘s Brian of Nazareth? We have an order for him to be released.“

Terry Jones, et al.: Life of Brian (1979)
Identifying humans and Entity Authentication

Goal: Identify a subject (user or process!) and verify identity

Classes of authentication:

• *User authentication (login)*
• *Computer network authentication (identity management)*
• *Identity verification service*

Authentication cardinality:

• One-way authentication
  — Computer authenticates user
  — ATM authenticates cardholder
  — Browser authenticates Web server

• Two-way (mutual) authentication
  — ePass <-- reader
  — UMTS cellphone < -- > network
  — Online bank < -- > account holder (w/ certificates)
Authentication factors

Different factors can be used to authenticate a user

- **Knowledge factors**
  - Passwords
  - Answers to „security questions“
  - ...

- **Possession factors**
  - Security token
  - Smart card
  - Keys/certificates
  - ...

- **Inherence factors**
  - Biometric factors
  - Signature
  - ...

- *Sometimes: other properties (e.g. location)*

**Authenticating a device**

Conventional: knowledge (key)
Advanced: Possession (smart card)
Recent: Inherence (PUF)
Sometimes: location (ATM)
Entity Authentication

Factor verification:
- **direct** (Alice vs. Bob) or
- mediated by an **arbiter** („TTP“, Kerberos, Shibboleth)

Basic requirements:
- Strength of secret determined by its entropy (passwords, biometry)
- Provision and management: factors must remain secret (*impersonation*), be adjustable, possibility for revocation
- Monitoring, detection and reaction of/to malicious authentication attempts

Multi-factor authentication:
- Combines different factors (*examples*)
  - ATM card (possession) and PIN (knowledge)
  - Password (knowledge) and mobileTAN (possession of cell phone)
- Requires independence of factors
- Increases security only as much as weakest factor (security question?)
- (*not to confuse with fall-back authentication – as secure as weakest factor…)
Reminder: The Dolev - Yao Adversary Model

Mallory has full control over the communication channel
- Intercept/eavesdrop on messages (passive)
- Relay messages
- Suppress message delivery
- Replay messages
- Manipulate messages
- Exchange messages
- Forge messages

But:
- Mallory can’t break (secure) cryptographic primitives!
Authentication – Case Studies

Authentication „in RL“
- Identity cards

User authentication
- Knowledge-based: passwords
- Inherence-based: Biometry

Network authentication:
- Kerberos
Individuals authenticating individuals

Consider how you authenticate „in RL“ (@home, bank, …)

What one *is* (inheritance)
- hand geometry
- finger print
- picture
- hand-written signature
- retina-pattern
- voice
- typing characteristics

has
- paper document
- metal key
- magnetic-strip card
- smart card (chip card)
- calculator

knows
- password, passphrase
- answers to questions
- calculation results for numbers

eID-card
Knowledge-based authentication: passwords

Simple approach

User → (dog,bone) → Server

<table>
<thead>
<tr>
<th>Login</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>dog</td>
<td>bone</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Yes → grant access
No → deny access
Password based authentication

Simple approach – **security problems**

User 🦜💻

(dog,bone) 🐶 🐕

Server 🛡️

Login | Password
--- | ---
... | ...
dog | bone
... | ...

Attacker might get access!
Password based authentication

Enhanced approach using one way (hash) functions

User → Server

Server calculates $h(bone)$

<table>
<thead>
<tr>
<th>Login</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>dog</td>
<td>$h(bone)$</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Grant access if the calculated hash matches the one stored.
Deny access otherwise.
Password based authentication

Enhanced approach using one way (hash) functions

User → (dog, bone) → Server

- Server calculates $h(\text{bone})$
- If $h(\text{Bone}) = h(\text{bone})$, grant access
- If $h(\text{Bone}) \neq h(\text{bone})$, deny access

Slightly reduced risk, if attacker gets access.
Problems of hashed password storage

Possible attack:
—pre-computation (rainbow tables)

countermeasures:
—Hash rounds
  • store: \( h^{1000}(pw) \)
  • linear overhead per string (computation, storage)
—Salting
  • Use (long) random value
  • store: \( h(salt,pw) \), salt

\[
\begin{array}{|c|c|}
\hline
\text{Login} & \text{Password} \\
\hline
\ldots & \ldots \\
dog & h(\text{bone}) \\
\ldots & \ldots \\
\hline
\end{array}
\]

\[
\rightarrow \text{pre-computation has to be done for each possible salt, |rounds|}
\]

\[
\text{\$ sudo less /etc/shadow | grep strufe} \\
\text{strufe:$6$m40rV3LS$kAf.4WUEwr7[...].}
\]
Problems of hashed password storage

Remaining attack:
- dictionary attack
- problem: people do not choose passwords randomly
- often names, words, or predictable numbers are used
- attacker uses dictionaries for brute force attack
- prominent program: John the Ripper
  - supports dictionary attacks and password patterns

Possible solutions:
- enforce password rules
  - consider usability
- pre-check (monitor) passwords (e.g., using John)
- require password managers
- train people to "generate" good passwords
  - Example: sentence → password
    - "This is the password I use for Google mail" → "TitpIu4Gm"
  - use password managers
A CRYPTO NERD’S IMAGINATION:

His laptop’s encrypted.
Let’s build a million-dollar cluster to crack it.

No good! It’s 4096-bit RSA!

Blast! Our evil plan is foiled!

WHAT WOULD ACTUALLY HAPPEN:

His laptop’s encrypted.
Drug him and hit him with this $5 wrench until he tells us the password.

Got it.
Reminder: Attacks on Authentication

- Man-in-the-middle attack
- Replay attack

Identify/authenticate parties pro-actively/externally

Ensure freshness!
Trouble with Authentication

Verification of factors over networks is difficult, possible with crypto.

Entity authentication is more than exchange of authentic messages:
— Even if Bob receives authentic messages from Alice during a communication, he can not be sure, if:
  • Alice is actually participating in this specific moment, or if
  • Eve is replaying old messages from Alice

— Especially important, when authentication is only performed at connection-setup time:
  • Example: transmission of a (possibly encrypted) PIN when logging in

— Two principle means to ensure timeliness in cryptographic protocols:
  • Timestamps (require loosely synchronized clocks)
  • Random numbers/Nonces (challenge-response exchanges)
Over the Net: Knowledge-based: Challenge/Response

Prerequisite: Both parties agree on common secret key

Here: Bob wants to authenticate Alice

Alice:
Key $K_{ID}$, Identification ID
Send Login-Information

Bob
Key $K_{ID}$ for ID
Generate RAND (Challenge)

(1) ID
(2) RAND
(3) C

$E(RAND, K_{ID}) = C$

Test: $C'$ = C?
Knowledge-based: Challenge - Response

Assuming the Dolev-Yao adversary:

• Plaintext space for Challenges RAND has to be large!
  — Mallory can intercept and store all tuples (RAND,C)
  — She hence can replay old response

• Cipher has to be known plaintext secure
Inherence factors: Biometry

Identifying an individual by physiological characteristics

Requirements:
- **Universality**: Applies to all individuals
- **Uniqueness**: Different for all individuals
- **Stable**: Property does not change over lifetime
- **Measurable**: reliably, easily with (cheap) sensors
- **Acceptance**: by users
- **Resistant to forgery**
Inherence factors: Biometry

Two-stage process:
- Enrollment: Registration of users
- Verification: Capture and compare biometric data with samples stored during enrollment

Verification is always noisy (statistical hypothesis tests)
- False positives: unauthorized individual is authenticated (FAR) → Security at risk!
- False negatives: authorized individual is rejected (FRR) → Problems for acceptance and usability
- FAR & FRR can be adapted by choice of features/characteristics

**Advantages:**
(+) Cannot be lost/forgotten
(+) Hard to copy

**Disadvantages:**
(-) Very hard to change/reset
(-) Error rate
Inherence factors: Biometry
Inherence factors: Biometry

- Storage of **minutiae**: Branches, endpoints, etc.
- Pattern (coordinates and angle) of minutiae considered to be unique
- Compare measured minutiae with samples stored at enrollment
- Problem: Some minutiae may be missing (noisy, incomplete sensing)!

- Main biometric factor (Face recognition, retina scans increasingly used)
Arbited Authentication, Single Sign On

Goals of Single Sign On-Concepts:

• User is authenticated only once (centrally),
• no separate authentication upon requests to use different services within an administrative domain

Kerberos (MIT, 1980s):

• SSO for services within a „realm“
• Authenticate subjects („principals“): users, computers, servers
• Exchange session keys for principals based on Needham-Schroeder
• Underlying cryptographic primitive of symmetric encryption (DES in Kerberos V. 4, from V. 5 on other algorithms allowed)
Single Sign On: Kerberos

Objectives of Kerberos:
• Security: prevent impersonation of users when accessing a service
• Reliability: service use requires authentication --> reliability and availability
• Transparency: authentication beyond password transparent to user
• Scalability: the system has to support a large number of clients and servers

Design of Kerberos:
• A single trusted server per domain (Key distribution center, KDC)
• Tasks of the KDC are:
  — Authenticating the clients of its domain
  — Issue tickets as authentication tokens

Authentication of a Principal
• Idea: Pre-Shared Secrets between Principal and KDC
  — For users: hashed (MD5) passwords, master key $K_A$ is derived
  — For servers: shared, secret master key $K_S$
Kerberos

Content of a Ticket:

- Each ticket is valid only for a principal $C$ (e.g. Joe) on the specific server $S$ (e.g. NFS), for a specific time:

  - $T_{c,s} = S, C, addr, timestamp, lifetime, K_{c,s}$ with:
    - $S$: Name of the Server,
    - $C$: Name of the requesting client,
    - $addr$: IP address of the requesting client,
    - $timestamp$: current time,
    - $lifetime$: lifetime of the ticket,
    - $K_{c,s}$: Session key for the communication between $S$ and $C$
Kerberos (Protocol, simplified)

- User Joe logs in to local PC with a password

- Local PC (client) C sends ID and Nonce to KDC and requests a ticket for the TGS:
  \[\text{Joe} \rightarrow \text{KDC: Joe, TGS, } \text{Nonce1}\]

- KDC extracts master key of the user from its database and issues a ticket \(T_{\text{Joe,TGS}}\) to authorize utilization of TGS
  \[\text{KDC} \rightarrow \text{Joe: } \{K_{\text{Joe,TGS}, \text{Nonce1}}\}_K\text{Joe}\{T_{\text{Joe,TGS}}\}_K\text{TGS}\]

- Client requests Joe to enter Kerberos password, derives \(K_{\text{Joe}}\) and extracts: \(K_{\text{Joe,TGS}, \text{Nonce1}}\)
Kerberos (Protocol, simplified, ctd.)

- Client requests ticket at TGS to use the NFS server:
  \[
  \text{Joe} \rightarrow \text{TGS: } \{ A_{\text{Joe}} \} K_{\text{Joe,TGS}}, \{ T_{\text{Joe,TGS}} \} K_{\text{TGS}}, \text{ NFS, Nonce2}
  \]
  where \( A_{\text{Joe}} = \text{Joe, IP-Addr, timestamp} \) is called „Authenticator“

- TGS checks Authenticator and sends a ticket for the NFS server to Joe:
  \[
  \text{TGS } \rightarrow \text{Client: } \{ K_{\text{Joe,NFS}}, \text{ Nonce2} \} K_{\text{Joe,TGS}}, \{ T_{\text{Joe,NFS}} \} K_{\text{NFS}}
  \]

- Joe uses ticket at NFS server:
  \[
  \text{Client } \rightarrow \text{NFS: } \{ A_{\text{Joe}} \} K_{\text{Joe,NFS}}, \{ T_{\text{Joe,NFS}} \} K_{\text{NFS}}
  \]

- For mutual authentication:
  \[
  \text{NFS } \rightarrow \text{Client: } \{ \text{timestamp+1} \} K_{\text{Joe,NFS}}
  \]
Kerberos

1. Request TGT
2. TGT, Session Key
3. Request SGT
4. SGT, Session Key
5. Request Service
6. Service Authenticator
Inter-Realm Authentication

Extending Kerberos to multiple realms:

- Establish mutual trust between TGS of different realms: $K_{TGS1,TGS2}$

```
1. Request TGT
2. TGT, Session Key
3. Request TGT_{rem}
4. TGT_{rem}, Session Key
5. Request SGT
6. SGT, Session Key
7. Service Request
8. Service Authenticate
```
Final Use Case Study

On sense and sensibility of

- Blockchain
- Bitcoin
- Smart Contracts

...problems in the context of entity authentication
Protocols and Cryptography (*B*‘*coin & Smart Contracts*)

Consensus in the physical world

- Alice and Bob make an agreement…
- *Ownership*: A villager trades his hut for some whiskey, the other villagers register and (the ledger) remember(s)…
Protocols and Cryptography (*B‘coin & Smart Contracts*)

Consensus in IT – the goal that is hard to achieve

- Simple, at absence of malice, perfect memory

- Simple, if less than half of the participants forget
  \[ \rightarrow \text{Majority vote} \]

- Slightly complicated but possible if less than $1/3$ of the participants actively lie
  \[ \rightarrow \text{Byzantine fault tolerance, consensus protocols} \]

- How do we know how many *individuals participate*?
The Sybil Attack (Douceur 2001)

Meet Sybil Dorsett and her dissociative identity disorder

How do we agree on a consensus now, who are the honest parties?

- Delegated to third parties: Proof by certificate
- Direct identity validation:
  - Solve a puzzle each participant can only solve once during a given period: Proof of work
Consensus on Possession – A Registry / Ledger

1: Everybody gets 3 coins

2: Transactions A->H, B->D, E->A

How do we achieve consensus?

3: At epoch end solve puzzle(state, transactions, „me+1“)

4: First shouts solution! (G: „Solution is X!“)
   Everybody: Agree on new state, go to step 2

Are we sure? Let's agree on the longest block chain. After six or seven rounds, nobody will have enough electricity to change my and all subsequent accepted blocks…
„Smart“ Contracts

Replace transactions to participants with simple programs, e.g.:

„Here‘s a coin.

If I there‘s a transaction from A to somebody
then give this coin to A“

(Once this happened, the coin is gone!).
A few Observations on Blockchain-Technology

• The ledger is public (so are all transactions)
• The ledger grows constantly (~210GB/~1TB)

• Blockchains are limited in speed (approx. 6 vs. 20 Tx/s)

• Trust is based on
  — burning electricity (and the assumption that „bad guys“ can’t provide more electricity than „good guys“)
  — the majority agrees to accept the longest chain (oh, unless...)¹

• „Smart“ contracts rely on trusted input and their effect (other than blockchain transactions) has to be enforced by external means.
• Coins spent by „smart“ contracts are spent – either the contract holds a coin, or you have to trust that the committing participant has more coins
• „Smart“ contracts are code. Programmers make mistakes. (Trust me.)

• It’s solely necessary to defeat the Sybil attack (uncertified nodes) – which does not apply for any business model of a serious company.

Summary
You know means for confidentiality & integrity other than crypto ;-) 

You can distinguish between authorization, authentication, access control

You can explain ACMs, ACLs and capabilities

You can distinguish DAC and MAC

You know the different general strategies of IBAC and RBAC

You can explain different classes of authentication and factors

You know about cardinality and multi-factor authentication

You can explain some knowledge-based authentication schemes, strategies

You know about biometry and its advantages and disadvantages