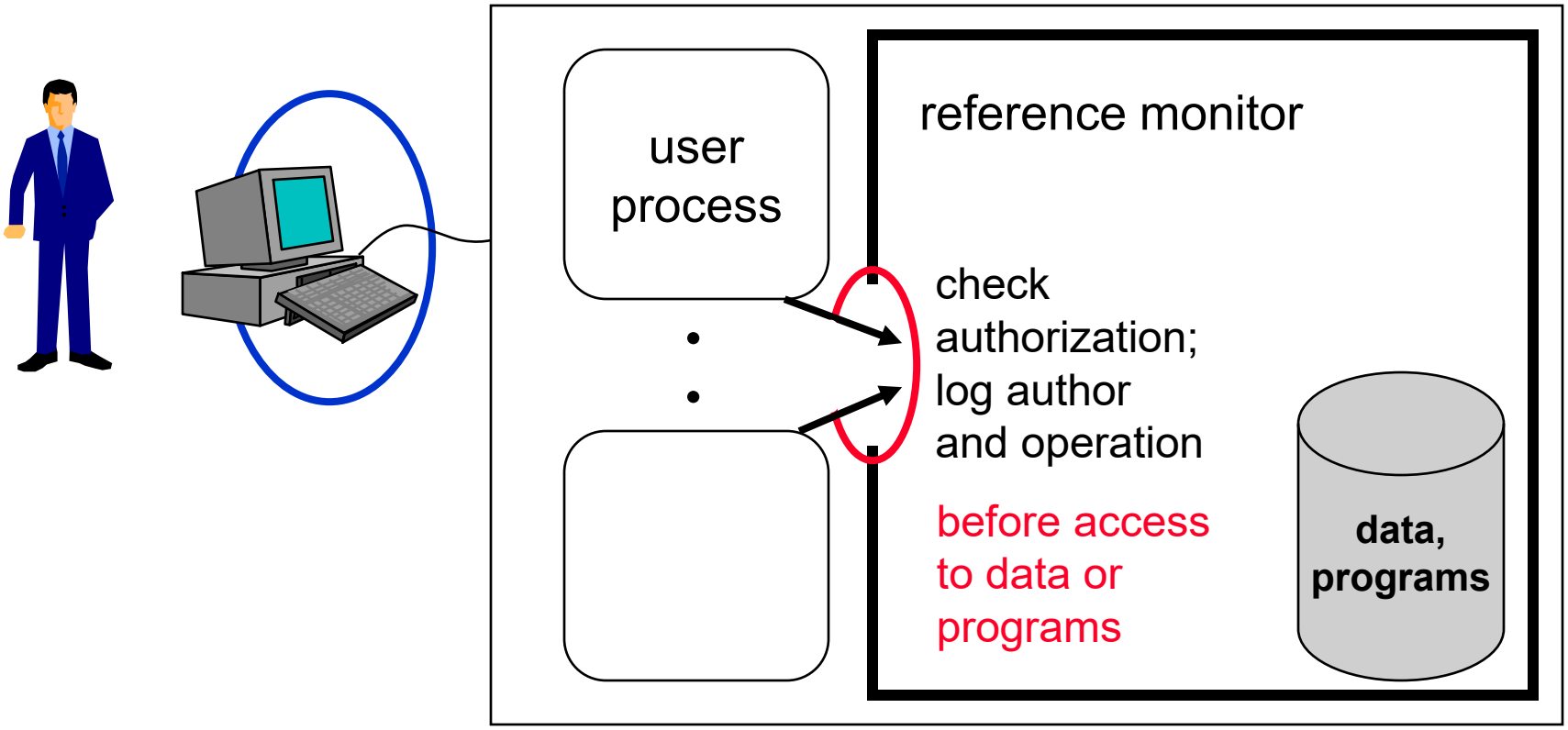
AUTHENTICATION

Slides are from Andreas Pfitzmann, Andreas Westfeld, Sebastian Clauß, Mike Bergmann and myself (Stefan Köpsell)

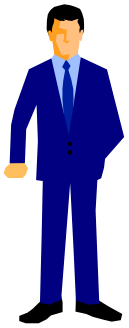
Why autentication: Admission and access control

Admission control communicate with authorized partners only



Access control subject can only exercise operations on objects if authorized.

Identification of human beings by IT-systems



What one *is*
- hand geometry
- finger print
- picture
- hand-written signature
- retina-pattern
- voice
- typing characteristics

has
- paper document
- metal key
- magnetic-strip card
- smart card (chip card)
- calculator

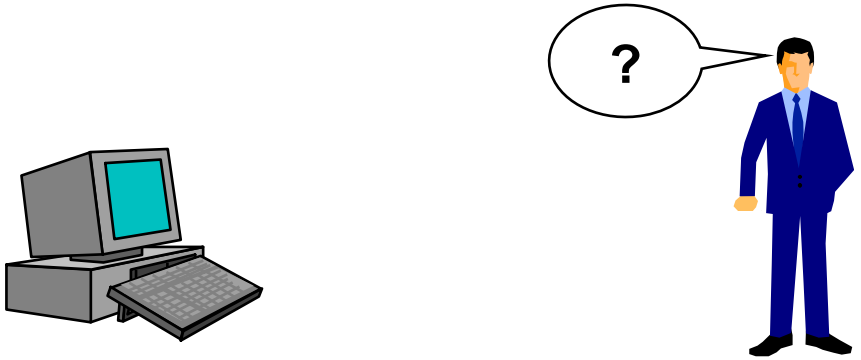
knows
- password, passphrase
- answers to questions
- calculation results for numbers

→ **eID-card**

New German eID Card

PIN protects access to chip

Identification of IT-systems by human beings

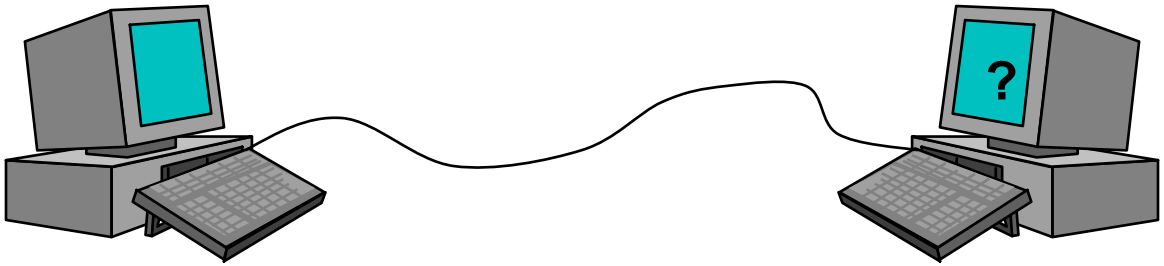


What it *is*
- casing
- seal, hologram
- pollution

knows
- password
- answers to questions
- calculation results for numbers

Where it *stands*

Identification of IT-systems by IT-systems



What it *knows*

- password
- answers to questions
- calculation results for numbers
- **cryptography**

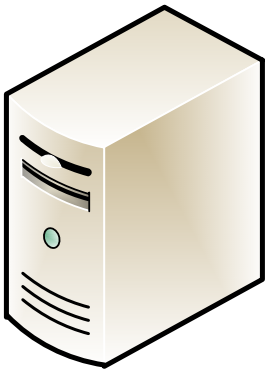
Wiring *from where*

Password based authentication

- Simple approach

User

(dog,bone)



Server

?

yes

grant access

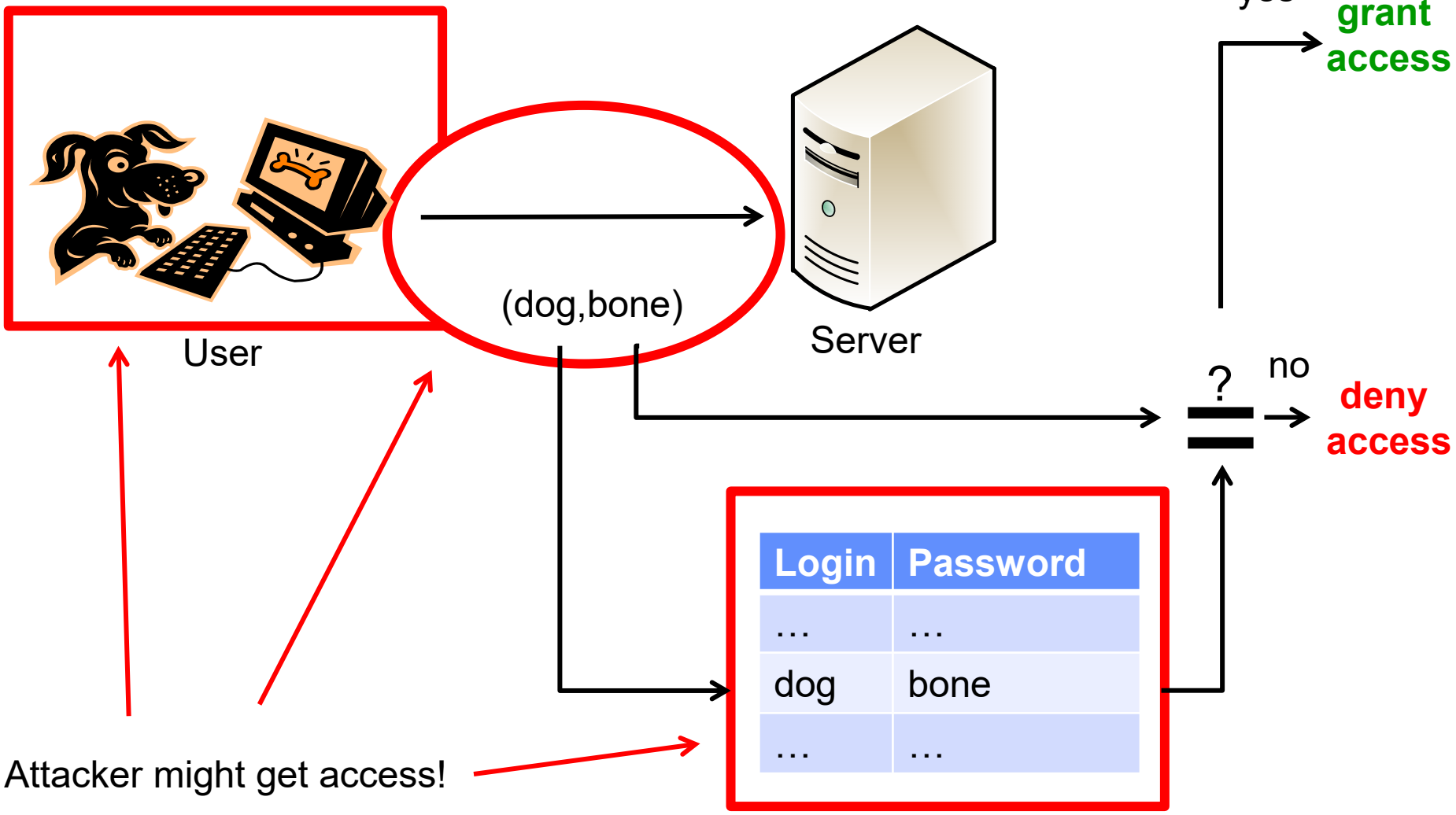
no

deny access

Login	Password
…	…
dog	bone
…	…

Password based authentication

- Simple approach – **security problems**

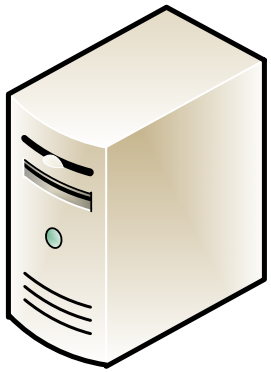


Password based authentication

- Enhanced approach using one way (hash) functions

User

(dog,bone)



Server

yes

grant access

calculates *h*(bone)

?

no

deny access

Login	Password
…	…
dog	***h*(bone)**
…	…

One-way functions – cryptographic hash functions

- One-way function *f:*
  - calculating $f(x)=y$ is easy
  - calculating $f^{-1}(y)=x$ is hard
    - computation / storage
  - open question: Do one-way functions exist?
- Cryptographic hash function *h*
  - might have different properties depending on the use case
  - collision resistance:
    - it is hard to find $h(y)=h(x)$ with $y \neq x$
    - note: *h* is usually not *collision free*, because $|h(x)| \ll |x|$
  - preimage resistance / one-way function / secrecy
    - given $h(x)$ it is hard to find x
  - second-preimage resistance / weak collision resistance / binding
    - given $h(x)$ it is hard to find $h(y)=h(x)$ with $y \neq x$
  - Note:
    - *h* is not necessarily a “random extractor”
    - only one of “secrecy” and “binding” can be information theoretic secure

Examples for cryptographic hash functions

- MD5
  - Message-Digest Algorithm
  - developed by Ronald Rivest (April 1992)
  - produces 128 bit hash values
  - can process arbitrary long inputs
  - **today MD5 is broken!**
- SHA-1
  - Secure Hash Standard
  - published 1993 as FIPS PUB 180 by US NIST
  - produces 160 bit hash values
  - **today SHA-1 is insecure!**
- SHA-2
  - set of hash functions, with hash values of 224, 256, 384, 512 bit
  - published 2001 as FIPS PUB 180-2 by NIST (current version: FIPS 180-4)
  - **SHA-2 hash functions are believed to be secure**
- SHA-3
  - result of the NIST Cryptographic Hash Algorithm Competition started November 2007
  - 3 selection rounds, 5 finalists
  - October 2012: Keccak is winner
  - FIPS 202: “SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions” (08/15)

MD5 Hash in the Wild

- United States Cyber Command (USCYBERCOM)
  - mission statement:= "*USCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyberspace and deny the same to our adversaries.*"

MD5 Hash in the Wild

mission statement:= "*USCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyber-space and deny the same to our adversaries.*"

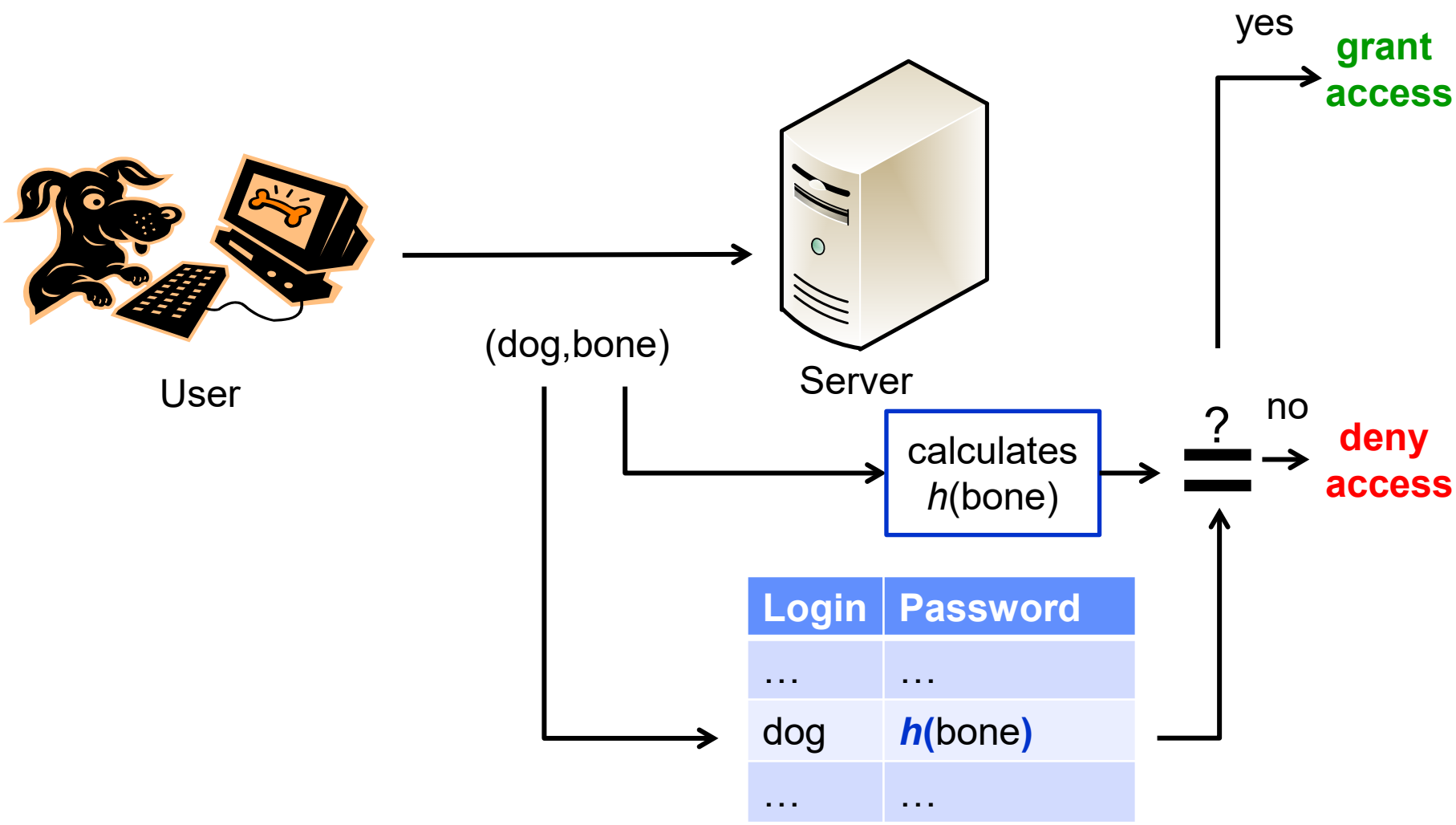
mission statement:= "*USCYBERCOM plans, coordinates, integrates, synchronizes and conducts activities to: direct the operations and defense of specified Department of Defense information networks and; prepare to, and when directed, conduct full spectrum military cyberspace operations in order to enable actions in all domains, ensure US/Allied freedom of action in cyber-space and deny the same to our adversaries.*"

MD5(mission statement)=
9ec4c12949a4f31474f299058ce2b22a

(Remember: MD5 is broken → find other interesting mission statements…)

Password based authentication

- Enhanced approach using one way (hash) functions

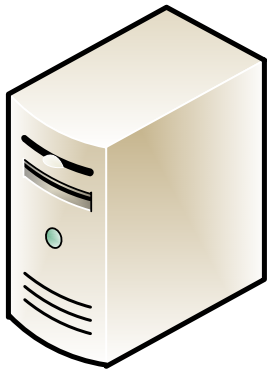


Password based authentication

- Enhanced approach using one way (hash) functions

User

(dog,bone)



Server

yes

grant access

calculates *h*(bone)

?

no

deny access

Login	Password
…	…
dog	*h*(bone)
…	…

Slightly reduced risk, if attacker gets access.

Remaining problems of password based authentication based one way functions

- Brute Force attack
  - function *h*() is public
  - value of *h*(*x*) is known to the attacker

  ➔ try all possible values for *x*

  Considerations:
    - usually >> 1 Mio. *h*(x)/s on ordinary hardware
    - assumption: password uses only small letters
    - password length = 8

Login	Password
…	…
dog	*h*(bone)
…	…

time needed: $\frac{26^8}{1\,000\,000 \cdot 60 \cdot 60} \approx 58\text{h}$

- first countermeasures (against remote attacks):
  - limit false attempts
- first password rules:
  - use a large alphabet (small and capitalised letters, numbers, specials)
  - use a long password

Remaining problems of password based authentication based one way functions

- first password rules:
  - use a large alphabet
    - (small, capitalised letters, numbers, specials)
    - time needed: $\frac{(26+26+10+30)^8}{1\,000\,000 \cdot 60 \cdot 60 \cdot 24 \cdot 365.25} \approx 162a$
  - use a long password

Login	Password
…	…
dog	*h*(bone)
…	…

- remaining possible attacks:
  - increase in computation power
    - distributed approach
    - GPU
    - Moore's law
  - pre-computation:
    - attacker creates lockup table
    - search time (example above):

      $\mathrm{ld}\left((26+26+10+30)^8\right) < 53$ comparisons

Remaining problems of password based authentication based one way functions

- remaining possible attack:
  - pre-computation
- countermeasure:
  - salt!
  - $h(x) \rightarrow h(\mathbf{salt},x)$
  - salt:
    - long (e.g. 128 bit) random value
    - some part is unique for the system (i.e. 104 bit)
    - some part is randomly chosen by the system for each entry in the password table (i.e. 24 bit)
      - NOT stored at the system
    - verification: iterate over all possible salt values

Login	Password
…	…
dog	*h*(bone)
…	…

➔ pre-computation has to be done *for each possible salt*

Hashfunktionen zum Speichern von Paßwörtern

übliche Anforderung an Hashfunktion:

- möglichst effiziente Verarbeitung großer Datenmengen

Problem:

- fördert Effizienz von Brute-Force-Angriffen

Spezialfall Paßwörter:

- Eingabe typsicherweise klein (<512 bit)
- gewisse Wartezeit für Login-Durchführung akzeptabel
  - ~ 1 Sekunde

  ➔ Paßwort-Hashfunktion muß nicht besonders effizient sein

Erschweren von Brute-Force-Angriffen:

- Paßwort-Hashfunktion darf nicht *effizient* implementierbar sein

Hashfunktionen zum Speichern von Paßwörtern

Paßwort-Hashfunktion darf nicht *effizient* implementierbar sein

- Software-Implementierungen:
  - Berücksichtigung aktueller CPU-Fähigkeiten
    - Multi-Core / Multi-Threaded
    - SIMD / Vector Extensions (AVX512)
    - Krypto-Erweiterungen (AES / SHA Befehle)
    - Cache-Größen (L1, L2, L3 Cache)
    - Branch Prediction
    - …
  - Berücksichtigung von verbreiteter „Spezial-Hardware“
    - Grafikkarten

- Hardware-Implementierungen
  - FPGA
  - spezial ASICs
    - Bitcoin-Mining

- zukunftsfähig
  - leichte Anpassung (Parametrisierung) an zukünftige (Hardware)-Verbesserungen

Hashfunktionen zum Speichern von Paßwörtern

Praktische Umsetzungen

- **bcrypt**
  - Niels Provos, David Mazières: „A Future-Adaptable Password Scheme“, USENIX, 1999
  - basiert auf Blowfish
    - symmetrische Blockchiffre

```
round_keys=EksBlowfishSetup(cost, salt, input) //
ineffizient!
hash="OrpheanBeholderScryDoubt" // 3 x  64-bit
blocks
loop (64)
   {
      hash=Blowfish_ECB(round_keys,hash)
   }
```

  - guter Schutz gegen Software / GPU basierte Brute-Force-Angriffe
  - schlechter Schutz gegen ASIC-basierte Brute-Force-Angriffe

Hashfunktionen zum Speichern von Paßwörtern

Praktische Umsetzungen

- **`PBKDF2`** (Password-Based Key Derivation Function 2)
  - ursprünglich in RSA Laboratories PKCS#5-Standard
    - gedacht für Ableitung symmetrischer Schlüssel aus Paßwort
    - übernommen als RFC 2898
    - anerkannt vom NIST in SP 800-132 (Dezember 2010)
  - *h*=`PBKDF2` (passwd, salt, iterations)

```
{
  h=Hash(passwd||salt||iterations);
  loop(iterations-1)
    {
      h=h XOR Hash(passwd||h);
    }
  return h;
}
```

  - guter Schutz gegen CPU-basierte Software Brute-Force-Angriffe
  - schlechter Schutz gegen ASIC/FPGA/GPU-basierte Brute-Force-Angriffe

PBKDF2— Speicherverbrauch erhöhen

- *h*=`PBKDF2` (passwd, salt, iterations)

```
{
  i=0
  h[i++]=Hash(passwd||salt||iterations);
  loop(iterations-1)
    {
       h[i+1]=h[i] XOR Hash(passwd||h[i]);
       i++;
    }
  sort(h[]);
  i=0;
  loop(iterations/2)
    {
      res=res + h[i] * h[i+1];
      i+=2;
    }
  return res;
}
```

Warning: Hand crafted crypto! - unverified -

Hashfunktionen zum Speichern von Paßwörtern

- Praktische Umsetzungen
  - **`scrypt`**
    - Colin Percival: “Stronger Key Derivation Via Sequential Memory-Hard Functions”, 2009
    - veröffentlicht in RFC 7914
    - Ziel: Hardware-Implementierung kostspielig machen
    - Lösungsansatz:
      - Speicherbedarf für effiziente Umsetzung stark erhöhen
    - Umsetzung:
      - Algorithmus benötigt großen Vektor pseudozufälliger Elemente, auf die in pseudozufälliger Reihenfolge zugegriffen wird
    - Kosten parameterisierbar

Hashfunktionen zum Speichern von Paßwörtern

- Praktische Umsetzungen
  - **`Argon2`**
    - Alex Biryukov, Daniel Dinu, Dmitry Khovratovich: “Argon2: the memory-hard function for password hashing and other applications”, 2015
    - Sieger der Password Hashing Competition (PHC)
      - Community getriebener Wettbewerb (2013-2015)
    - ähnliche Ziele / Überlegungen wie scrypt
    - bisher wenige bekannte Kryptanalyse

Remaining problems of password based authentication based one way functions

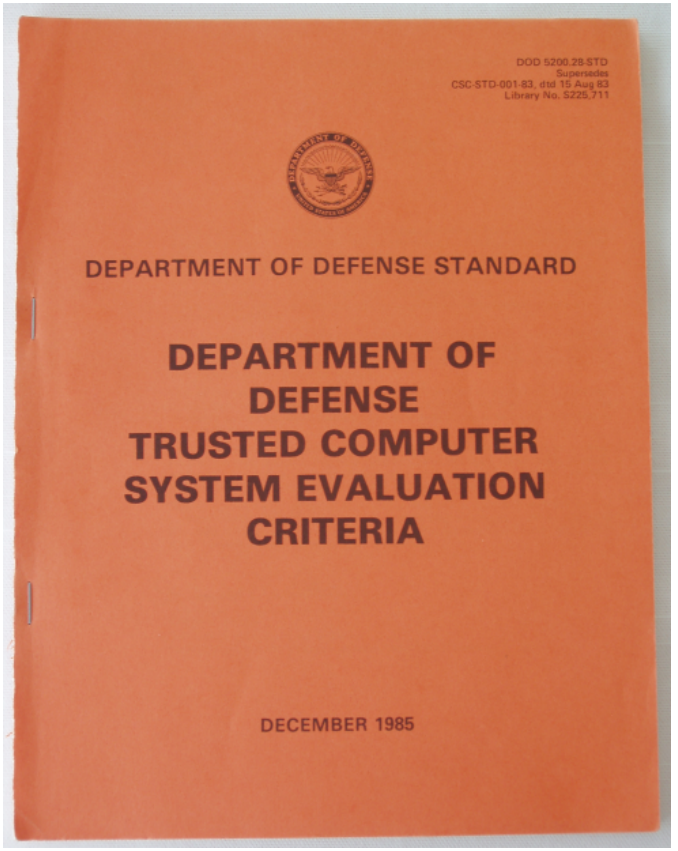
- remaining possible attack:
  - **dictionary attack**
  - problem: people do not chose passwords **randomly**
  - often names, words or predictable numbers are used
  - http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time
  - attacker uses dictionaries for brute force attack
  - prominent program: *John the Ripper*
    - supports dictionary attacks and password patterns
- possible solutions:
  - enforce password rules
    - consider usability
  - pre-check passwords (e.g. using John)
  - train people to “generate” good passwords
    - Example: sentence → password
    - “This is the password I use for Google mail” → “TitpIu4Gm”

Login	Password
…	…
dog	*h*(salt,bone)
…	…

Password based authentication

- **(non protocol related) security problems:**
  - phising, i.e. faked UI for entering secret information
  - today: mostly Internet based attacks
  - but: local attacks possible as well
    - faked login / lock screen
    - solution: “trusted path” / Secure Attention Key

      3.2.2.1.1 The TCB [Trusted Computing Base] shall support a trusted communication path between itself and user for initial login and authentication. Communications via this path shall be initiated exclusively by a user.
      [Department of Defense: “Trusted Computer System Evaluation Criteria”, CSC-STD-001-83, 15. August 1983 – called “Orange Book”]
    - well known implementations:
      - Windows: Ctrl+Alt+Del
      - Linux: Ctrl+Alt+Pause
        - could be freely chosen in principle



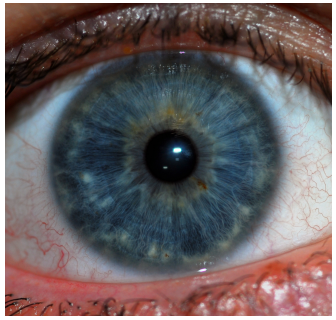
[http://en.wikipedia.org/wiki/File:Orange-book-small.PNG]

Biometrics for Authentication

- *Physiological* or *behavioural* characteristics (of a human being) are measured and compared with reference values to
  - **verify,** that a given subject is the one it claimed to be
    - claimed “identity” is known to the system by other means
  - **identify**, a subject within a given set of (known) subjects
    - “identity” should be derived from biometrics
    - usually more difficult compared to verification

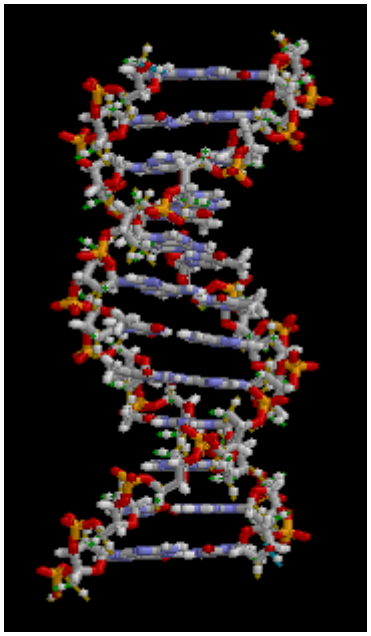
Biometrics: Physiological / Behavioural Characteristics

[Pictures are mostly from Wikipedia]



Iris / Retina

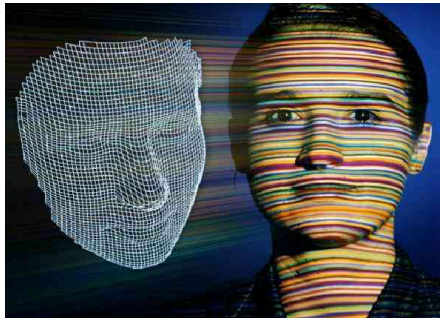
Fingerprint



DNA

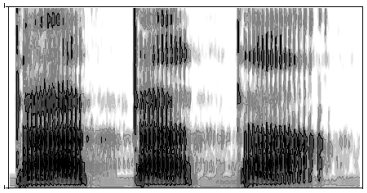
Thermography: facial thermograms

Hand geometry



[http://www.bromba.com/knowhow/IBS2005.pdf]

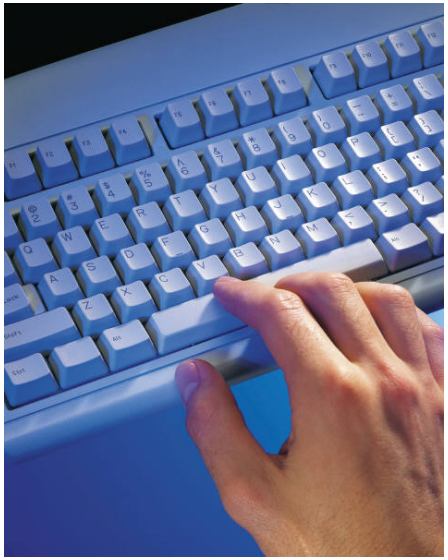
(3D) Face geometry



Voice spectrogram

Handwriting: appearance, dynamics of writing

Gait



Key strokes: dynamics of writing (speed, pressure etc.)

Biometric characteristics: Requirements

- universal: everyone has it
- unique
- stable over time
- measurable
- acceptable
- analysable
- resistant against cloning / faking

Biometrics: Pros and Cons

- Pros:
  - Cannot be divulged or lost/forgotten
  - can be utilised “on the fly”
  - Hard to copy
- Cons:
  - Cannot be renewed
  - Person related data requires special protection (privacy)
  - Invasion (of privacy)
  - Error rate

Biometrics: Pros and Cons

- Pros:
  - Cannot be divulged or lost/forgotten
    - but could be stolen

Safety Risks of Biometrics

Biometrics: Pros and Cons

- Pros:
  - Cannot be divulged or lost/forgotten
    - but could be stolen:
      - http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm
    - could become „unusable“ due to
      - ageing
      - incidents
      - disease
  - can be utilised “on the fly”
    - privacy problems (unnoticeable measurement of Biometrics)
  - Hard to copy
    - depends on the Biometric system used
    - many systems are easy to cheat
    - ftp://ftp.ccc.de/pub/documentation/Fingerabdruck_Hack/fingerabdruck.mpg

Demonstration of Fingerprint Cloning by CCC

Biometrics: Pros and Cons

- Pros:
  - Cannot be divulged or lost/forgotten
    - but could be stolen:
      - https://www.theregister.co.uk/2005/04/04/fingerprint_merc_chop/
      - could become „unusable“ due to
      - ageing
      - incidents
      - disease
  - can be utilised “on the fly”
    - privacy problems (unnoticeable measurement of Biometrics)
  - Hard to copy
    - depends on the Biometric system used
    - many systems are easy to cheat
    - ftp://ftp.ccc.de/pub/documentation/Fingerabdruck_Hack/fingerabdruck.mpg
    - cloning of e.g. fingerprints might be in the interest of law enforcement
      - access to biometrically secured devices

Biometric Systems: Types of Failures

- False Accept Rate (FAR) / False Match Rate (FMR):
  - **Security problem!**
- False Reject Rate (FRR) / False nonmatch Rate (FNR):
  - Usability / acceptance problem
- Receiver Operating Characteristic (ROC):
  - curve of FAR against FRR
- Equal Error Rate (EER):
  - rate for FAR=FRR
  - can be seen from the ROC curve

ROC Curve and Security Problems of Biometrics

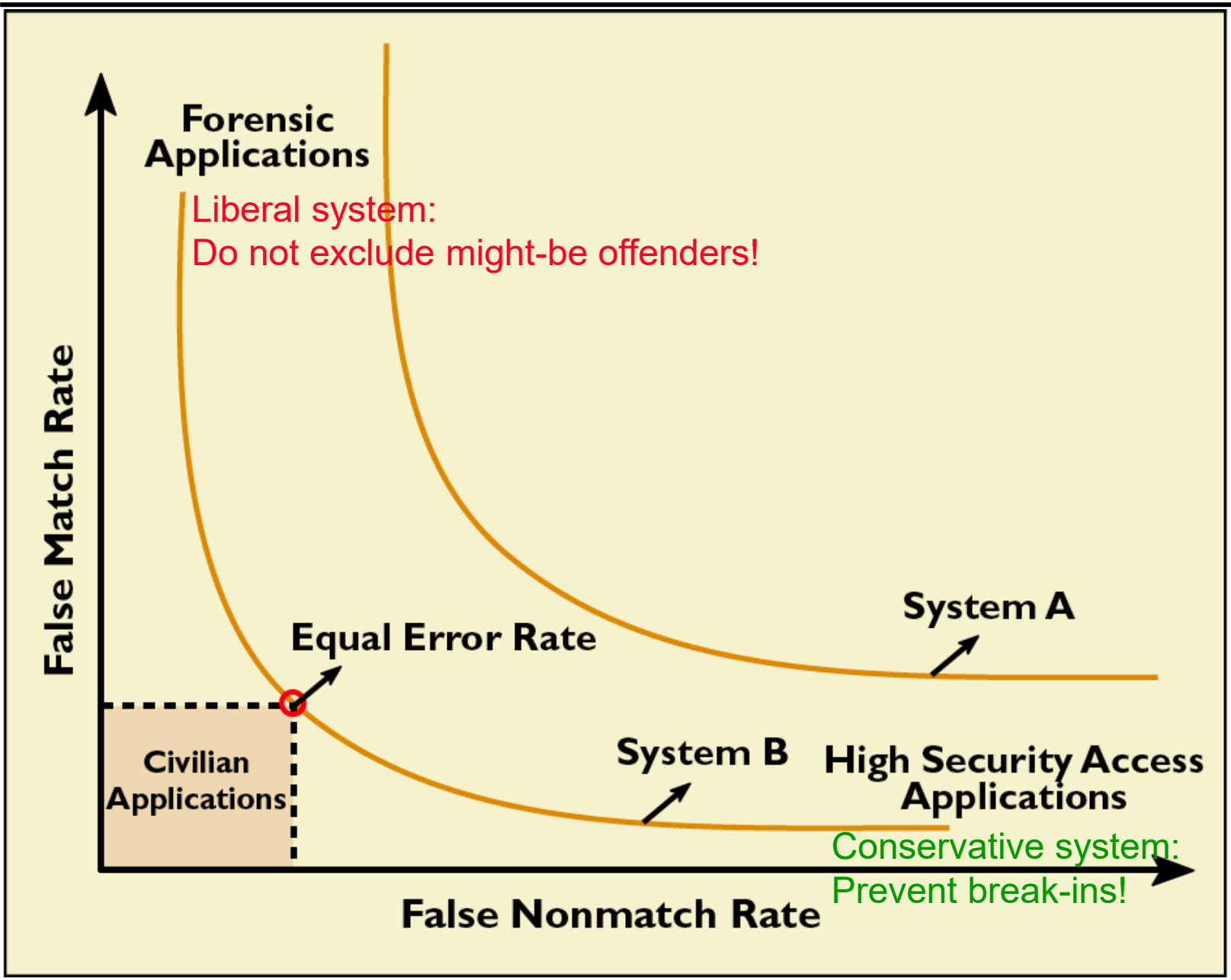


Figure taken from:
Anil Jain, Lin Hong, Sharath Pankanti: Biometric Identification; Communications of the ACM 43/2 (2000) 91-98

Low FMR causes high FNR and vice versa !

Biometric Systems: Types of Failures

- False Accept Rate (FAR):
  - **Security problem!**
- False Reject Rate (FRR):
  - Usability / acceptance problem
- Receiver Operating Characteristic (ROC):
  - curve of FAR against FRR
- Equal Error Rate (EER):
  - error rate for FAR=FRR
  - can be seen from the ROC curve
- Failure To Enroll Rate (FTE):
  - Usability / acceptance problem
- Failure To Capture Rate (FTC):
  - Usability / acceptance problem

Enhanced Security: Multi-biometric Systems

Multi-sample

capacitive

Multi-sensor

Multi-modal

Hybrid

optical

Multi-algorithm

Multi-instance

Minutiae

structure

right eye

left eye