Systems Security Group | Lancaster University

Access Control

Learning Objectives

- Understand the main stages of access control (AC)
- Familiarise with mechanisms in each stage of AC
- Learn about AC models, policies and mechanisms

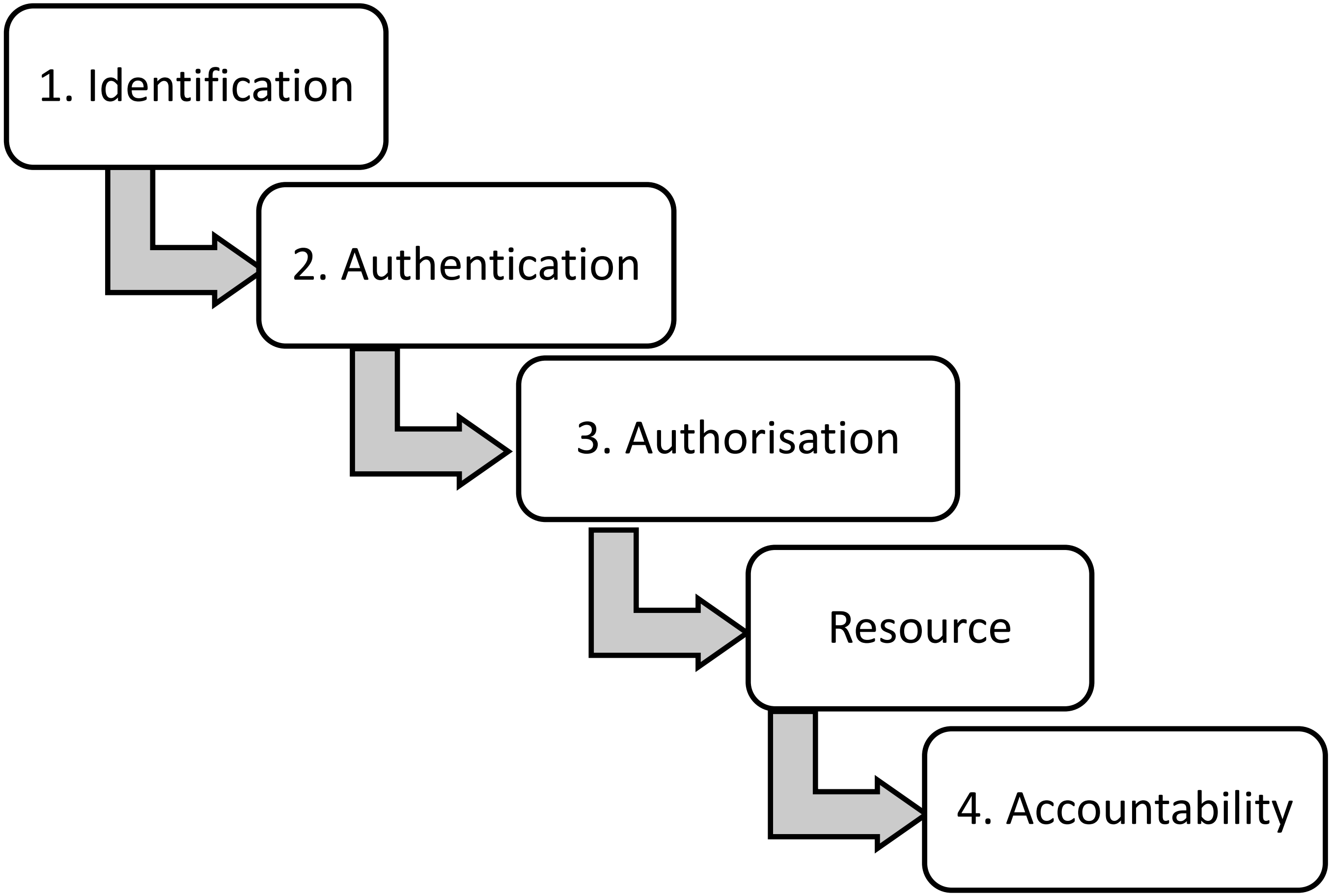
Access controls

- Set of security features that control how users and systems communicate and interact with other systems and resources
- Offer protection against unauthorised access to system resources
- Determine the level of authorisation after a successful authentication

Definitions

- **Access:** the flow of information between a subject and an object
- **Subject:** an active entity that requests access to an object or the data within an object
- **Object:** a passive entity that contains information
- Relationship is defined by the object owner

Steps for a subject to access an object



Identification

- Ensure that a subject is the entity it claims to be
- Identification information may be public information
  - User name, account number, etc.
- Creation of identities should consider
  - Uniqueness for accountability
  - Naming conventions
  - Not shared between several subjects
  - Issuance: Which authority validated or proved the identity?

Identity management (IdM)

- Identity management (IdM) describes the management of individual identifiers
- Different products to identify, authenticate and authorise users through automated means
  - Account management
    - Creation of an account
    - Offer management of privileges
    - Decommission of an account
  - Password management
  - Single Sign-on
  - Profile update

Authentication

- Authentication is private information – 3 factors
  - Something a person knows
    - Authentication by knowledge
  - Something a person has
    - Authentication by ownership
  - Something a person is
    - Authentication by characteristic
- Strong authentication or two-factor authentication include two of the above three categories.

Password ?

Physical Access ?

Biometrics ?

Password plus Authenticator

Password attacks

- Electronic monitoring
- Access the password file
- Brute force attack
- Dictionary attack

Password attacks

- Rainbow tables
  - Use tables that contain all possible passwords already in a hash format
- Social engineering
  - An attacker convinces an individual that she has the necessary authorization to access specific resources.
- Tools to verify password strength analysis have different name depending on who is using them
  - Security professionals use password checker
  - Hackers use password cracker

Example: UNIX-style password

- How should we store passwords?
  - In cleartext?
  - Encrypted? → Would need a key, to protect it would be a challenge
  - Hashed?

Password hashing

- Instead of user password, store H(password)
- When a user enters password, compute its hash and compare with entry in password file
- Hash function H must have some properties

Dictionary attack

- Password file /etc/passwd is world-readable
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can compute H(word) for every word in the dictionary and see if the results is in the password file

Salt

- Users with the same password have different entries in the password file
- Example, assuming ’user1’ with password ‘mypass’
- Hashed value will be H(‘mypass’+salt)

```
 user    alg   salt                        md5
user1:$1$cvASsn/U$ 76d47e44c7bf1419ef207d0cc679f2bb
```

```
import hashlib
H=hashlib.md5()
H.update("mypass")
H.hexdigest()
H.update("mypass"+"cvASsn/U")
H.hexdigest()
```

Advantages of salting

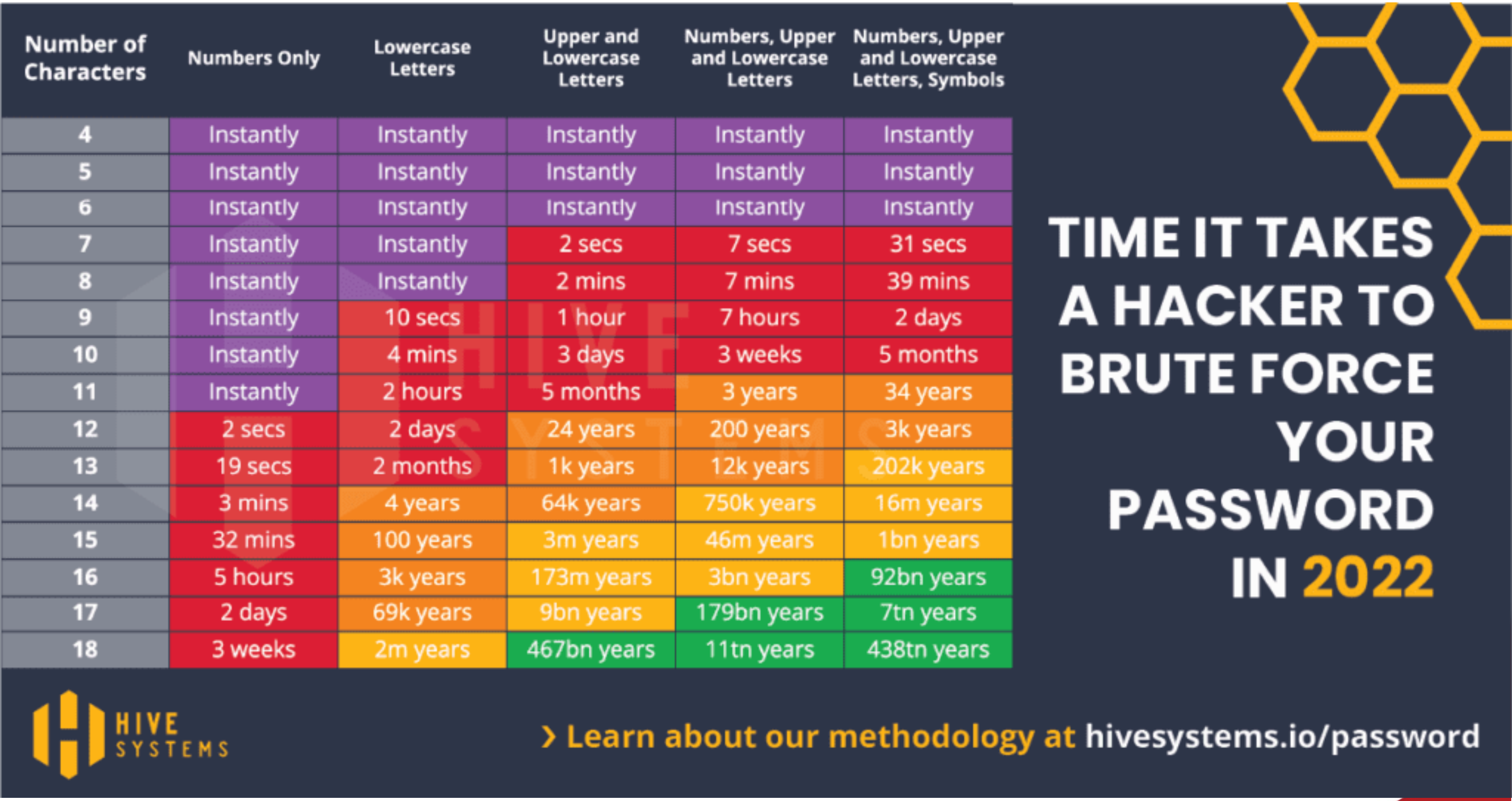
- Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
- With salt, attacker must compute hashes of all dictionary words once for each password entry
  - With 1 byte of random salt, same password can hash to 2^8 different hash values

Shadow passwords

- Hashed passwords are not stored in a world-readable file
- Store hashed passwords in /etc/shadow file, which is only readable by the system administrator (root)
- Add expiration dates for passwords

```
user1:x:1000:1000:name:/home/user1:/bin/bash
```

Time to crack a password



Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	2 secs	7 secs	31 secs
8	Instantly	Instantly	2 mins	7 mins	39 mins
9	Instantly	10 secs	1 hour	7 hours	2 days
10	Instantly	4 mins	3 days	3 weeks	5 months
11	Instantly	2 hours	5 months	3 years	34 years
12	2 secs	2 days	24 years	200 years	3k years
13	19 secs	2 months	1k years	12k years	202k years
14	3 mins	4 years	64k years	750k years	16m years
15	32 mins	100 years	3m years	46m years	1bn years
16	5 hours	3k years	173m years	3bn years	92bn years
17	2 days	69k years	9bn years	179bn years	7tn years
18	3 weeks	2m years	467bn years	11tn years	438tn years

TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD IN 2022

HIVE SYSTEMS

> Learn about our methodology at hivesystems.io/password

Biometrics

- Verify the identify by analysing unique personal attributes or behaviour
  - Physiological: What you are
  - Behavioural: What you do
- Perform accurate and repeatable measurements
- False Rejection Rate (FRR): Type I error
- False Acceptance Rate (FAR): Type II error
- The lower the number, the more accurate the system is

Biometrics

- Fingerprint, facial scan
- Retina scan

and more…

- How about their cost?
- What's the user acceptance?

Authorisation

- Access criteria
  - Trust in the subject
  - Subject’s need-to-know
- Criteria can be enforced by
  - Roles
  - Physical or logical location
  - Time of day

Authorisation

- Default to 'No Access' → Start from the ground up, slowly give out privileges based on necessity

- Authorisations creep: regularly review the principle of Least Privilege

- Least Privilege: every subject must be able to access only objects that are necessary for its legitimate purpose.

Single Sign-On (SSO)

- Enter credentials once
- Reduce time to authenticate to resources
- Streamline account management
- Issues
  - Interoperability
  - Potentially only one layer of security
- Technologies
  - Kerberos (https://web.mit.edu/Kerberos/)
  - SESAME (https://www.cosic.esat.kuleuven.be/sesame/html/sesame_what.html)
  - Security Domains
  - Social login

Accountability

- Accountability is tracked by recording user, system and application activities
- Used to track back individuals, detect intrusions, produce reports and legal resource material
- Huge amount of data – use of tools (e.g., audit-reduction tools) to review audit information

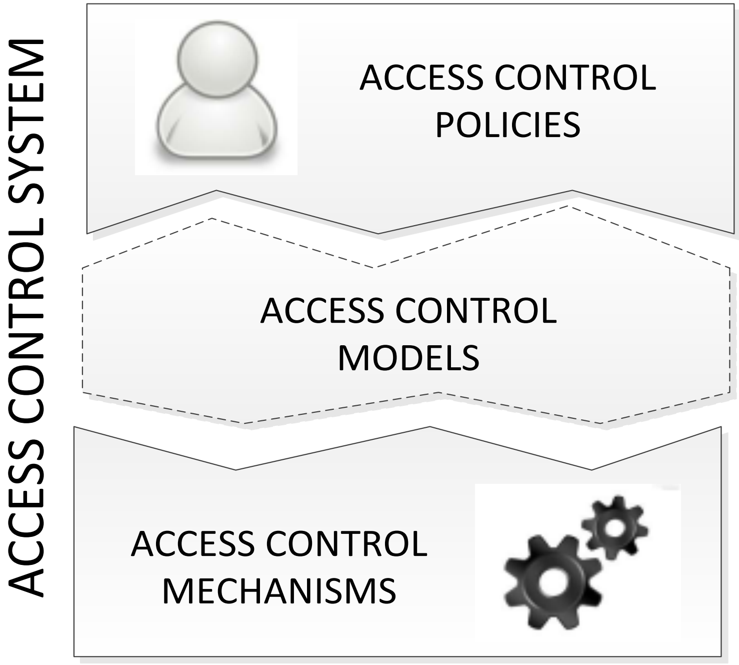
Access control review

- Identification
  - A subject may provide identification information, e.g., username
- Authentication
  - Verify identification information, e.g., password, biometric
- Authorisation
  - Determine what operations subjects have on objects
- Accountability
  - Monitoring and logging information to track subject activities with objects

Access control (AC) systems

- AC Policies enforced through AC Mechanisms

- AC Models bridges the gap between AC Policies and AC Mechanisms



Types of access controls policies

- Mandatory Access Control (MAC)
- Discretionary Access Control (DAC)
- Role Based Access Control (RBAC)
- Attribute Based Access Control (ABAC)

Mandatory access control (MAC)

- Use of a labelling mechanism to enforce a multilevel security model,
  e.g., Unclassified < Confidential < Secret < Top Secret
- Implemented by the operating system
- Security labels are attached to all subjects and objects
- Users will be denied unless their clearance is equivalent or higher that the classification of the object
- Implemented in SE Linux, and trusted Solaris

Bell-LaPadula model

- Enforces confidentiality
- Is a subject-object model: use of subjects, objects and access operations (read, write, read/write)
- How it works?
  - The subject's clearance is compared with the object's classification
  - Specific rules are applied to control how the subject-object interactions take place

Bell-LaPadula rules

- Simple security (no read up)
  - A subject at a given security level cannot read data that reside at a higher security level
- *-property (no write down)
  - A subject in a given security level cannot write information to a lower security level
- Strong *-property
  - A subject that has read and write capabilities can only perform those functions at the same security level. Nothing higher and nothing lower.

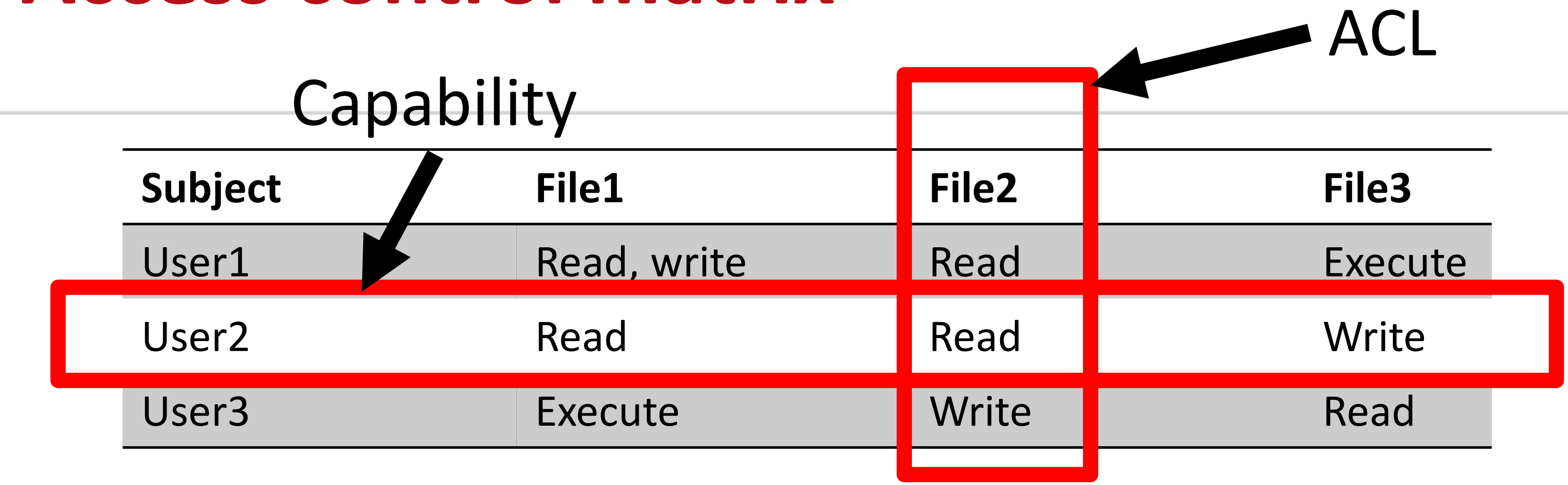
Biba model

- Describes a set of rules that are designed to ensure data integrity
  - "read-up, write-down" model
- Simple integrity property (no read down)
  - A subject at a given level of integrity must not read data at a lower integrity level
- *- integrity property (no write up)
  - A subject at a given level of integrity must not write data at a higher level of integrity
- Invocation property
  - A process from below cannot request higher access.

Discretionary access control (DAC)

- The owner of the resource decides which subjects can access the resource
- Implemented via access control lists (ACLs)
- Used in most operating systems, Linux, Unix, Windows
- Based on sets that define security subjects (s), security objects (o) and access privileges (a)
- Access rules are defined as tuples (o, s, a)

Access control matrix



ACL

Capability

Subject	File1	File2	File3
User1	Read, write	Read	Execute
User2	Read	Read	Write
User3	Execute	Write	Read

- ACL
  - …
  - File2 – User1: Read, User2: Read, User3: Write
  - …
- Capability
  - …
  - User2 – File1: Read, File2: Read, File3: Write
  - …

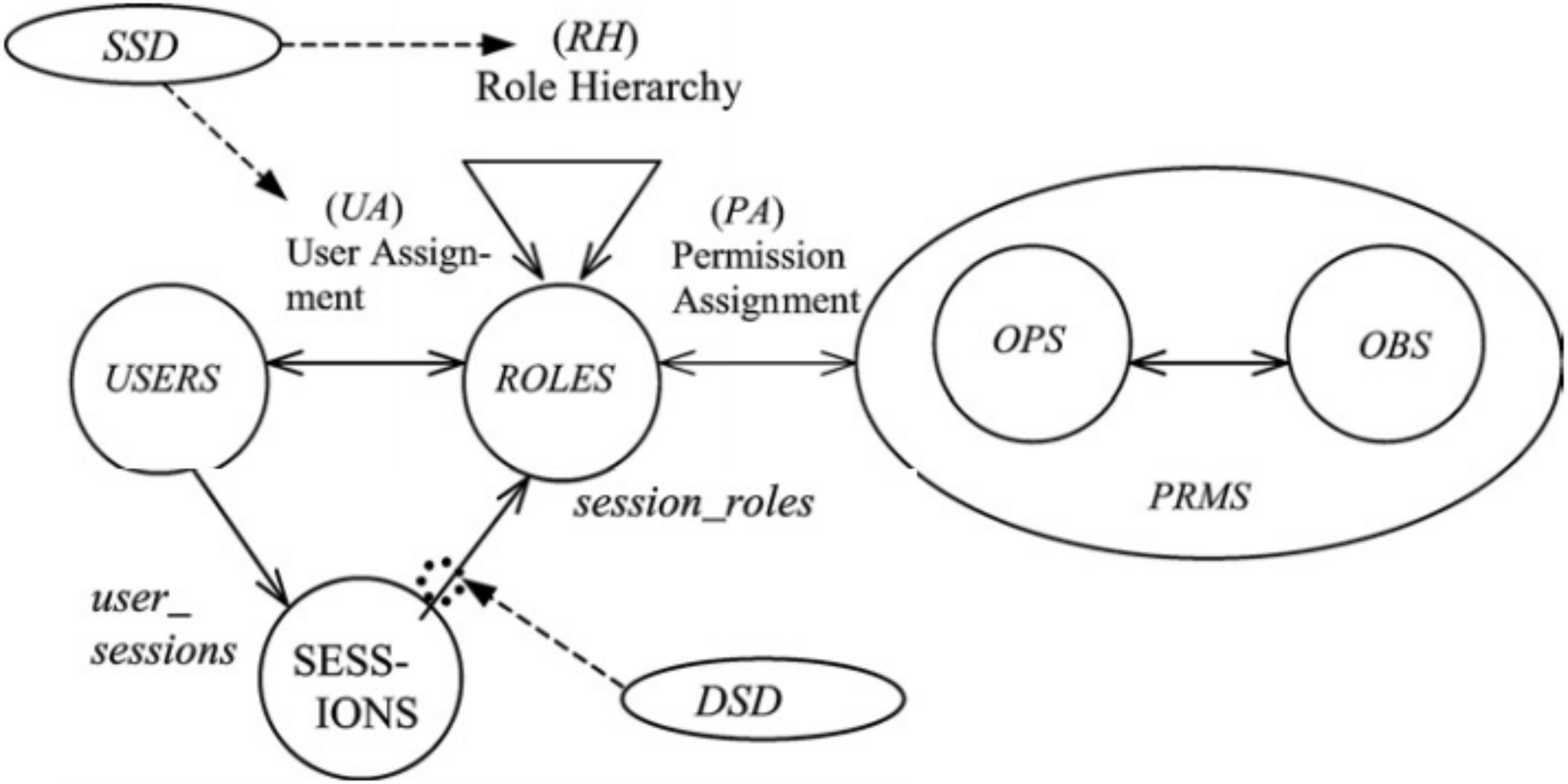
Role based access control (RBAC)

- Centrally administrated set of controls
- Supports the principles of least privilege and separation of duties.
- Useful in high employee turnover environments
- Has been standardised by the American National Standards organisation – ANSI INCITS 359-2004 (http://profsandhu.com/journals/tissec/ANSI+INCITS+359-2004.pdf)

Separation of Duties (SoD)

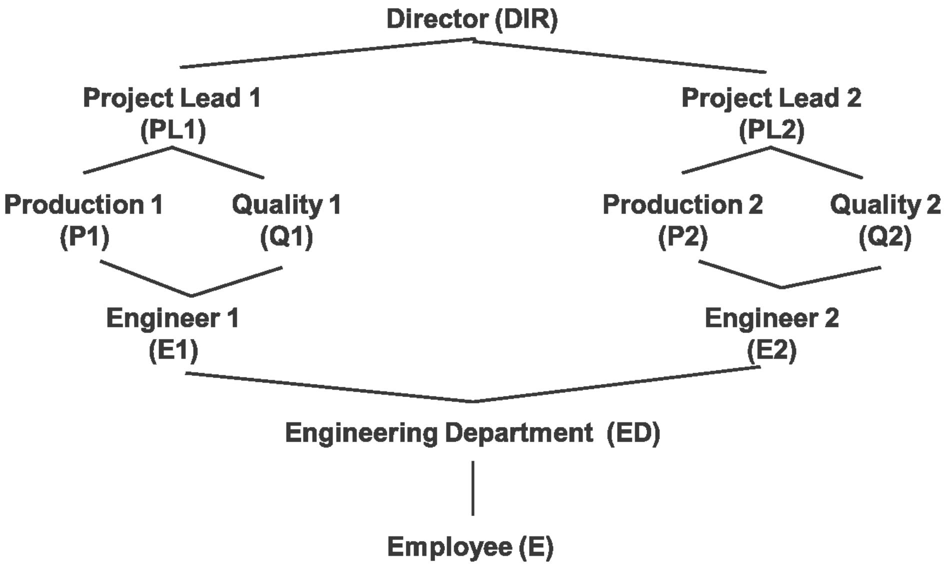
- Security method to manage conflict of interest and fraud
- Restricts the power held by an individual
- Example:
  - Accounting Employee A: Maintains cash balances per books
  - Assistant Cashier B: Maintains custody of cash on hand
  - Assistant control C: Makes monthly comparisons: reports any differences to the controller
  - A ← Separation of Duties→ B

The RBAC model



Family of RBAC models

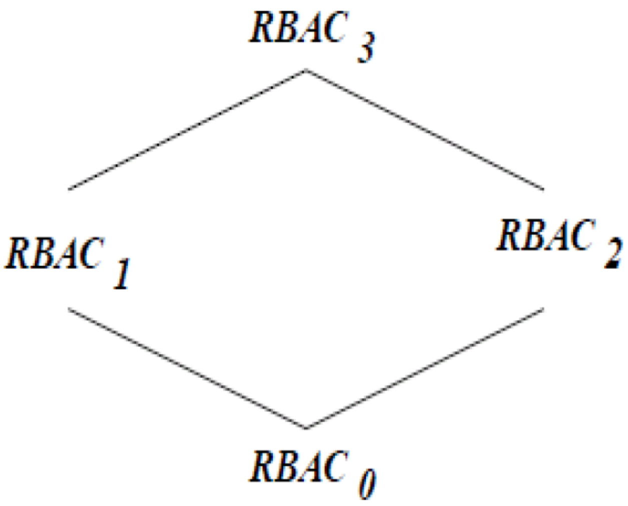
- Hierarchical
  - Support of hierarchies
  - Senior roles on top
  - Junior roles at the bottom



- Support of Constraints
  - Static separation of duties
  - Dynamic separation of duties

Family of RBAC models

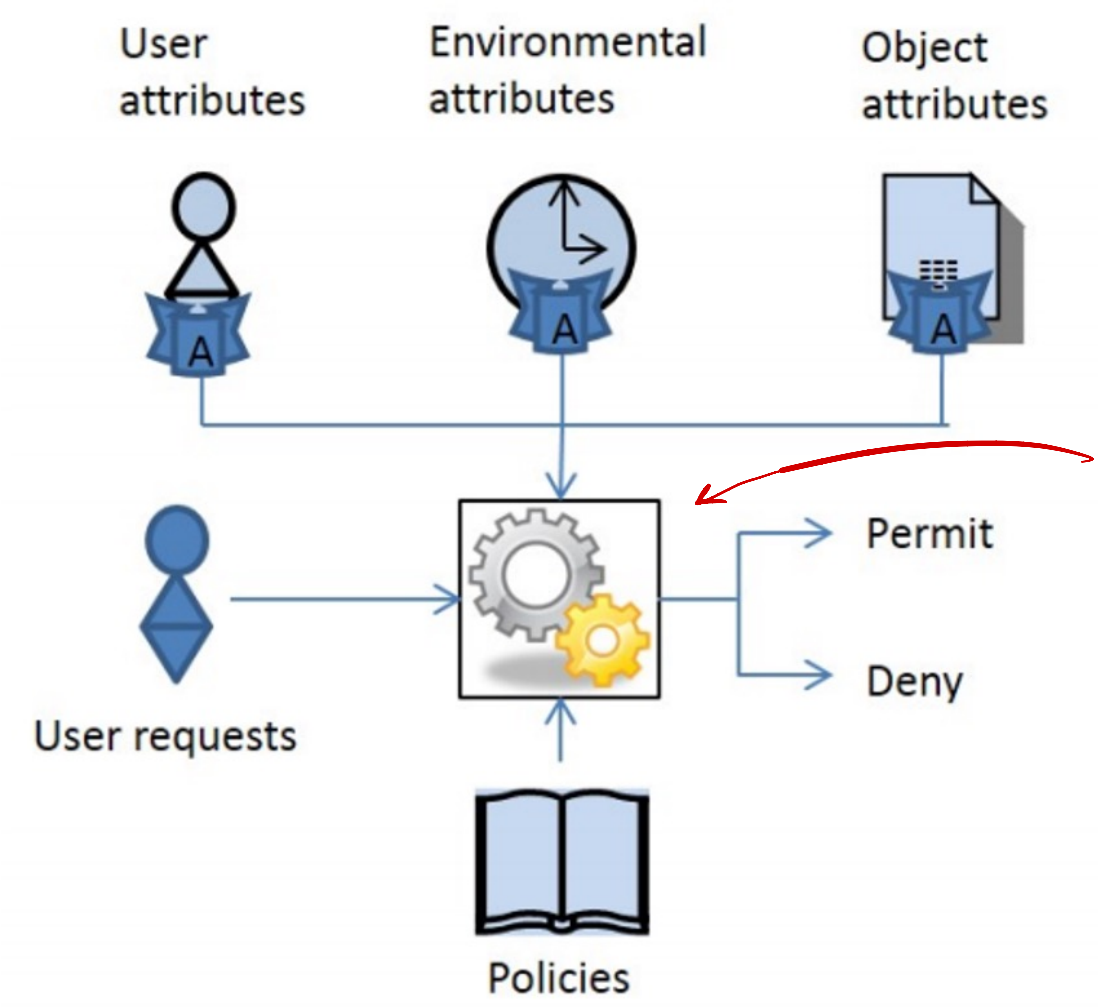
Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes



Attribute based access control (ABAC)

- Logical access control methodology
- Authorisations are determined by evaluating attributes of elements, including environment conditions against rules.
- Standards proposed by NIST in Special Publication 800-162 (https://csrc.nist.gov/publications/detail/sp/800-162/final)

ABAC mechanism



Collects all conditions to come to a conclusion

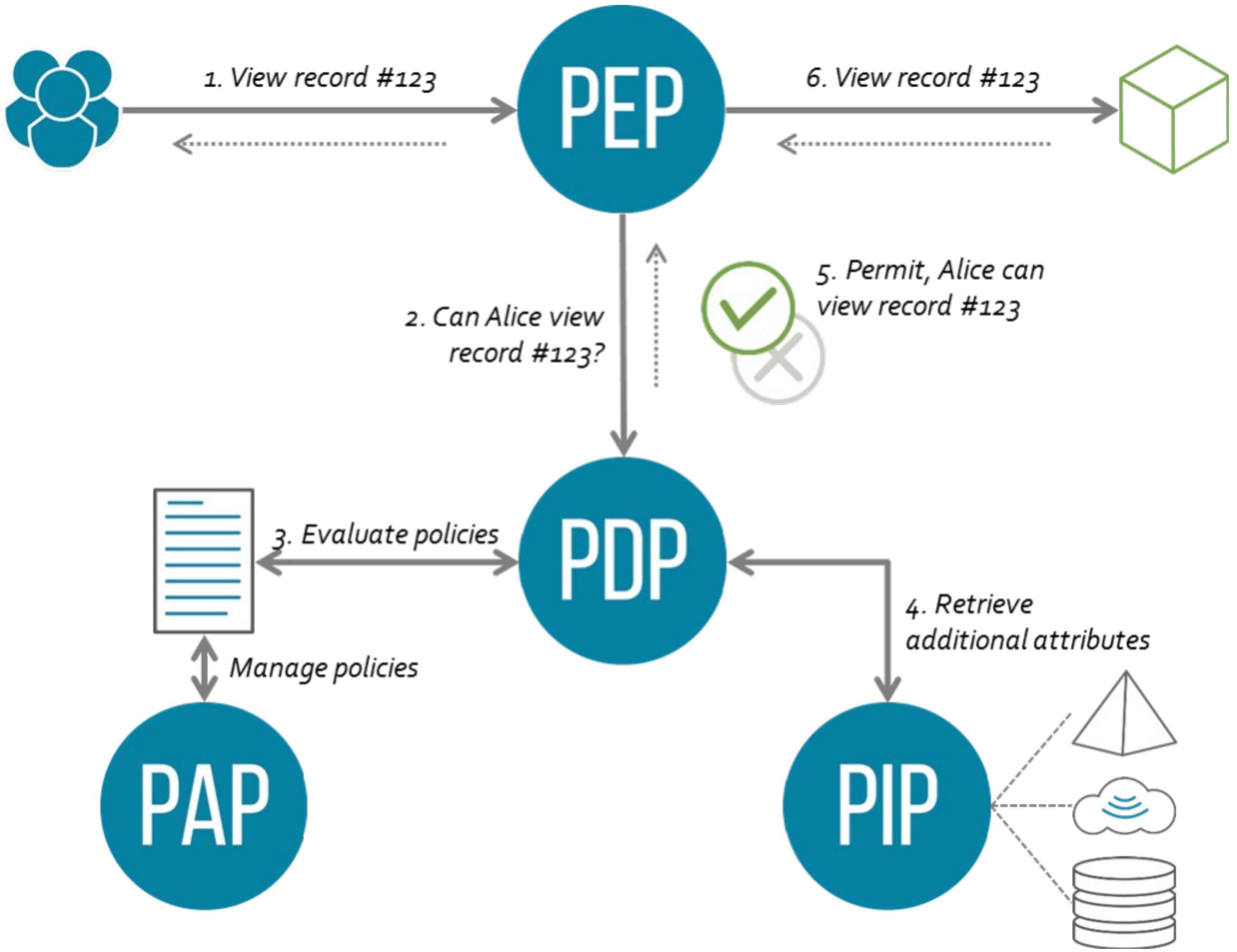
ABAC Frameworks

- Frameworks provide useful guidelines when considering implementation of AC systems

- Main ABAC frameworks
  - Extensible Access Control Markup Language (XACML)
  - Next Generation Access Control (NGAC)

- Provide operations to manage policies, evaluate decision, enforce policies, etc.

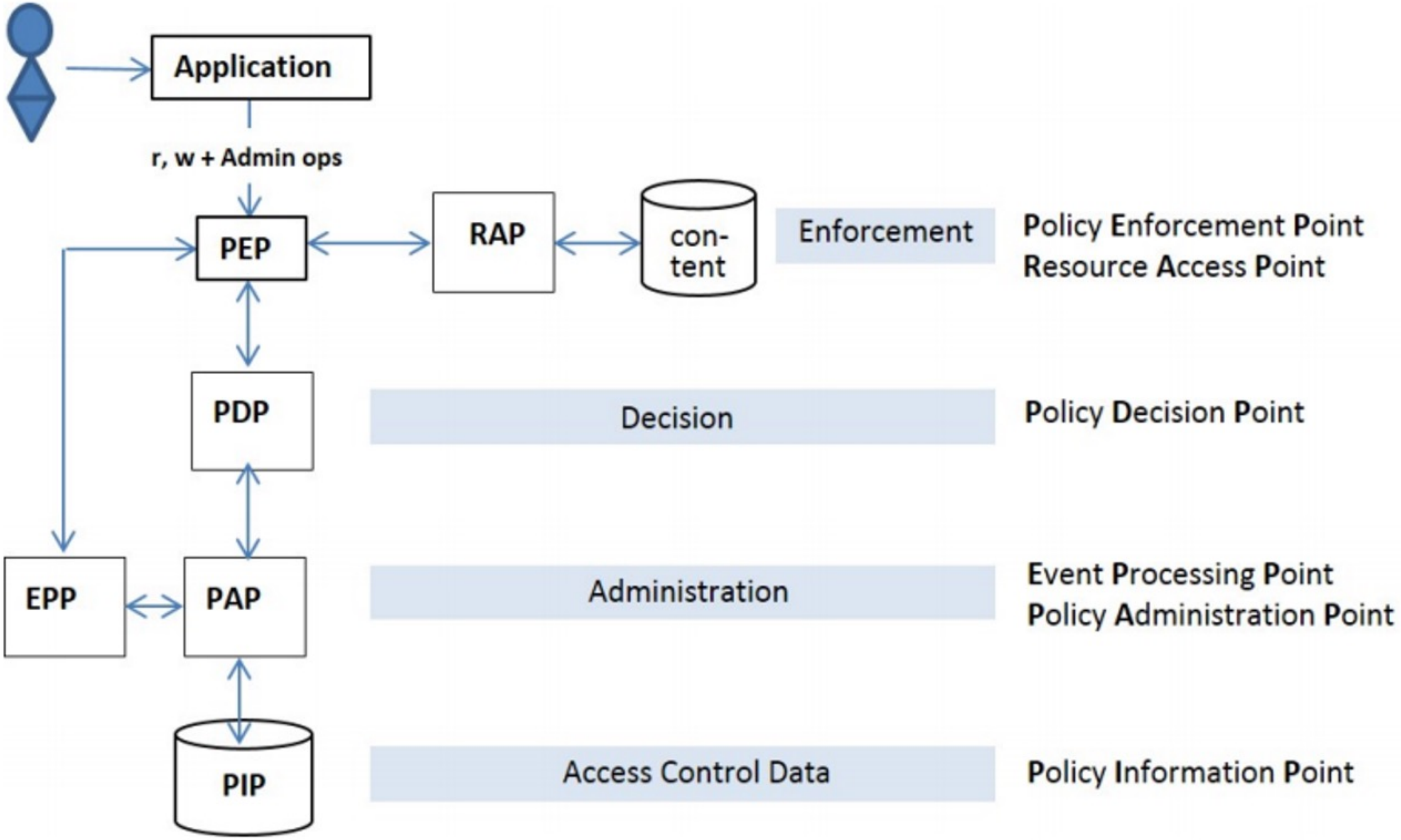
XACML Reference architecture


1. View record #123
PEP
6. View record #123
2. Can Alice view record #123?
5. Permit, Alice can view record #123
PDP
3. Evaluate policies
4. Retrieve additional attributes
Manage policies
PAP
PIP

NGAC

- An attempt to standardise the ABAC mechanism
- Recommended by NIST
- Able to express and enforce a wide range of policies and defined in accordance to ABAC to meet its requirements

NGAC standard function architecture



Application

r, w + Admin ops

PEP

RAP

con-
tent

Enforcement

Policy Enforcement Point
Resource Access Point

PDP

Decision

Policy Decision Point

EPP

PAP

Administration

Event Processing Point
Policy Administration Point

PIP

Access Control Data

Policy Information Point

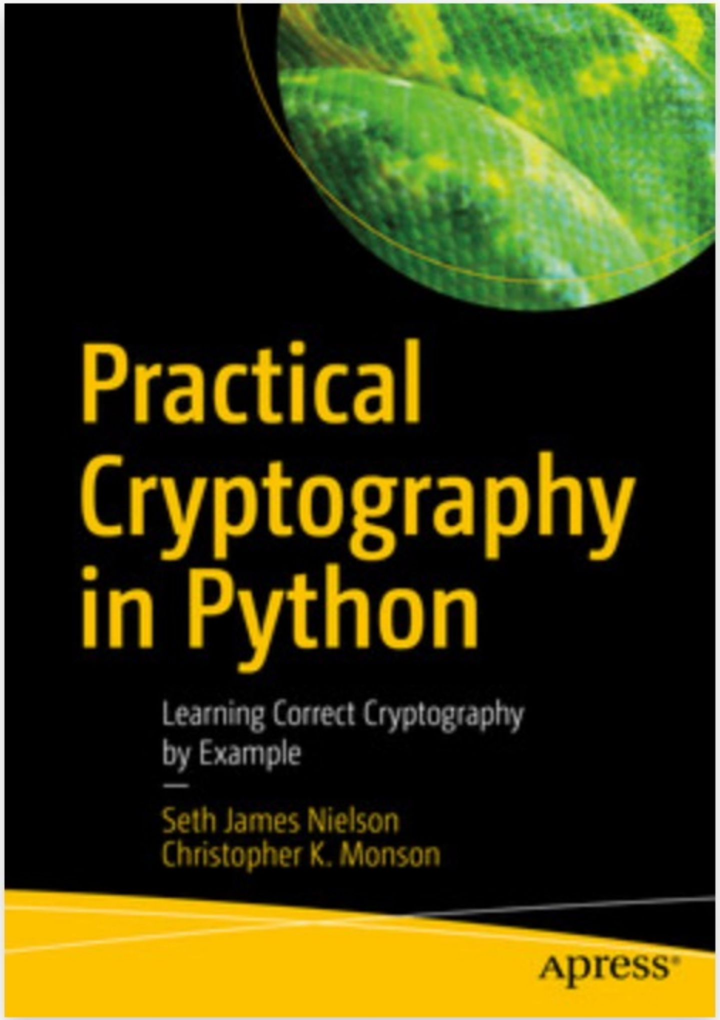
Questions?

References

- Security Engineering, Chapter on Access Control, https://www.cl.cam.ac.uk/~rja14/book.html
- All in one - CISSP, 5th edition, Chapter 4: Access Control
- NIST SP 800-162 https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf
- NIST SP 800-178 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-178.pdf
- ANSI INCITS 359-2004 https://www.cs.purdue.edu/homes/ninghui/readings/AccessControl/ANSI+INCITS+359-2004.pdf

Week 13 Symmetric encryption

Recommended reading


Practical Cryptography in Python
Learning Correct Cryptography by Example
Seth James Nielson
Christopher K. Monson
Apress

The book is available to you via the library

Technology stack

- Python 3
  Link to a Python Cheat Sheet

- cryptography.io
  Link to the library

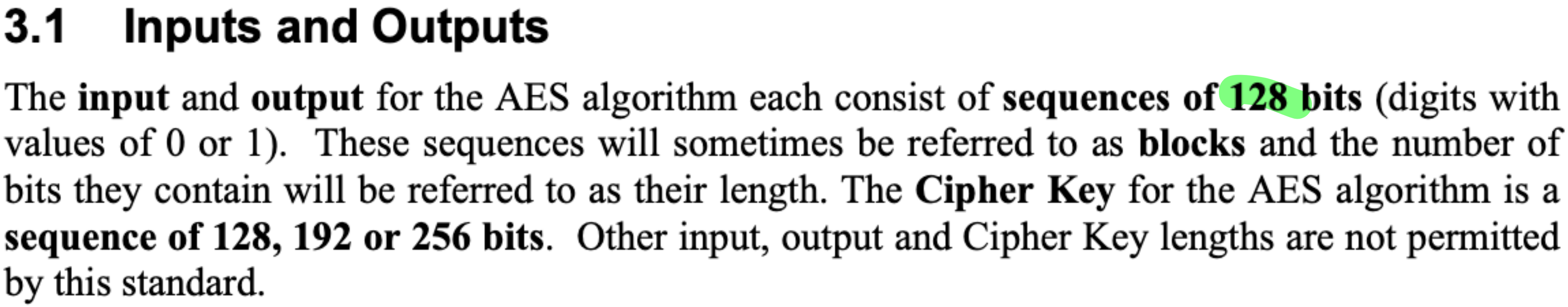
Topics

- AES – ECB
- Encrypt a B&W file in AES-ECB
- Padding
- AES-CTR

Recommended reading: Chapters 3 from the book of "Practical Cryptography in Python"

Collisions in MD5

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

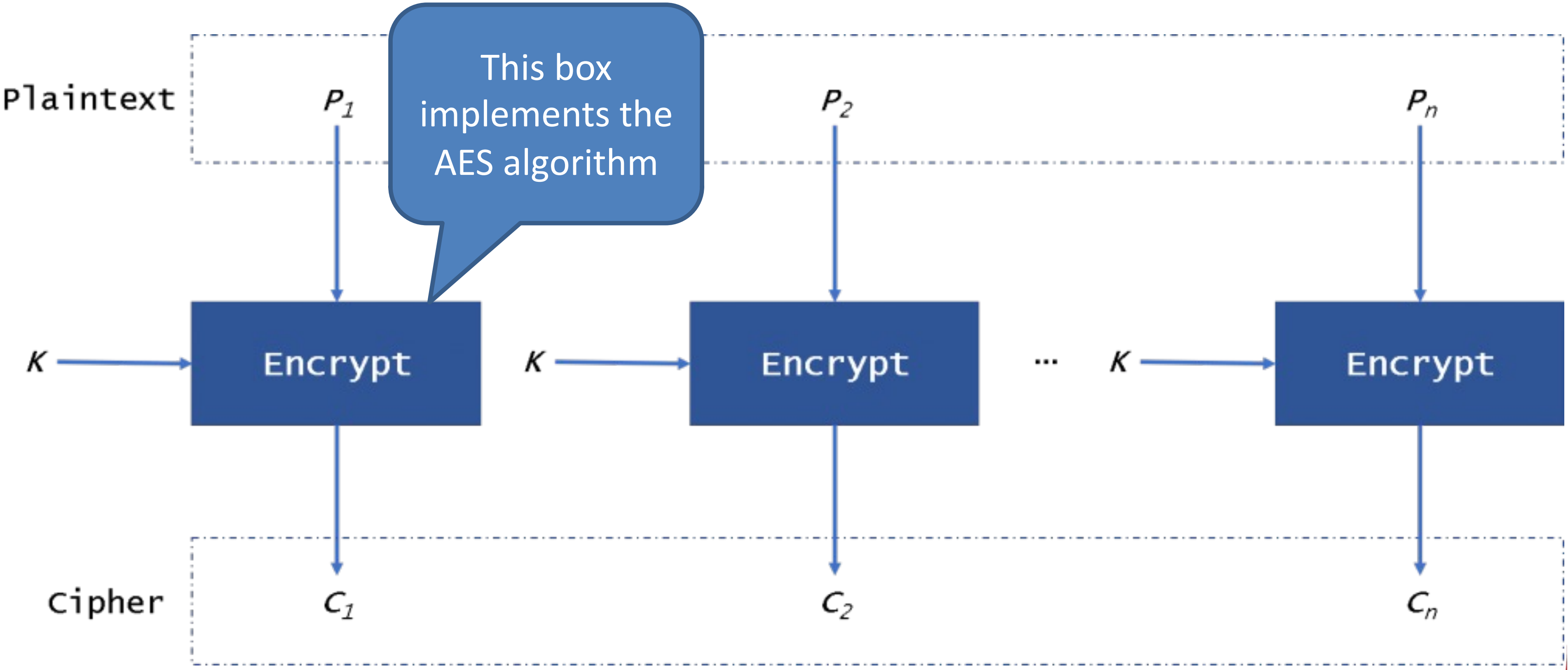


3.1 Inputs and Outputs

The **input** and **output** for the AES algorithm each consist of **sequences of 128 bits** (digits with values of 0 or 1). These sequences will sometimes be referred to as **blocks** and the number of bits they contain will be referred to as their length. The **Cipher Key** for the AES algorithm is a **sequence of 128, 192 or 256 bits**. Other input, output and Cipher Key lengths are not permitted by this standard.

If sequence ≤ 128, padding added to equal 128

AES – Electronic Code Book (ECB) - Encrypt

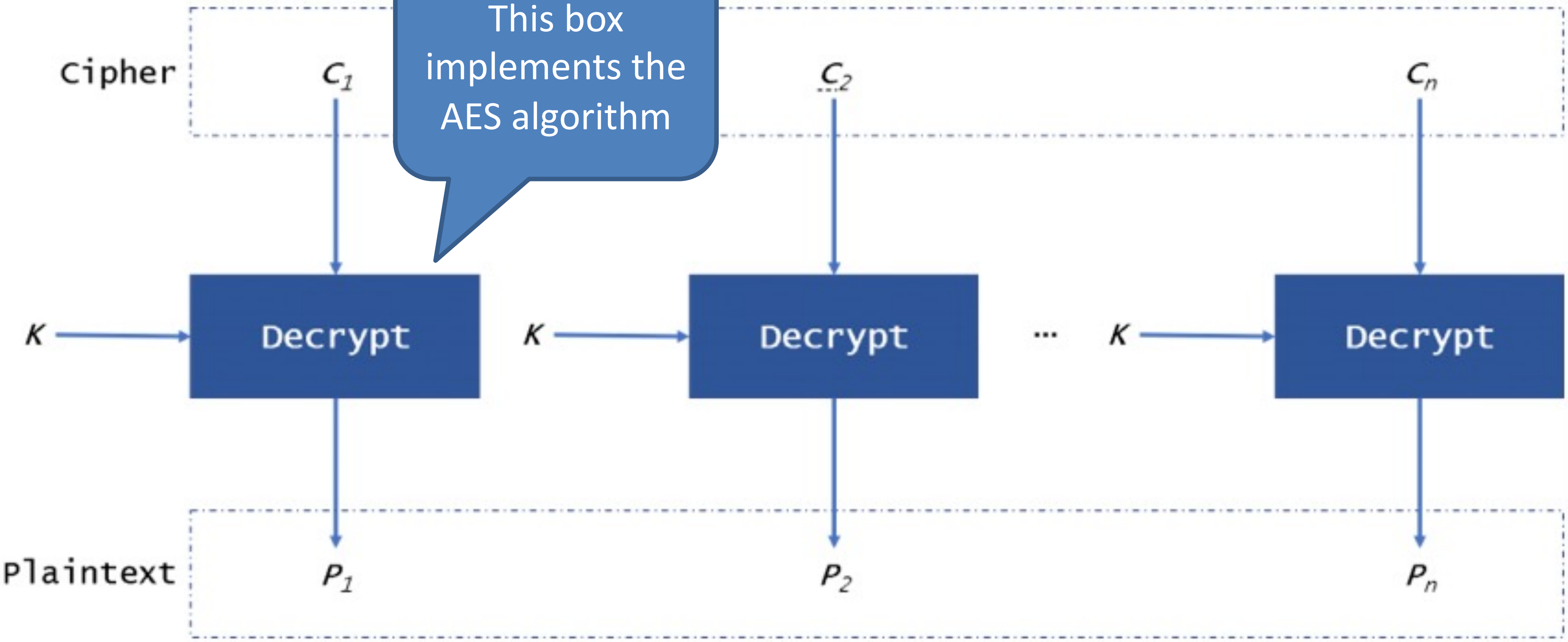


This box implements the AES algorithm

Issue w/ diagram → same key ?

https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/

AES – Electronic Code Book (ECB) - Decrypt



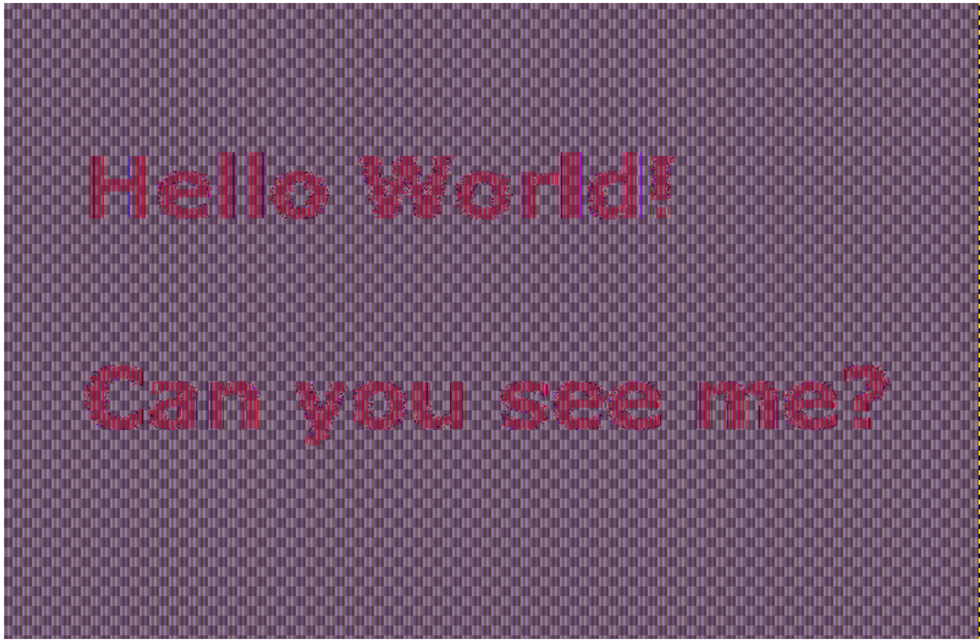
This box implements the AES algorithm

https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/

Example

Hello World!

Can you see me?



Are we forgetting something?

```
from cryptography.hazmat.primitives.ciphers import Cipher,
algorithms, modes
import os

def SimpleECB():
        key = os.urandom(32)
        aesCipher = Cipher(algorithms.AES(key), modes.ECB())
        aesEncryptor = aesCipher.encryptor()
        aesDecryptor = aesCipher.decryptor()

        message = b"Hello world"

        cipherText = aesEncryptor.update(message)
        print(cipherText)

        plainText = aesDecryptor.update(cipherText)
        print(plainText)
```

Padding – PKCS7

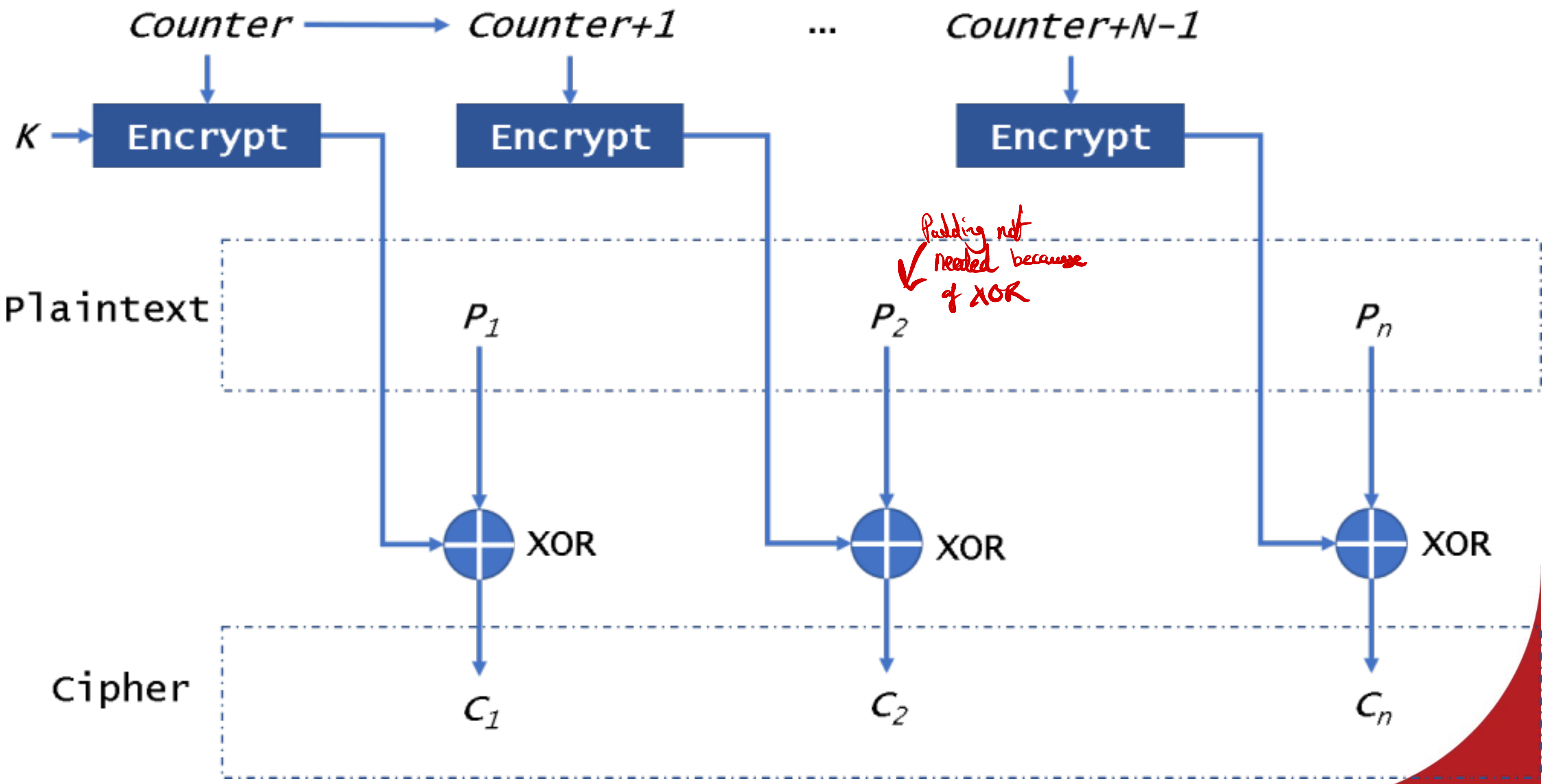
```
def SimpleECB():
        …
        message = b"Hello world"
        padder = padding.PKCS7(128).padder()
        unpadder = padding.PKCS7(128).unpadder()

        paddedMessage = padder.update(message) +
padder.finalize()
        cipherText = aesEncryptor.update(paddedMessage)
        print(cipherText)
        plainText = aesDecryptor.update(cipherText)
        plainText = unpadder.update(plainText) +
unpadder.finalize()
        print(plainText)
```

AES – Counter- CTR

not a block cipher, but a string cipher

only counter is encrypted,



Padding not needed because of XOR

Structure of your code…

Modules you want to import

```
import XYZ
```

List of functions you implement

```
def  myFunction():
     # TODO

     return # TODO
```

Have a main section to call your functions

```
if __name__ == “__main__”:
     x = myFunction()
```