

Authentication

LASEC

Who are you?

P. Steiner, The New Yorker



"On the Internet, nobody knows you're a dog."

- ◆ On the internet, nobody knows you're a dog

Authentication

- ◆ Can't secure your data if you can't identify and authenticate your users
- ◆ Before obtaining privileges, users must indicate who they are (identification) and prove it (authentication)
- ◆ We can authenticate a user with the help of
 - something she knows (passwords)
 - something she possesses (tokens)
 - something she is (biometry)

Biometrics

Biométrie: généralités

- Biometrie: measuring human beings

- ◆ Morphology:

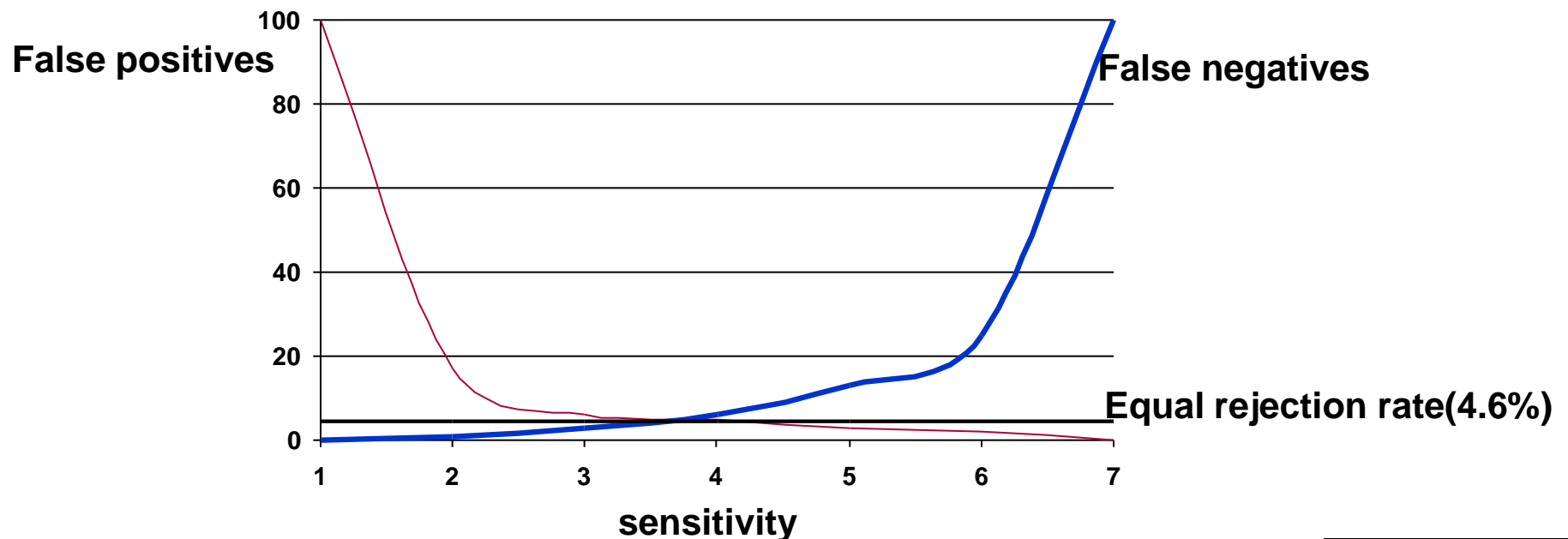
- Finger print
- Shape of hand
- Shape of head
- Iris
- Retina
- Shape of ear
- DNA

- ◆ Behavior:

- Dynamics of signature (speed, pressure, direction)
- Voice
- Keyboard usage

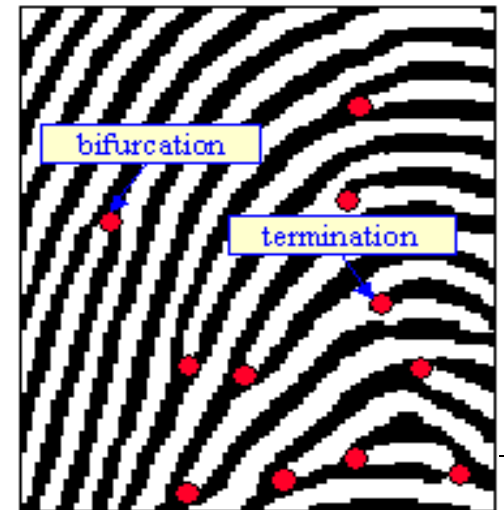
Biometrics: rejection rate

- ◆ Can't have a perfect biometric system
 - If it is too sensitive it generates false negatives
 - If not sensitive enough, too many false positives
- ◆ The quality of a biometric system is defined by its equal rejection rate



Example: fingerprints

- ◆ The fingerprint is scanned
- ◆ Interesting points (minutiae) are extracted (x and y coordinates plus direction)
- ◆ The list of minutiae is send to a server
- ◆ The liste is compared with a previously stored list
- ◆ The number of matching minutiae tells how close the match is



Biometrics: discussion

- ◆ Information is never identical
 - Not possible to hash
 - Risk of theft
 - ◆ Some sensors never reveal the information (e.g smart phones)
 - Can't change a stolen finger!
- ◆ Some sensors can be fooled or replaced
- ◆ Ideal applications
 - Supervised Physical access control

Tow factor authentication

Token: something you own

- ◆ Bingo cards (can be copied)

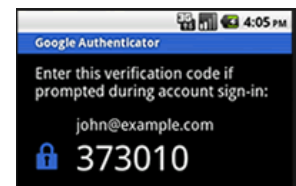
- Proof that the user owns the card (or a copy)
- Grid-type list: no off-line attacks



B I N G O				
7	25	44	57	62
15	22	40	50	70
11	30	FREE SPACE	46	74
2	28	37	55	68
10	27	39	59	75

- ◆ One Time Password (OTP) token:

- Displays a 6 digit number that changes every minute or every use
- Proof that the user owns the token



- ◆ Mobile phone

- Confirmation code sent by SMS



- ◆ Small calculator

- The user enters a challenge (displayed on the screen) and the calculator displays a response to give to the server
- Proof that the owner owns the calculator AND that he has read the challenge



Confirmation of transactions

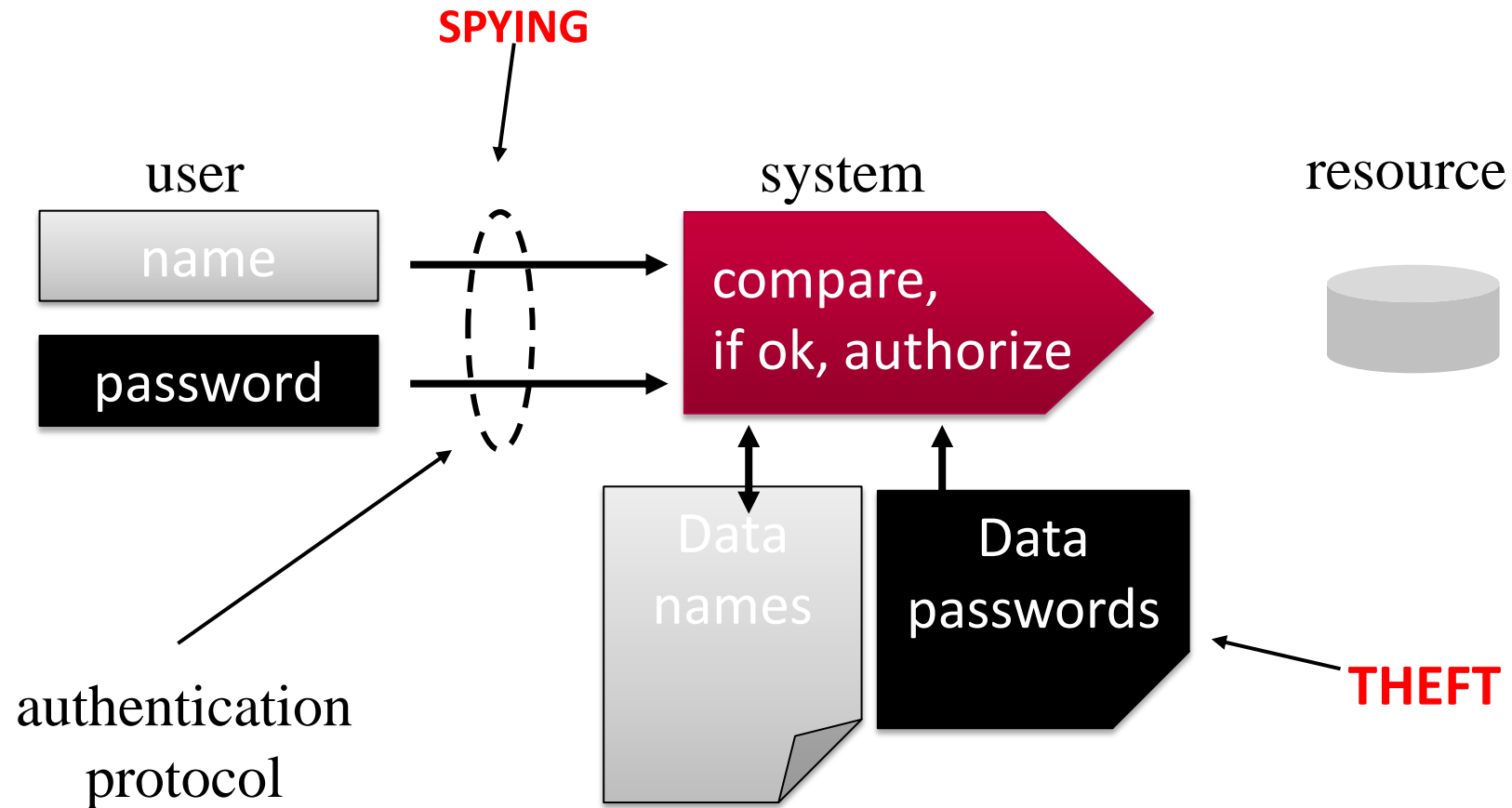
- ◆ Some types of two factor authentications can be used to confirm a payment:
- ◆ To confirm that you want to pay to a certain account:
 - The bank can confirm the account number by SMS with a validation code that you must type if you agree
 - You can type the account number into the calculator and get a validation code for the payment.
- ◆ OTP tokens and bingo cards can not be used to confirm a payment.
 - The codes they generate are not related to the payment.

Passwords

User name and password

- ◆ We use the username for identification, and the password for authentication
- ◆ To avoid authenticating ourselves for each operation, we use centralized authentication systems (operating system, domain controller, authentication server)
- ◆ Once it has authenticated the user, the system assigns him privileges giving access to certain resources

Classical model: risks



Passwords storage

- ◆ Passwords are never stored as such. The risk of theft would be too high
- ◆ Instead of passwords, we store a hash
- ◆ The hash must be unique and irreversible
- ◆ By comparing the hash of the password provided with the stored hash, we can know if both have been created using the same password

Hashing for different OSes

- ◆ Ubuntu Linux (Yakketi Yak):
 - SHA-512, 5000 iterations, 48 bits of salt
- ◆ OS-X 10.8 Mountain Lion and following
 - SHA-512, variable number of iterations (0.1s), 256 bits of salt
- ◆ Windows Vista and later (NTLM hash)
 - MD4, 1 iteration, no salt

Cracking passwords

- ◆ The cracker must first obtain a copy of the password's hashes
- ◆ Since he cannot inverse the hashes he will
 - guess passwords (dictionary) or generate random passwords (brute force)
 - generate the hashes of those words
 - compare them with the stolen hashes to see if he guessed right
- ◆ If users had passwords that could not be guessed, they would never be cracked!

Cracking tools

- ◆ Cracking programs generate hashes from words using a dictionary or by enumerating all the character combinations
- ◆ A powerful PC can generate a tens of millions of hashes per second depending on the type of has
- ◆ Graphic cards (GPU) can even generate up to billions of hashes per seconds
- ◆ Windows password cracking: Ophcrack, Hashcat
- ◆ Unix cracking: John the ripper
- ◆ GPU cracking: Hashcat

Time-Memory Trade-Off

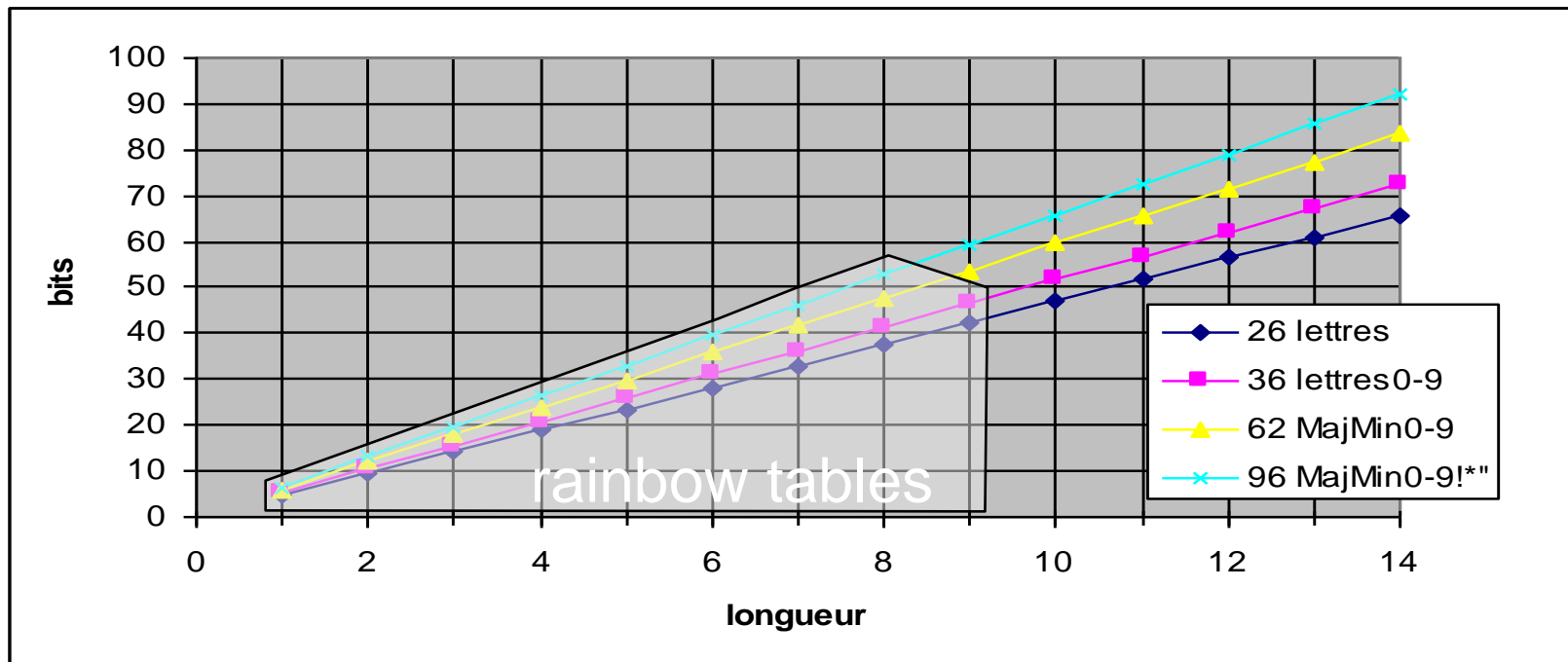
- ◆ If the hashes contain no salt, they can be generated in advance
- ◆ Using a trade-off technique only a fraction of the hashes needs to be stored. The others can be recreated with little effort during the cracking operation
- ◆ Examples: Ophcrack,
 - Ophcrack: 2.5 TB of tables, 60 seconds to crack any 8 character password (numbers, letters, 33 special chars)

Ophcrack



Password guidance

- ◆ Use complex passwords (at least 9 characters, mixed case, with numbers and special characters)
- ◆ Length matters.



Password guidance

- ◆ Use a password manager
- ◆ Good password don't need to be difficult to remember:
 - bobby@epfl.ch, ftp://cisco.com, Loupyes-tu?, 96.7Rhone-FM
 - N,rdr,jnrr! (Non, rien de rien, je ne regrette rien!)
 - correcthorsebatterystaple
 - ◆ https://imgs.xkcd.com/comics/password_strength.png