

Cyber Security

Identification, authentication, authorization

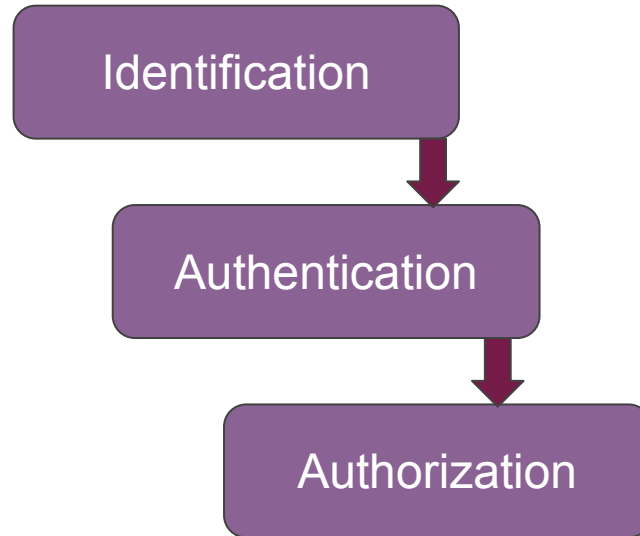
Grégoire Payen de La Garanderie



Access Control

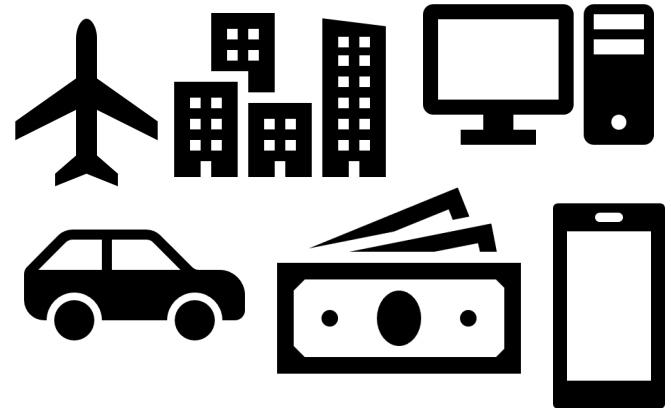


User



Access to the resources:

- Claim the identity
- Verify the credentials
- Check permissions
- Grant access



Identification

Terminology

- Subject — An active entity within a system (physical person, script, etc)
- Principal — An entity that can be granted access (represented by a username, userid, pin, etc)

Say your name



Enter your username



john.doe

Present yourself



Identification

The subject (a person, a script, etc) identifies itself to the system as a principal (represented by a username, userid, pin, etc).

Say your name



Enter your username



john.doe

Present yourself



Authentication

The system verify the identity of the user.

Present credentials



Enter your password



john.doe

Scan your fingerprints



Authentication

Terminology

- Subject — An active entity within a system (physical person, script, etc)
- Principal — An entity that can be granted access (represented by a username, userid, pin, etc)
- Object — Resource that some principals may access or use

Permissions (e.g. Unix)

-rwxr-x--- johndoe compsci

Access lists (e.g. Apache)

```
<RequireAll>  
  Require all granted  
  Require not ip 10.252.46.165  
</RequireAll>
```

Authentication

The system checks that the principal has the permissions to access an object.

Permissions (e.g. Unix)

`-rwxr-x--- johndoe compsci`

Access lists (e.g. Apache)

`<RequireAll>`

`Require all granted`

`Require not ip 10.252.46.165`

`</RequireAll>`

Credentials

Terminology

- What you know?
 - Passwords, pin numbers
- What you have?
 - Authentication key, passport, ticket, mobile phone
- Who you are?
 - Biometrics (fingerprints, DNA, face recognition)





The problem with Passwords

Top 20 most popular passwords (ranking 2018)

1. 123456 (Unchanged)	6. 111111 (New)	11. princess (New)	16. football (Down 7)
2. Password (Unchanged)	7. 1234567 (Up 1)	12. admin (Down 1)	17. 123123 (Unchanged)
3. 123456789 (Up 3)	8. sunshine (New)	13. welcome (Down 1)	18. monkey (Down 5)
4. 12345678 (Down 1)	9. qwerty (Down 5)	14. 666666 (New)	19. 654321 (New)
5. 12345 (Unchanged)	10. iloveyou (Unchanged)	15. abc123 (Unchanged)	20. !@#\$\$%^&* (New)

Source: <https://bit.ly/2Cq3O8e>

123456 has been used by almost 3% of people.

The problem with Passwords

Common Security Guidelines



- Adopt long passphrases
- Avoid easy to guess passwords
- Use combination of a-z, A-Z, 0-9 and symbols
- Do not write down passwords
- Avoid using the same password for multiple services

However — when Internet users log on to as many as 25 password-protected sites per day, remembering a different 14-character password is a Herculean mental exercise.

The problem with Passwords

Never write down plain passwords

But store them in a password-protected password agent

E.g.

Websites: LastPass, DashLane, 1Password

Tools: Roboform, PasswordSafe, Keepass

Authentication keys

Authentication keys (e.g. SSH keys)

- Similar to passwords, but
- RSA-based keys
- Subject create private/public key
- Share the public key with services
- Per device RSA key

```
> cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCr+LGFvYZh75uN
0of9i0sbAVJdXbby6gWXVwofk0AMV73MxrYRxecUWKDpsIFYL9+y
k0MpMl4a4zx2l4cs3RmfRNIq9Aiz9/F5h32pti3oU9EW8dB0hcSe
a4Zaqq8wSKBEE0KopqWm4CeTU/ARsiuS6KQqrlsq/08MMejPQpBJ
tj7oQFetj95mdnr8rR8Mf7qkj9X9VnjZr5lpZRkr6bu5ukcj+zR
kD8+XMLpmIPyhVpW8KLXEPdZ7Fq528wguAGB/RiCL8wceoU2S06d
XuxTbPUik/UgFp93weGAxvPHbg9vdIzCV6telWGHajzUyWdMPRm0
en6r6v9ym6tfEX451AZoxb6wT+JJiLdEXug9xUVn8BP3nB9AvZeF
2ogY5day9w+ECbEE0dZBAz5ZQ65Wf6WXZFU4Apbq/6cnDkTuM13E
hN0sdnG0UXwfa1QSfhUxqeMP3XZU4+sCdcXYDtLj6bk75Q5wvpXB
Vx5juM9hpfach/slB3vrtLyyaNNYubupVZmLH0W6nxuuu5gw1540
cLXEA3ZxBpWg8Ss0Ev1no9SEbuMcmIGlucMrXMBGGRLk1YqUf0mM
slgkxMazm/1n4qPi8zn3lb9tSxoP/V97QKr32zmKgrSPYnqAfta
+TiXHGTVL8wNBldr0r2oc7Nd+30CrXV6eF2dQfhA2GT7HQ== gre
```

Authentication keys

Advantages

- Public key leak are inconsequential
- Compromised device access can be revoked

```
> cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCr+LGFvYZh75uN
0of9i0sbAVJdXbby6gWXVwofk0AMV73MxrYRxecUWKDpsIFYL9+y
k0MpMl4a4zx2l4cs3RmfRNIq9Aiz9/F5h32pti3oU9EW8dB0hcSe
a4Zaqq8wSKBEE0KopqWm4CeTU/ARsiuS6KQqrlsq/08MMejPQpBJ
tj7oQFetj95mdnr8rR8Mf7qkj9X9VnjZr5lpZRkr6bu5ukcj+zR
kD8+XMLpmIPyhVpW8KLXEPdZ7Fq528wguAGB/RiCL8wceoU2S06d
XuxTbPUik/UgFp93weGAxvPHbg9vdIzCV6telWGHajzUyWdMPRm0
en6r6v9ym6tfEX451AZoxb6wT+JJiLdEXug9xUVn8BP3nB9AvZeF
2ogY5day9w+ECbEE0dZBAz5ZQ65Wf6WXZFU4Apbq/6cnDkTuM13E
hN0sdnG0UXwfa1QSFhUxqeMP3XZU4+sCdcXYDtLj6bk75Q5wvpXB
Vx5juM9hpfacH/slB3vrtLyYaNNYUbuPvZmLH0W6nxuuu5gw1540
cLXEA3ZxBpWg8Ss0Ev1no9SEbuMCmIGlucMrXMBGG6Rlk1YqUfOmM
slgkxMazm/ln4qPi8zn3lb9tSxoP/V97QKr32zmKgrSPYnqAfqta
+TiXHGTVL8wNBldr0r2oc7Nd+30CrXV6eF2dQfhA2GT7HQ== gre
```

Authentication keys

Demo time: SSH keys

Setting up SSH keys for your client

<code>ssh-keygen</code>	(create key for your client)
<code>ssh-copy-id xzpq33@mira.dur.ac.uk</code>	(send public key to server)
<code>ssh xzpq33@mira.dur.ac.uk</code>	(ssh'ing using your key)

Public key: <code>.ssh/id_rsa.pub</code>	(publicly share)
Private key: <code>.ssh/id_rsa</code>	(do not share)
Server authorized keys: <code>.ssh/authorized_keys</code>	(server side)

Security Keys

Authentication keys weakness: Compromised client

Solution: Physical security keys

- Static password token
(not recommended)
- Asynchronous tokens
(one-time passwords)
- Challenge-response tokens



One time password:
For a well known bank



Car keys



One time password: Yubikey

Biometrics

History of fingerprint



1982
Ink & paper



1990s
Optical



1990s
Capacitive



1997
First swip
sensor



2007
Slap sensor

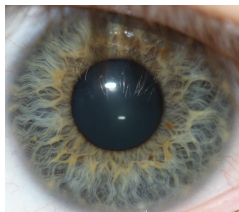


2010
Touchless
swipe sensor

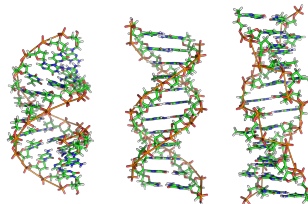
Biometrics



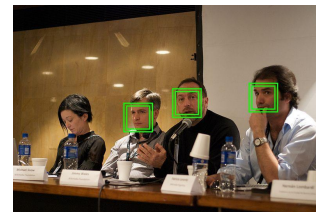
Fingerprints



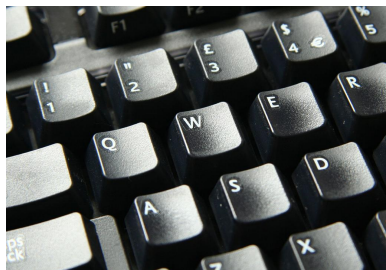
Iris recognition



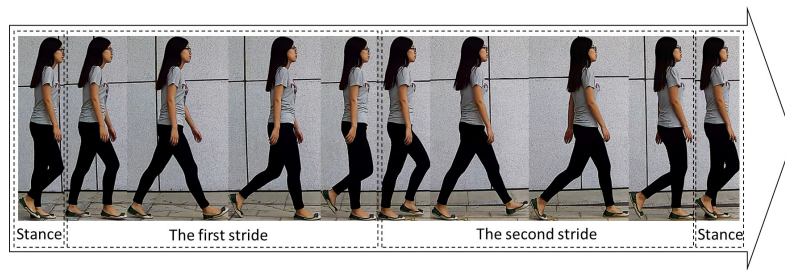
DNA matching



Face recognition



Keystroke & Mouse
biometrics



Gait recognition

Advantages

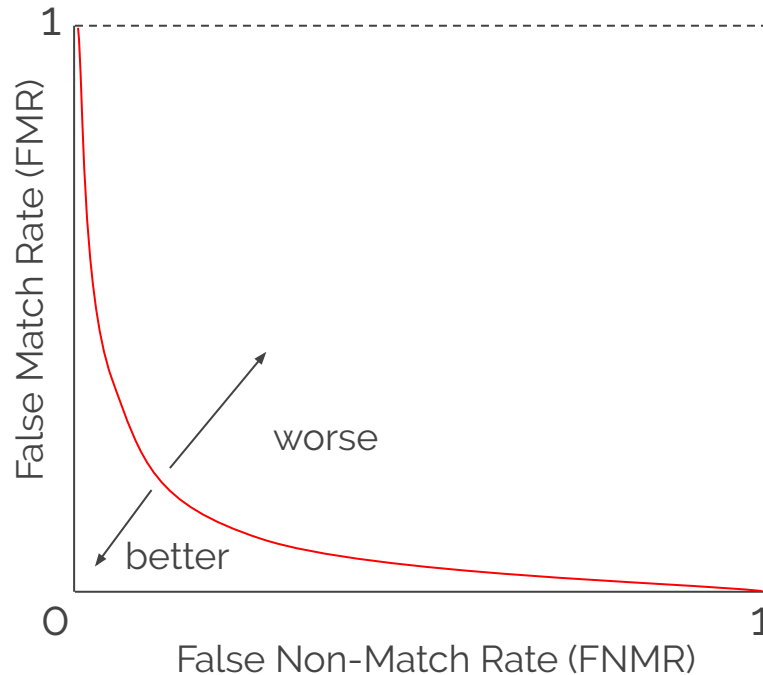
Non-repudiation: a way to guarantee that an individual who accesses a certain facility cannot later deny using it

Disadvantages

Uncertainty: Compromise between false-positives and false-negatives.



Receiver Operating Characteristic (ROC) curve

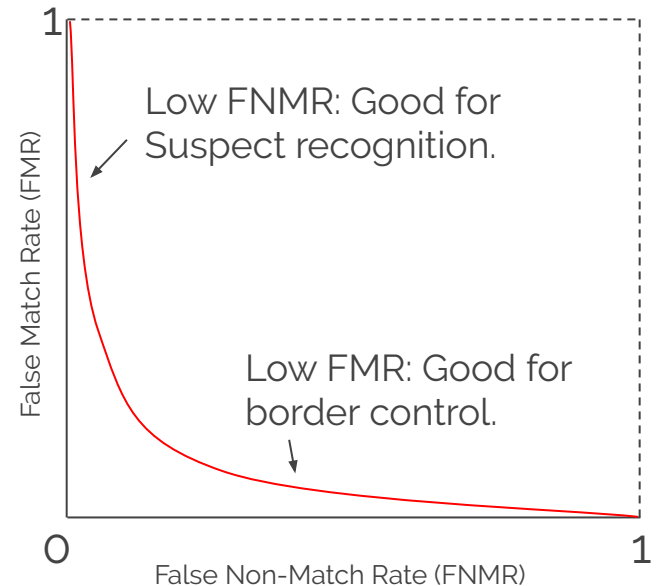


FMR = number of false positives / total matches

FNMR = number of false negatives / total matches

Performance Policy

- Prefer low FMR
E.g. automatic border control
Refer to human on negative result
- Prefer low FNMR
E.g. suspect recognition on CCTV
Refer to human on positive result



Two-factor authentication

Two-factor authentication

Combine two authentication factors from:

- “What you know”: password, pin
- “What you have”: mobile phone, authentication key

Best Practice!

Example: Bitcoin

- Principal: Public key
- Authentication factor: Public/private key
- Authorization mechanism:
each object (transaction) output has an
associated script controlling permissions:

Standard transaction (pay-to-pubkey-hash):

scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>

Funds freezed until specified time:

scriptPubKey: <expiry time> OP_CHECKLOCKTIMEVERIFY OP_DROP OP_DUP OP_HASH160

<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>



Example: Bitcoin

Standard transaction (pay-to-pubkey-hash):

scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash>

OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>

Funds freezed until specified time:

scriptPubKey: <expiry time> OP_CHECKLOCKTIMEVERIFY

OP_DROP OP_DUP OP_HASH160 <pubKeyHash>

OP_EQUALVERIFY OP_CHECKSIG

scriptSig: <sig> <pubKey>

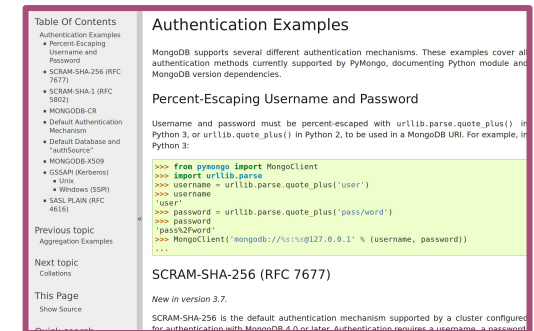
Zero-knowledge Password Proof (ZKPP) Durham University

Objective: Do not reveal anything in the client/server communications about the password
Otherwise we are vulnerable to replay attacks.

Most common ZKPP approach: Challenge-response auth

- Server generate unique challenge value: nonce
- Server send nonce to the client
- Client compute response = hash(nonce + password)
- Client send response back to server
- Server compare the response with hash(nonce + stored password)

More info: SCRAM-SHA-256 authentication



The screenshot shows the 'Authentication Examples' section of the MongoDB documentation. It includes a 'Table of Contents' on the left with links to 'Authentication Examples', 'Default Authentication Mechanism', 'Default Database and "authSource"', 'MONGODB-X509', 'GSSAPI (Kerberos)', 'SASL PLAIN (RFC 4616)', 'Previous topic', 'Next topic', 'Collations', 'This Page', and 'Show Source'. The main content area is titled 'Authentication Examples' and explains that MongoDB supports several authentication mechanisms. It then focuses on 'Percent-Escaping Username and Password', stating that usernames and passwords must be percent-escaped using `urllib.parse.quote_plus()` in Python 3 or `urllib.quote_plus()` in Python 2. A code block shows the following Python code:

```
>>> from pymongo import MongoClient
>>> import urllib.parse
>>> username = urllib.parse.quote_plus('user')
>>> password = urllib.parse.quote_plus('pass/word')
>>> MongoClient('mongodb://%s:%s@127.0.0.1' % (username, password))
...

```

 Below the code, it mentions 'SCRAM-SHA-256 (RFC 7677)' and notes that it is new in version 3.7. At the bottom, it states that SCRAM-SHA-256 is the default authentication mechanism supported by a cluster configured for authentication with MongoDB 4.0 or later, and that authentication requires a username and password.

Example: EMV payment

Standard used by all credit cards

EMV standard: Initially written in 1993

Over 3600 pages of protocol specification

Requirements varies from bank to banks

Protocol evolution as attacks gets more sophisticated.



Example: EMV payment

Card authentication mechanism

- Static data authentication (offline)
- Dynamic data authentication (offline)
- Combined DDA with application cryptogram generation (offline)
- Cryptogram (online)



Multiple cardholder verification mechanism (CVM):

- No CVM required (e.g. motorway toll)
- Signature (common in some countries, e.g. US)
- Offline PIN (no internet, pin verified by the card)
- Online PIN (internet, pin verified by the bank)

Example: EMV payment

SDA: Static Data Authentication

Offline card payment

Used by old card & terminal

Lowest common denominator

Vulnerable to skimming attack

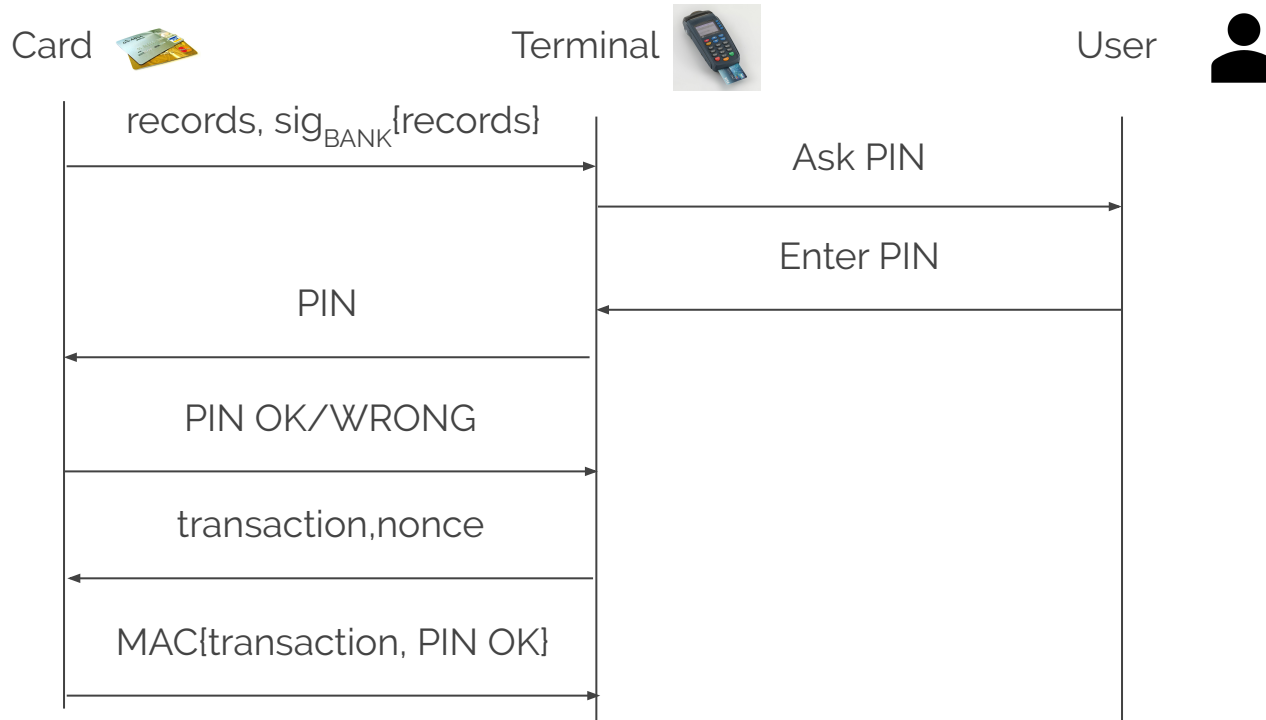
During transaction, terminal records the static data

A cloned card is created with the same static data.



Example: EMV payment

SDA: Static Data Authentication



* MAC: Message Authentication Code, computed by the card from a unique key

Example: EMV payment

SDA: Static Data Authentication & yes card attack

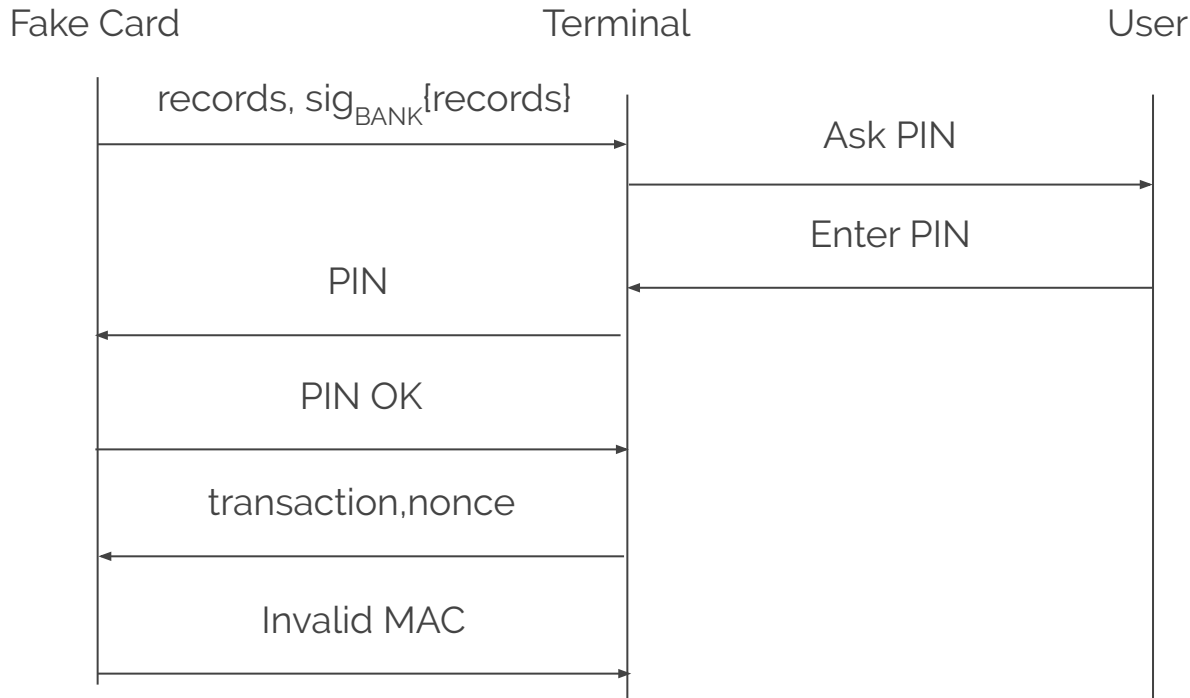
An attacker can get records, $\text{sig}_{\text{BANK}}\{\text{records}\}$ by listening to a valid transaction.

Then the attacker can create a fake card using $\text{sig}_{\text{BANK}}\{\text{records}\}$ and generate an invalid MAC. For offline transaction, the merchant cannot verify the MAC anyway. By the time the merchant send the transactions to the bank, the attacker will be long gone!

Problem: Static password

Example: EMV payment

SDA: Static Data Authentication



* MAC: Message Authentication Code, computed by the card from a unique key

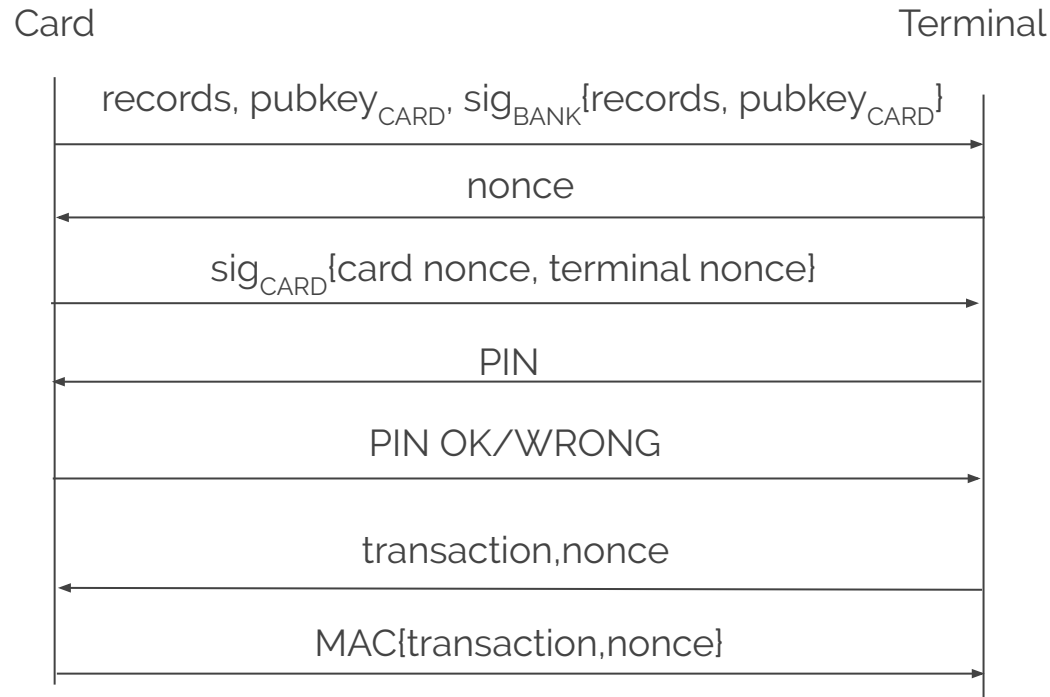
Example: EMV payment

DDA: Dynamic Data Authentication

Use challenge-response authentication to generate data unique to the transaction.

Example: EMV payment

DDA: Dynamic Data Authentication



* MAC: Message Authentication Code, computed by the card from a unique key

Example: EMV payment

DDA: Dynamic Data Authentication

YES card clone not possible:

Because $\text{sig}_{\text{CARD}}[\text{card nonce, terminal nonce}]$ is different at every transaction.

However, card answer to PIN check is not authenticated either.

A wedge between a stolen card and terminal can pretend that the password is always correct.

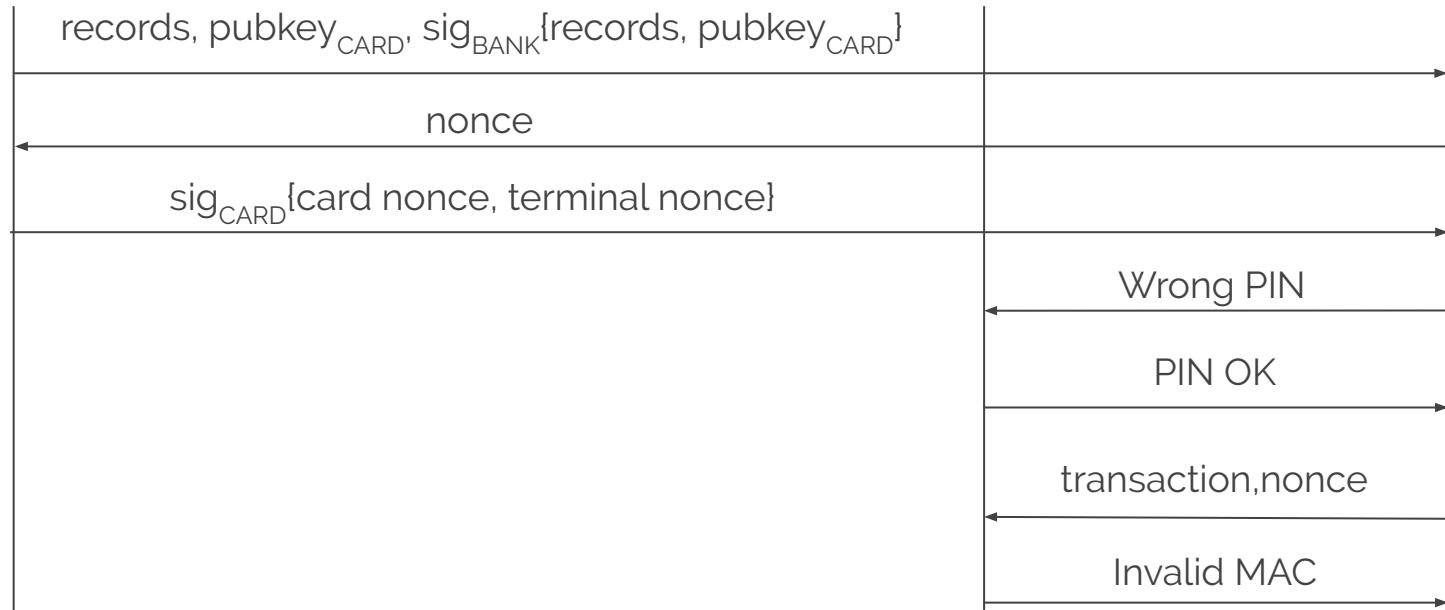
Example: EMV payment

DDA: Dynamic Data Authentication

Stolen Card

Wedge

Terminal



* MAC: Message Authentication Code, computed by the card from a unique key

Example: EMV payment

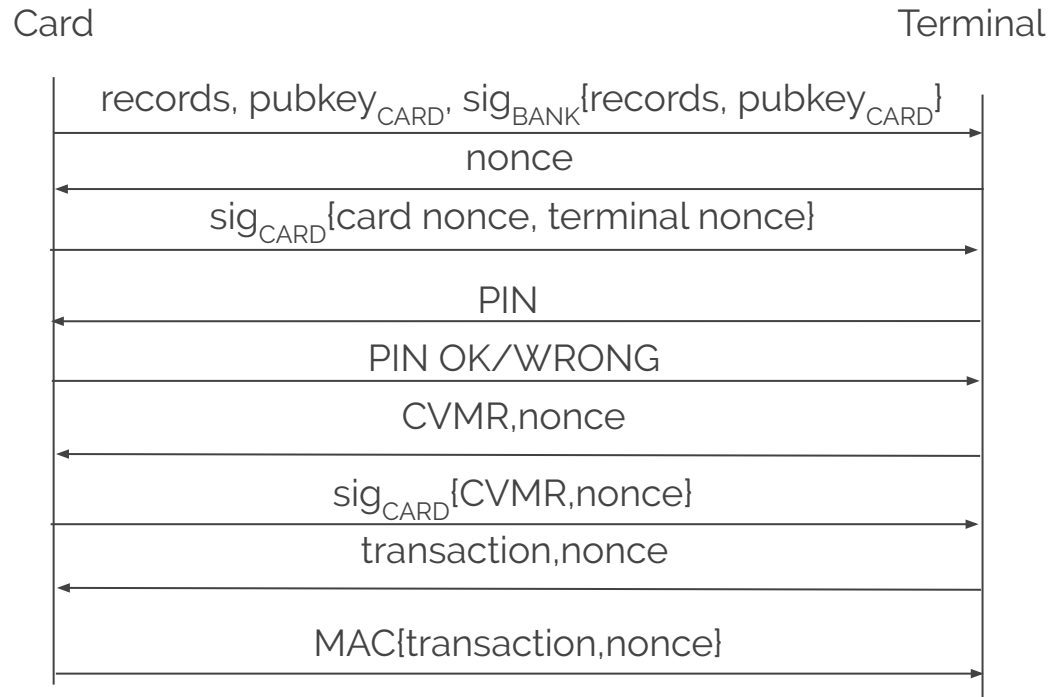
CDA: Combined DDA/Application Cryptogram Generation

Solution: Second card authentication step after PIN check.

The terminal send a message called CVMR representing the terminal view of the operation (PIN OK, PIN Wrong, signature, etc) for the card to compare with its own point of view.

Example: EMV payment

CDA: Combined Data Authentication



* MAC: Message Authentication Code, computed by the card from a unique key

Example: EMV payment

Takeaway:

Do not send static auth data (e.g. unencrypted passwords)

Use challenge-response to prevent replay attacks

Make sure that authentication is verified at all steps.

For more details:

<https://www.lightbluetouchpaper.org/2009/08/25/defending-against-wedge-attacks/>