

Authentication protocols

Identity attributes

- ▷ Set of attributes for setting apart individuals
 - ♦ Name
 - ♦ Numerical identifiers
 - Fixed for life
 - Variable with context
 - ♦ Address
 - ♦ Photo
 - ♦ Identity of relatives
 - Usually parents
 - ♦ ...

Authentication: Definition

- ▶ Proof that an entity has a claimed identity attribute
 - Hi, I'm Joe
 - Prove it!
 - Here are my Joe's credentials
 - Credentials accepted/not accepted
- Hi, I'm over 18
 - Prove it!
 - Here is the proof
 - Proof accepted/not accepted

Authentication: proof types

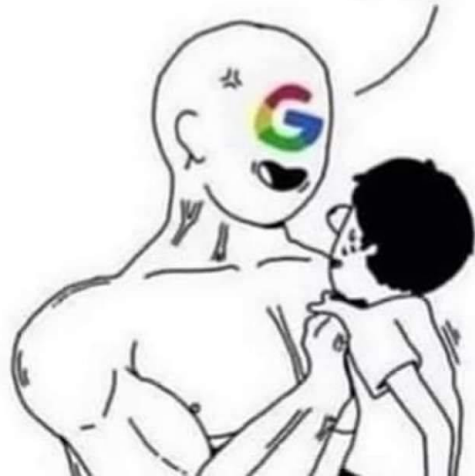
- ▷ Something we know
 - ♦ A secret memorized (or written down...) by Joe
- ▷ Something we have
 - ♦ An object/token solely held by Joe
- ▷ Something we are
 - ♦ Joe's Biometry
- ▷ Multi-factor authentication
 - ♦ Join or consecutive use of different proof types

Multi-factor verification jokes

me: *enters password
correctly on new device*

google:

Did you just sign in?



Authentication: goals

- ▷ Authenticate interactors
 - ◆ People, services, servers, hosts, networks, etc.
- ▷ Enable the enforcement of authorization policies and mechanisms
 - ◆ Authorization \Rightarrow authentication
- ▷ Facilitate the exploitation of other security-related protocols
 - ◆ e.g. key distribution for secure communication

Authentication: requirements

▷ Trustworthiness

- ♦ How good is it in proving the identity of an entity?
- ♦ How difficult is it to be deceived?
- ♦ Level of Assurance (LoA) (NIST, eIDAS, ISO 29115)
 - LoA 1 - Little or no confidence in the asserted identity
 - LoA 2 - Some confidence in the asserted identity
 - LoA 3 - High confidence in the asserted identity
 - LoA 4 - Very high confidence in the asserted identity

▷ Secrecy

- ♦ No disclosure of secrets used by legitimate entities

Authentication: requirements

▷ Robustness

- ◆ Prevent attacks to the protocol data exchanges
- ◆ Prevent on-line DoS attack scenarios
- ◆ Prevent off-line dictionary attacks

▷ Simplicity

- ◆ It should be as simple as possible to prevent entities from choosing dangerous shortcuts

▷ Deal with vulnerabilities introduced by people

- ◆ They have a natural tendency to facilitate or to take shortcuts

Authentication:

Entities and deployment model

▷ Entities

- ♦ People
- ♦ Hosts
- ♦ Networks
- ♦ Services / servers

▷ Deployment model

- ♦ Along the time
 - Only when interaction starts
 - Continuously along the interaction
- ♦ Directionality
 - Unidirectional
 - Bidirectional (or mutual)

Authentication interactions:

Basic approaches

▷ Direct approach

- ◆ Provide **credentials**
- ◆ Wait for verdict
- ◆ Authenticator checks credentials against what it knows

▷ Challenge-response approach

- ◆ Get **challenge**
- ◆ Provide a **response** computed from the **challenge** and the **credentials**
- ◆ Wait for verdict
- ◆ Authenticator checks response for the challenge provided and the credentials it knows

Authentication of people:

Direct approach w/ known password

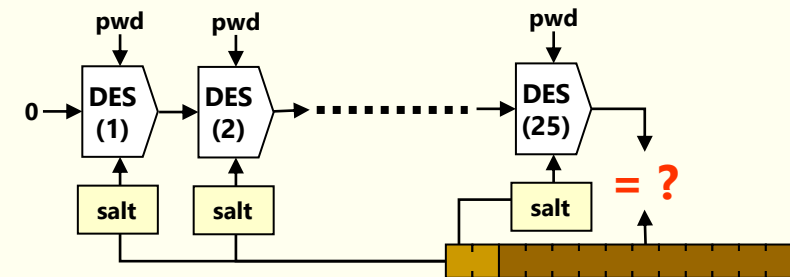
- ▷ A password is matched with a stored value
 - ♦ For a claimed identity (username)

- ▷ Personal stored value:
 - ♦ Transformed by a unidirectional function
 - Key Derivation Function (KDF)
 - Preferably slow!
 - Bcrypt, scrypt, Argon2, PBKDF2
 - ♦ UNIX: DES hash + salt
 - ♦ Linux: KDF + salt
 - ♦ Windows: digest function

$\text{DES hash} = \text{DES}_{\text{pwd}}^{25}(0)$

$\text{DES}_k^n(x) = \text{DES}_k(\text{DES}_k^{n-1}(x))$

Permutation of 12 subkeys' bit pairs with salt (12 bits)



Authentication of people:

Direct approach w/ known password

▷ Advantage

- ♦ Simplicity!
- ♦ Sharing!

▷ Problems

- ♦ Usage of predictable passwords
 - They enable dictionary attacks
- ♦ Different passwords for different systems
 - To prevent impersonation by malicious admins
 - But our memory has limits!
- ♦ Exchange along insecure communication channels
 - Eavesdroppers can easily learn the password
 - e.g. Unix remote services, PAP



Top 10 2023 at Portugal by NordPass

- 1 - 123456
- 2 - 12345
- 3 - 123456789
- 4 - 12345678
- 5 - benfica
- 6 - portugal
- 7 - sporting
- 8 - 1234567890
- 9 - password
- 10 - 1234567

source: <https://nordpass.com/most-common-passwords-list/>
Image <https://www.pinterest.com/networkboxusa/it-humor>

Password selection jokes



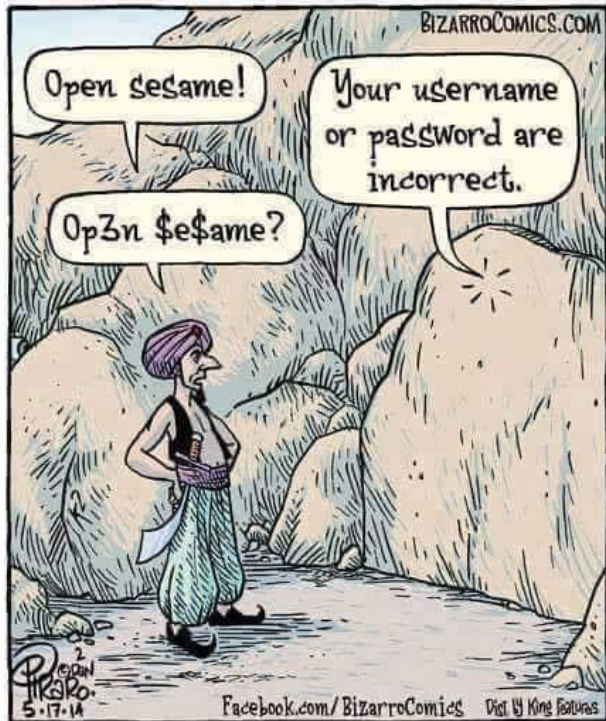
Dear IT,

the more "secure" you try to make our passwords by making them impossible to remember, the more likely I am to save them all in a big word doc named "Passwords"

Signed,
Everyone



Sorry, but your password must contain an uppercase letter, a number, a haiku, a gang sign, a hieroglyph, and the blood of a virgin.



Identification, Authentication and Authorization

Password bloopers



Authentication of people:

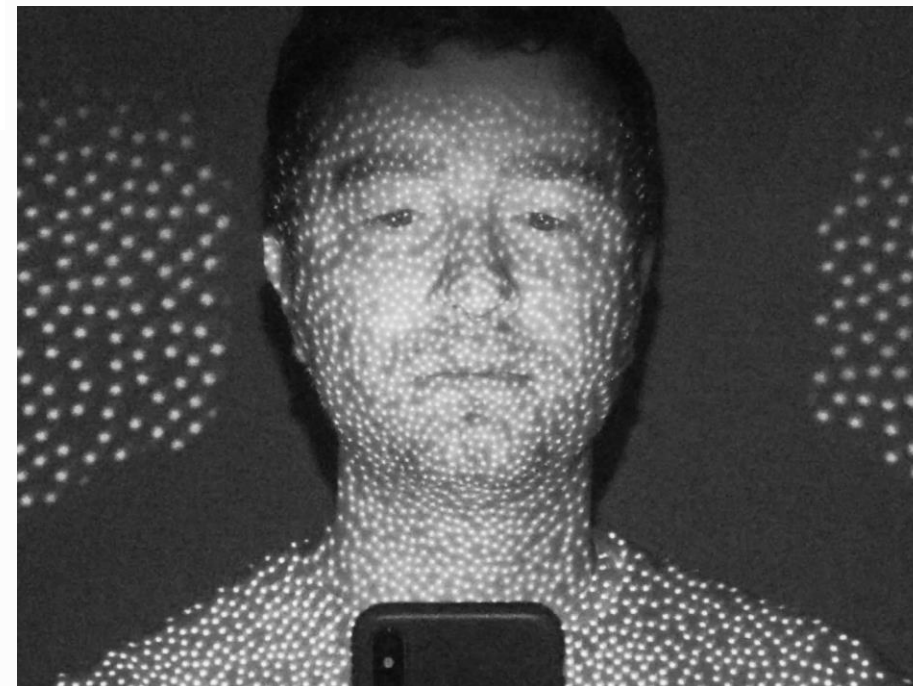
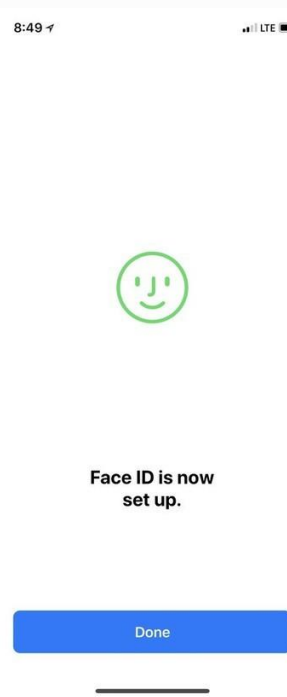
Direct approach with biometrics

▷ People get authenticated using body measurements

- ♦ Biometric samples or features
- ♦ Common modalities
 - Fingerprint
 - Facial recognition
 - Palm print
 - Iris scan
 - Voice recognition
 - DNA

▷ Measures are compared with personal records

- ♦ Biometric references (or template)
- ♦ Registered in the system with a previous enrolment procedure



Infrared Camera

An infrared camera reads the dot pattern, captures an infrared image, then sends the data to the secure enclave in the A11 Bionic chip to confirm a match.

Flood Illuminator

Invisible infrared light helps identify your face even when it's dark.

Dot Projector

More than 30,000 invisible dots are projected onto your face to build your unique facial map.



Fingerprint sensor

An optical sensor.

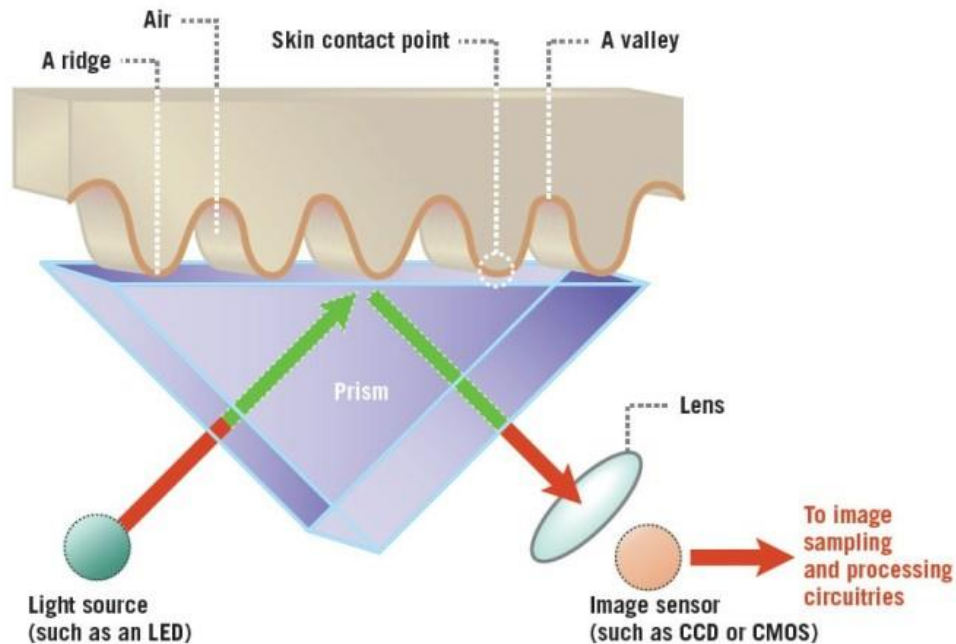
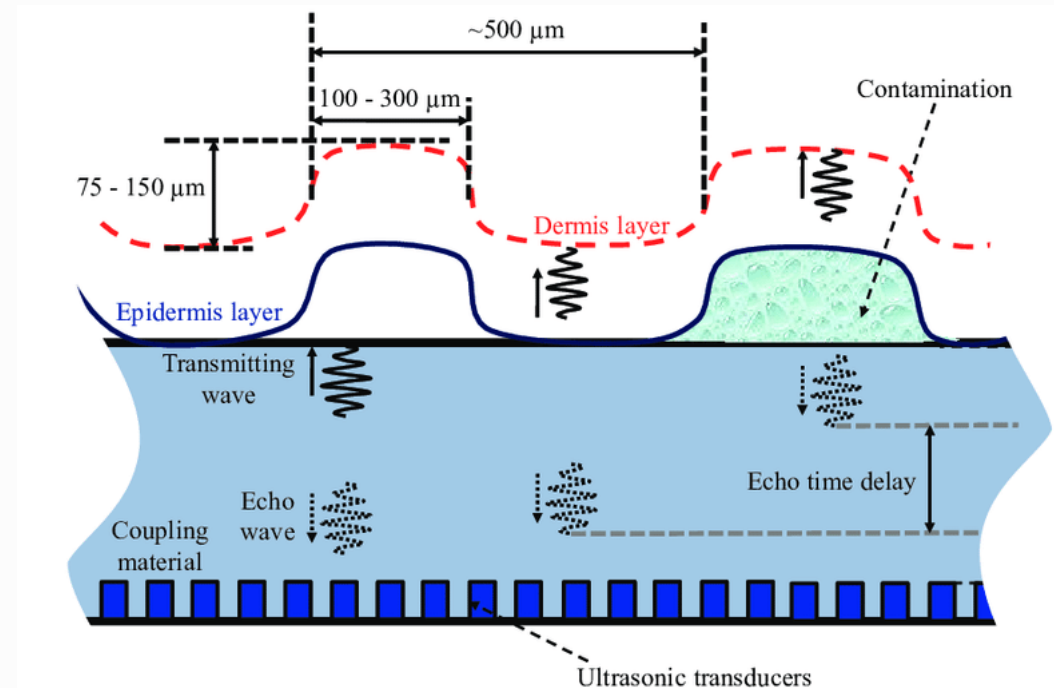


Figure 2



► Easy to bypass: [youtube/watch?v=hJ35ApLKpN4](https://www.youtube/watch?v=hJ35ApLKpN4)

Biometrics: advantages

- ▷ Convenient: people do not need to use memory
 - ♦ Just be their self
- ▷ People cannot choose weak passwords
 - ♦ In fact, they don't choose anything
- ▷ Credentials cannot be transferred to others
 - ♦ One cannot delegate their own authentication
- ▷ Stealth identification
 - ♦ Interesting for security surveillance

Biometrics: problems

- ▷ Usability
 - ◆ Comfort of people, ergonomic
 - ◆ Exploitation scenario
- ▷ Biometrics are still being improved
 - ◆ In many cases they can be easily cheated
 - ◆ Liveness detection
- ▷ People cannot change their credentials
 - ◆ Upon their robbery
- ▷ It can be risky for people
 - ◆ Removal of body parts for impersonation of the victim



Image source: <https://biometrics.mainguet.org/types/tongue.htm>

Biometrics: problems

- ▷ **Sensitivity tuning**
 - ♦ Reduction of FRR (annoying)
 - ♦ Reduction of FAR (dangerous)
 - ♦ Tuning is mainly performed with the target population
 - Not with attackers!
- ▷ Not easy to deploy remotely
 - ♦ Requires trusting the remote sample acquisition system
- ▷ Can reveal personal sensitive information
 - ♦ Diseases
- ▷ **Credentials cannot be (easily) copied to others**
 - ♦ In case of need in exceptional circumstances

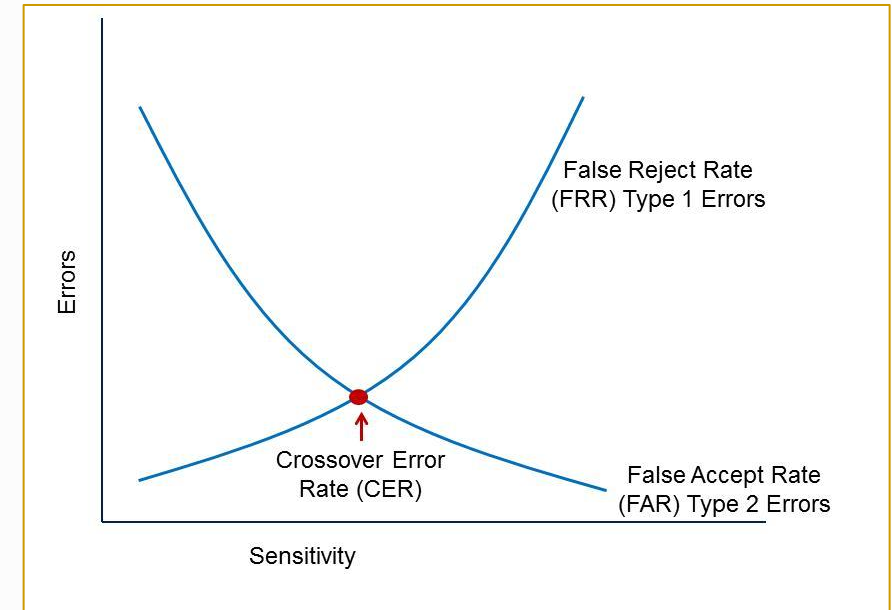


Image source: <http://www.pearsonitcertification.com/articles/article.aspx?p=1718488>

Authentication of people:

Direct approach with OTPs

▷ One-time password (OTP)

- ◆ Credential that can be used only once

▷ Advantage

- ◆ OTPs can be eavesdropped
- ◆ Eavesdroppers cannot impersonate the OTP owner
 - True for passive eavesdroppers
 - False for active attackers!

Authentication of people:

Direct approach with OTPs

▷ Problems

- ♦ Interactors need to know which password they should use at different occasions
 - Requires some form of synchronization
- ♦ People may need to use extra resources to maintain or generate one-time passwords
 - Paper sheets
 - Computer programs
 - Special devices, etc.

Authentication of people:

OTPs and secondary channels

- ▶ OTPs are codes sent through secondary channels
 - ♦ A secondary channel is a channel that is not the one where the code is going to be used
 - SMS, email, Twitter, Firebase, QR codes, NFC, etc.
 - ♦ The secondary channel provides the synchronization
 - Just-in-time provision of OTP

1.

- ▶ Two authentications are possible
 - ♦ Confirm a secondary channel provided by a profile owner
 - In order to trust that that channel belongs to the profile owner
 - ♦ Authenticate the owner of a profile
 - Which is bound to a secondary channel

Authentication of people:

OTPs produced from a shared key

▷ HOTP (Hash-based One Time Password, RFC 4226)

- ♦ OTP generated from a counter and a shared key
- ♦ Counters are updated independently

↳ serve de pulc maquina

▷ TOTP (Time-based One Time Password, RFC 6238)

- ♦ OTP generated from a timestamp and a shared password
- ♦ TOTP is HOTP with timestamps instead of counters
- ♦ Clocks need a rough synchronization



HOTP (HMAC-based one-time password, RFC 4226)

- ▷ Numeric OTP computed from shared key K and synchronized counter C
 - ♦ Hash key and counter
 - And increase counter
 - ♦ From hash, get a (floating) portion of 31 contiguous bits
 - Dynamic Binary Code (DBC)
 - ♦ Compute a d -long decimal number
 - $d \geq 6$

- ▷ Issues
 - ♦ Counter synchronization upon a failure
 - If the authenticator keeps it after a failure, exhaustive search attacks are viable
 - If the authenticator always increments it, DoS attacks are possible
 - ♦ Acceptance windows
 - Mitigates minor desynchronizations, but decreases security

TOPT (Time-based one-time password, RFC 6238)

▷ HOTP with a counter derived from time

▷ $C_T = \left\lfloor \frac{T - T_0}{T_x} \right\rfloor$

- ♦ T – initial time
- ♦ T_0 – initial time
- ♦ T_x – time interval (default: 30 seconds)

▷ $\text{TOTP}(K) = \text{HOTP}(K, C_T)$

Token-based OTP generators: RSA SecurID



- ▶ Personal authentication token
 - ◆ Or software modules for handhelds (PDAs, smartphones, etc.)
- ▶ It generates a unique number at a fixed rate
 - ◆ Usually one per minute (or 30 seconds)
 - ◆ Bound to a person (User ID)
 - ◆ Unique number computed with:
 - A 64-bit key stored in the token
 - The actual timestamp
 - A proprietary digest algorithm (SecurID hash)
 - An extra PIN (only for some tokens)

RSA SecurID

- ▷ OTP-based authentication
 - ♦ A user combines their User ID with the current token number
$$\text{OTP} = \text{User ID}, \text{Token Number}$$
- ▷ An RSA ACE Server does the same and checks for match
 - ♦ It also knows the person's key stored in the token
 - ♦ There must be a synchronization to tackle clock drifts
 - RSA Security Time Synchronization
- ▷ Robust against dictionary attacks
 - ♦ Keys are not selected by people

Yubikey

▷ Personal Authentication Device

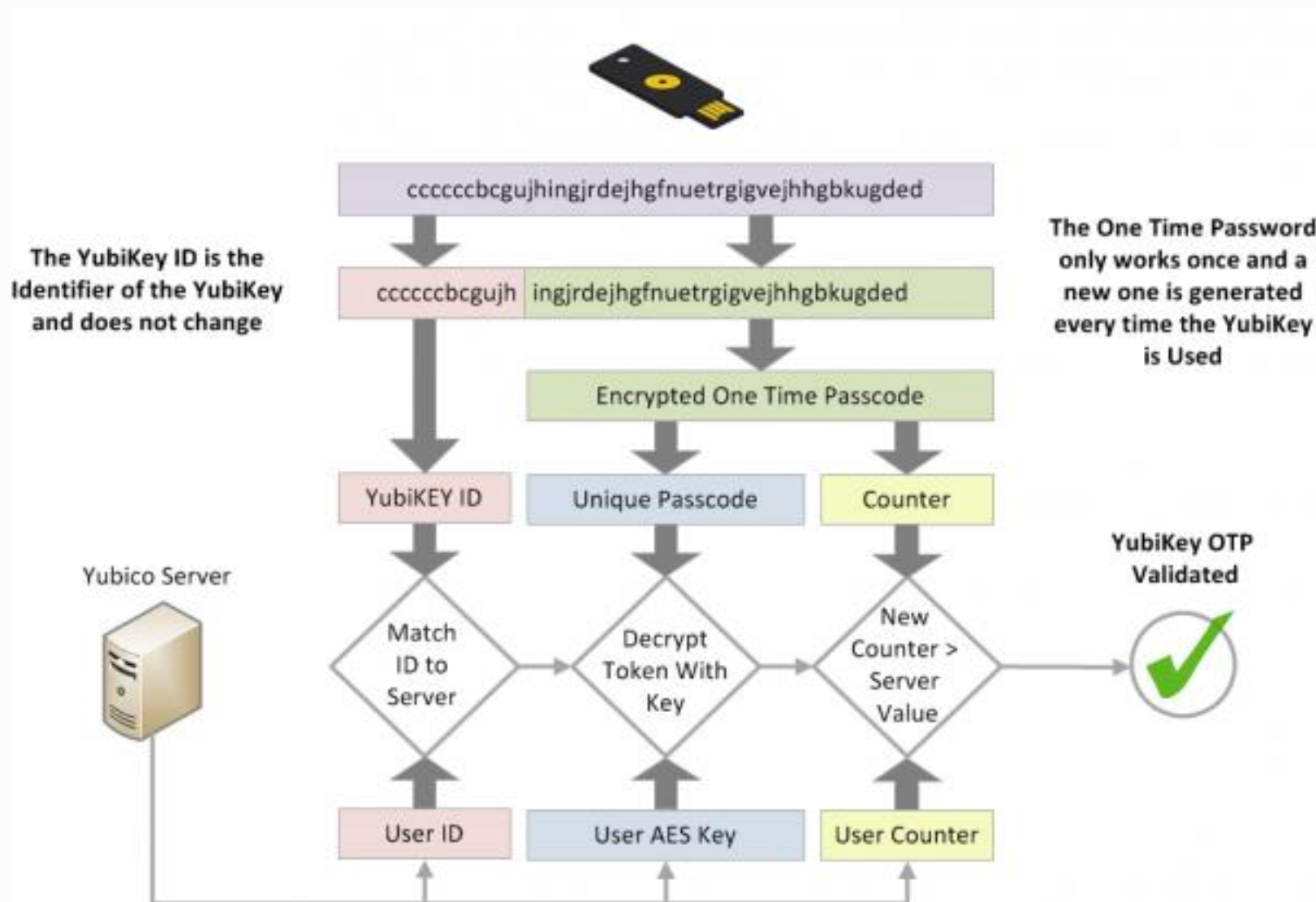
- ◆ USB and/or NFC

▷ Activation generates a 44 characters key

- ◆ Emulates a USB keyboard (besides an own API)
- ◆ Supports HOTP (events) or TOPT (Temporal)
- ◆ If a challenges is provided, user most touch the button to obtain a result
- ◆ Several algos, including AES 256



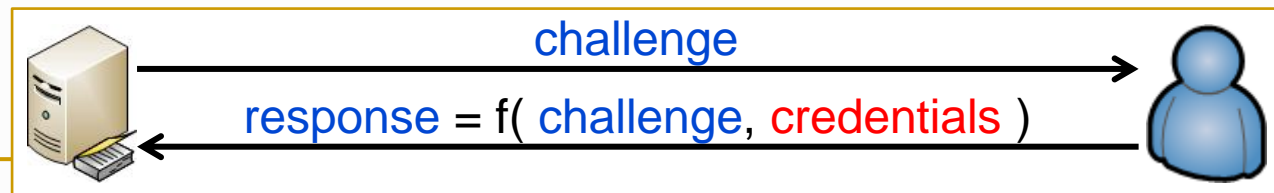
cccjggkhcbbirdrfdnlngghhfgtrnnlgedjlfrbdeut



Challenge-response approach:

Generic description

- ▷ The authenticator provides a challenge
- ▷ The entity being authenticated transforms the challenge
 - ♦ With its authentication credentials
- ▷ The result (response) is sent to the authenticator
- ▷ The authenticator checks the response
 - ♦ Produces a similar result and checks if they match
 - ♦ Transforms the result and checks if it matches the challenge or a related value



Challenge-response approach:

Generic description

▷ Advantage

- ♦ Authentication credentials are not exposed

▷ Problems

- ♦ People may require means to compute responses
 - Hardware or software
- ♦ The authenticator may have to have access to shared secrets
 - How can we prevent them from using the secrets elsewhere?
- ♦ Offline dictionary attacks
 - Against recorded challenge-response dialogs
 - Can reveal secret credentials (passwords, keys)

Challenge-response protocols: selection of challenges

- ▷ Challenges cannot be repeated for the same entity
 - ♦ Same challenge → same response
 - ♦ An active attacker can impersonate a user using a previously recorded protocol run
- ▷ Challenges should be nonces
 - ♦ Nonce: number used only once
 - ♦ Stateful services can use counters
 - ♦ Stateless services can use (large) random numbers
 - ♦ Time can be used, but with caution
 - Because one cannot repeat a timestamp

Authentication of people:

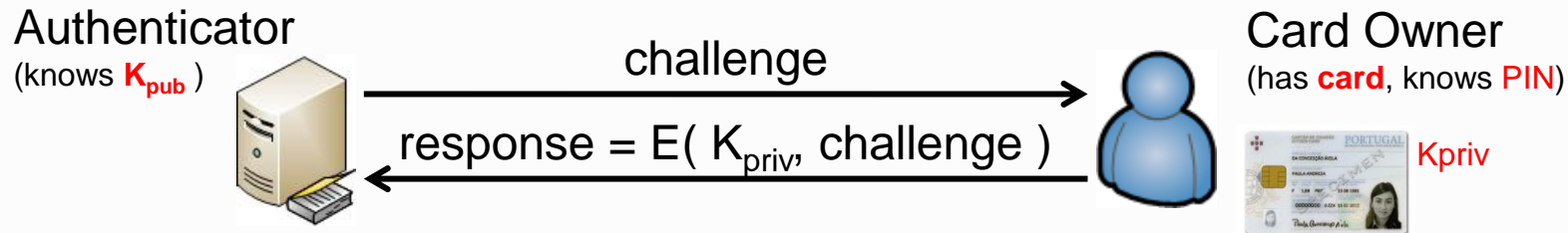
Challenge-response with smartcards

- ▶ Authentication credentials
 - ♦ The smartcard
 - e.g. Citizen Card
 - ♦ The private key stored in the smartcard
 - ♦ The PIN to unlock the private key
- ▶ The authenticator knows
 - ♦ The corresponding public key
 - ♦ Or some personal identifier
 - Which can be related with a public key through a (verifiable) certificate



Authentication of people:

Challenge-response with smartcards



▷ Signature-based protocol

- ♦ The authenticator generates a random challenge
 - Or a value not used before
- ♦ The card owner ciphers the challenge with their private key
 - PIN-protected
- ♦ The authenticator decrypts the result with the public key
 - If the output matches the challenge, the authentication succeeds

▷ Encryption-based protocol

- ♦ Possible when private key decryption is available

Authentication of people:

Challenge-response with memorized password

- ▷ Authentication credentials
 - ◆ Passwords selected by people
- ▷ The authenticator knows
 - ◆ All the registered passwords; or
 - ◆ A transformation of each password
 - Preferable option
 - Preferably combined with some local value (salt)
 - Preferable using a tunable function (e.g. iterations)

Authentication of people:

Challenge-response with memorized password

- ▷ The authenticator generates a random challenge
- ▷ The person computes a function of the challenge and password
 - ♦ e.g. a joint digest: $\text{response} = \text{digest}(\text{challenge}, \text{password})$
 - ♦ e.g. an encryption $\text{response} = E_{\text{password}}(\text{challenge})$
- ▷ The authenticator does the same (or the inverse)
 - ♦ If the output matches the response (or the challenge), the authentication succeeds
- ▷ Examples
 - CHAP, MS-CHAP v1/v2, S/Key

PAP & CHAP

(RFC 1334, 1992, RFC 1994, 1996)

- ▷ Protocols used in PPP (Point-to-Point Protocol)
 - ♦ Unidirectional authentication
 - Authenticator is not authenticated
- ▷ PPP developed in 1992
 - ♦ Mostly used for dial-up connections
- ▷ PPP protocols are used by PPTP VPNs
 - ♦ e.g. vpn.ua.pt

PAP & CHAP

(RFC 1334, 1992, RFC 1994, 1996)

▷ PAP (PPP Authentication Protocol)

- ♦ Simple UID/password presentation
- ♦ Insecure cleartext password transmission

▷ CHAP (CHallenge-response Authentication Protocol)

Aut → U: authID, challenge

U → Aut: authID, MD5(authID, pwd, challenge), identity

Aut → U: authID, OK/not OK

- ♦ The authenticator may require a reauthentication anytime

MS-CHAP (Microsoft CHAP)

(RFC 2433, 1998, RFC 2759, 2000)

▷ Version 1

A → U: authID, **C**
U → A: **R1, R2**
A → U: OK/not OK

$R1 = DES_{LMPH}(C)$
 $R2 = DES_{NTPH}(C)$

$LMPH = DEShash(pwd')$
 $NTPH = MD4(pwd)$

$pwd' = capitalized(pwd)$

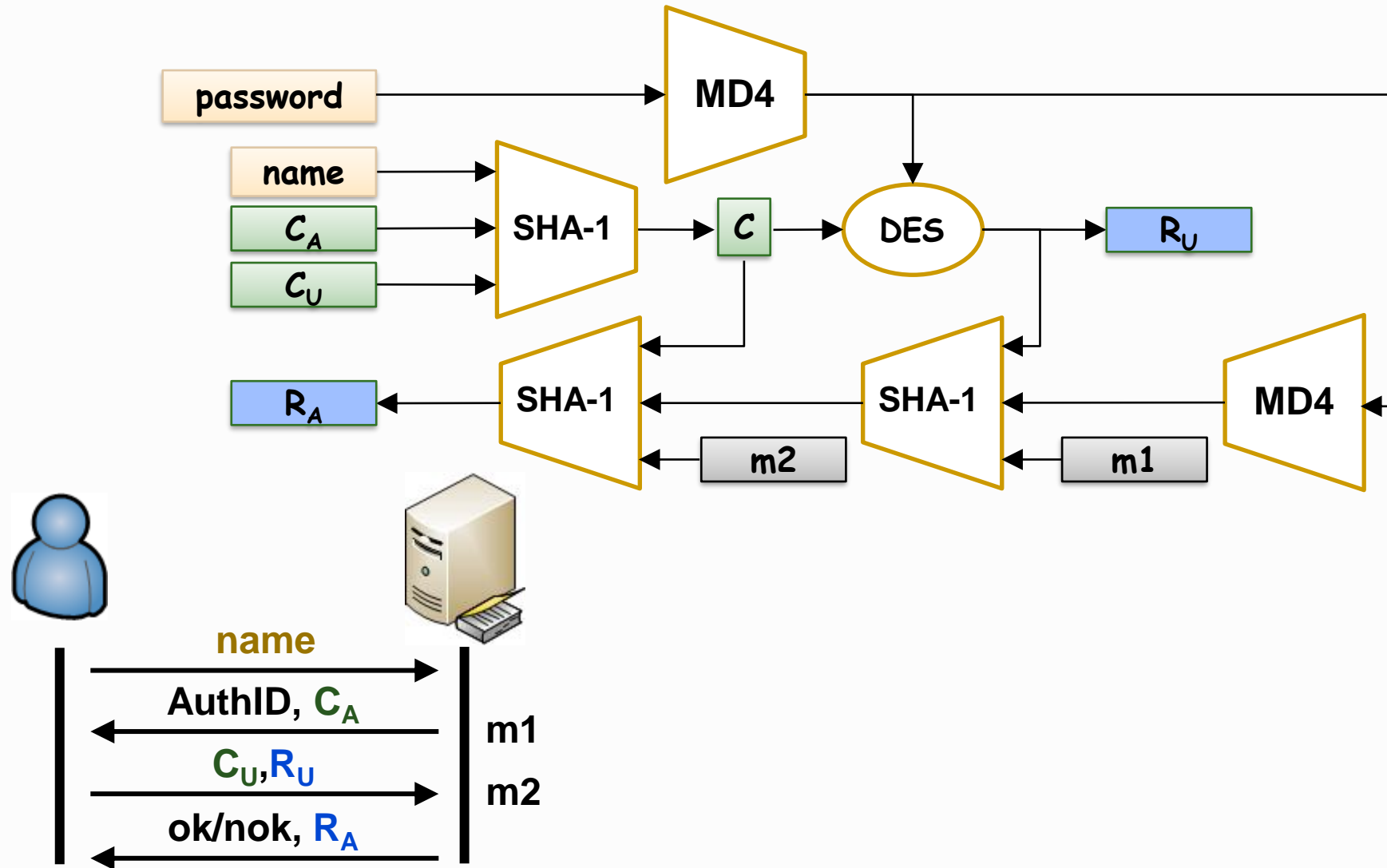
▷ Version 2

A → U: authID, **C_A** ← **m1**
U → A: **C_U, R1** ← **m2**
A → U: OK/not OK, **R2**

$R1 = DES_{PH}(C)$
 $C = SHA(C_U, C_A, username)$
 $PH = MD4(password)$
 $R2 = SHA(SHA(MD4(PH), R1, m1), C, m2)$

- Mutual authentication
- Passwords can be updated

MS-CHAP v2



Authentication of people:

Generation of OTPs with challenges

- ▷ OTPs can be produced from a challenge received
 - ♦ The fundamental protocol is password-based
 - But passwords are OTPs
 - ♦ OTPs are produced from a challenge
 - ♦ One can use several algorithms to handle OTPs

Authentication of people:

OTPs selected from shared data

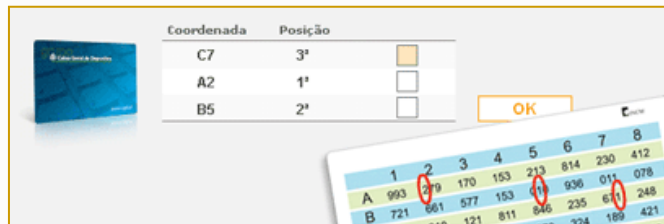
Co-located Matrix

► Advantage:

- ♦ Shared data can be random
- ♦ No long-term short secrets to protect

► OTPs build from printed data

- ♦ Example: online bank codes



► Selection of an OTP from a printed / saved list

TPW list generated 2020-12-01 14:10 on ubuntu

```
000 jZos ZoWc 056 Q%8x reaG 112 Pwvj ZZKE 168 b!qB ZxJ= 224 MABc m8dj
001 e6+w ZTpi 057 bUzv 67yK 113 dct6 M3m3 169 WL7T szrM 225 UxZz BgJa
002 Z5/n bWH2 058 wKps yVeW 114 gY=X X3R% 170 MFL: /3aH 226 6+w2 v2Mj
003 FOCV E=/4 059 uFUT j2Ag 115 tF6u E8mh 171 fk9H 2bvb 227 7z=8 xOP0
004 yAOF YbvS 060 NExv %hmn 116 E380 HQ85 172 Ls9u 8DXn 228 4Tow uKkz
005 SBP4 NP8r 061 P4pZ JrL9 117 qdXp FYxc 173 4ZbJ RMXL 229 G4eL od:N
006 K=Ze O7sp 062 K2ys +Wvb 118 wvcC =85E 174 zPnj 2rA8 230 54f= P5xH
007 9Bqp 9E97 063 yBP= rE39 119 sIdC mqDi 175 5Kek oaI+ 231 2mwJ uAJV
008 mhmP BjMX 064 Yayh Y=uM 120 3bcx 4cSB 176 Z:Q6 dsCn 232 9kB: :xap
009 3KLK w2ck 065 t6FW er=a 121 dE:9 97vL 177 :aFX buOX 233 eR3O mben
010 %MXc 3x7: 066 J+y: DpVR 122 BWMz 65GX 178 ZTFQ IJHS 234 D+rQ WPwO
011 jRTZ Ftsu 067 wERC JF5Q 123 XaYP XUFN 179 A3tN 9p5v 235 o2m% PmS=
012 :UtJ xzLb 068 bCuN eGIX 124 =AWj %J%p 180 MCre BFgp 236 dz3+ EZRd
013 PgVK M+vS 069 mo+d taKT 125 wODU =mwS 181 56IO VhB6 237 3RNm xFHY
014 NT/s F8mf 070 XiSO rNDz 126 qCnE 5Kp/ 182 6=im Rht4 238 rLKG N:3c
015 Hj+Z nmCK 071 UA8U qECw 127 pafC foFR 183 RM7Q 9Twe 239 d4z2 HTuc
016 =OKR rInR 072 PjGj n:oS 128 DSdk ufcM 184 eKmQ avXK 240 PnPM NthB
017 hKbD UrZX 073 XYTD ZKfn 129 dFAC /zVX 185 wEtS X3P5 241 gTiF 69k4
018 tCbN wfiR 074 q4B2 uPh9 130 lx2: XsGF 186 RgQe OnoQ 242 sy/U bpZJ
019 8R2V MNx5 075 =qv2 oMr9 131 gc2Y YSWd 187 WAF9 5Ac/ 243 =vS5 pXxx
020 9wC7 Q8Vh 076 E/2P e5I= 132 GhI6 P4bP 188 u/K4 MyTy 244 PLKS roct
021 g2Pg qTuF 077 TaFC /cs7 133 b=aP UeSQ 189 3FOC /9nd 245 ACj4 4s:A
022 9uxI P:dy 078 ILZ6 Tvpi 134 wLiT AgSS 190 vhty RMe2 246 GmvO Cp2N
023 qtUo GAX3 079 bD2x GRet 135 tDoH 7qXH 191 oHnH y5KG 247 /mcs Gair
024 ZXFO 8HAV 080 W%Ig c:T= 136 VgNd Evz6 192 CT6a HrAc 248 yozd vKtf
025 E8/N 7kxE 081 eRwK nzx8 137 =Gsw EHWg 193 %Xm/ =pbj 249 z:BY BW5L
026 guVQ Ljhb 082 xfeT z9ta 138 9OOX RtQT 194 f4gE NHOB 250 %zBj jsvd
027 Y%8n :y6f 083 a3ob rjvf 139 KfMj VE6k 195 %38V 3ZEh 251 %ZNR vDBY
028 rDmL X9pH 084 q89w pFka 140 Ym+ hmaY 196 =Tho ORDr 252 QAOt tbe7
029 awpG M6Rt 085 7/no X/m3 141 gGub j:hZ 197 Ppjo AwgM 253 c3F4 Q4oD
030 Jt:w DGUX 086 5iWg L=Am 142 4iBd IpBy 198 %mdT opDB 254 hwQw BQoO
031 aFh2 uPSP 087 cMxK 6tjU 143 dsRp NTN/ 199 rtm2 OGN+ 255 fxeC Q%mb
032 mx4E GIVC 088 guQR h+Kd 144 Mk6X S/qJ 200 LHQo rOVN 256 o5zt +xMm
033 xh7V CYgj 089 Viw/ AaFq 145 Otm3 %NfF 201 IfsK JGkK 257 OTKn Co4p
034 ZjBZ xW:j 090 bDcc mMUb 146 YR28 OxTH 202 d5zO tWbb 258 ceb+ s=2%
035 RwhA wcV: 091 KuOk nF/G 147 8AKW 99u7 203 dMOJ d:/I 259 moQh RoOK
036 XRsS NPGR 092 PSY3 expc 148 onxi /gBe 204 L3WG AH8K 260 SMuY 9ArI
037 B6VS EKT/ 093 TiRy 5ZQj 149 ozwu O2br 205 EzB6 =Uc9 261 AGDP pYdt
038 %J6/ 6HR7 094 /HQz jeUC 150 NVR7 tsgm 206 46f9 g9BX 262 i%7h RcnF
039 eYQa fwAg 095 Rzd2 /3kf 151 Ewer s9/f 207 Va2M Lr=S 263 dn8j 955y
040 Wt4A 3TQA 096 XjtZ ARep 152 bxc5 /+Sv 208 :+md drVx 264 tjYf uP22
041 4Pwp W68U 097 kZBy t8WW 153 zFgv Xz8q 209 ayGk e8oF 265 %WZ BcR=
042 Myi3 j9k9 098 b:Kx PuI6 154 Zsp2 UaNN 210 9hv6 Vu3: 266 XC%Q GpOG
043 QH58 kQhy 099 z3w7 B8Qa 155 PPOg ih8e 211 kv8n kn+o 267 /:IK Ik3w
044 zYCi y5NZ 100 4wdV =ak 156 bhy4 UkfN 212 qL8a cDz= 268 2ei5 dY43
045 s=Dz a2F2 101 BITZ JP9Q 157 Ufv% T:Wx 213 LTwv 96a9 269 ABM 3mN2
046 7AFp Rctz 102 4WPg HNko 158 XsED ywk+ 214 ywp+ Xq2P 270 7yoU fB6w
047 ao8H 5PHh 103 g6mJ T3YK 159 mTuL ZsrQ 215 ST2: qzCF 271 uA:4 Q+bJ
048 /vgM /h%c 104 wsiw x3/U 160 nHGY aIB+ 216 CnOH Wf6P 272 =aJb w97Z
049 TznT mhbL 105 UTNP 6vjE 161 :Cin i7:4 217 IFJh x5cZ 273 5dSN evT=
050 6Tha :rNG 106 zKAu 8Qt6 162 /ECm Z6yy 218 8U5W Xu% 274 eim+ eWz%
051 DM9z wb37 107 tojS KwGB 163 PbA7 3rja 219 N=8= pCiU 275 +Qm% %zZo
052 NUze JzMn 108 66n8 jhKk 164 O:I7 :2qO 220 CPuy =y3K 276 Qs4k i5HP
053 RfE% 7:BN 109 Pt9u +/p+ 165 =vuu O:dt 221 O5vF XaoK 277 +2KS +rvR
054 mJqq hmK= 110 Aped PQbv 166 9KAR Fazz 222 umBy =t5d 278 u47J XSUO
055 FI/d nFk= 111 PF+k FdFZ 167 x%a= 222P 223 iC2I =2eJ 279 xP6i 3TCQ
```

S/Key (RFC 2289, 1998)

▷ Authentication credentials

- ♦ A password (**pwd**)

▷ The authenticator knows

- ♦ The last used one-time password (**OTP**)
- ♦ The last used **OTP index**
 - Defines an order among consecutive OTPs
- ♦ A **seed** value for each person's OTPs
 - The seed is similar to a UNIX salt

S/Key setup

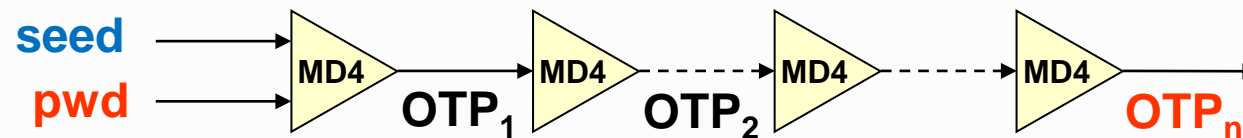
▷ The authenticator defines a random seed

▷ The person generates an initial OTP as:

$$\text{OTP}_n = h^n(\text{seed}, \text{pwd}), \text{ where } h = \text{MD4}$$

♦ Some S/Key versions also use MD5 or SHA-1

▷ The authenticator stores seed, n and OTP_n as authentication credentials



S/Key authentication protocol

- ▷ Authenticator sends seed & index of the person
 - ♦ They act as a challenge
- ▷ The person generates index-1 OTPs in a row
 - ♦ And selects the last one as result
 - ♦ $\text{result} = \text{OPT}_{\text{index}-1}$
- ▷ The authenticator computes $h(\text{result})$ and compares the result with the stored $\text{OPT}_{\text{index}}$
 - ♦ If they match, the authentication succeeds
 - ♦ Upon success, stores the recently used index & OTP
 - index-1 and $\text{OPT}_{\text{index}-1}$

S/Key

▷ Advantages

- ◆ Users passwords are unknown to authenticators
- ◆ OTPs can be used as ordinary passwords

▷ Disadvantages

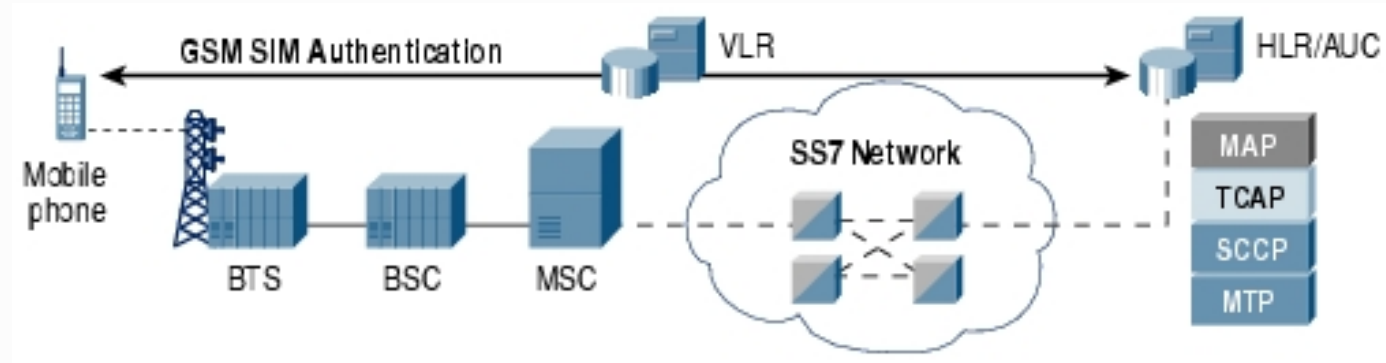
- ◆ People need an application to compute OTPs
- ◆ Passwords can be derived using dictionary attacks
 - From data stored in authenticators
 - From captured protocol runs

Authentication of people:

Challenge-response with shared key

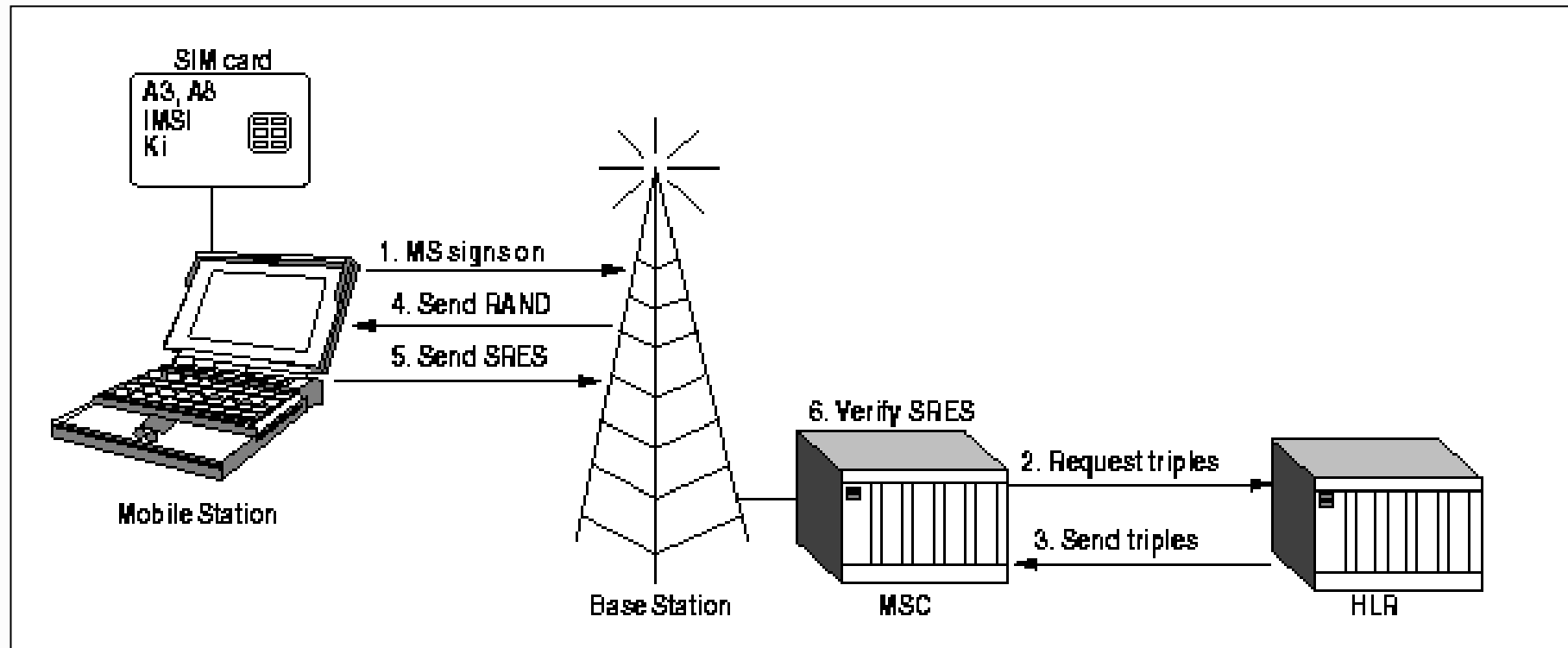
- ▶ Uses a shared key instead of a password
 - ◆ Robust against dictionary attacks
 - ◆ Requires some token to store the key
- ▶ Example:
 - ◆ GSM

GSM: authentication architecture



- ▷ Based on a secret key shared between the HLR and the station
 - ♦ 128 Ki, stored in the station's SIM card
 - ♦ Can only be used after entering a PIN
- ▷ Algorithms (initially not public):
 - ♦ A3 for authentication
 - ♦ A8 for generating a session key
 - ♦ A5 for encrypting the communication
- ▷ A3 and A8 implemented by SIM card
 - ♦ Can be freely selected by the operator

GSM: mobile station authentication



GSM: mobile station authentication

- ▷ MSC fetches trio from HLR
 - ♦ $RAND, SRES, K_c$
 - ♦ In fact more than one are requested
- ▷ HLR generates $RAND$ and corresponding trio using subscriber's K_i
 - ♦ $RAND$, random value (128 bits)
 - ♦ $SRES = A3(K_i, RAND)$ (32 bits)
 - ♦ $K_c = A8(K_i, RAND)$ (64 bits)
- ▷ Usually operators use COMP128 for A3/A8
 - ♦ Recommended by the GSM Consortium
 - ♦ $[SRES, K_c] = COMP128(K_i, RAND)$

Host authentication

- ▷ By name or address
 - ◆ DNS name, IP address, MAC address, other
 - ◆ Extremely weak, no cryptographic proofs
 - Nevertheless, used by many services
 - e.g. NFS, TCP *wrappers*
- ▷ With cryptographic keys
 - ◆ Keys shared among peers
 - With an history of usual interaction
 - ◆ Per-host asymmetric key pair
 - Pre-shared public keys with usual peers
 - Certified public keys with any peer

Service / server authentication

▷ Host authentication

- ♦ All co-located services/servers are indirectly authenticated

▷ Per-service/server credentials

- ♦ Shared keys
 - When related with the authentication of people
 - The key shared with each person can be used to authenticate the service to that person
- ♦ Per-service/server asymmetric key pair
 - Certified or not

TLS (Transport Layer Security, RFC 8446)

- ▷ Secure communication protocol over TCP/IP
 - ♦ Created upon SSL V3 (Secure Sockets Layer)
 - ♦ Manages per-application secure sessions over TCP/IP
 - Initially conceived for HTTP traffic
 - Actually used for other traffic types
- ▷ There is a similar version for UDP (DTLS, RFC 6347)
- ▷ Security mechanisms
 - ♦ Communication confidentiality and integrity
 - Key distribution
 - ♦ Authentication of communication endpoints
 - Servers (or, more frequently, services)
 - Client users
 - Both with asymmetric key pairs, typically with certified public keys

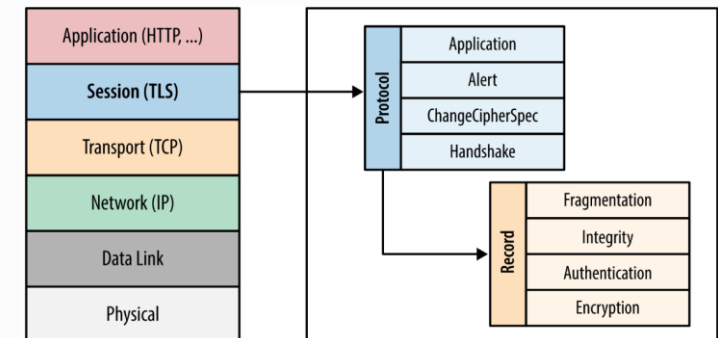
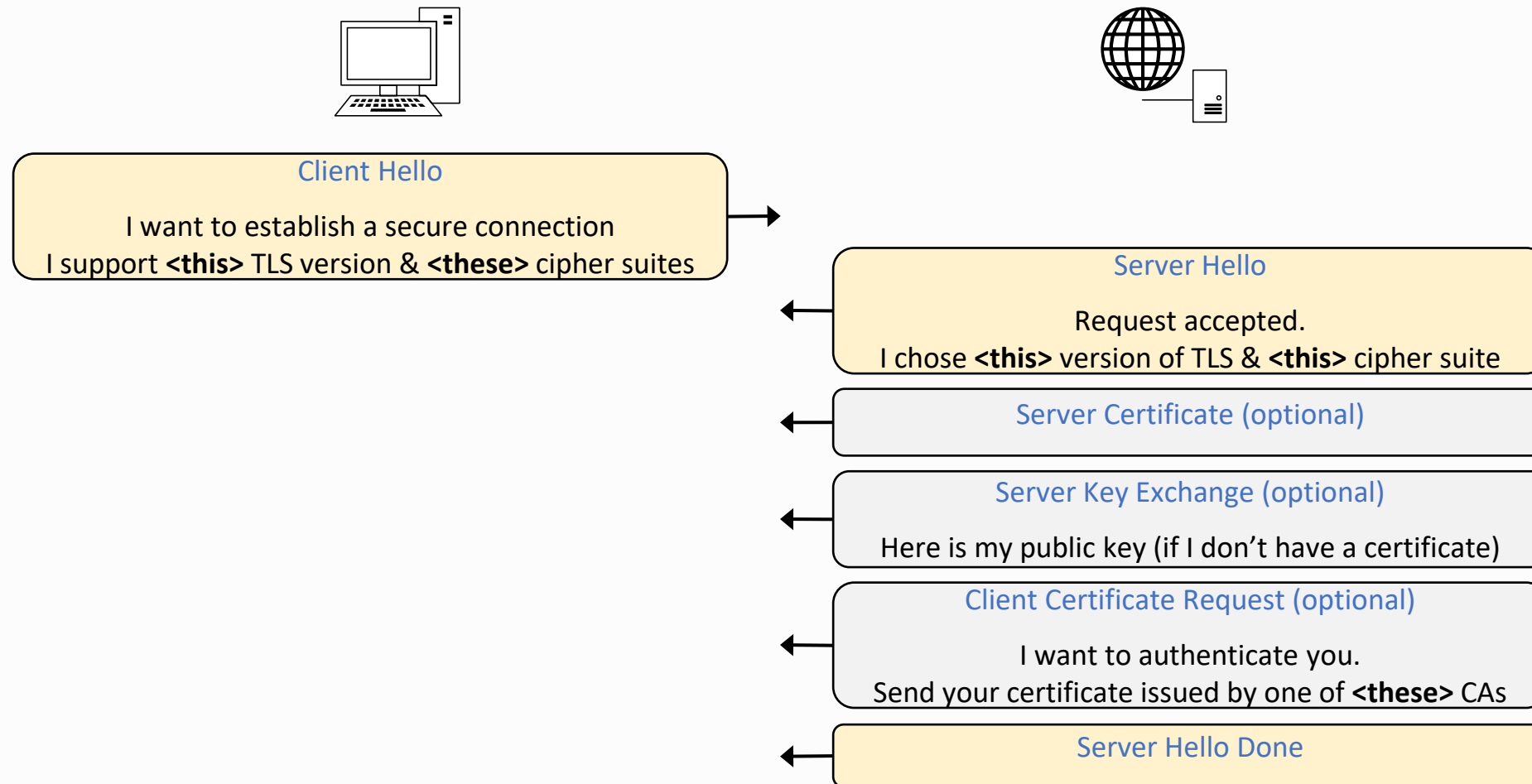
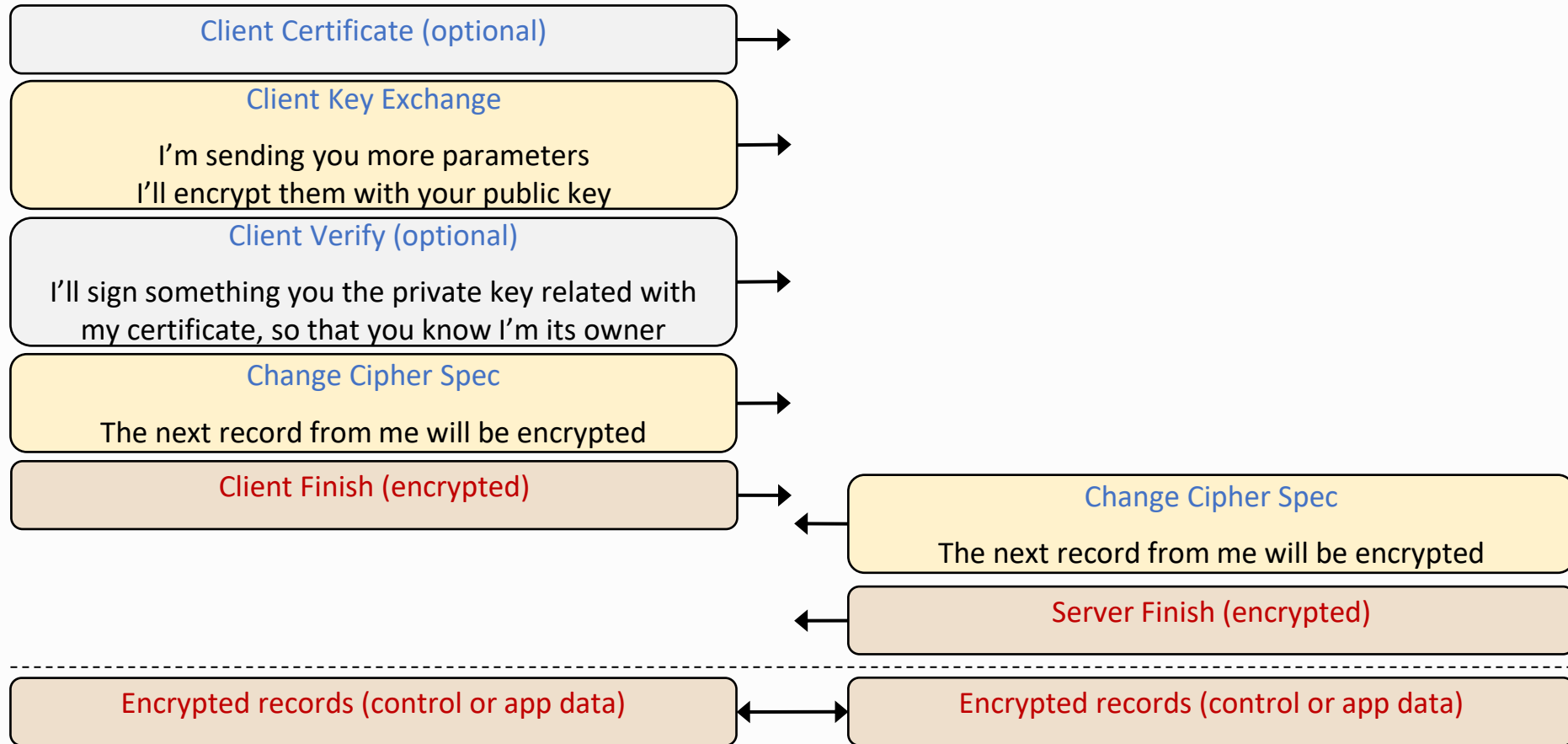
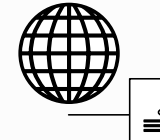
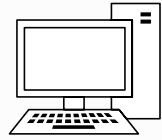


Image source: <https://hpbn.co/transport-layer-security-tls/>

TLS interaction diagrams (1st part)



TLS interaction diagrams (2nd part)



TLS Ciphersuites

- ▷ If a server supports a single algorithm, it not expected for all clients to also support it
 - ♦ More powerful/limited, older/newer
- ▷ The Ciphersuite concept allows the negotiation of mechanisms between client and server
 - ♦ Both send their supported ciphersuites, and select one they both share
 - ♦ TLS v1.3: O servidor escolhe
- ▷ Example: ECDHE-RSA-AES128-GCM-SHA256
- ▷ Format:
 - ♦ Key negotiation algorithm: ECDHE
 - ♦ Authentication algorithm: RSA
 - ♦ Cifra algorithm, and cipher mode: AES-128 GCM
 - ♦ Integrity control algorithm: SHA256

SSH (Secure Shell, RFC 4251)

▷ Alternative to telnet/rlogin protocols/applications

- ♦ Manages secure consoles over TCP/IP
- ♦ Initially conceived to replace telnet
- ♦ Actually used for other applications
 - Secure execution of remote commands (rsh/rexec)
 - Secure copy of contents between machines (rcp)
 - Secure FTP (sftp)
 - Creation of arbitrary secure tunnels (inbound/outbound/dynamic)

▷ Security mechanisms

- ♦ Communication confidentiality and integrity
 - Key distribution
- ♦ Authentication of communication endpoints
 - Servers / machines
 - Client users
 - Both with different techniques

SSH authentication mechanisms

▷ Server: with asymmetric keys pair

- ♦ Inline public key distribution
 - Not certified!
- ♦ Clients cache previously used public keys
 - Caching should occur in a trustworthy environment
 - Update of a server's key raises a problem to its usual clients

▷ Client users: configurable

- ♦ Username + password
 - By default
- ♦ Username + private key
 - Upload of public key in advance to the server

Single Sign-On (SSO)

- ▷ Unique, centralized authentication for a set of federated services
 - ♦ The identity of a client, upon authentication, is given to all federated services
 - ♦ The identity attributes given to each service may vary
 - ♦ The authenticator is called **Identity Provider (IdP)**
- ▷ Examples
 - ♦ SSO authentication @ UA
 - Performed by a central IdP (idp.ua.pt)
 - The identity attributes are securely conveyed to the service accessed by the user

Authentication metaprotocols

- ▷ Generic authentication protocols that encapsulate other authentication protocols
- ▷ Examples
 - ◆ EAP (Extensible Authentication Protocol)
 - Used in 802.1X (WiFi, enterprise mode)
 - e.g. PEAP (Protected EAP) and EAP-TLS run over EAP
 - ◆ ISAKMP(Internet Security Association and Key Management Protocol)
 - Formerly used in IPSec
 - e.g. IKE v1 (Internet Key Exchange) runs over ISAKMP

Authentication services

- ▶ Trusted third parties (TTP) used for authentication
 - ◆ But often combined with other related functionalities
- ▶ AAA services
 - ◆ Authentication, Authorization and Accounting
 - ◆ e.g. RADIUS

Key distribution services

▷ Services that distribute a shared key for authenticated entities

- ◆ That key can then be used by those entities to protect their communication and ensure source authentication

▷ Examples

- ◆ 802.1X (Wi-Fi, enterprise mode)
- ◆ Kerberos

