Manolis (Emmanouil Vasilomanolakis)

network security: authentication

Course plan

- Lecture 1: Intro (**Manolis**, **Carsten**) 30.01
- Lecture 2: **Crypto** essentials (**Carsten**) 06.02
- Lecture 3: **Authentication** (**Manolis**), lab bootcamp (TAs) 13.02
- Lecture 4: **TLS** (**Manolis**) 20.02
- Lecture 5: **Threat detection** (**Manolis**) 27.02
- Lecture 6: Hacking Lab day (**TAs**) – blue team 05.03
- Lecture 7: **IoT security** (**Manolis**) 12.03
- Lecture 8: **WIFI security** (**Manolis**) 19.03
- Lecture 9: **Private communication** (**Carsten**) 02.04
- Lecture 10: **When everything fails** (**Manolis**) 09.04
- Lecture 11: Hacking Lab day (**TAs**) – red team 16.04
- Lecture 12: Guest lecture (OT security, **Ludwig**) 23.04
- Lecture 13: **Exam preps** (**Carsten, Manolis**) 30.04

Outline

- **Introduction**
- **Authentication of humans**
  - NIST digital authentication model
  - Authenticators
- **Authentication of machines**
  - Needham–Schroeder
  - Kerberos
  - Active directory
- **Bootcamp: setting up your lab environments**
- **Lab exercise**

Authentication

- **Authentication of humans**
  - –How can you, with high confidence, authenticate
  - –Authenticators
  - –From authenticator -> crypto key
- **Authentication of machines**
  - –How can machines exchange keys
  - –(mutually) authenticate
  - –Identity and access management

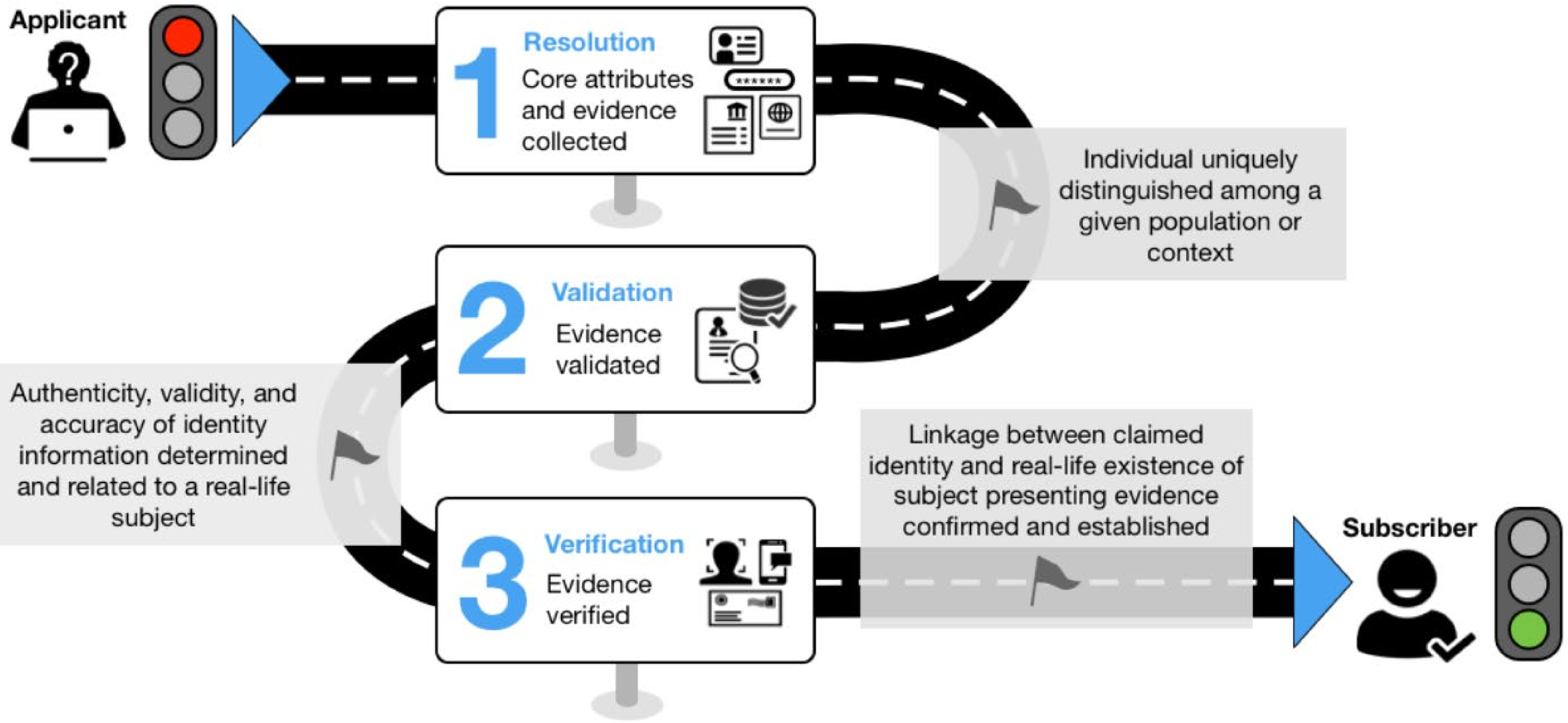
Outline

- Introduction
- **Authentication of humans**
  - NIST digital authentication model
  - Authenticators
- **Authentication of machines**
  - Needham–Schroeder
  - Kerberos
  - Active directory
- **Bootcamp: setting up your lab environments**
- **Lab exercise**

Digital Authentication

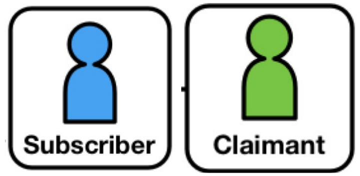
- Process of establishing confidence in user identities electronically presented to an information system
- The service **verifies** the **authenticity** of the **identity** and determines if that individual is **authorized** to perform a **transaction**
- The digital identity model provides different levels of complexity based on the classes of application:
  - Separate functions like issuing of credentials and providing of attributes are used in the model

The identity proofing user journey



Terminology

- **Registration Authority** - A trusted entity that establishes and vouches for the identity or attributes of a Subscriber to a CSP
  - The RA may be an integral part of a CSP, or it may be independent of a CSP, but has a relationship to the CSP(s)



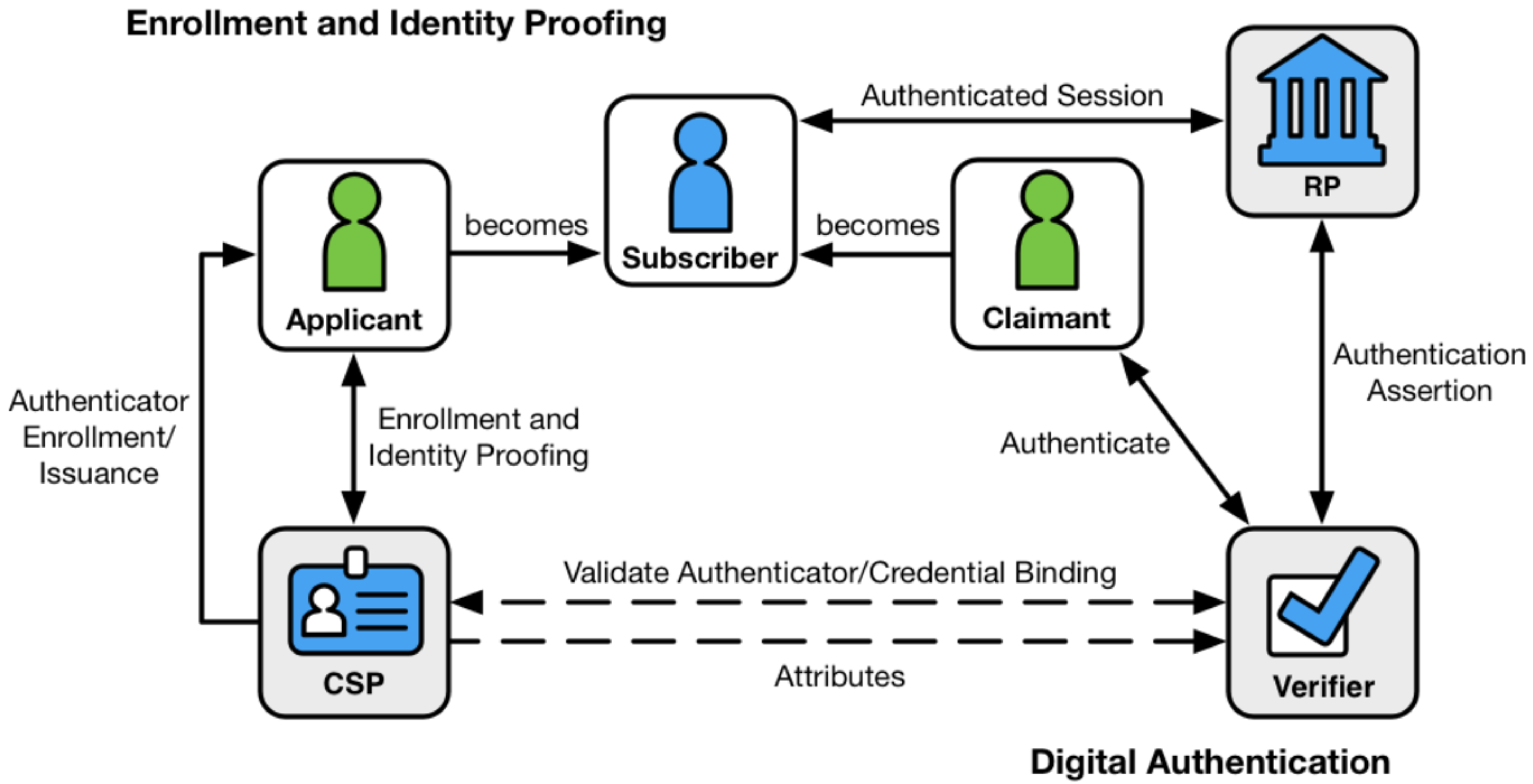
- **Subscriber/Claimant** - A party whose identity is to be verified using an authentication protocol

- **Relying Party** - An entity that relies upon the Subscriber's authenticators and credentials or a Verifier's assertion of a Claimant’s identity, to process a transaction or grant access to information or a system

- **Verifier - entity that verifies the Claimant’s identity**
  - by verifying the Claimant’s possession and control of an authenticator using an authentication protocol
  - To do this, the Verifier may also need to validate credentials that link the authenticator and identity and check their status

- **Credential Service Provider (CSP)** - A trusted entity that **issues** or **registers** Subscriber authenticators and issues electronic credentials to Subscribers
  - The CSP may encompass Registration Authorities (RAs) and Verifiers that it operates
  - A CSP may be an independent third party, or may issue credentials for its own use

Digital Identity Model



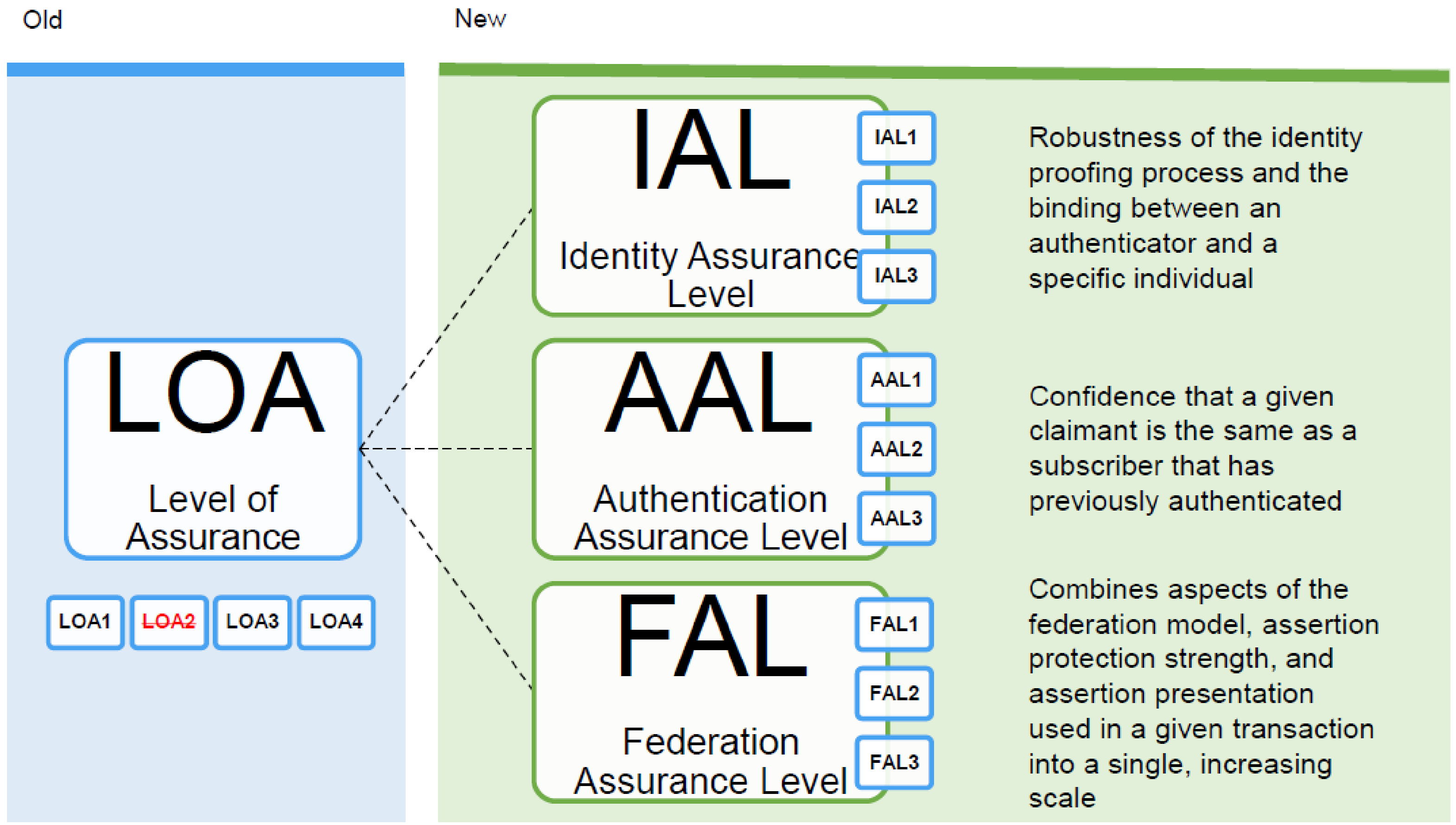
Preliminaries

- **Authenticator**: Something a user (claimant) possesses and controls, typically a cryptographic key or password that is used to authenticate the user's identity

- Three **Factors** as the cornerstone of authentication
  - Something you **know** (e.g., a password)
  - Something you **have** (e.g., an ID badge or a cryptographic key)
  - Something you **are** (e.g., a fingerprint or other biometric data)

- Multi-factor authentication
  - Uses more than one of the factors
  - Strength of authentication is largely determined by the number of factors

NIST – Assurance Levels



Identity Assurance Levels (IALs)

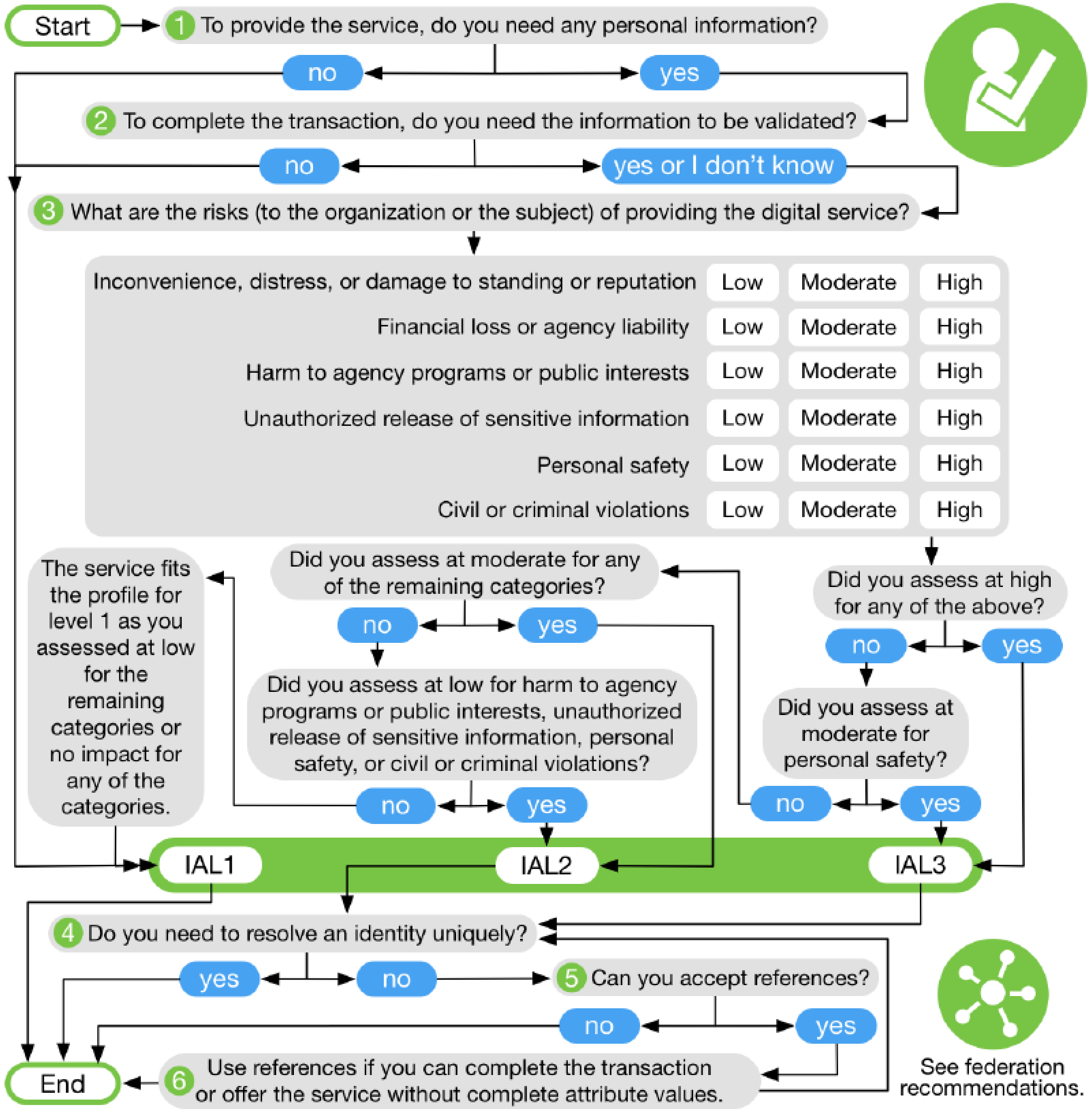
- Refers to the **robustness of the identity proofing process** and the binding between an authenticator and a specific individual

IAL	Description
1	Self-asserted attribute(s) – 0 to n attributes
2	Remotely identity proofed
3	In-person identity proofed (and a provision for attended remote)

Identity Assurance Levels (IALs)

- Refers to the robustness of the identity proofing process and the binding between an authenticator and a specific individual

Identity Assurance Level
IAL1: At IAL1, attributes, if any, are self-asserted or should be treated as self-asserted.
IAL2: At IAL2, either remote or in-person identity proofing is required. IAL2 requires identifying attributes to have been verified in person or remotely using, at a minimum, the procedures given in SP 800-63A.
IAL3: At IAL3, in-person identity proofing is required. Identifying attributes must be verified by an authorized CSP representative through examination of physical documentation as described in SP 800-63A.


Start
1 To provide the service, do you need any personal information?
no
yes
2 To complete the transaction, do you need the information to be validated?
no
yes or I don't know
3 What are the risks (to the organization or the subject) of providing the digital service?
Inconvenience, distress, or damage to standing or reputation
Financial loss or agency liability
Harm to agency programs or public interests
Unauthorized release of sensitive information
Personal safety
Civil or criminal violations
Low
Moderate
High
The service fits the profile for level 1 as you assessed at low for the remaining categories or no impact for any of the categories.
Did you assess at moderate for any of the remaining categories?
no
yes
Did you assess at low for harm to agency programs or public interests, unauthorized release of sensitive information, personal safety, or civil or criminal violations?
no
yes
Did you assess at high for any of the above?
no
yes
Did you assess at moderate for personal safety?
no
yes
IAL1
IAL2
IAL3
4 Do you need to resolve an identity uniquely?
yes
no
5 Can you accept references?
no
yes
End
6 Use references if you can complete the transaction or offer the service without complete attribute values.
See federation recommendations.

Authenticator Assurance Levels (AALs)

- Describes the robustness of **confidence** that a given **claimant** is **the same as a subscriber that has previously authenticated**

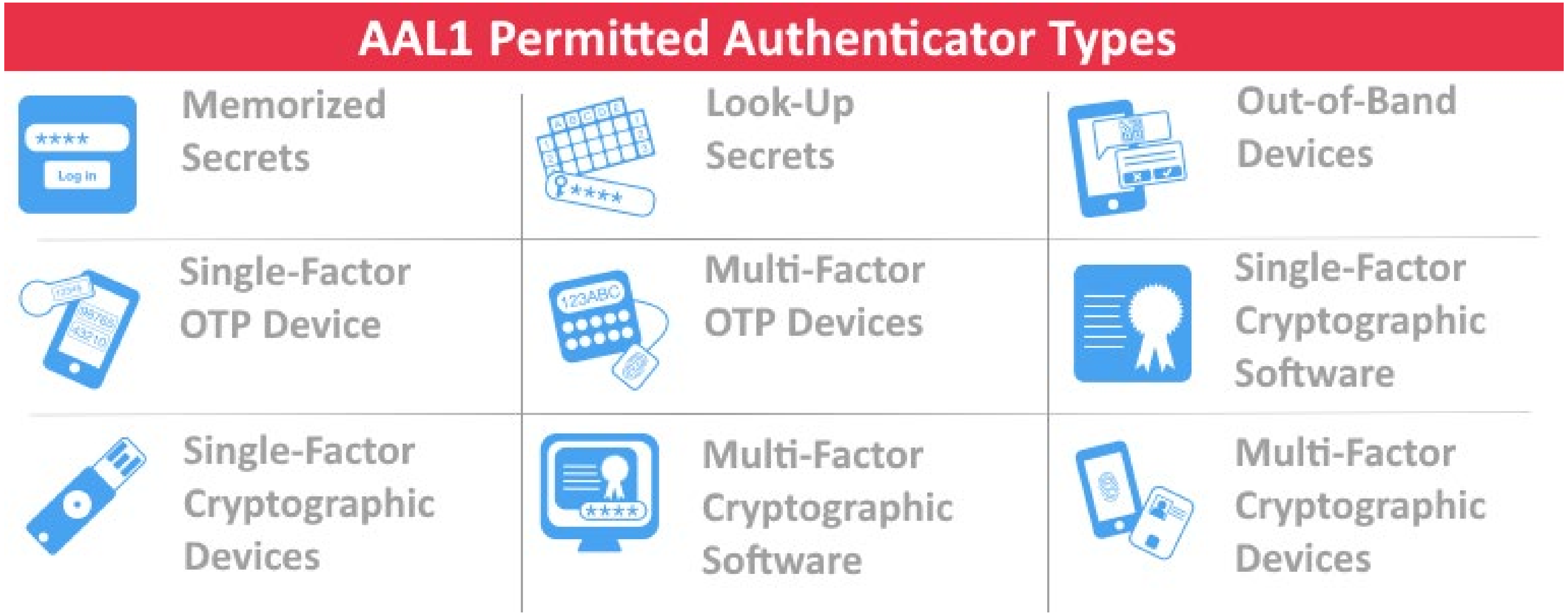
AAL	Description
1	Single-factor authentication
2	Two-factor authentication
3	Two-factor authentication with hardware authenticator

Authenticator Assurance Levels (AALs)

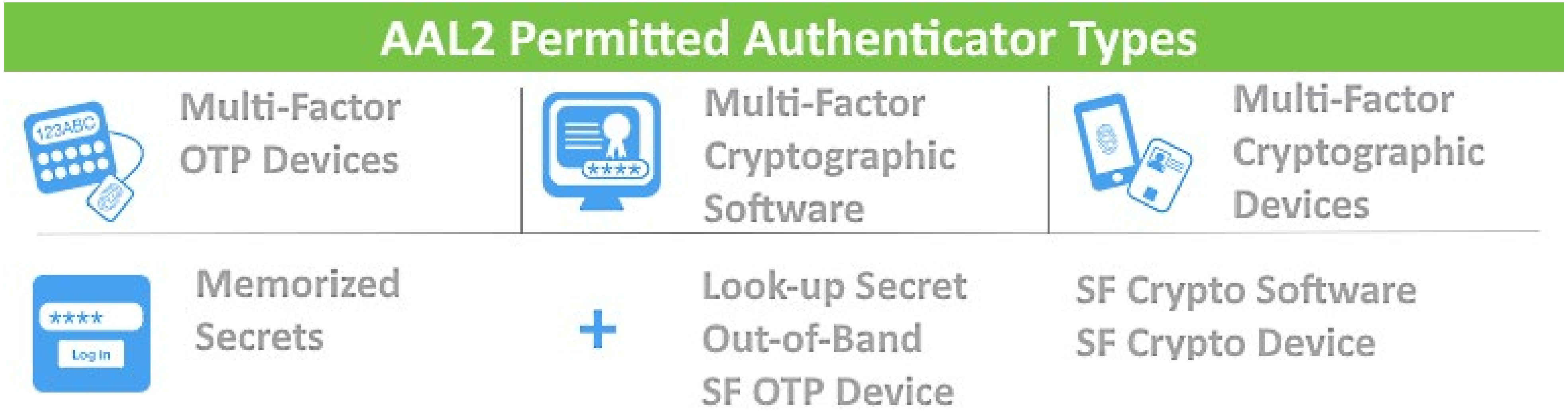
- Describes the robustness of confidence that a given claimant is the same as a subscriber that has previously authenticated

Authenticator Assurance Level
AAL1: AAL1 provides some assurance that the claimant controls an authenticator registered to the subscriber. AAL1 requires single-factor authentication using a wide range of available authentication technologies. Successful authentication requires that the claimant prove possession and control of the authenticator(s) through a secure authentication protocol.
AAL2: AAL2 provides high confidence that the claimant controls authenticator(s) registered to the subscriber. Proof of possession and control of two different authentication factors is required through a secure authentication protocol. Approved cryptographic techniques are required at AAL2 and above.
AAL3: AAL3 provides very high confidence that the claimant controls authenticator(s) registered to the subscriber. Authentication at AAL3 is based on proof of possession of a key through a cryptographic protocol. AAL3 is like AAL2 but also requires a “hard” cryptographic authenticator that provides verifier impersonation resistance.

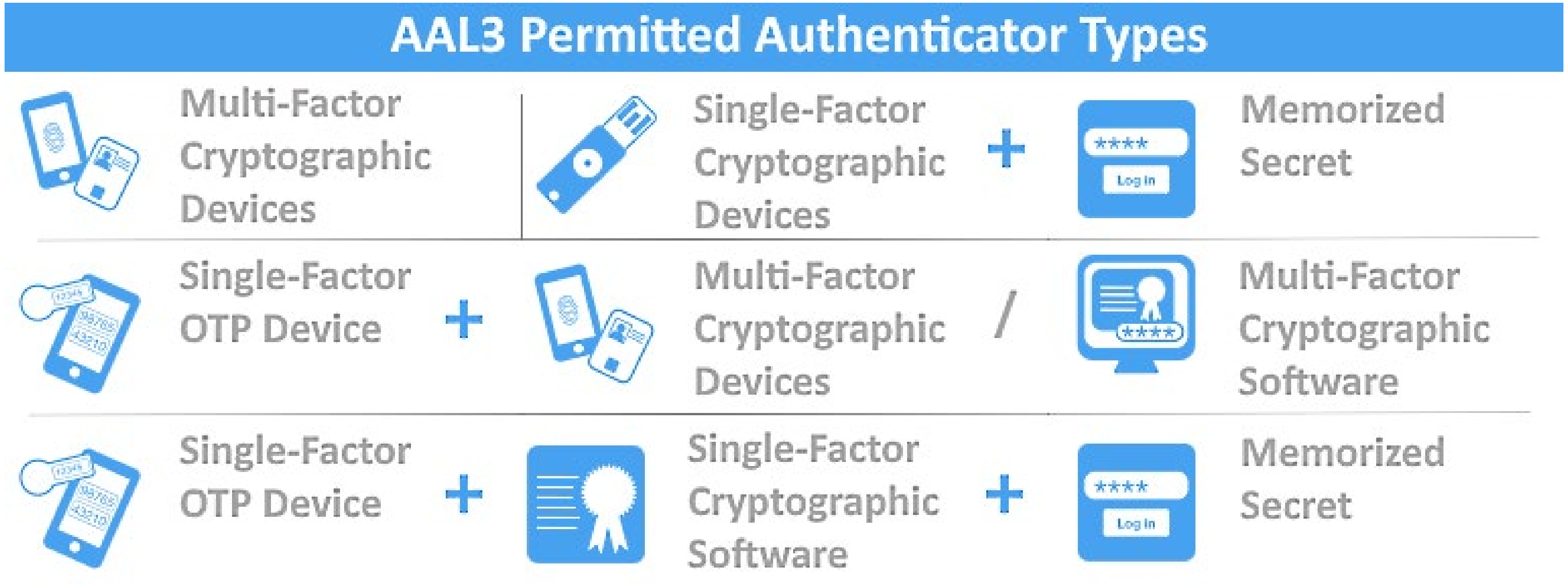
Permitted authenticator types AAL1

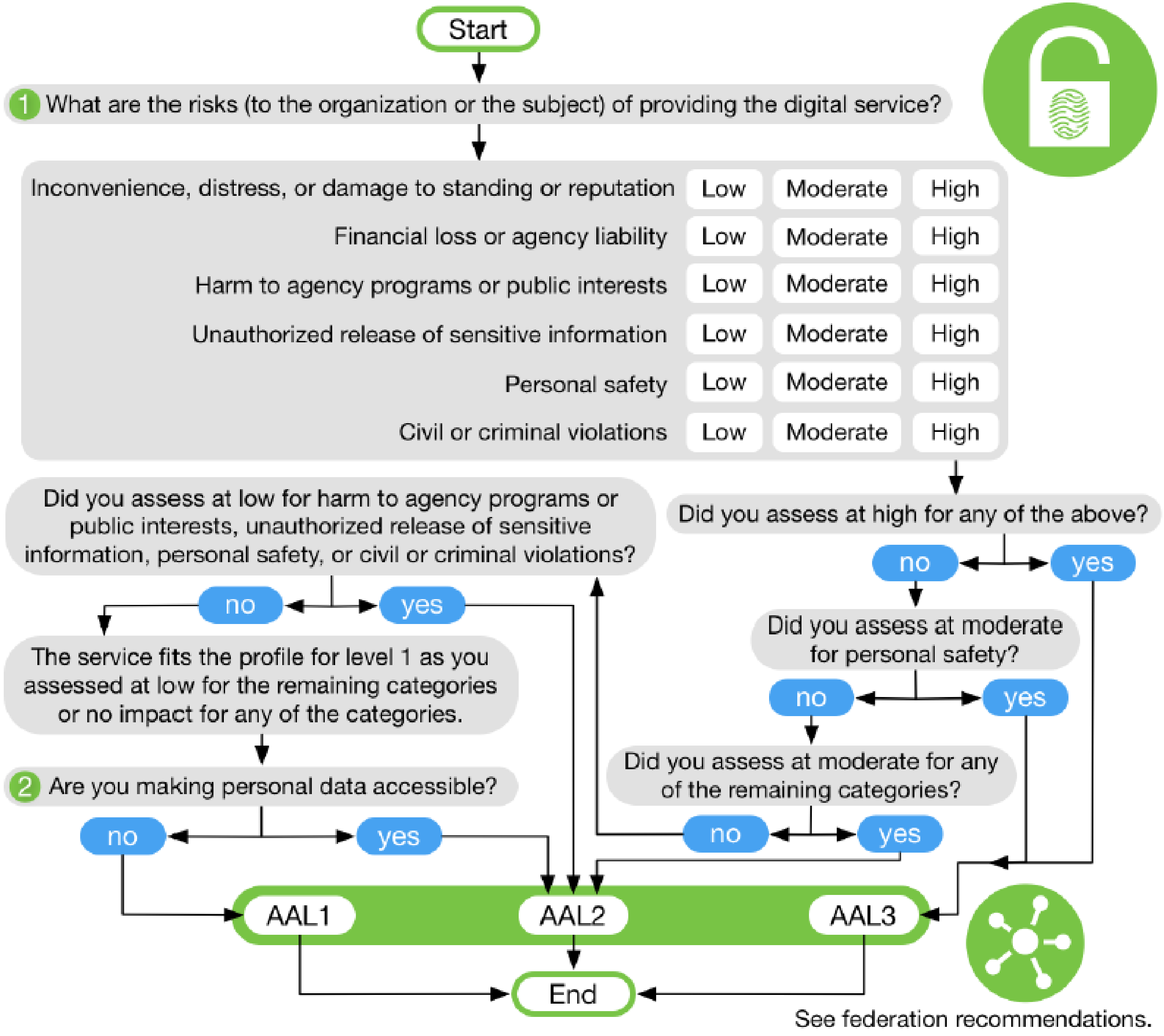


Permitted authenticator types AAL2



Permitted authenticator types AAL3




Start
1 What are the risks (to the organization or the subject) of providing the digital service?
Inconvenience, distress, or damage to standing or reputation Low Moderate High
Financial loss or agency liability Low Moderate High
Harm to agency programs or public interests Low Moderate High
Unauthorized release of sensitive information Low Moderate High
Personal safety Low Moderate High
Civil or criminal violations Low Moderate High
Did you assess at low for harm to agency programs or public interests, unauthorized release of sensitive information, personal safety, or civil or criminal violations?
no
yes
The service fits the profile for level 1 as you assessed at low for the remaining categories or no impact for any of the categories.
2 Are you making personal data accessible?
no
yes
Did you assess at high for any of the above?
no
yes
Did you assess at moderate for personal safety?
no
yes
Did you assess at moderate for any of the remaining categories?
no
yes
AAL1
AAL2
AAL3
End
See federation recommendations.

eIDAS Levels of Assurance (LoA)

- European (EU) alternative to NIST assurance levels
- Very similar to NIST
- **MitID uses** NSIS: the Danish version of **eIDAS**
- Other alternatives:
  - ISO/IEC 29115:2013
  - https://www.iso.org/standard/45138.html

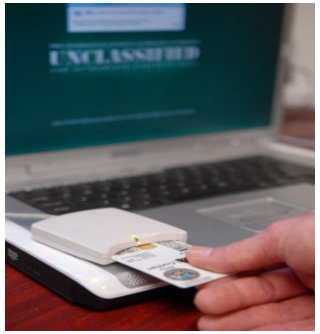
eIDAS Levels of Assurance (LoA)

- **Low**
  - **limited degree** of confidence in the claimed or asserted identity of a person
  - for instance, enrolment is performed by **self-registration in a web-page, without any identity verification**
- **Substantial:**
  - **substantial degree** of confidence in the claimed or asserted identity of a person
  - for instance, enrolment is performed by providing and verifying identity information, and authentication by using **a username and a password and a one-time password sent to your mobile phone**;
- **High:**
  - **higher degree** of confidence in the claimed or asserted identity of a person
  - for instance, enrolment is performed by registering in person in an office, and authentication by using a smartcard, like **a National ID Card**

Outline

- Introduction
- Authentication of humans
  - NIST digital authentication model
  - Authenticators
- Authentication of machines
  - Needham–Schroeder
  - Kerberos
  - Active directory
- Bootcamp: setting up your lab environments
- Lab exercise

EXAMPLES OF AUTHENTICATORS



Authentication factors

- Authenticators are characterized by the **number and types of authentication factors** that they use:
  - **Single-factor authenticator**
    - Only one of the three factors to achieve authentication
    - E.g., a password
    - No other additional factors are required to activate the authenticator
  - **Multi-factor authenticator**
    - Uses two or more factors to achieve authentication
    - E.g., a private key on a smart card that is activated via PIN is a multi-factor authenticator

Authenticator Types

- **Memorized Secret authenticator**:
  - Examples: Pin or Password
  - 8 characters in length if chosen by the subscriber or random and min. 6 characters if chosen by CSP
  - Something you know
- **Pre-registered Knowledge authenticator** :
  - **Not** accepted/supported anymore
  - Example: What is your favorite color ?
  - Something you know
- **Look-up Secret authenticator** :
  - Example: Danish nemid printed card
  - Something you have
- **Out of Band authenticator** :
  - Secondary channel of communication
  - Example: SMS to Cellphone of a secret (e.g., 6-digit code)
  - Something you have

Authenticator Types

- **Single-factor (SF) one-time password (OTP) Device** :
  - These devices use two persistent values: i) a symmetric key, ii) a nonce (acts as timer/counter)
  - Example: 6-digit PIN generator
  - Something you have

- **Multi-factor (MF) One-Time password(OTP) Device** :
  - These devices use two persistent values: i) a symmetric key, ii) a nonce (acts as timer/counter)
  - Example: 6-digit PIN generator
  - Something you have **but** activated by something you know or something you are (e.g., fingerprint)

- **Single-factor (SF) Cryptographic Device** :
  - Connect via USB or other direct way and provide authenticator output
  - Embedded symmetric/asymmetric keys
  - Something you have

Authenticator Types

- **Multi-factor (MF) Cryptographic Software**:
  - Key stored on disk or other “soft” media that requires activation through a 2nd factor authentication
  - Example: encrypted certificate
  - Something you have, but activated by something you know or something you are

- **Multi-factor (MF) Cryptographic Device**:
  - Connect via USB or other direct way and provide authenticator output
  - Activation via a 2nd factor authentication
  - Example: PIN activated USB
  - Something you have, but activated by something you know or something you are

attacks/problems on/with authenticators?

- Any ideas on attacking single factor authenticators?

- How about multi-factor?

Common attacks/problems on/with authenticators

- Password reset:
  - Password reset questions are very insecure!
- 2-factor optional
- 1 devices-does-it-all:
  - Saved password on mobile, sms to phone, authenticator on phone

Common attacks/problems on/with authenticators

- Multi-factor threats:
  - Social engineering: tricking user to give away the 2nd factor OTP
    - “to confirm your identity, we have sent you an SMS”
    - change phone number
  - Technical:
    - intercept SMS
    - Brute-force 2FA
    - Implementation bugs

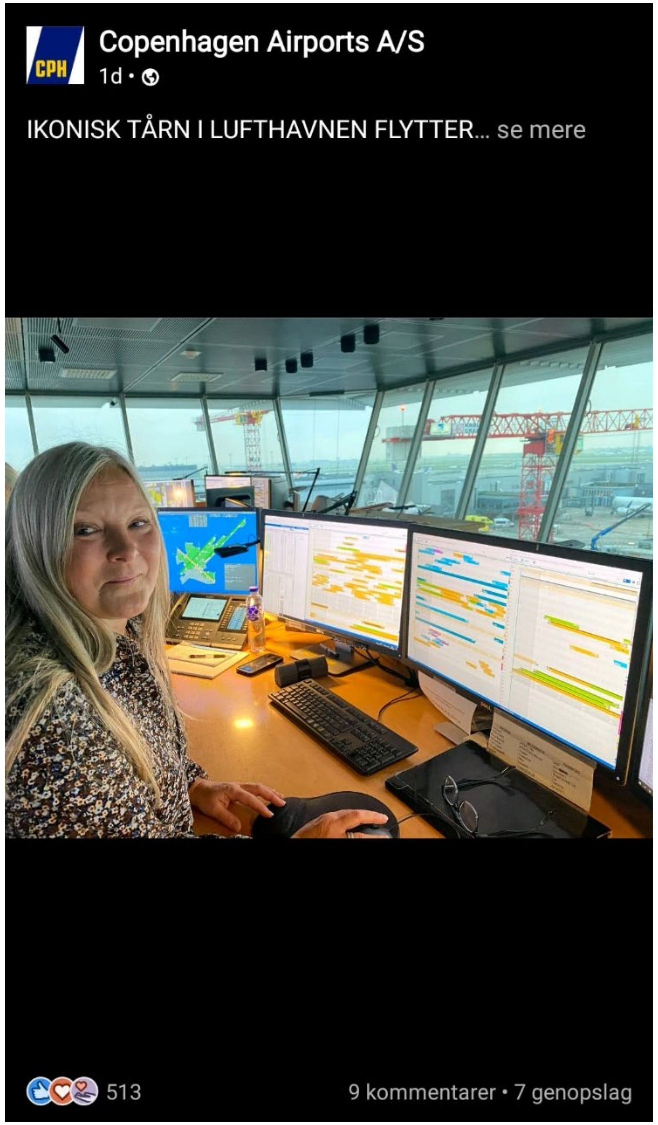
how scammers are attacking 2nd factor (5:20 min)

Common attacks against passwords

- **Password guessing**
- **Brute-force or dictionary attack**
- **Finding the hash of a password**
  - Bruteforcing, rainbow table attacks
- **Key loggers**

Common attacks against passwords

- **Password guessing**
- **Brute-force or dictionary attack**
- **Finding the hash of a password**
  – Bruteforcing, rainbow table attacks
- **Key loggers**
- **Social engineering**
  – SMS, email
- **Human error**



Common attacks against passwords

- Passwords introduce too many challenges but are also needed:
  - Future is slowly becoming passwordless
  - Passwords only for user authentication
    - Once authenticated crypto keys (e.g., certificates, cookies, JWTs, etc.)

- For user authentication passwords can be enhanced via:
  - Password managers
  - Single sign on
  - Multi factor
  - Bio metrics
  - Rate limit login attempts

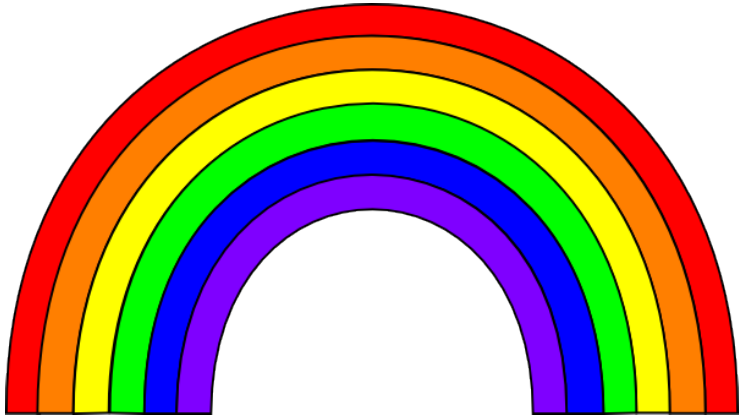
Attacks against stored passwords

- Passwords may be stored as:
  - **Plaintext**
  - Some hash value (derived from a **generic hash function + the password**)
    - E.g., SHA3(password) or MD5(password)
  - Some hash value (derived from a **generic hash function + the password + some random value**)
    - E.g., SHA3(password+random_value)
  - Some hash value (derived from **a password-hashing function + the password + some random value**)

- Saving passwords as the result of a hash function:
  - Fast
  - Problematic and hence insecure
    - Because they are **very fast**
    - Because **rainbow tables** can be used against them

Rainbow Tables

- A rainbow table is a lookup table offering a time-memory tradeoff used in recovering the plaintext password from a password hash generated by a hash function
  - Approach invented by Martin Hellman
- The concept behind rainbow tables is simple
  - Make one-way hash functions two way by making a list of outputs for all possible inputs up to a character limit



Rainbow Tables

- Rainbow Tables are **Large**
  - A rainbow table set for windows NTHASH (exactly 8 characters including only 0-10, a-z, A-Z, and the symbols !*) is 134.6GB
  - 9+ character rainbow tables can take up terabytes of space
  - Generating rainbow tables requires more time than a brute force attack
  - Requires access to the password hash
  - Salting passwords makes the approach unfeasible
- Rainbow Tables are **built once, but used many times**
- Rainbow Table lookups are **fast**
  - Password lookups become a table search problem
  - The brute force work is pre-computed

[Download] WPA-PSK Rainbow Tables

BY DO SON · MAY 6, 2017

Currently, the use of WPA as the encryption method for Access Points has greatly enhanced the security of wireless networks making it hard work to get into a victim network by an attacker.

However, this type of encryption has weaknesses that can be used to get the password. **WPA-PSK** may be compromised if subjected to a brute – force attack which by using dictionary words or passwords (which can become extremely large) ended up finding the key.

The problem with this process is time, every time an extracted keyword in the dictionary is read is necessary to create a hash of this and likewise compared with the hash of the original obtained key to the AP, in addition to this the hash belonging to the original password directly depends on network configuration, specifically AP name (SSID) and the length of the name, so it's not the same password hashing "password" for a network with SSID "D- link "to one with SSID" Linksys ".

For this reason, the WPA tables are helpful because they reduce considerably the time needed to test a certain number of passwords on a specific AP, then this list downloads some tables with their ESSID.

Download Links for WPA tables:

ESSID	Link
101	http://www.mediafire.com/?zadv0ppvzkdoiz9
3Com	http://www.mediafire.com/?adco3kuiiqiprkb
Airport	http://www.mediafire.com/?xdcrmiz96j87uip

Attacks against stored passwords

- Passwords may be stored as:
  - **Plaintext**
  - Some hash value (derived from a **generic hash function + the password**)
    - E.g., SHA3(password) or MD5(password)
  - Some hash value (derived from a **generic hash function + the password + some random value**)
    - E.g., SHA3(password+random_value)
  - Some hash value (derived from **a password-hashing function + the password + some random value**)

- Saving passwords as the result of a hash function:
  - Fast
  - Problematic and hence insecure
    - Because they are **very fast**
    - Because **rainbow tables** can be used against them

Argon2: Winner of the Password hashing competition

- Three variants:
  - Argon2i: data-**i**ndependent memory access (slower), resistant to side-channel attacks
  - Argon2d: data-**d**epending memory, resistant against GPU cracking
  - Argon2id: hybrid

1. H := **Hash**(password, salt, all parameters)
2. Fill a 2-dimension array B of ***MemParameter*** 1024-byte **blocks**
   - Fill column by column, with sequential dependency
   - Blocks B[i][0] and B[i][1] depend on H
   - Other blocks B[i][j] depend on B[i][j–1] and on **another block**
   - "depend on X" = "are a BLAKE2-based **hash** of stuff including X"
3. Repeat 2 ***TimeParameter*** times, **xoring** new blocks to old one
4. Return as a **tag** an xor of the last column's blocks

Argon2 example (https://antelle.net/argon2-browser/)

Argon2 in browser

Argon2 is a password-hashing function that summarizes the state of the art in the design of memory-hard functions and can be used to hash passwords for credential storage, key derivation, or other applications.

Here Argon2 library is compiled for browser runtime. Statistics, js library, source and docs on GitHub.

Password	Manolis
Salt	T@^f salting is good
Memory	4096 KiB
Iterations	50
Hash length	32
Parallelism	1
Type	(•) Argon2d ( ) Argon2i ( ) Argon2di

Run WebAssembly | Run WebAssembly in WebWorker | WebAssembly+SIMD | GitHub Repo

Result

```
[00.000] Testing Argon2 using Binaryen native-wasm
[00.000] Calculating hash...
[00.011] Params: pass=Manolis, salt=T@^f salting is good, time=50, mem=4096, hashLen=32, parallelism=1, type=0
[00.390] Encoded: $argon2d$v=19$m=4096,t=50,p=1$VEBeZiBzYWx0aW5nIGlzIGdvb2Q$BCiimyqLvS8j8GhwFNwGJVBtb12JAxWympEbyLk6ZEQ
[00.390] Hash: 0428a29b2a8bbd2f23f0687014dc0625506d6f5d890315b29a911bc8b93a6444
[00.390] Elapsed: 379ms
```

Attacks against stored passwords

- Passwords may be stored as:
  - **Plaintext**
  - Some hash value (derived from a **generic hash function + the password**)
    - E.g., SHA3(password) or MD5(password)
  - Some hash value (derived from a **generic hash function + the password + some random value**)
    - E.g., SHA3(password+random_value)
  - Some hash value (derived from **a password-hashing function + the password + some random value**)

- Saving passwords as the result of a hash function:
  - Fast
  - Problematic and hence insecure
    - Because they are **very fast**
    - Because **rainbow tables** can be used against them

Salting

- Salting defends against rainbow table attacks (and others)
  – Addition of random data in the input of a hash function
  – Even same passwords will have different hashes

Username	Password
user1	password123
user2	password123

Username	Salt value	String to be hashed	Hashed value = SHA256 (Password + Salt value)
user1	E1F53135E559C253	**password123**E1F53135E559C253	72AE25495A7981C40622D49F9A52E4F1565C90F048F59027BD9C8C8900D5C3D8
user2	84B03D034B409D4E	**password123**84B03D034B409D4E	B4B6603ABC670967E99C7E7F1389E40CD16E78AD38EB1468EC2AA1E62B8BED3A

Salting

- A salt is a **unique**, **randomly generated string** that is added to each password as part of the hashing. As the salt is unique for every user, an attacker has to crack hashes **one at a time** using the respective salt rather than calculating a hash once and comparing it against every stored hash. This makes cracking large numbers of hashes significantly harder, as the time required grows in direct proportion to the number of hashes

- Salting also protects against an attacker **pre-computing hashes** using rainbow tables or database-based lookups. Finally, salting means that it is impossible to determine whether two users have the same password without cracking the hashes, as the different salts will result in different hashes even if the passwords are the same

- Modern hashing algorithms such as **Argon2id**, **bcrypt**, and **PBKDF2** automatically salt the passwords, so no additional steps are required when using them

Outline

- **Introduction**
- **Authentication of humans**
  - NIST digital authentication model
  - Authenticators
- **Authentication of machines**
  - Needham–Schroeder
  - Kerberos
  - Active directory
- **Bootcamp: setting up your lab environments**
- **Lab exercise**

Key exchange & authentication protocols

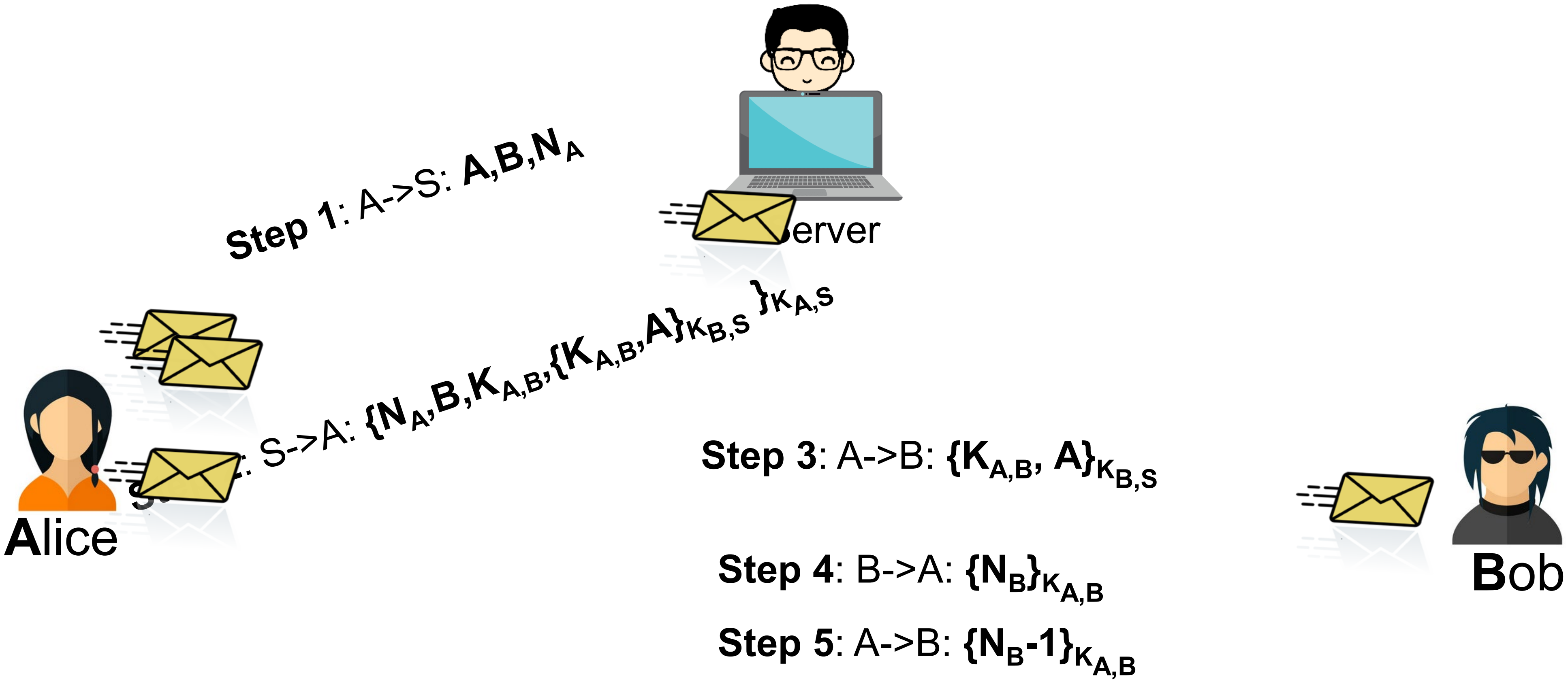
- In the following we will talk about two **key transport and authentication** protocols:
  - **Needham–Schroeder**
  - **Kerberos**
- Mostly connected to symmetric encryption
- The basis for Microsoft Active Directory

Needham–Schroeder protocol

- Assumes a central trusted server (S)
- Users A and B
  - have already established a secure channel with S

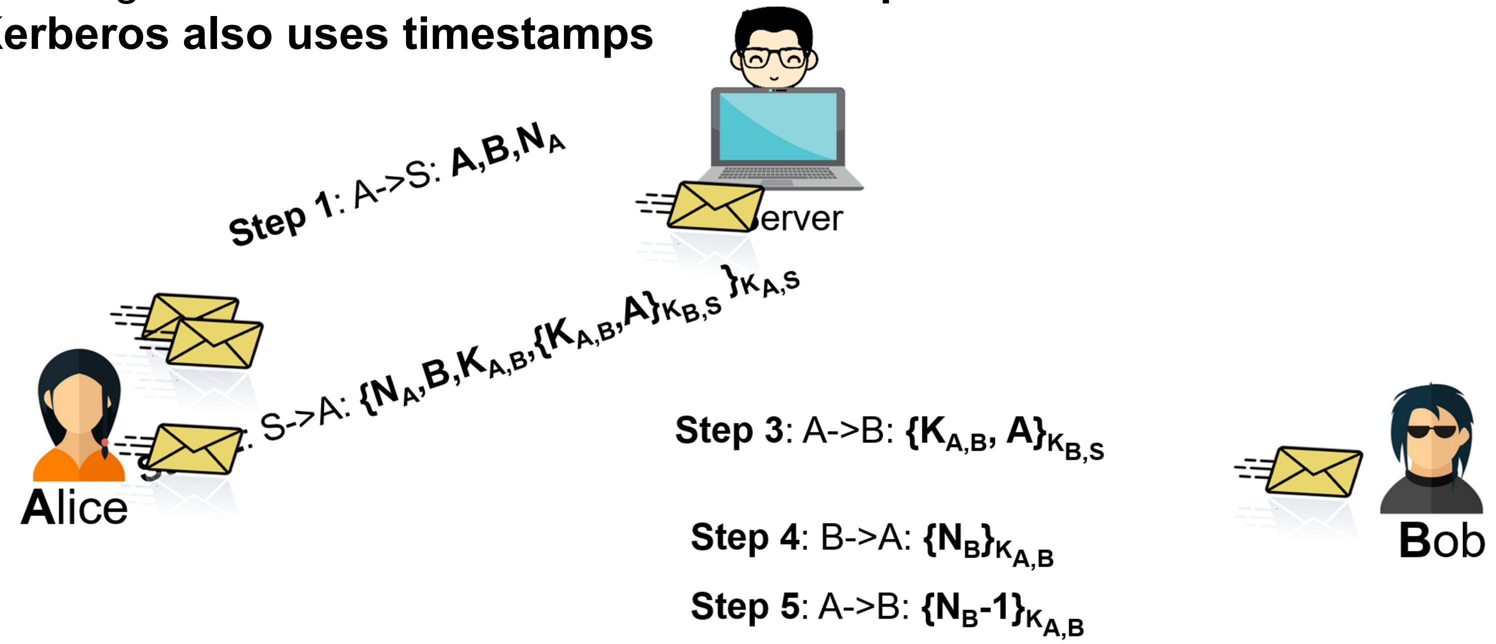
- Needham–Schroeder is important as:
  - Introduces the ticket concept
  - Basis for Kerberos

Needham–Schroeder protocol



Needham–Schroeder protocol

- Notes:
  - After step 3, A and B have **established** a shared **key** $\{K_{A,B}\}$
  - Steps 4 and 5 are part of the mutual **authentication**
  - In step 5 we need $\{N_B - 1\}$ since the encrypted N_B has been already transmitted
  - The main weakness of this protocol is related to the **freshness** of the keys
    - Denning-Saco variation introduced **timestamps**
    - **Kerberos also uses timestamps**



Outline

- **Introduction**
- **Authentication of humans**
  - NIST digital authentication model
  - Authenticators
- **Authentication of machines**
  - Needham–Schroeder
  - Kerberos
  - Active directory
- **Bootcamp: setting up your lab environments**
- **Lab exercise**

The Kerberos protocol

- Network **authentication** protocol
- Based on Needham-Schroeder **symmetric key** protocol (and Denning-Saco)
- MIT in late 1980s (v.5 in 1993)
  – Latest version krb5-1.20 **(Nov 2022)**
- Some variation used in most modern OSs
- **Centralized** architecture
  – Trusted third party key distribution system

Kerberos key components

- Key Distribution Center (KDC)
  - – Ticket granting service (**TGS**)
    - Ticket granting ticket (TGT)
  - – Authentication Service (**AS**)

- Master key shared by KDC with each user (principal)
- When Alice logs into her machine, her station asks the KDC for a session key for Alice. The KDC also gives her a TGT
- Alice’s workstation retains only the session key and the TGT
- Alice’s workstation uses the TGT to receive other tickets from the TGS

Key Distribution Centre (KDC)

- Runs on a physically secure node in the network
- Database of keys for all users
- Creates and hands out keys for each transaction (session) between clients
- Single Point of Failure

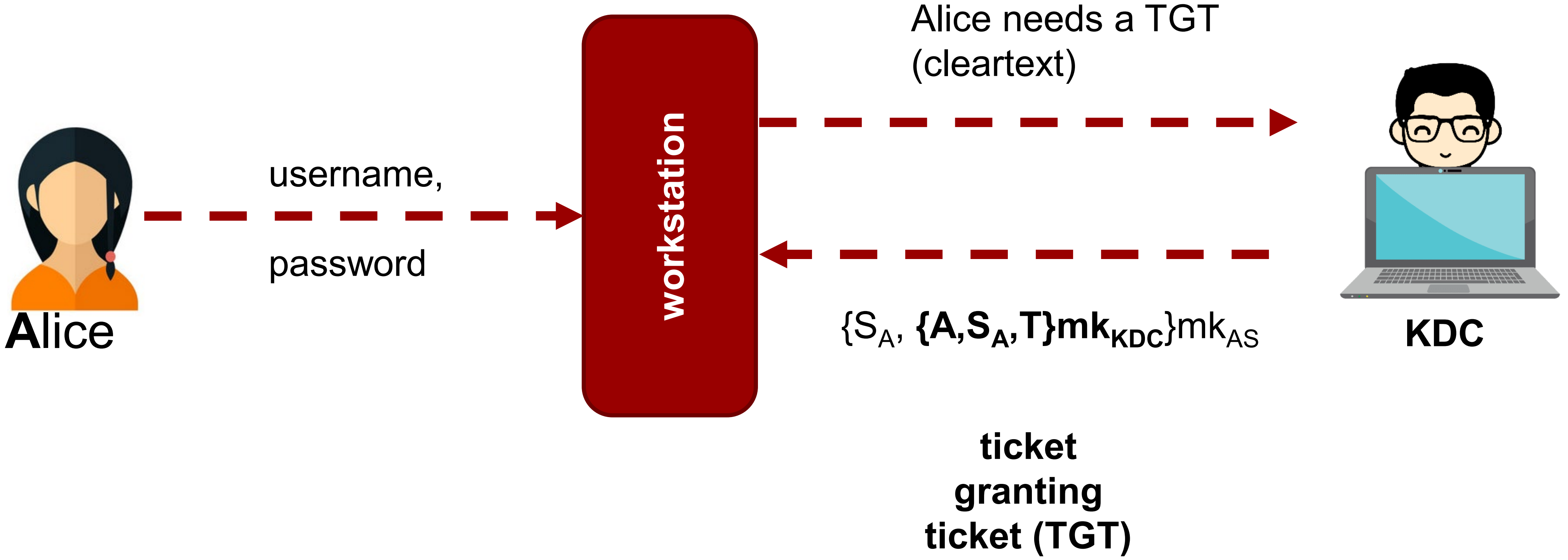
Tickets and operations

- Alice requests KDC to talk to Bob
- KDC creates a session key k_{AB} for Alice and Bob to use for the session
- KDC encrypts k_{AB} with Alice's master key mk_{AS}
- KDC also encrypts k_{AB} and some identifying info (A) about Alice with Bob's master key mk_{BS}
  - $\mathbf{\{A, t, k_{AB}\}\ mk_{BS}}$
  - t is a timestamp (for avoiding replay attacks)
  - this is called a ticket
- Only Alice and Bob know $\mathbf{k_{AB}}$
- k_{AB} and the ticket are Alice's credentials to Bob

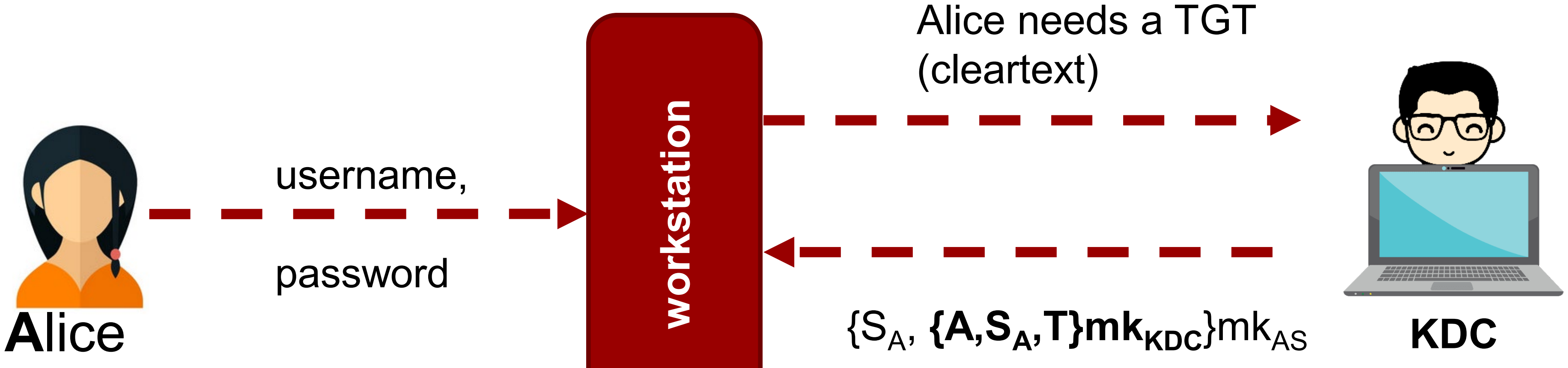
Kerberos configuration

- KDC has a database for users (principals) and their master keys
  - E.g., for Alice there is a master key mk_{AS}
- All data encrypted with the (super secret) KDC master key mk_{KDC}
- Btw: secret keys are derived from users' passwords (via some cryptographic technique; e.g., the hash of password)
- Kerberos traditionally used DES
  - now moved away -> AES

Kerberos: initiating a session

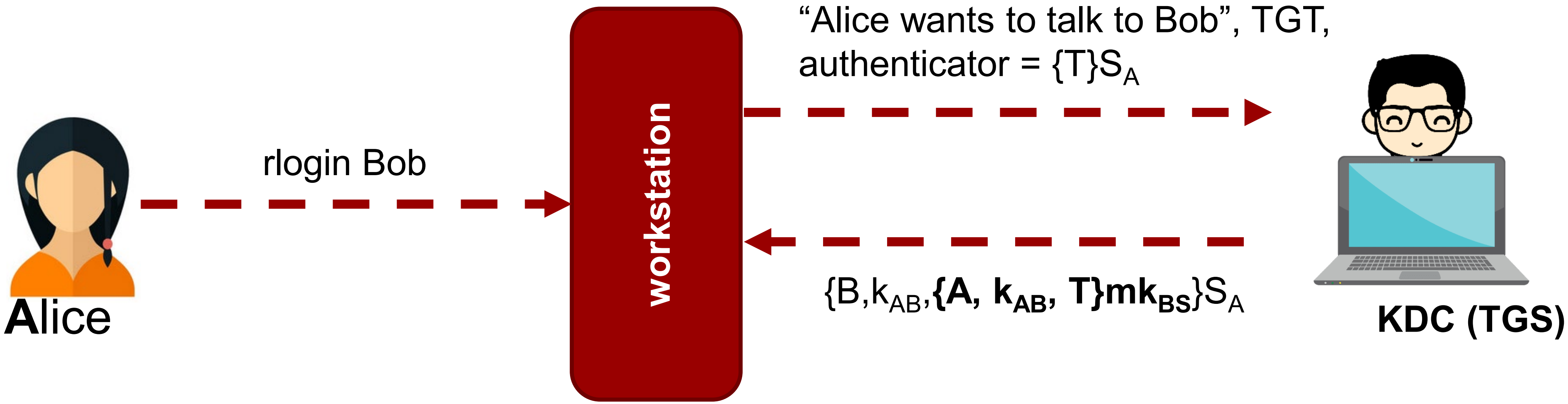


Kerberos: initiating a session

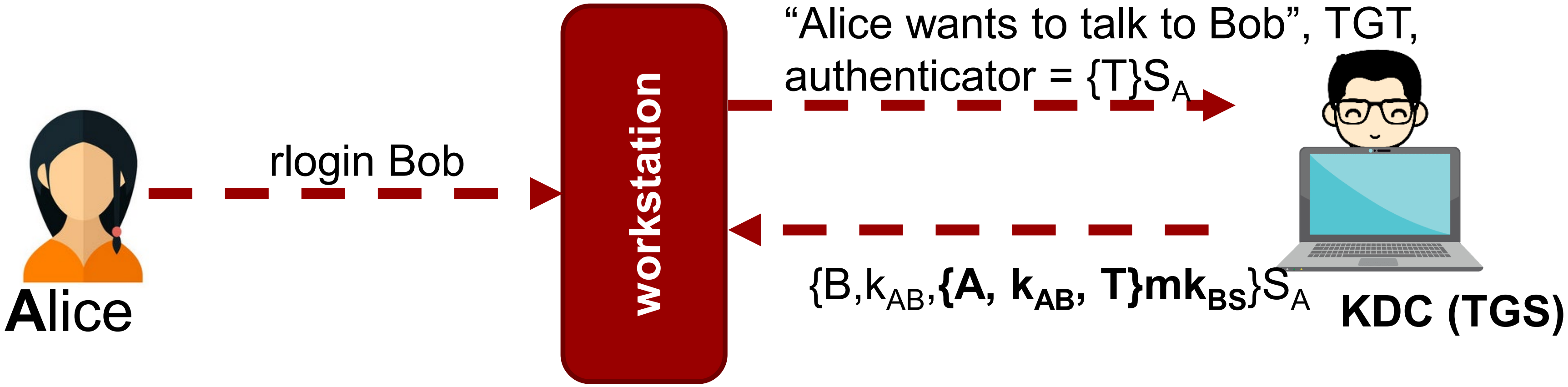


- TGT: $\mathbf{\{A,S_A,T\}mk_{KDC}}$
- T is a timestamp
- $\mathbf{mk_{KDC}}$ the KDC (super secret) master key. By using this KDC doesn't have to remember S_A
- mk_{AS} can be decrypted via alice's passwd
- S_A is the Session key for Alice to use for (upcoming) secure communications with KDC
- TGT will be used for authenticating Alice to the KDC during the session
- S_A also means that the workstation can forget Alice's password
- Use of the TGT informs the KDC to use S_A instead of mk_{AS}

Kerberos: setting up a session with Bob

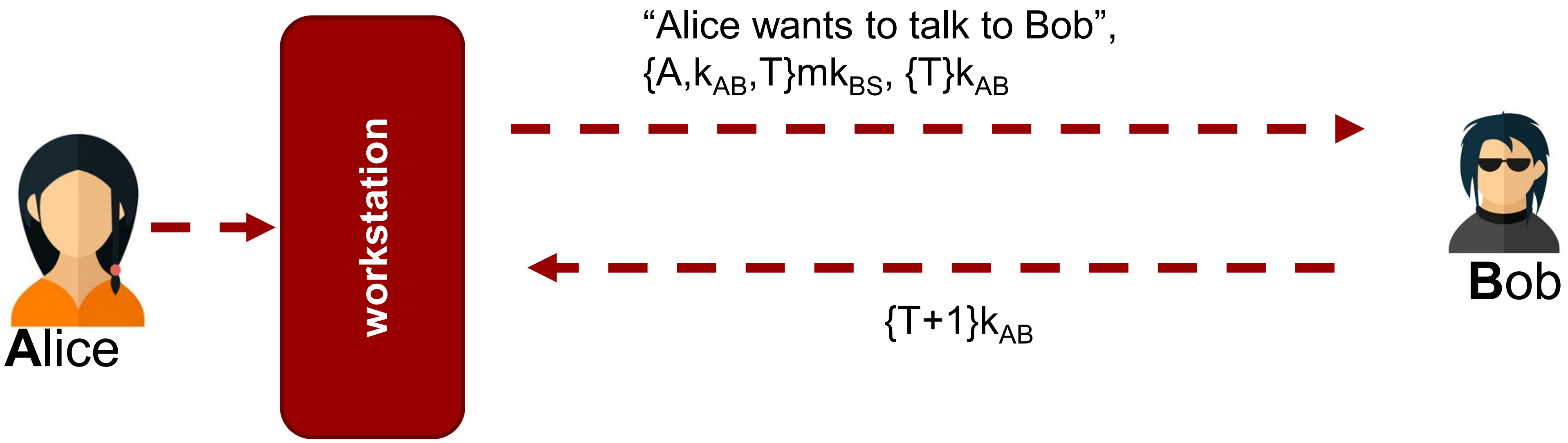


Kerberos: setting up a session with Bob

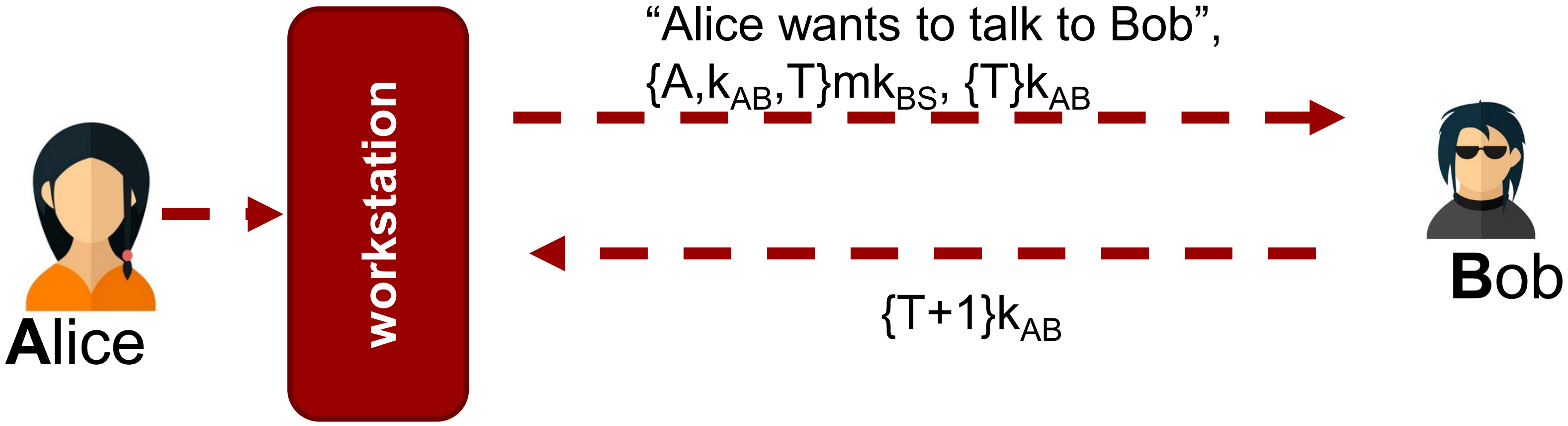


- Ticket for Alice to use with Bob **{A, k_{AB}, T}mk_{BS}**
- T is timestamp (notation same but different every time)
- Bob indirectly authenticated (because **mk_{BS}**)

Kerberos: Alice talks to Bob (yaaaay)



Kerberos: Alice talks to Bob (yaaaay)

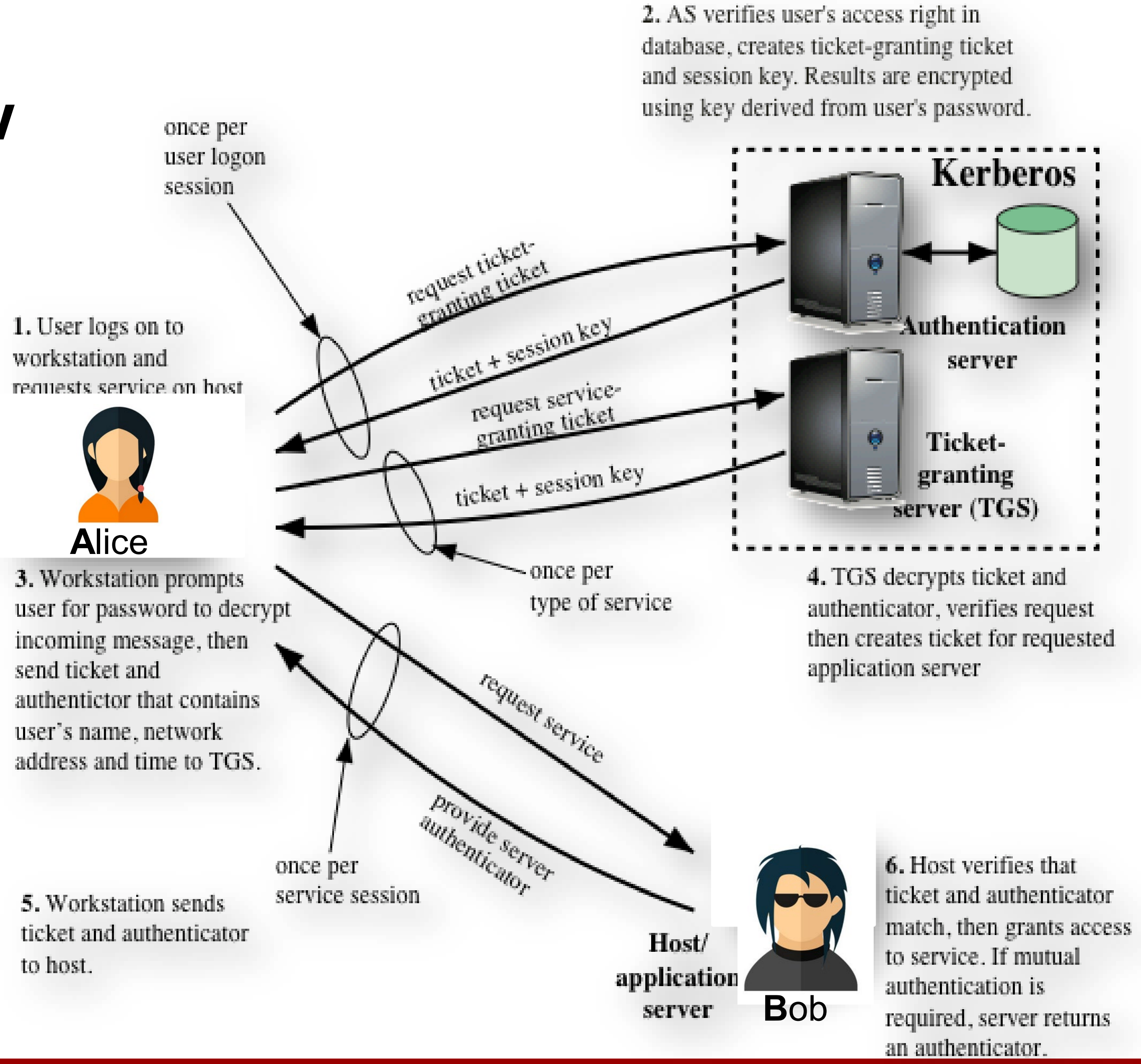


- Ticket: $\{A,k_{AB},T\}mk_{BS}$
- Replay attacks:
  - Timestamp (~5 min)
  - Stores requests in the allowable time and checks for repeats
- Alice and bob can now interact
  - Unencrypted or encrypted based on their requirements

Kerberos: sum up

- Authentication protocol using symmetric key cryptography
- Key distribution centre
  - Single point of failure
  - Shares master keys with participants
- KDC performs key generation and distribution
- Tickets allow authentication guarantees
- Timestamps against replay attacks

Overview



2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
once per user logon session
Kerberos
Authentication server
Ticket-granting server (TGS)
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
1. User logs on to workstation and requests service on host
Alice
once per type of service
4. TGS decrypts ticket and authenticator, verifies request then creates ticket for requested application server
3. Workstation prompts user for password to decrypt incoming message, then send ticket and authentictor that contains user's name, network address and time to TGS.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to host.
Host/ application server
Bob
6. Host verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Outline

- **Introduction**
- **Authentication of humans**
  - NIST digital authentication model
  - Authenticators
- **Authentication of machines**
  - Needham–Schroeder
  - Kerberos
  - Active directory
- **Bootcamp: setting up your lab environments**
- **Lab exercise**

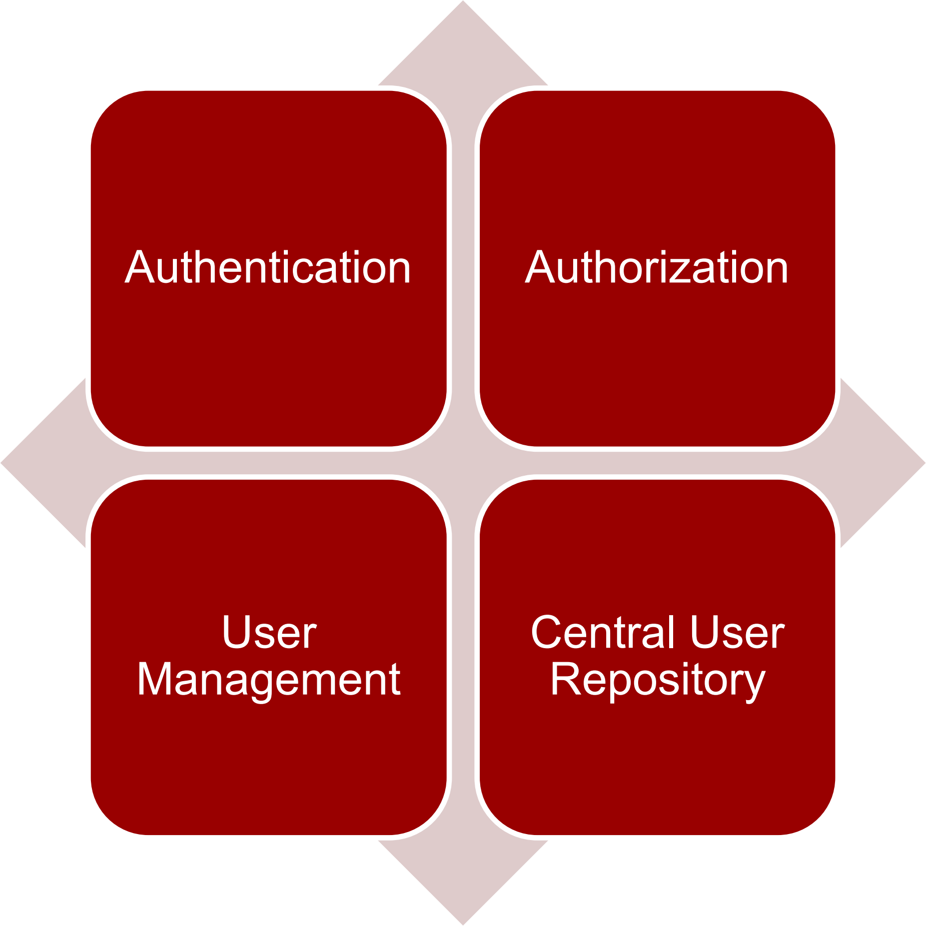
Directory Services?

- A directory service is a **collection of software and processes that stores information** about an enterprise, subscribers, or both

- An example of a directory service is the Domain Name System (DNS), which is provided by DNS server

- The DNS server stores only two types of information: names and IP addresses
- An LDAP directory service can store information on many other kinds of real-world and conceptual objects. E.g., usernames, emails, group memberships, network drives, PCs, printers.

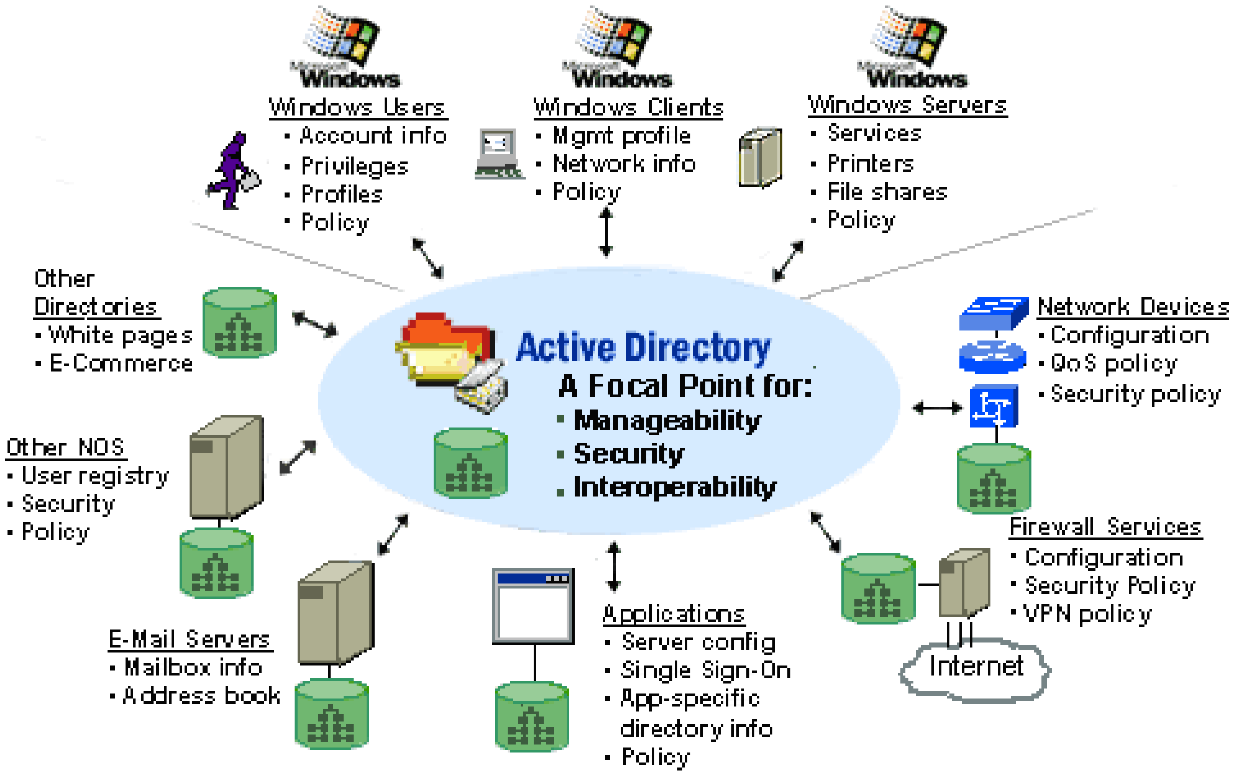
Microsoft active directory history



Directory (Store) Services	Access Management
Active Directory & ADAM	Active Directory Federation Services
Extended Directory Services	Authorization Manager
PKI / CA	Enterprise Single Sign On
Services for Unix / Services for Netware	ISA Server

Microsoft active directory history

- An **Enterprise IAM solution** by Microsoft
- Introduced in 1999
- A self-hosted, on-premise service on a Microsoft Server OS (2003, 2008, 2012, 2012R2, 2016, 2019)
- Installed as a Role / Feature on the Microsoft Server environment



Ref: Microsoft

Microsoft active directory

- Active Directory **stores information about objects on the network**; makes this information easy for administrators and users to manage

- This data store, also known as the directory, contains information about Active Directory objects

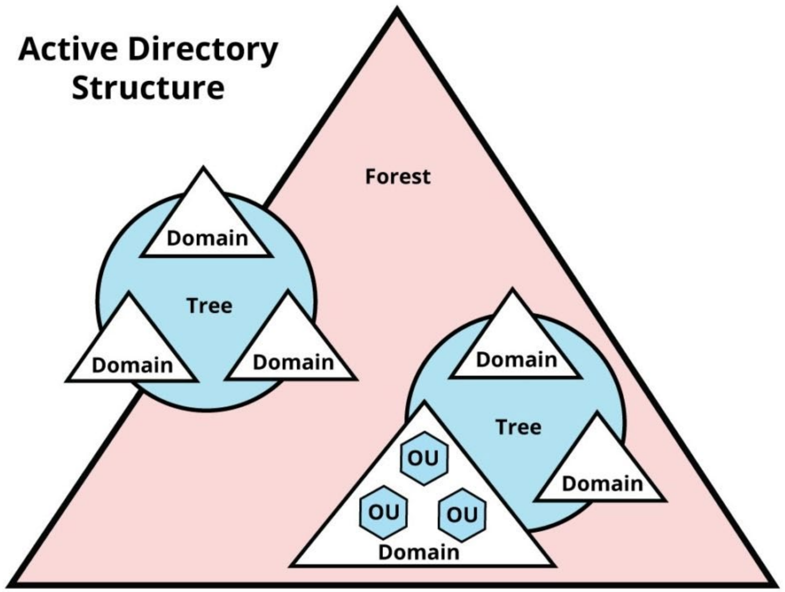
- These objects typically include **shared resources** such as servers, volumes, printers, and the network users and computer accounts

Some terminology

- **Domain Controller:** A server with Active Directory service

- **Domain**: A **domain** is **defined** as a logical group of network objects (computers, users, devices) that share the same **Active Directory** database, e.g., dtu.dk, danskebank.dk

- **Forest**: An **Active Directory forest** (**AD forest**) is the top most logical container in an **Active Directory** configuration that contains domains, users, computers, and group policies.
  – a forest is a collection of domains



OU – Organizational Unit

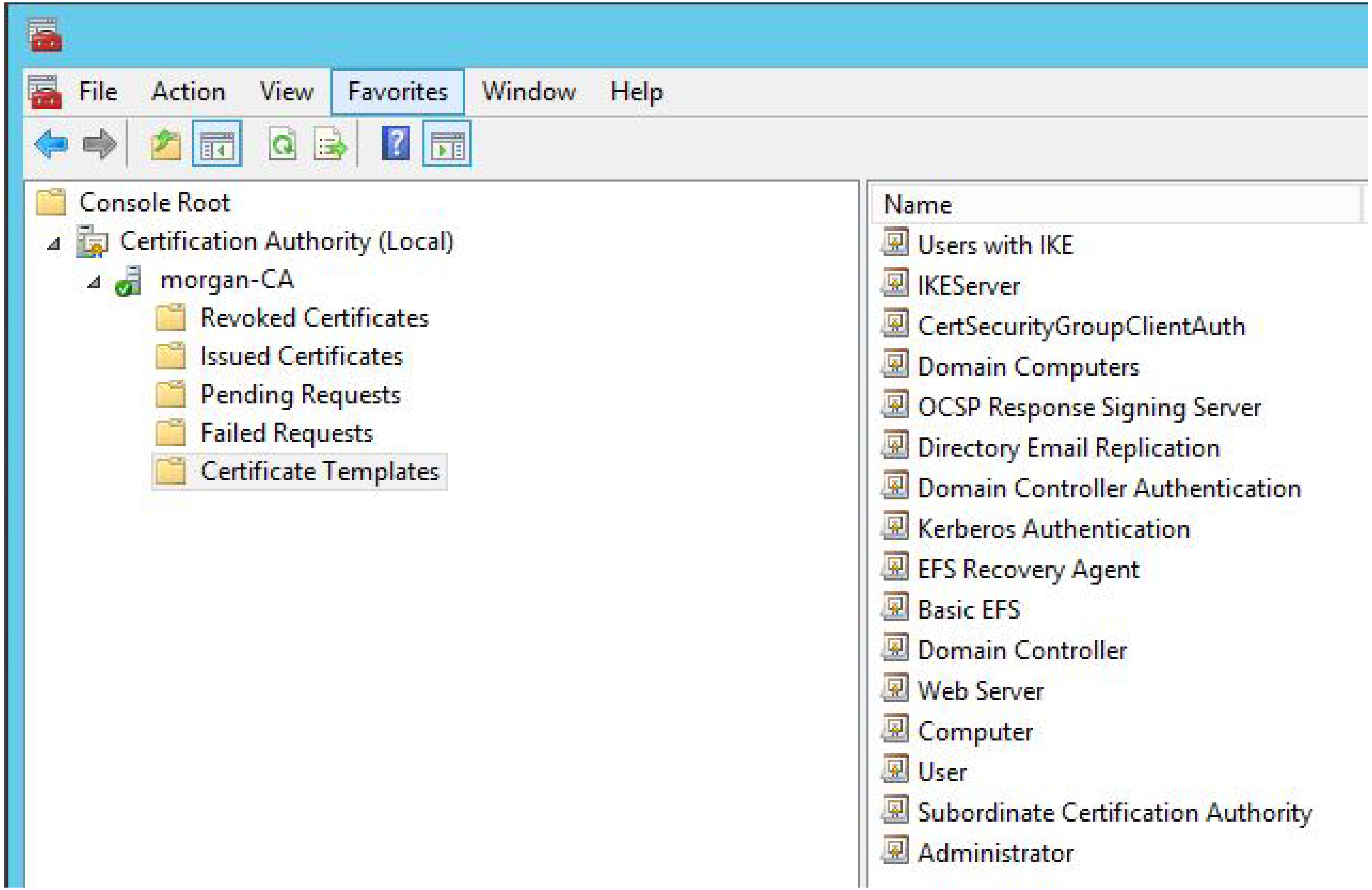
Domain Controller – Services (1)

- **Domain Services:**
  - allows **admins to manage and store information** about resources from a network, as well as application data, in a distributed database
  - helps **admins manage network's elements** (computers and end users) and reorder them into a custom hierarchy

- **Lightweight Directory Service:**
  - provides flexible support for directory-enabled applications, without the dependencies and domain-related restrictions of Active Directory Domain Services (AD DS)

- **LDAP**
  - stands for Lightweight Directory Access Protocol. It is a lightweight client-server protocol for accessing directory services, specifically X.500-based directory services
  - Usually runs over TCP/IP

- **Certificate Services**
  - allows to **build a** public key infrastructure (**PKI**) and provide public key cryptography, digital certificates, and digital signature capabilities for your organization


File
Action
View
Favorites
Window
Help
Console Root
Certification Authority (Local)
morgan-CA
Revoked Certificates
Issued Certificates
Pending Requests
Failed Requests
Certificate Templates
Name
Users with IKE
IKEServer
CertSecurityGroupClientAuth
Domain Computers
OCSP Response Signing Server
Directory Email Replication
Domain Controller Authentication
Kerberos Authentication
EFS Recovery Agent
Basic EFS
Domain Controller
Web Server
Computer
User
Subordinate Certification Authority
Administrator

Domain Controller – Services (2)

- **Active Directory Federation Service**
  - authenticates user access to multiple applications, even on different networks, using single sign-on (SSO)
  - An SSO only requires the user to sign in once, rather than use multiple dedicated authentication keys for each service

- **Rights Management Service**
  - provides with management and development tools that work with security technologies, including encryption, certificates, and authentication, to create reliable information protection solutions

So, what can Active Directory be used for?

- Authentication
- Authorization
- Access Management
- User Management
- Certificate Management
- Shared resource Management
- Policy Management

Active Directory Authentication Service

- AD uses Kerberos 5 as the authentication protocol for client-server auth
- Kerberos has 3 components: client, server and a trusted key issue authority (KDC)
- In AD, KDC (Key Distribution Center) is integrated
- KDC performs 2 main functions:
  - Authentication Service (AS)
  - Ticket Granting Service (TGS)

Active Directory Group Policy (1)

Group Policy

- Feature of the Microsoft Windows NT OS family that **controls the working environment of user accounts and computer accounts**

- Provides **centralized management and configuration** of OS, applications, and users' settings in an AD environment

- A set of Group Policy configurations is called a **Group Policy Object** (**GPO**)

- A version of Group Policy called Local Group Policy (LGPO or LocalGPO) allows Group Policy Object management without Active Directory on standalone computers

Active Directory Group Policy (2)

Group Policy

- Mainly used for creation and deployment of policies in a domain
- IT Admins best friend – simple policy management
- gpupdate /force

Example: AD Group policies

- The command **gpresult /V**
- gives a lot of information regarding the AD
  - Settings
  - Applied group policy objects
  - Security groups you are part of

Active Directory Security Risks

- Active Directory forms an essential part to all steps of the Cyber Kill Chain
- Common Vulnerabilities include:
  - Use of Kerberos (hashing, golden ticket, silver ticket)
  - Supports the weak NTLM encryption
  - Brute-force attacks
  - Malware, phishing in windows platforms

Summary

- **Introduction**
- **Authentication of humans**
  - NIST digital authentication model
  - Authenticators
- **Authentication of machines**
  - Needham–Schroeder
  - Kerberos
  - Active directory
- **Bootcamp: setting up your lab environments**
- **Lab exercise**

The lab exercise: goals and learning points

- Introduction to a privilege escalation attack
- An attack on Microsoft Active Directory
- An exploit that targets the mode of operation (and implementation) of AES
- Setting up VMs and performing network attacks