

# Authentication protocols



© André Zúquete /  
João Paulo Barraca

Security

1

## 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
  - ♦ ...



© André Zúquete /  
João Paulo Barraca

Security

2

## 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

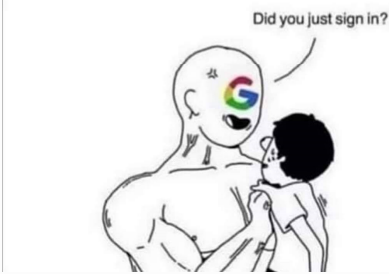
- ▷ Something we know
  - ♦ A secret memorized (or written down...) by Joe
- ▷ Something we have
  - ♦ An object/token solely held by Joe
- ▷ Something we are
  - ♦ Joe's Biometry
- ▷ Multi-factor authentication
  - ♦ Join or consecutive use of different proof types



## Multi-factor verification jokes

me: \*enters password correctly on new device\*

google:



© André Zúquete /  
João Paulo Barraca

Security

5

## 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



© André Zúquete /  
João Paulo Barraca

Security

6

## 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



© André Zúquete /  
João Paulo Barraca

Security

7

## 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



© André Zúquete /  
João Paulo Barraca

Security

8

## 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)



© André Zúquete /  
João Paulo Barraca

Security

9

## 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



© André Zúquete /  
João Paulo Barraca

Security

10

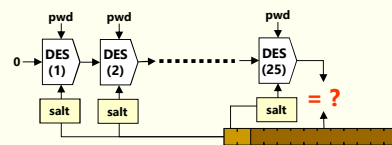
## Authentication of people: Direct approach w/ known password

- ▷ 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

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)



© André Zúquete /  
João Paulo Barraca

Security

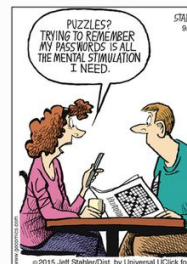
## Authentication of people: Direct approach w/ known password

### ▷ Advantage

- ♦ Simplicity!

### ▷ 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 15 2019 by Splashdata

- 1 - 123456
- 2 - 123456789
- 3 - qwerty
- 4 - password
- 5 - 1234567
- 6 - 12345678
- 7 - 12345
- 8 - iloveyou
- 9 - 111111
- 10 - 123123
- 11 - abc123
- 12 - qwerty123
- 13 - 1q2w3e4r
- 14 - admin
- 15 - qwertyuiop

source: <https://www.teampassword.com/blog/top-50-worst-passwords-of-2019>  
 Image <https://www.pinterest.com/networkboxusa/it-humor>



© André Zúquete /  
João Paulo Barraca

Security

12

## Password selection jokes

Someone figured out my PASSWORD  
Now I have to rename my dog.

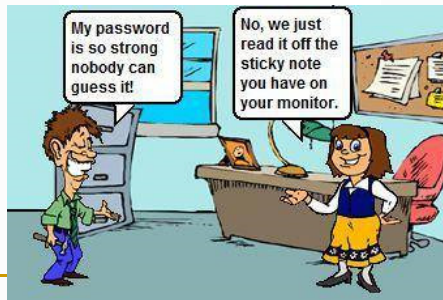
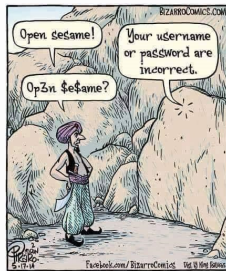


Dear IT,  
the more "secure" you try to make our passwords by making them impossible to remember, the more likely I am to save them all in a big word doc named "Passwords"

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.



© André Zúquete /  
João Paulo Barraca

Security

13

## Password bloopers



© André Zúquete /  
João Paulo Barraca

Security

14

## 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



## Biometrics: advantages

- ▷ Convenient: people do not need to use memory
  - ♦ Just be their self
- ▷ People cannot chose weak passwords
  - ♦ In fact, they don't chose 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



© André Zúquete /  
João Paulo Barraca

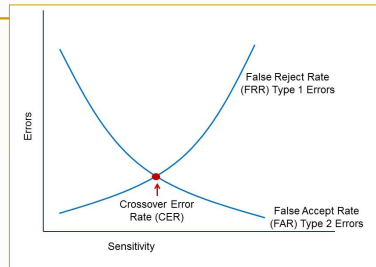
Image source: <https://biometrics.maignet.org/types/tongue.htm>

Security

17

## 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



© André Zúquete /  
João Paulo Barraca

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

Security

18

## 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
- ▷ 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
- ▷ 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



## 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)



© André Zúquete /  
João Paulo Barraca

Security

23

## RSA SecurID

- ▷ OTP-based authentication
  - ♦ A user combines their User ID with the current token number  
OTP = User ID, 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



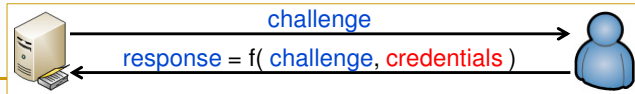
© André Zúquete /  
João Paulo Barraca

Security

24

## 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



© André Zúquete /  
João Paulo Barraca

Security

25

## 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)



© André Zúquete /  
João Paulo Barraca

Security

26

## 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



© André Zúquete /  
João Paulo Barraca

Security

27

## 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

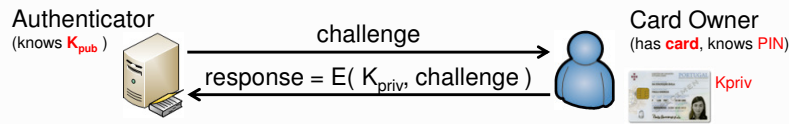


© André Zúquete /  
João Paulo Barraca

Security

28

## 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



© André Zúquete /  
João Paulo Barraca

Security

29

## 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)



© André Zúquete /  
João Paulo Barraca

Security

30

## 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<sub>A</sub>

U → A: C<sub>U</sub>, R1

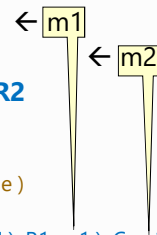
A → U: OK/not OK, R2

$R1 = DES_{PH}(C)$

$C = SHA(C_U, C_A, username)$

$PH = MD4(password)$

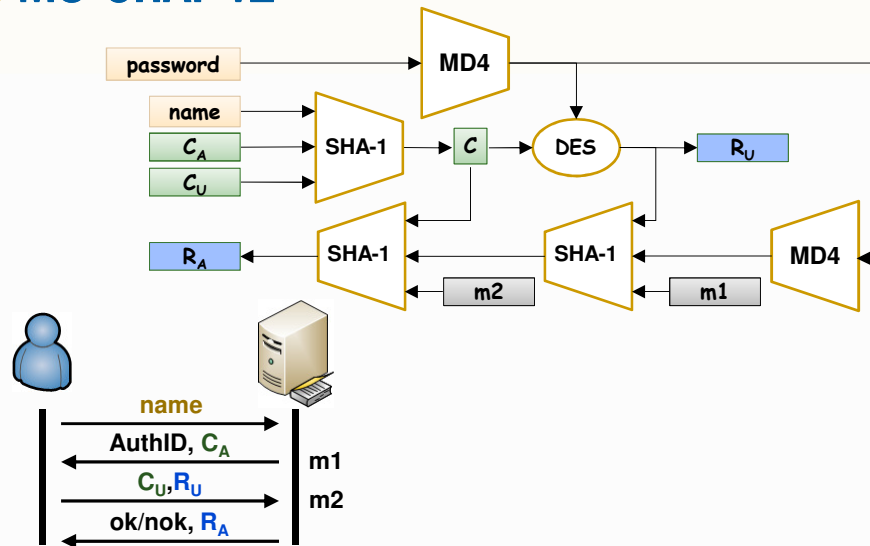
$R2 = SHA(SHA(MD4(PH), R1, m1), C, m2)$



- Mutual authentication
- Passwords can be updated



## MS-CHAP v2



© André Zúquete /  
João Paulo Barraca

Security

35

## 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



© André Zúquete /  
João Paulo Barraca

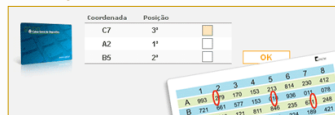
Security

36

# Authentication of people: OTPs selected from shared data

- ▷ 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 3070 3a7e 054 0f8e e9ed 112 7e93 8888 169 h4p8 2a7e 224 M40a w4d1
001 e47e 8791 037 608e 679e 113 d6e6 85a3 169 M47V e47e 225 04a8 8a7e
002 837a 8a82 038 608e 679e 114 6fde 2381 170 8071 /a4d 226 47e4 7393
003 0007 8e74 039 a407 154e 115 1f7e 85a3 171 8a7e 1a7e 227 7e7e 8c00
004 4a07 7393 040 8a7e 1a7e 116 8280 8071 172 1a7e 8a7e 228 47e4 7393
005 8884 8884 041 1a7e 1a7e 117 8071 8a7e 173 4a07 1a7e 229 8a7e 1a7e
006 8e74 7393 042 8a7e 1a7e 118 8a7e 1a7e 174 8a7e 1a7e 230 4a07 1a7e
007 8884 8884 043 8a7e 1a7e 119 8a7e 1a7e 175 8a7e 1a7e 231 8a7e 1a7e
008 8a7e 1a7e 044 8a7e 1a7e 120 8a7e 1a7e 176 8a7e 1a7e 232 8a7e 1a7e
009 8a7e 1a7e 045 8a7e 1a7e 121 8a7e 1a7e 177 8a7e 1a7e 233 8a7e 1a7e
010 8a7e 1a7e 046 8a7e 1a7e 122 8a7e 1a7e 178 8a7e 1a7e 234 8a7e 1a7e
011 8a7e 1a7e 047 8a7e 1a7e 123 8a7e 1a7e 179 8a7e 1a7e 235 8a7e 1a7e
012 8a7e 1a7e 048 8a7e 1a7e 124 8a7e 1a7e 180 8a7e 1a7e 236 8a7e 1a7e
013 8a7e 1a7e 049 8a7e 1a7e 125 8a7e 1a7e 181 8a7e 1a7e 237 8a7e 1a7e
014 8a7e 1a7e 050 8a7e 1a7e 126 8a7e 1a7e 182 8a7e 1a7e 238 8a7e 1a7e
015 8a7e 1a7e 051 8a7e 1a7e 127 8a7e 1a7e 183 8a7e 1a7e 239 8a7e 1a7e
016 8a7e 1a7e 052 8a7e 1a7e 128 8a7e 1a7e 184 8a7e 1a7e 240 8a7e 1a7e
017 8a7e 1a7e 053 8a7e 1a7e 129 8a7e 1a7e 185 8a7e 1a7e 241 8a7e 1a7e
018 8a7e 1a7e 054 8a7e 1a7e 130 8a7e 1a7e 186 8a7e 1a7e 242 8a7e 1a7e
019 8a7e 1a7e 055 8a7e 1a7e 131 8a7e 1a7e 187 8a7e 1a7e 243 8a7e 1a7e
020 8a7e 1a7e 056 8a7e 1a7e 132 8a7e 1a7e 188 8a7e 1a7e 244 8a7e 1a7e
021 8a7e 1a7e 057 8a7e 1a7e 133 8a7e 1a7e 189 8a7e 1a7e 245 8a7e 1a7e
022 8a7e 1a7e 058 8a7e 1a7e 134 8a7e 1a7e 190 8a7e 1a7e 246 8a7e 1a7e
023 8a7e 1a7e 059 8a7e 1a7e 135 8a7e 1a7e 191 8a7e 1a7e 247 8a7e 1a7e
024 8a7e 1a7e 060 8a7e 1a7e 136 8a7e 1a7e 192 8a7e 1a7e 248 8a7e 1a7e
025 8a7e 1a7e 061 8a7e 1a7e 137 8a7e 1a7e 193 8a7e 1a7e 249 8a7e 1a7e
026 8a7e 1a7e 062 8a7e 1a7e 138 8a7e 1a7e 194 8a7e 1a7e 250 8a7e 1a7e
027 8a7e 1a7e 063 8a7e 1a7e 139 8a7e 1a7e 195 8a7e 1a7e 251 8a7e 1a7e
028 8a7e 1a7e 064 8a7e 1a7e 140 8a7e 1a7e 196 8a7e 1a7e 252 8a7e 1a7e
029 8a7e 1a7e 065 8a7e 1a7e 141 8a7e 1a7e 197 8a7e 1a7e 253 8a7e 1a7e
030 8a7e 1a7e 066 8a7e 1a7e 142 8a7e 1a7e 198 8a7e 1a7e 254 8a7e 1a7e
031 8a7e 1a7e 067 8a7e 1a7e 143 8a7e 1a7e 199 8a7e 1a7e 255 8a7e 1a7e
032 8a7e 1a7e 068 8a7e 1a7e 144 8a7e 1a7e 200 8a7e 1a7e 256 8a7e 1a7e
033 8a7e 1a7e 069 8a7e 1a7e 145 8a7e 1a7e 201 8a7e 1a7e 257 8a7e 1a7e
034 8a7e 1a7e 070 8a7e 1a7e 146 8a7e 1a7e 202 8a7e 1a7e 258 8a7e 1a7e
035 8a7e 1a7e 071 8a7e 1a7e 147 8a7e 1a7e 203 8a7e 1a7e 259 8a7e 1a7e
036 8a7e 1a7e 072 8a7e 1a7e 148 8a7e 1a7e 204 8a7e 1a7e 260 8a7e 1a7e
037 8a7e 1a7e 073 8a7e 1a7e 149 8a7e 1a7e 205 8a7e 1a7e 261 8a7e 1a7e
038 8a7e 1a7e 074 8a7e 1a7e 150 8a7e 1a7e 206 8a7e 1a7e 262 8a7e 1a7e
039 8a7e 1a7e 075 8a7e 1a7e 151 8a7e 1a7e 207 8a7e 1a7e 263 8a7e 1a7e
040 8a7e 1a7e 076 8a7e 1a7e 152 8a7e 1a7e 208 8a7e 1a7e 264 8a7e 1a7e
041 8a7e 1a7e 077 8a7e 1a7e 153 8a7e 1a7e 209 8a7e 1a7e 265 8a7e 1a7e
042 8a7e 1a7e 078 8a7e 1a7e 154 8a7e 1a7e 210 8a7e 1a7e 266 8a7e 1a7e
043 8a7e 1a7e 079 8a7e 1a7e 155 8a7e 1a7e 211 8a7e 1a7e 267 8a7e 1a7e
044 8a7e 1a7e 080 8a7e 1a7e 156 8a7e 1a7e 212 8a7e 1a7e 268 8a7e 1a7e
045 8a7e 1a7e 081 8a7e 1a7e 157 8a7e 1a7e 213 8a7e 1a7e 269 8a7e 1a7e
046 8a7e 1a7e 082 8a7e 1a7e 158 8a7e 1a7e 214 8a7e 1a7e 270 8a7e 1a7e
047 8a7e 1a7e 083 8a7e 1a7e 159 8a7e 1a7e 215 8a7e 1a7e 271 8a7e 1a7e
048 8a7e 1a7e 084 8a7e 1a7e 160 8a7e 1a7e 216 8a7e 1a7e 272 8a7e 1a7e
049 8a7e 1a7e 085 8a7e 1a7e 161 8a7e 1a7e 217 8a7e 1a7e 273 8a7e 1a7e
050 8a7e 1a7e 086 8a7e 1a7e 162 8a7e 1a7e 218 8a7e 1a7e 274 8a7e 1a7e
051 8a7e 1a7e 087 8a7e 1a7e 163 8a7e 1a7e 219 8a7e 1a7e 275 8a7e 1a7e
052 8a7e 1a7e 088 8a7e 1a7e 164 8a7e 1a7e 220 8a7e 1a7e 276 8a7e 1a7e
053 8a7e 1a7e 089 8a7e 1a7e 165 8a7e 1a7e 221 8a7e 1a7e 277 8a7e 1a7e
054 8a7e 1a7e 090 8a7e 1a7e 166 8a7e 1a7e 222 8a7e 1a7e 278 8a7e 1a7e
055 8a7e 1a7e 091 8a7e 1a7e 167 8a7e 1a7e 223 8a7e 1a7e 279 8a7e 1a7e

```

# 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
  - An seed value for the 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:  
$$OTP_n = h^n(\text{seed}, \text{pwd}), \text{ where } h = \text{MD4}$$
  - ♦ Some S/Key versions also use MD5 or SHA-1
- ▷ The authenticator stores seed, n and  $OTP_n$  as authentication credentials



© André Zúquete /  
João Paulo Barraca

Security

39

## 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
  - ♦  $\text{result} = OPT_{\text{index}-1}$
- ▷ The authenticator computes h(result) and compares the result with the stored  $OPT_{\text{index}}$ 
  - ♦ If they match, the authentication succeeds
  - ♦ Upon success, stores the recently used index & OTP
    - index-1 and  $OPT_{\text{index}-1}$



© André Zúquete /  
João Paulo Barraca

Security

40

## 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

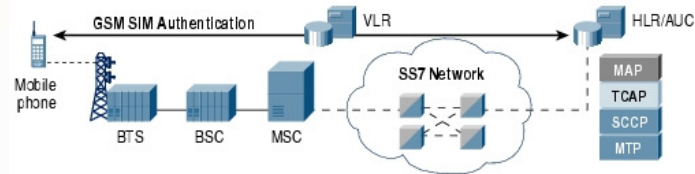
- ♦ Robust against dictionary attacks
- ♦ Requires some token to store the key

### ▷ 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

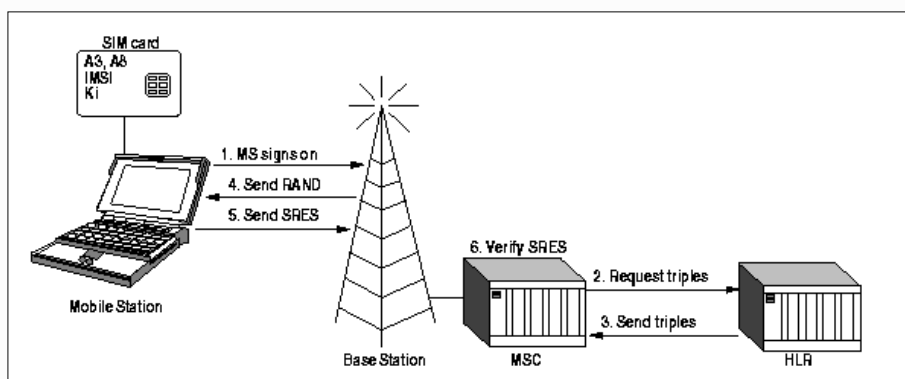


© André Zúquete /  
João Paulo Barraca

Security

43

## GSM: mobile station authentication



© André Zúquete /  
João Paulo Barraca

Security

44

## GSM: mobile station authentication

- ▷ MSC fetches trio from HLR
  - ♦ **RAND, SRES, Kc**
  - ♦ In fact more than one are requested
- ▷ HLR generates RAND and corresponding trio using subscriber's Ki
  - ♦ **RAND**, random value (128 bits)
  - ♦ **SRES = A3 (Ki, RAND)** (32 bits)
  - ♦ **Kc = A8 (Ki, RAND)** (64 bits)
- ▷ Usually operators use COMP128 for A3/A8
  - ♦ Recommended by the GSM Consortium
  - ♦ **[SRES, Kc] = COMP128 (Ki, 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 5246)

### ▷ 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)

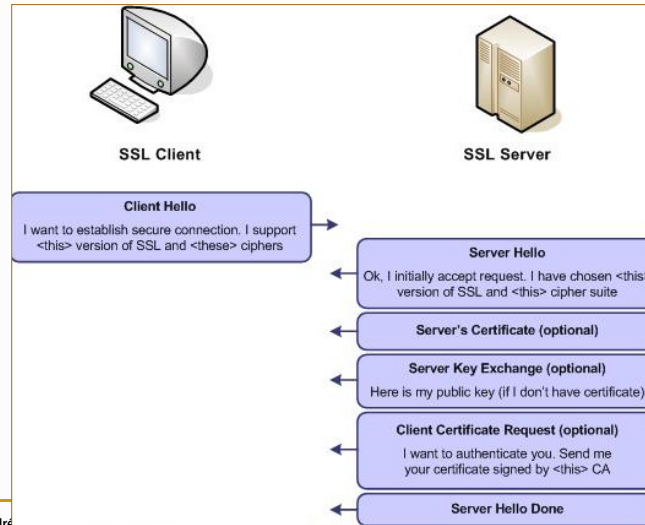
### ▷ Security mechanisms

- ♦ Communication confidentiality and integrity
  - Key distribution
- ♦ Authentication of communication endpoints
  - Servers (or, more frequently, services)
  - Client users
  - Both with asymmetric key pairs and certified public keys





## SSL/TLS interaction diagrams (1<sup>st</sup> part)

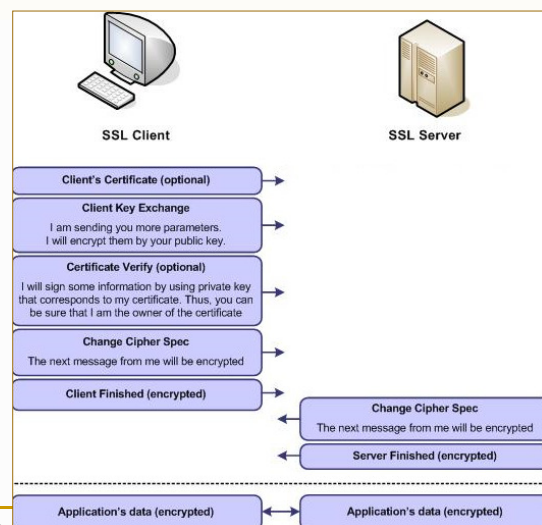


© André Zúquete /  
João Paulo Barraca

Security

49

## SSL/TLS interaction diagrams (2<sup>nd</sup> part)



© André Zúquete /  
João Paulo Barraca

Security

50

# 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



## Authentication metaprotocols

- ▷ Generic authentication protocols that encapsulate other specific authentication protocols
- ▷ Examples
  - ♦ EAP (Extensible Authentication Protocol)
    - Used in 802.11 (Wi-Fi)
  - ♦ ISAKMP (Internet Security Association and Key Management Protocol)
    - Used in IPSec



## 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 at UA
    - Performed by a central IdP (idp.ua.pt)
    - The identity attributes are securely conveyed to the service accessed by the user



## 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

