Security and Cryptography 1

Stefan Köpsell, Thorsten Strufe

Module 8:Access Control and Authentication

Disclaimer: large parts from Stefan Katzenbeisser, Günter Schäfer

Dresden, WS 16/17

Reprise from the last modules (30k ft)

Security goals and services describe and implement protection from threats

History has been an arms race between cryptography and cryptanalysis

Each success for the cryptanalysis community has helped make ciphers more secure

Different flavors of ciphers with different properties aim at confidentiality

Secure pseudo-random numbers are essential for the security of ciphers

MACs and signatures aim at providing integrity

Keys can be agreed upon apriori, exchanged, or agreed upon online

Stream- and block ciphers are commonly symmetric and have different properties

Asymmetric crypto allows for public keys and has many different applications

Access Control

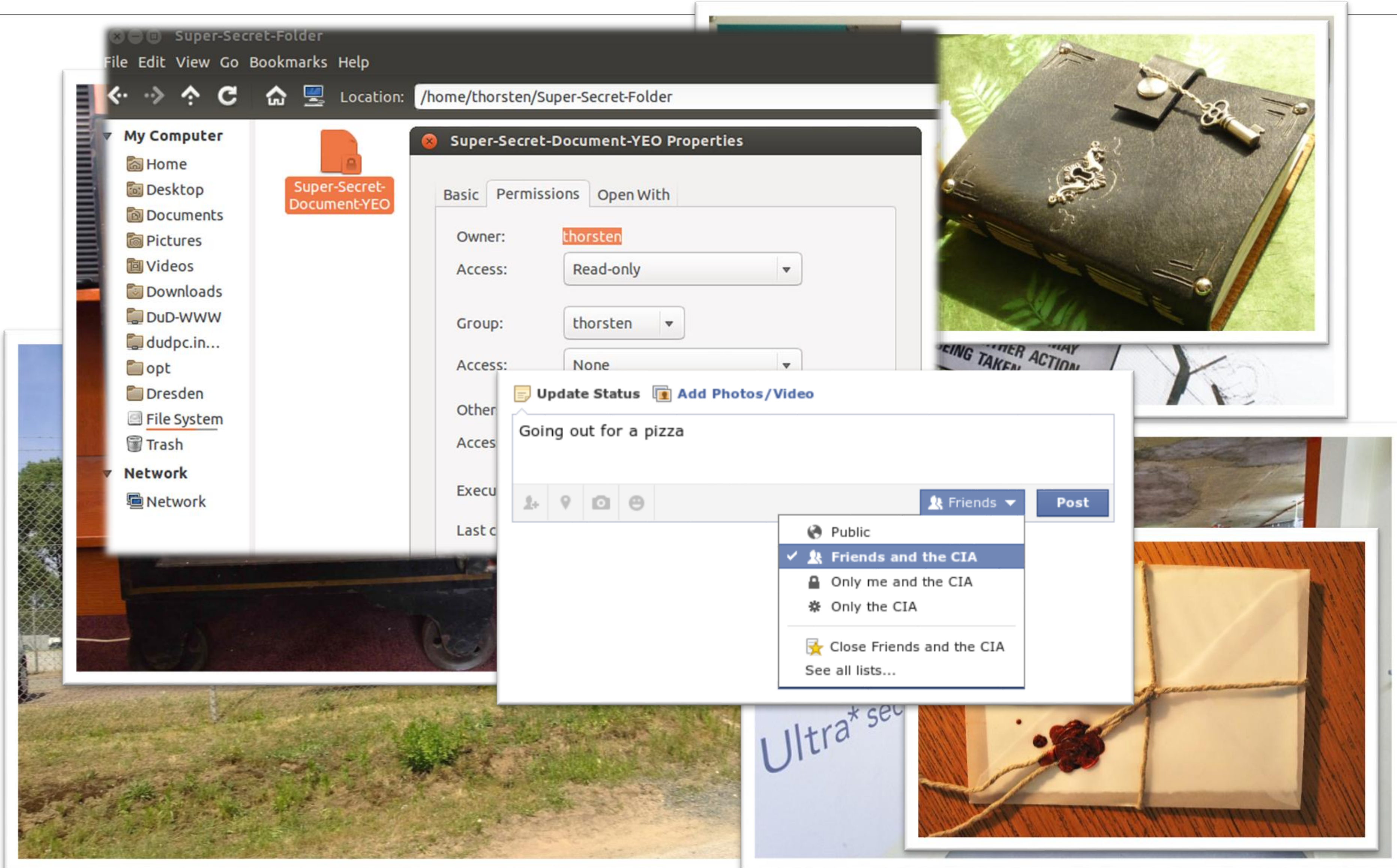
Recall security goals of confidentiality and integrity

So far, using crypto:

- Conceal information in seemingly random noise
- Prove absence of tampering by signature

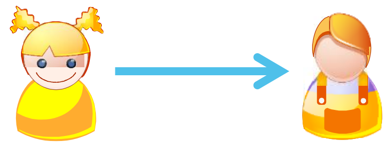
- *How does this solution relate to „real life“?*

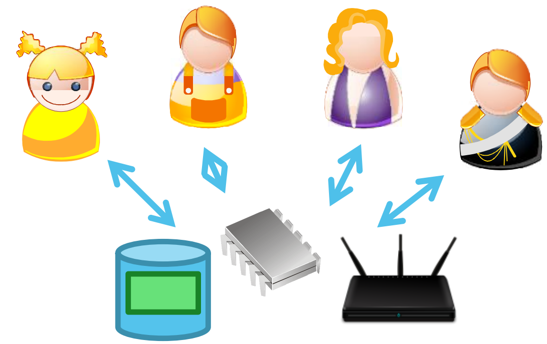
Confidentiality and Integrity „in RL“



Physical access control

Objects vs. Subjects





Subjects have controlled access to objects

- Prevents information disclosure
- Prevents tampering

Requires some gatekeeper:

- Identification of subjects (Authentication)
- Explicit instructions (Policy, <policy descriptions>, authorization)
- Controlling (and granting) access

Some Terminology

Def: ***Access control*** comprises mechanisms to enforce *mediation* on *subject* ***requests for access*** to *objects* as defined in a *security policy*.

Def: A ***subject*** is an *active entity* that can *initiate a request* for resources and utilize these resources to complete some task

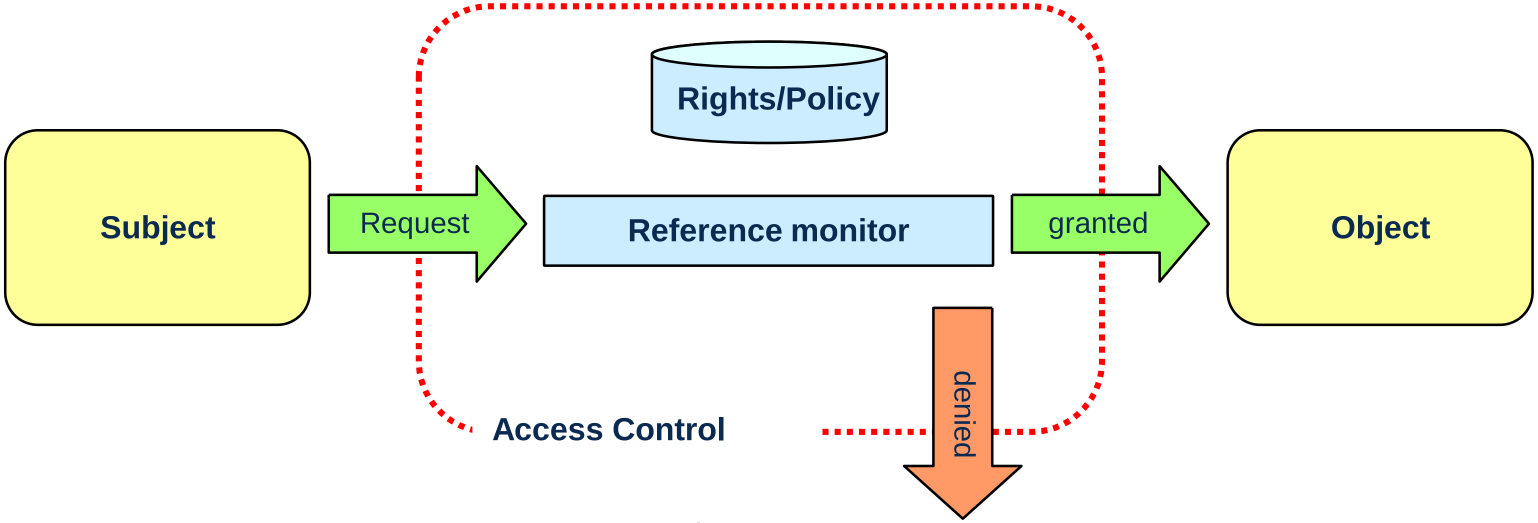
Def: An ***object*** is a resource that is used to store, access, or process information

Def: An ***operation*** (action) is an instance of access, commonly a *utilization, retrieval,* or *manipulation event,* of a *subject* on an *object*

Objects historically had the notion of files, or repositories
Subjects commonly processes (local or remote)
Operations historically*: "r,w,x"*

Common Concept of Access Control

„Reference monitor“ is a concept to detail decision process:



RM not necessarily a physical/logical component in the system

AC/RM may be implemented on different levels:

- *Online application*: control access to functions/data
- *Databases*: control access to tables, columns
- *OS*: control access to resources (files, devices)

Security Objectives, Levels, and Categories

A ***security objective*** is a *statement of intent* to *counter a given threat* or *enforce* a given organisational *security policy*.

A ***security level*** is defined as a *hierarchical attribute* with *entities* of a system to denote their degree of sensitivity

- Examples:
  - Military: unclassified < confidential < secret < top secret
  - Commercial: public < sensitive < proprietary < restricted

A ***security category*** is defined as a nonhierarchical *grouping of entities* to help denote their degree of sensitivity

- Example (commercial): department A, department B, administration, etc.

--> Security categories facilitate the "Need-to-know" principle

Security Labels and Policies

A ***security label*** is defined as an *attribute* that is *associated with system entities* to denote their hierarchical sensitivity level *and* security categories

Security labels that denote the security sensitivity of:

- Subjects are called *clearances*
- Objects are called *classifications*

The ***security policy*** of a system defines the *conditions* under which *subject accesses* to *objects* are mediated by the system reference monitor functionality

- To be derived from the organizational policy (IPRs, procedures)
- Compliance to be monitored (on introduction, regularly)

Classes of Security Models

Access control models

- Identity-based access control (IBAC)
- Role-based access control (RBAC)
- Attribute-based access control (ABAC)

Information flow models (e.g. Chinese Wall model)

- Multilevel security models (e.g. Bell-La Padula model)

Non-interference models

General types of access control:

- Discretionary
- Mandatory

Types of Access Control: DAC / MAC

Discretionary Access Control

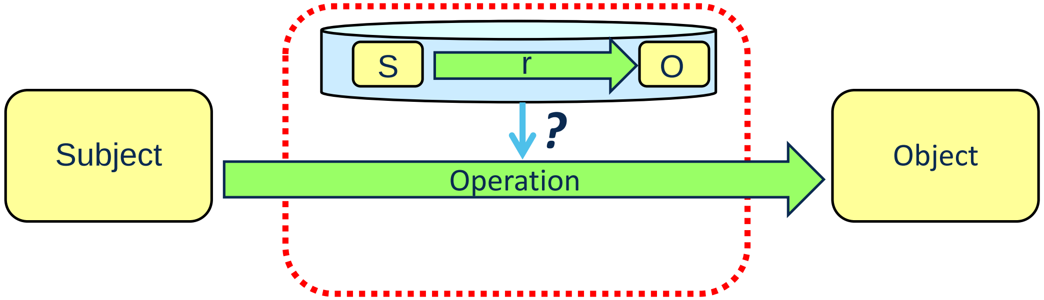
- Owner is responsible for security of her objects
- Authorization per object
- No system-wide security properties
- Rights commonly to be granted: read, write, execute (*NIX, win)
- --> commonly challenged by lack of competence, overview

Mandatory Access Control

- System-wide (usually: rule-based) security policy configuration
- User may change authorization, but system policy dominates
- --> commonly challenged by lack of overview, BOfH

IBAC: Access Control Matrix

Task: Configuration of authorizations (rights of ***subjects*** on ***objects***)



Define: Set of objects *O*, set of subjects *S*, set of rights *R* (e.g. rwx…)

Access Control Matrix defines mapping M : S x O → 2^R (e.g.: {true,false})

Advantages of ACM:

- Intuitive, flexible
- Easy to implement

Disadvantages of ACM:

- Huge, sparse
- static

	o1	o2	o3	o4	o5
s1	{ read, write }		{ read, write }		{ send, receive }
s2				{ send, receive}	
s3		{ owner, execute }		{ signal }	

Access Control Description Schemes

Access Control Lists (ACLs)

- Columns of the ACM: list of authorizations on an object
- `ACL(o1) = {(s1,{r,w}), (s2,{r}); ACL(o2) = {(s3,{r,w,x});` …

(*NIX: subjects only identified as owner, group, others)

```
-r-x--xrwx  1 thorsten www-data    0 Jan 13 10:14 Super-Secret-Document-YEO
```

- Assessing authorizations to an object is simple
- Assessing authorizations granted to a subject is difficult

Capabilities

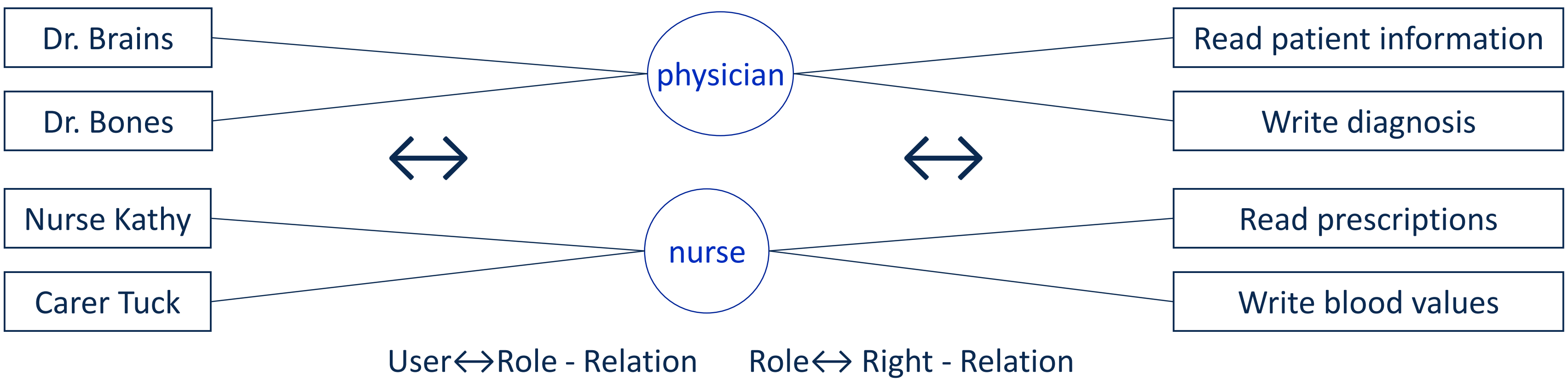
- Rows of ACM: list of objects and rights granted to a subject
- `CL(s3) = {(o2,{o,x}), (o4,{s,r});` …
- Advantages/disadvantages inverse to ACLs…

From IBAC to RBAC

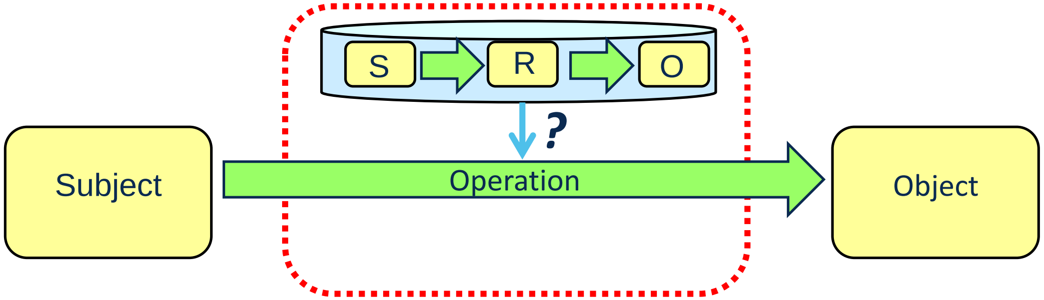
Complexity of IBAC yields problems of overview and adaptation

Subjects usually act in „roles“ (specificly in organizations)

Introduce indirection of the role abstraction:



Role-based Access Control



Extend IBAC:

- Set of subjects *S*
- Set of roles *R*
- Set of objects *O*
- Set of permissions *P*

Define mappings $sr: S \rightarrow 2^R$; $pr: R \rightarrow 2^P$

Sessions are dynamic role assignments (a subject is active in a role)

Subject is assigned permissions from role for the session accordingly

Role hierarchies and constraints extend RBAC

Entity Authentication

„Who's Brian of Nazareth? We have an order for him to be released."

Terry Jones, et al.: Life of Brian (1979)

Identifying humans and Entity Authentication

Goal: Identify a subject (user or process!) and verify identity

Classes of authentication:

- *User authentication (login)*
- *Computer network authentication (identity management)*
- *Identity verification service*

Authentication cardinality:

- One-way authentication
    - Computer authenticates user
    - ATM authenticates cardholder
    - Browser authenticates Web server
- Two-way (mutual) authentication
    - ePass <--> reader
    - UMTS cellphone < -- > network
    - Online bank < -- > account holder (w/ certificates)

Authentication factors

Different factors can be used to authenticate a user

- Knowledge factors
  - Passwords
  - Answers to „security questions“
  - …
- Possession factors
  - Security token
  - Smart card
  - Keys/certificates
  - …
- Inherence factors
  - Biometric factors
  - Signature
  - …
- *Sometimes: other properties (e.g. location)*

Authenticating a device

Conventional: knowlede (key)

Advanced: Possession (smart card)

Recent: Inherence (PUF)

Sometimes: location (ATM)

Entity Authentication

Factor verification:

- ***direct*** (Alice vs. Bob) or
- mediated by an ***arbiter*** („TTP“, *Kerberos, Shibboleth*)

Basic requirements:

- Strength of secret determined by its entropy (passwords, biometry)
- Provision and management: factors must remain secret (*impersonation*), be adjustable, possibility for revocation
- Monitoring, detection and reaction of/to malicious authentication attempts

Multi-factor authentication:

- Combines different factors (*examples?*)
  - ATM card (possession) and PIN (knowledge)
  - Password (knowledge) and mobileTAN (possession of cell phone)
- Requires independence of factors
- Increases security only as much as weakest factor (security question?)
- *(not to confuse with fall-back authentication – as secure as weakest factor…)*

Trouble with Authentication

Verification of factors over networks is difficult, possible with crypto

Entity authentication is more than exchange of authentic messages:

- Even if Bob receives authentic messages from Alice during a communication, he can not be sure, if:
  - Alice is actually participating *in this specific moment,* or if
  - Eve is *replaying* old messages from Alice
- Especially important, when authentication is only performed at connection-setup time:
  - Example: transmission of a (possibly encrypted) PIN when logging in
- Two principle means to ensure timeliness in cryptographic protocols:
  - *Timestamps* (require loosely synchronized clocks)
  - *Random numbers/Nonces* (challenge-response exchanges)

Authentication – Case Studies

Authentication „in RL“

- Identity cards

User authentication

- Knowledge-based: passwords
- Inherence-based: Biometry

Network authentication:

- Kerberos

Individuals authenticati

Consider how you authenticate „in RL“ ((@

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

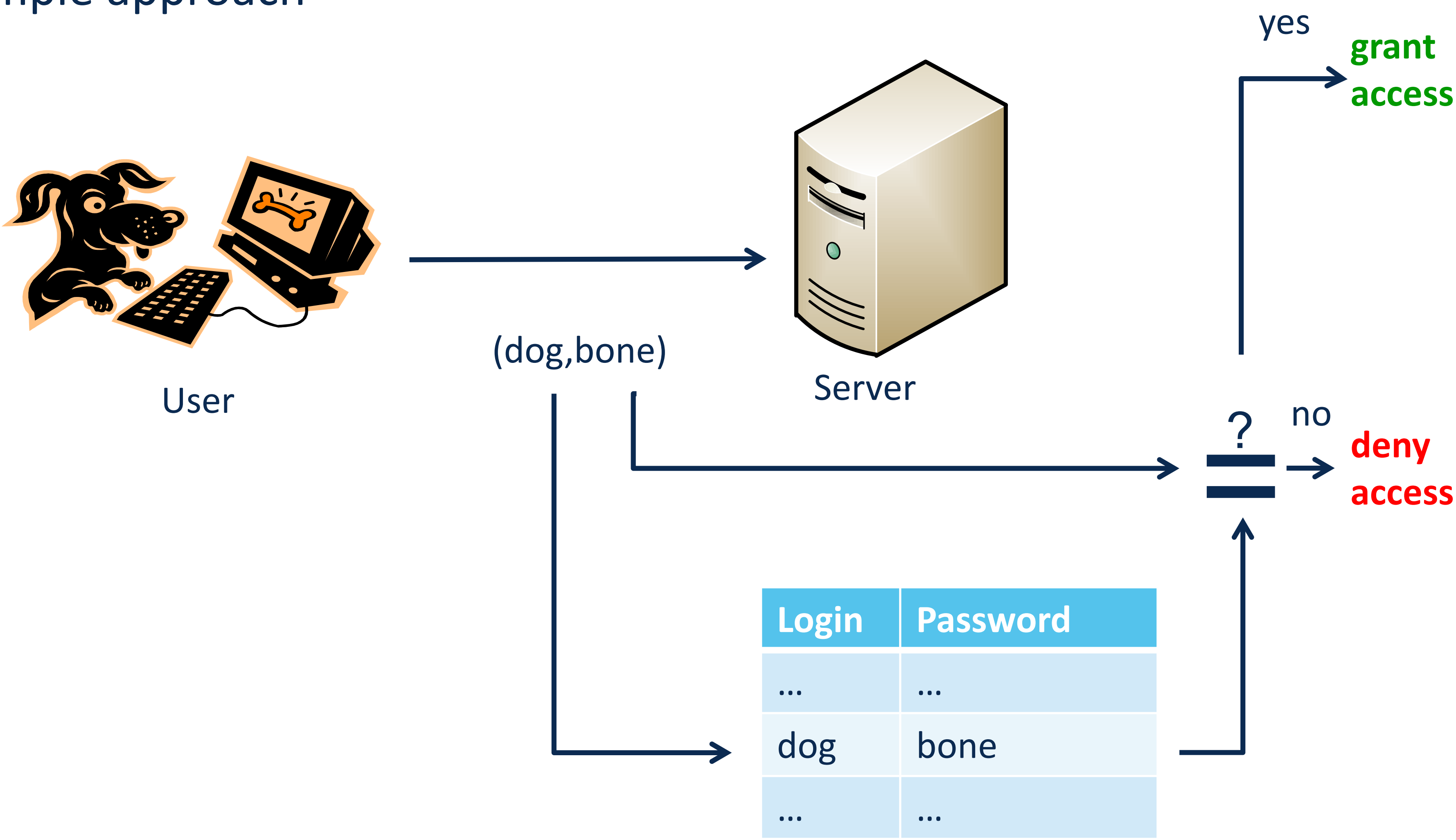
has (possession)
- paper document
- metal key
- magnetic-strip card
- smart card (chip card)
- calculator

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

finger print, picture, hand-written signature, paper document, smart card (chip card), password, passphrase → **eID-card**

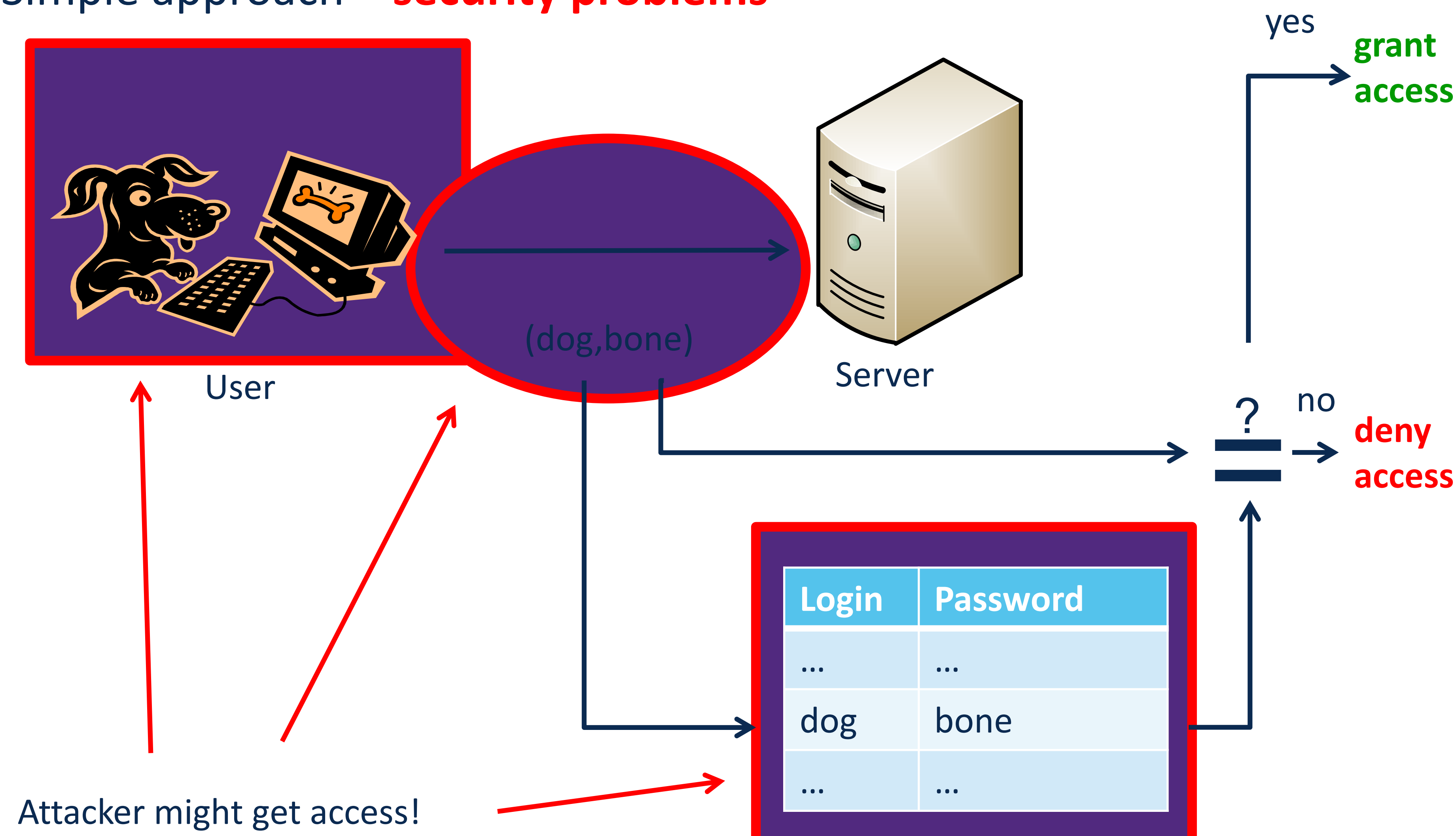
Knowledge-based authentication: passwords

Simple approach

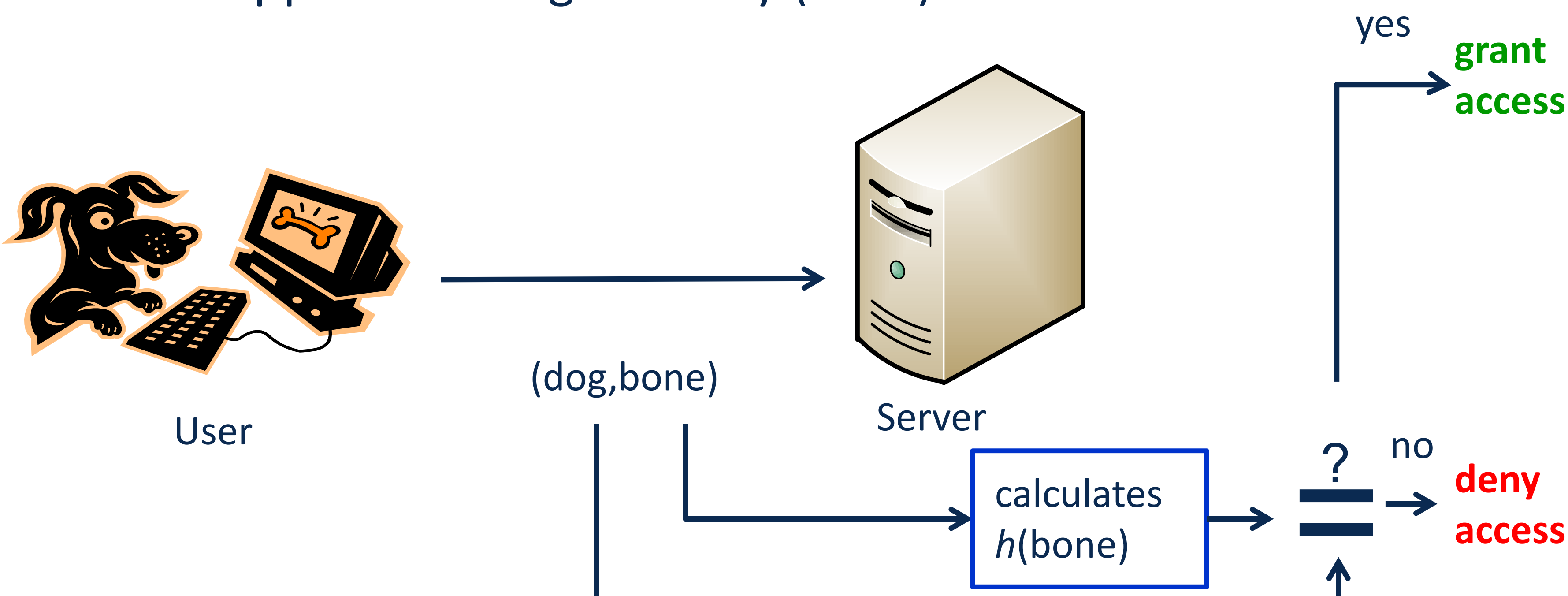


Password based authentication

Simple approach – **security problems**

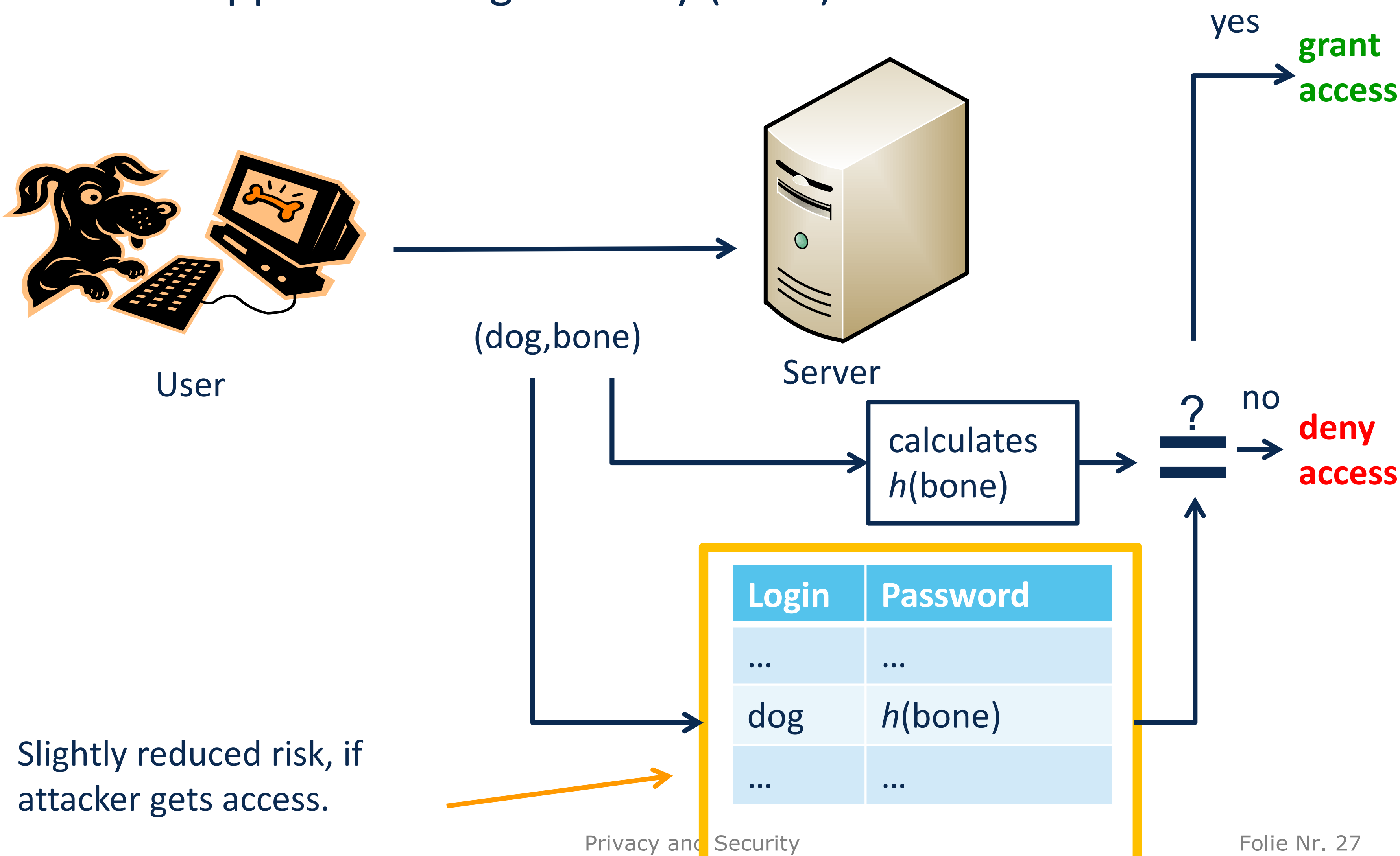


Enhanced approach using one way (hash) functions



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

Enhanced approach using one way (hash) functions



Login	Password
…	…
dog	*h*(bone)
…	…

Slightly reduced risk, if attacker gets access.

Problems of hashed password storage

Login	Password
…	…
dog	*h*(bone)
…	…

Possible attack:

- pre-computation (rainbow tables)

countermeasures:

- Hash rounds
  - store: h^{1000}(pw)
  - linear overhead per string (computation, storage)
- Salting
  - Use (long) random value
  - store: h(salt,pw), salt

```
$ sudo less /etc/shadow | grep strufe
strufe:$6$m40rV3LS$kAf.4WUEwr7[…]
```

- ➔ pre-computation has to be done for each possible salt,|rounds|

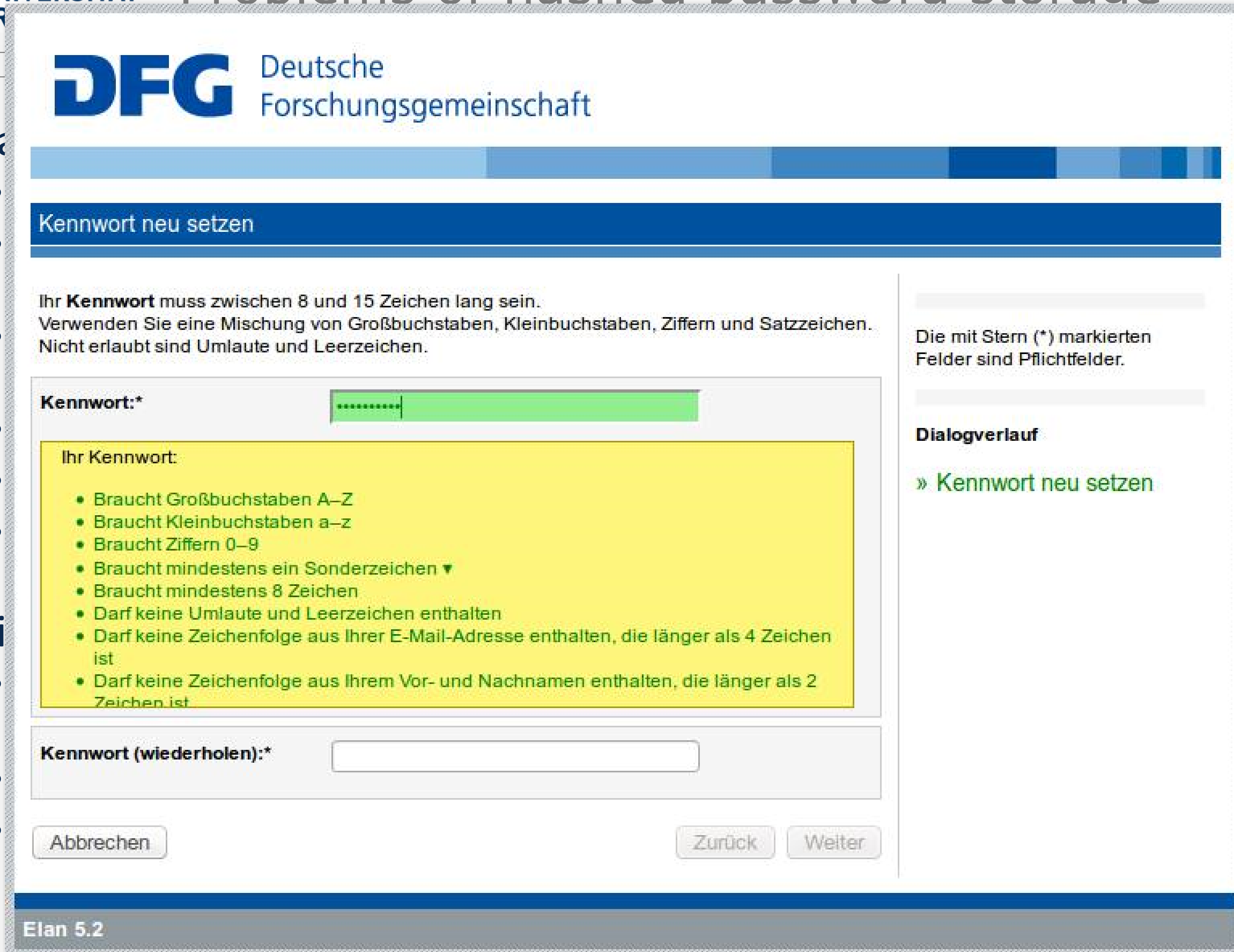
Problems of hashed password storage

Rema

Possi

ne)

me



DFG Deutsche Forschungsgemeinschaft

Kennwort neu setzen

Ihr **Kennwort** muss zwischen 8 und 15 Zeichen lang sein.
Verwenden Sie eine Mischung von Großbuchstaben, Kleinbuchstaben, Ziffern und Satzzeichen.
Nicht erlaubt sind Umlaute und Leerzeichen.

Kennwort:*

Ihr Kennwort:

- Braucht Großbuchstaben A–Z
- Braucht Kleinbuchstaben a–z
- Braucht Ziffern 0–9
- Braucht mindestens ein Sonderzeichen ▾
- Braucht mindestens 8 Zeichen
- Darf keine Umlaute und Leerzeichen enthalten
- Darf keine Zeichenfolge aus Ihrer E-Mail-Adresse enthalten, die länger als 4 Zeichen ist
- Darf keine Zeichenfolge aus Ihrem Vor- und Nachnamen enthalten, die länger als 2 Zeichen ist

Kennwort (wiederholen):*

Abbrechen | Zurück | Weiter

Die mit Stern (*) markierten Felder sind Pflichtfelder.

Dialogverlauf

» Kennwort neu setzen

Elan 5.2

Problems of hashed password storage

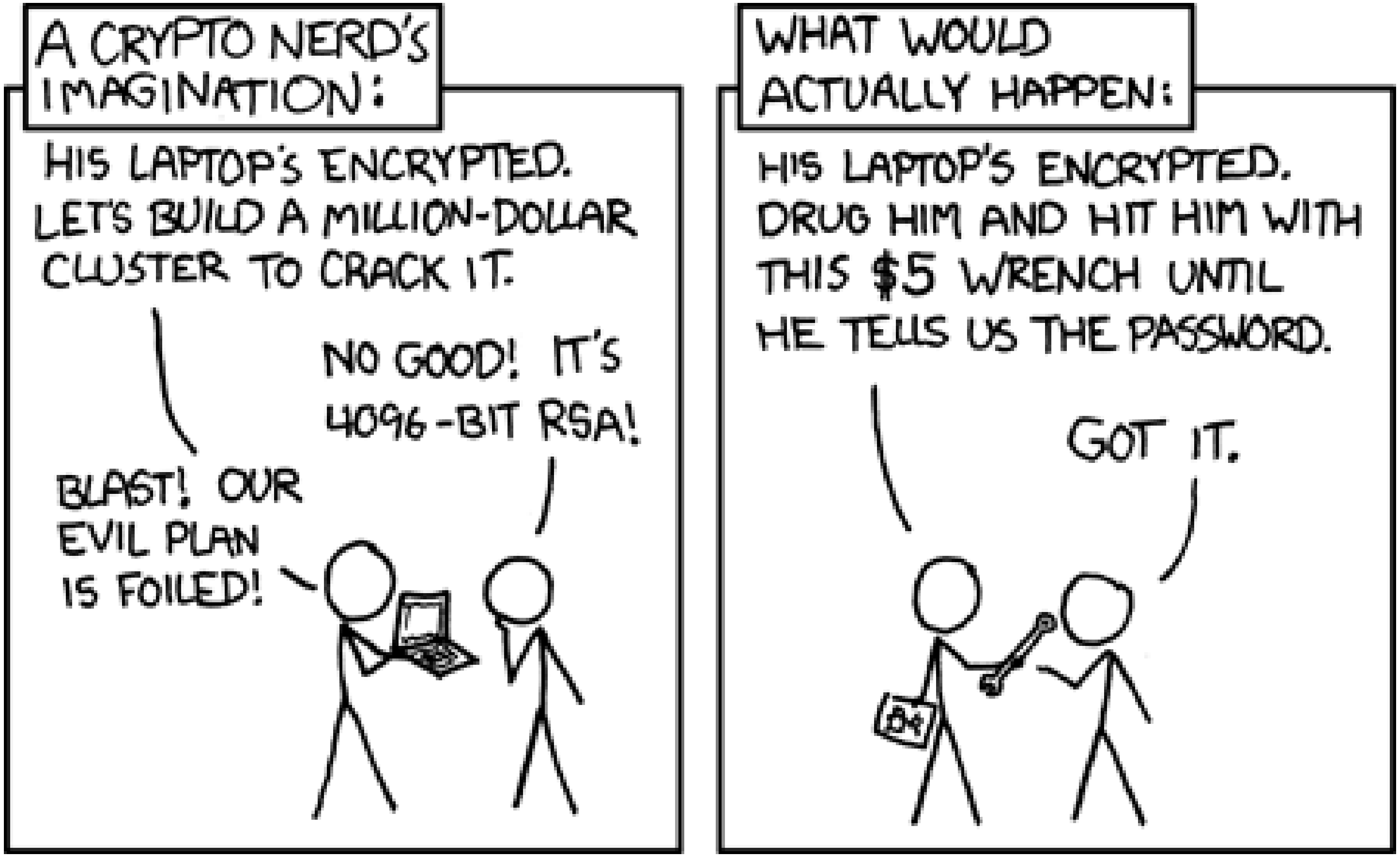
Login	Password
…	…
dog	*h*(salt,bone)
…	…

Remaining attack:

- dictionary attack
- problem: people do not chose passwords randomly
- often names, words or predictable numbers, recently personal information are used
- http://www.whatsmypass.com/the-top-500-worst-passwords-of-all-time
- attacker uses dictionaries and social networks for brute force attack
- prominent program: John the Ripper
  - supports dictionary attacks and password patterns

Possible solutions:

- enforce password rules
  - consider usability
- pre-check (monitor) passwords (e.g. using John)
- train people to "generate" good passwords
  - Example: sentence → password
  - "This is the password I use for Google mail" → "TitpIu4Gm"


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Knowledge-based: Challenge - Response

Prerequisite: Both parties agree on a common secret key

Here: Bob wants to authenticate Alice

Alice:		Bob
Key K_{ID}, Identification ID		Key K_{ID} for ID
Send Login-Information	(1) ID →	
		Generate RAND (Challenge)
	← (2) RAND	
E(RAND, K_{ID}) = C	(3) C →	E(RAND, K_{ID}) = C‘
		Test: C‘ = C?

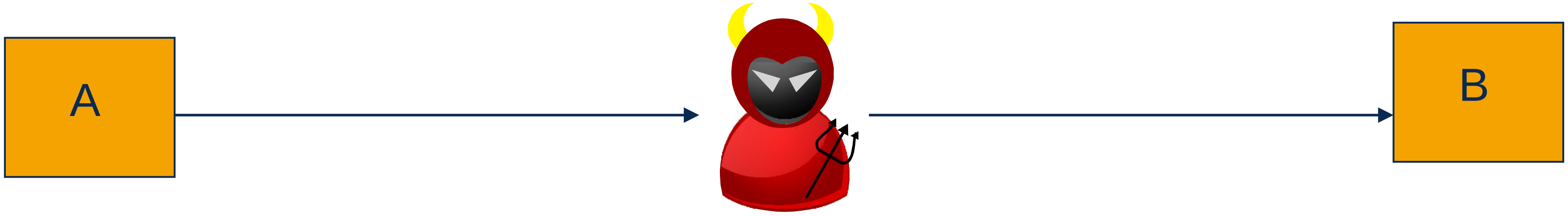
Knowledge-based: Challenge - Response

Assuming the Dolev-Yao adversary:

- Plaintext space for Challenges RAND has to be large!
  - Mallory can intercept and store all tuples (RAND,C)
  - She hence can replay old response

- Cipher has to be known plaintext secure

- Man-in-the-middle attacks, Replay attacks?



Arbited Authentication, Single Sign On

Goals of Single Sign On-Concepts:

- User is authenticated only once (centrally),
- no separate authentication upon requests to use different services within an administrative domain

Kerberos (MIT, 1980s):

- SSO for services within a „realm“
- Authenticate subjects („principals“): users, computers, server
- Exchange session keys for principals based on Needham-Schroeder
- Underlying cryptographic primitive of symmetric encryption (DES in Kerberos V. 4, from V. 5 on other algorithms allowed)

Single Sign On: Kerberos

Objectives of Kerberos:

- *Security:* prevent impersonation of users when accessing a service
- *Reliability:* service use requires authentication --> reliability and availability
- *Transparency:* authentication beyond password transparent to user
- *Scalability:* the system has to support a large number of clients and servers

Design of Kerberos:

- A **single trusted server** per domain (Key distribution center, KDC)
- Tasks of the KDC are:
  - Authenticating the clients of its domain
  - Issue tickets as authentication tokens

Authentication of a Principal

- **Idea: Pre-Shared Secrets** between Principal and KDC
  - For users: hashed (MD5) passwords, master key K_A is derived
  - For servers: shared, secret master key K_S

Kerberos

Content of a Ticket:

- Each ticket is valid only for a principal *C* (e.g. Joe) on the specific server *S* (e.g. NFS), for a specific time:

- $T_{c,s}$ = *S, C, addr, timestamp, lifetime,* $K_{c,s}$ with:
  - *S:* Name of the Server,
  - *C:* Name of the requesting client,
  - *addr:* IP address of the requesting client,
  - *timestamp:* current time,
  - *lifetime*: lifetime of the ticket,
  - $K_{c,s}$: Session key for the communication between *S* and *C*

Kerberos (Protocol, simplified)

- User Joe logs in to local PC with a password

- Local PC (client) C sends ID and Nonce to KDC and requests a ticket for the TGS:

  Joe → KDC: Joe, TGS, *Nonce1*

- KDC extracts master key of the user from its database and issues a ticket $T_{Joe,TGS}$ to authorize utilization of TGS

  KDC → Joe: $\{K_{Joe,TGS}, Nonce1\}_{KJoe}, \{T_{Joe,TGS}\}_{KTGS}$

- Client requests Joe to enter Kerberos password, derives K_{Joe} and extracts: $K_{Joe,TGS}$, *Nonce*1

Kerberos (Protocol, simplified, ctd.)

- Client requests ticket at TGS to use the NFS server:

  Joe→ TGS: $\{A_{joe}\}\, K_{Joe,TGS}$, $\{T_{Joe,TGS}\}\, K_{TGS}$, *NFS, Nonce*2

  where A_{joe}=*Joe, IP-Addr, timestamp* is called „Authenticator“

- TGS checks Authenticator and sends a ticket for the NFS server to Joe:

  TGS → Client: $\{K_{Joe,NFS}$, *Nonce*2$\}\, K_{Joe,TGS}$, $\{T_{Joe,NFS}\}\, K_{NFS}$

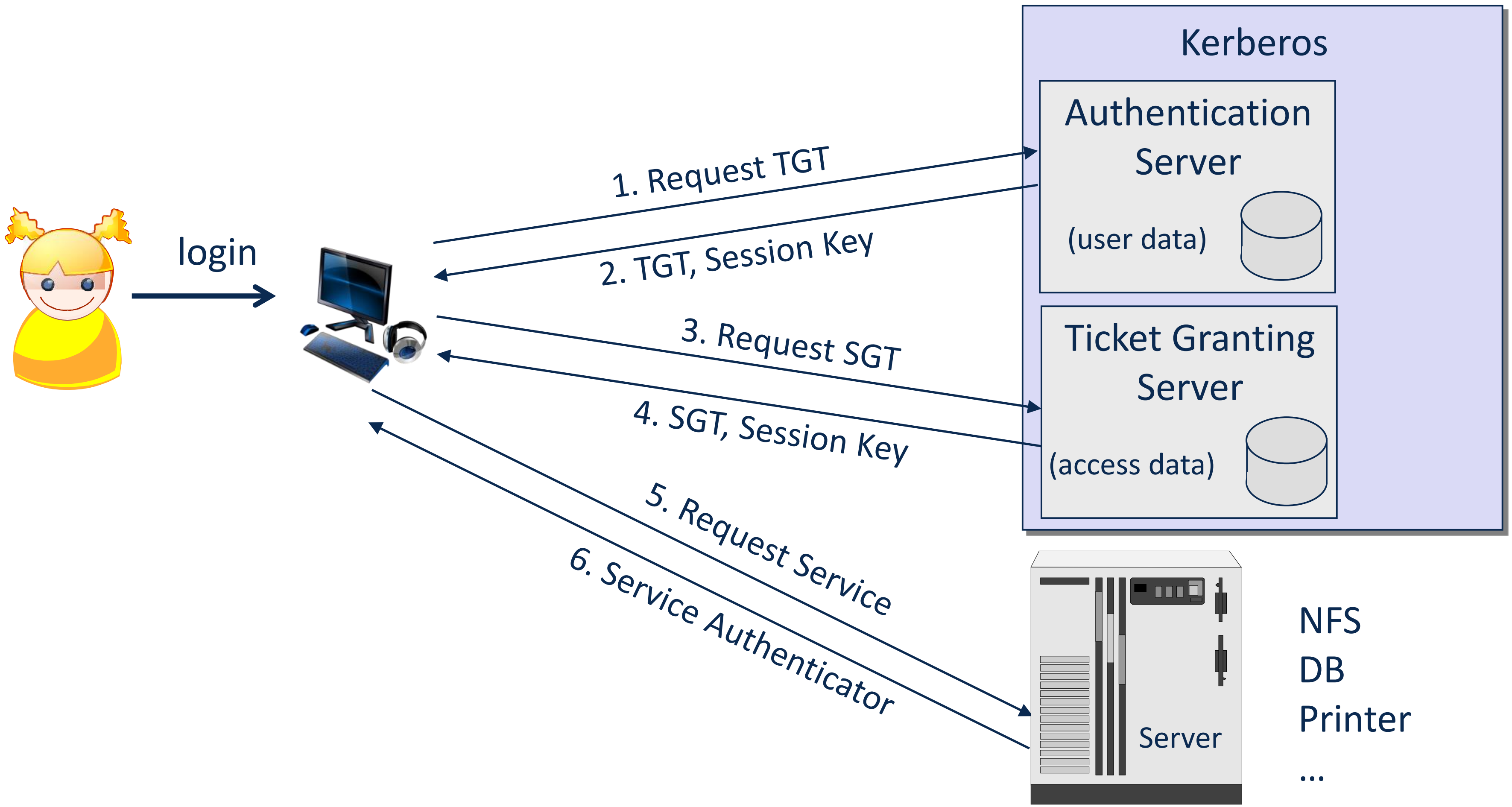
- Joe uses ticket at NFS server:

  Client → NFS: $\{A_{Joe}\}_{KJoe,NFS}$, $\{T_{Joe,NFS}\}_{KNFS}$

- For mutual authentication:

  NFS → Client: $\{timestamp+1\}_{KJoe,NFS}$

Kerberos



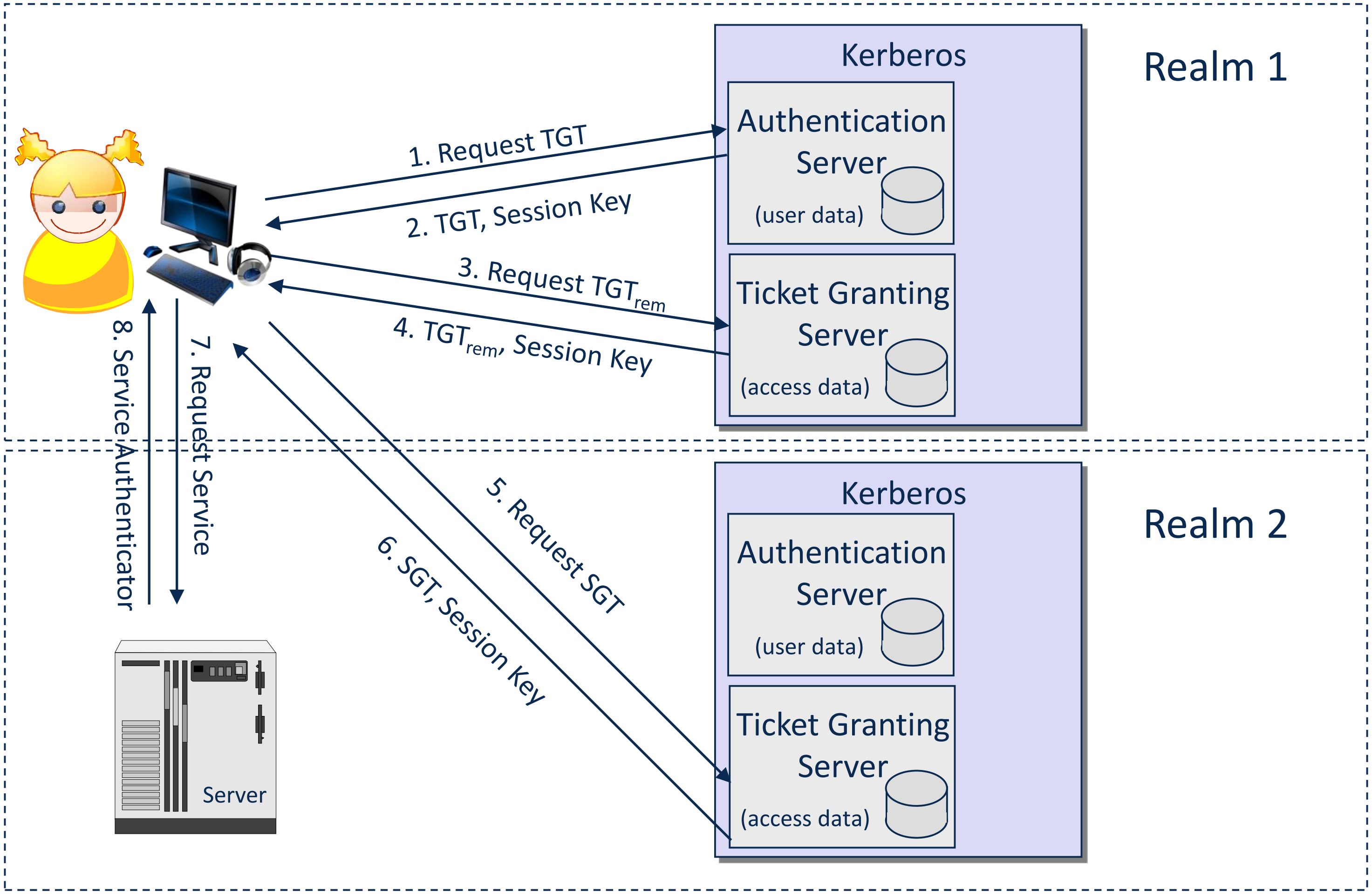
Kerberos
Authentication Server
(user data)
Ticket Granting Server
(access data)
login
1. Request TGT
2. TGT, Session Key
3. Request SGT
4. SGT, Session Key
5. Request Service
6. Service Authenticator
Server
NFS
DB
Printer
…

Inter-Realm Authentication

Essentially the „mother of all SSO/auth-AC systems“

Extending Kerberos to multiple realms:

- Establish mutual trust between TGS of different realms: $K_{TGS1,TGS2}$



Summary

You know means for confidentiality & integrity other than crypto ;-)

You can distinguish between authorization, authentication, access control

You can explain ACMs, ACLs and capabilities

You can distinguish DAC and MAC

You know the different general strategies of IBAC and RBAC

You can explain different classes of authentication and factors

You know about cardinality and multi-factor authentication

You can explain some knowledge-based authentication schemes, strategies

You about biometry and ist advantages and disadvantages