Table of contents I

1 Security protocols

2 Building a key establishment protocol (*)

3 Messages, communication, protools

4 An example: Needham-Schroeder Public Key protocol

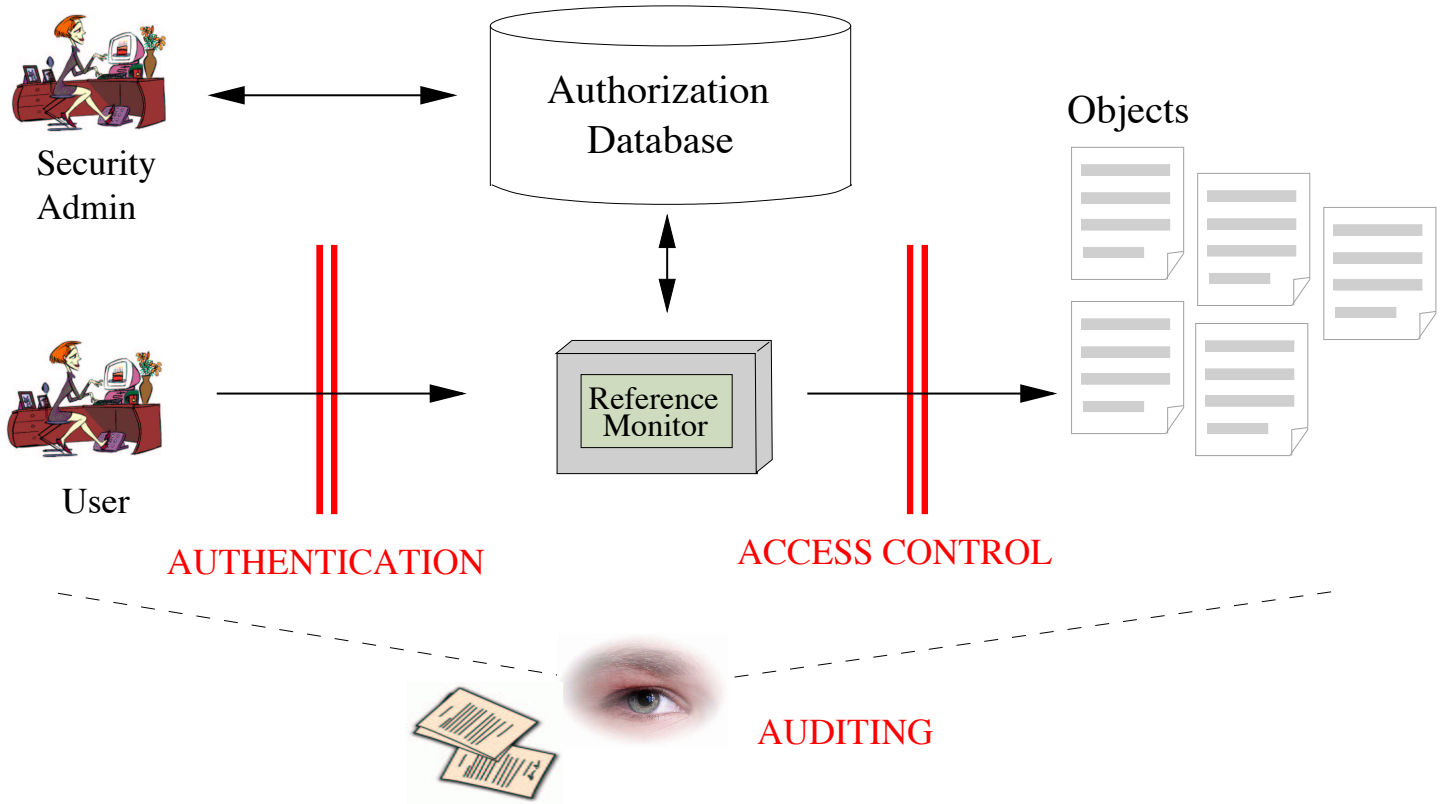
5 Examples of protocols and possible vulnerabilities (*)

6 Prudent engineering of security protocols (*)

7 Kerberos

Authentication/access control in centralized systems

- System knows who the user is, i.e. user authentication performed.
- OS designed so that access requests pass through a gatekeeper.



What mechanisms are suitable for securing distributed systems?

Distribution

- Until recently, authentication by assertion was standard. Clients authenticate users and servers enforce security policy based on user ID.
- Example: Unix *rlogin*. Suppose I rlogin to a remote machine under my own name where a *.rhosts* file names my machine. *rlogin* asserts my identity to the remote machine's daemon, who does not require a password.
  Unsafe! Consider an attacker who can convince rlogin he is me or rewrites a rogue version of rlogin to assert my identity.
- Sending passwords (e.g., HTTP Basic Authentication) is not much better. Problems?
  - Time consuming to repeatedly enter passwords.
  - Sending passwords on the network is dangerous and error prone.

Kerberos

- Protocol for authentication in open distributed environments.
- In Greek mythology, Kerberos is the 3-headed dog guarding the entrance to Hades.
  Modern Kerberos intended to have three components to guard a network's gate: authentication, accounting, and audit. Last two heads never implemented.

- Developed as part of Project Athena (MIT, 1980's)
  - Version 4 (1989) still used at many sites.
  - Version 5 (1993) now standard (IETF RFC 1510).
    - Version 5 plus extensions for public key certificates used for network authentication in Windows 2000, XP, Vista, etc.
    - January 2019: krb5-1.17 release.

Kerberos — requirements

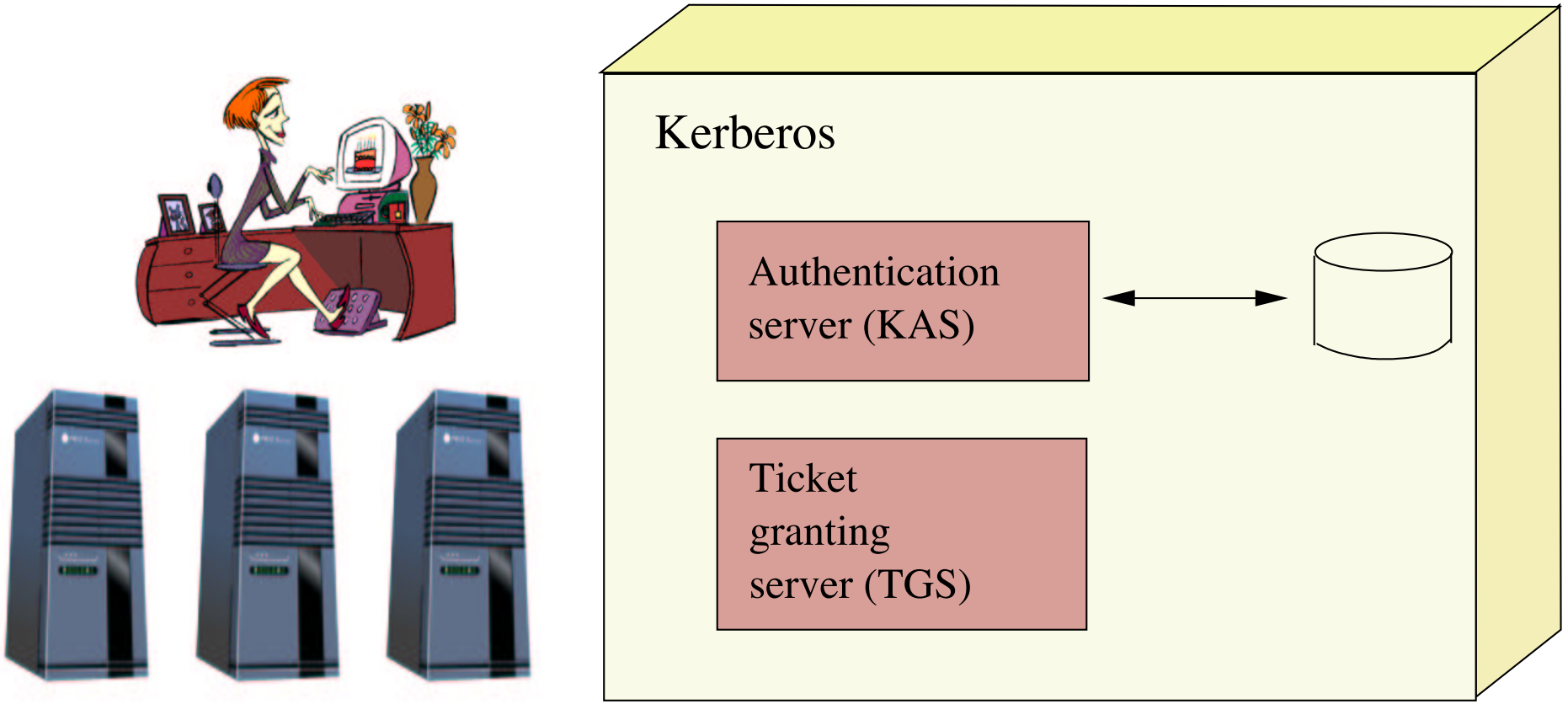
Secure: An eavesdropper should not be able to impersonate users.

Reliable: As many services depend on Kerberos for access control, it must be highly reliable and support a distributed architecture, where one system can back up another.

Transparent: Each user should enter a single password to obtain network services and should be unaware of underlying protocols. Nowadays, this is referred to as single sign-on.

Scalable: The system should scale to support large numbers of users and servers; this suggests a modular, distributed architecture.

Kerberos version 4: architecture

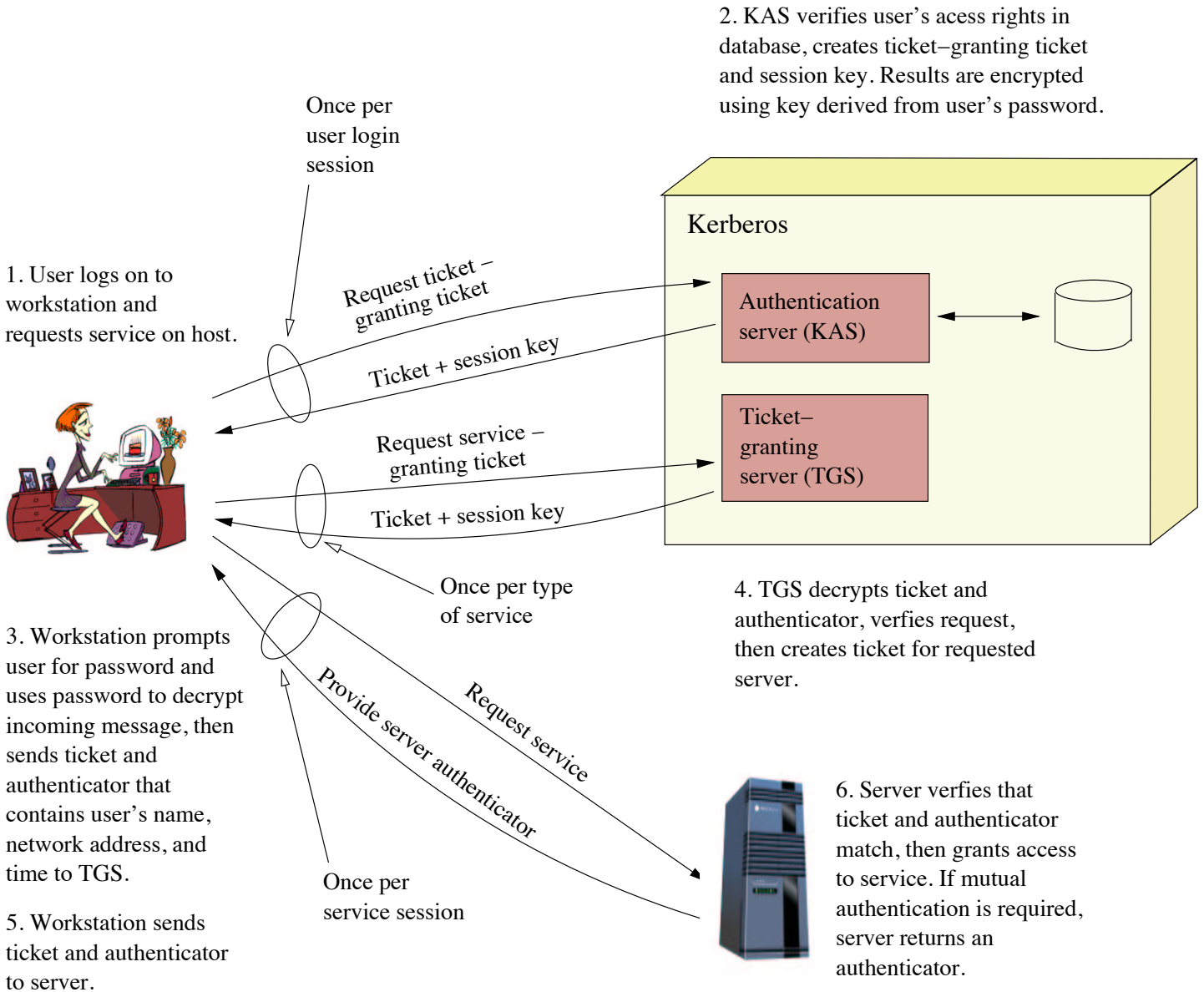


Authentication using KAS, the Kerberos Authentication Server.

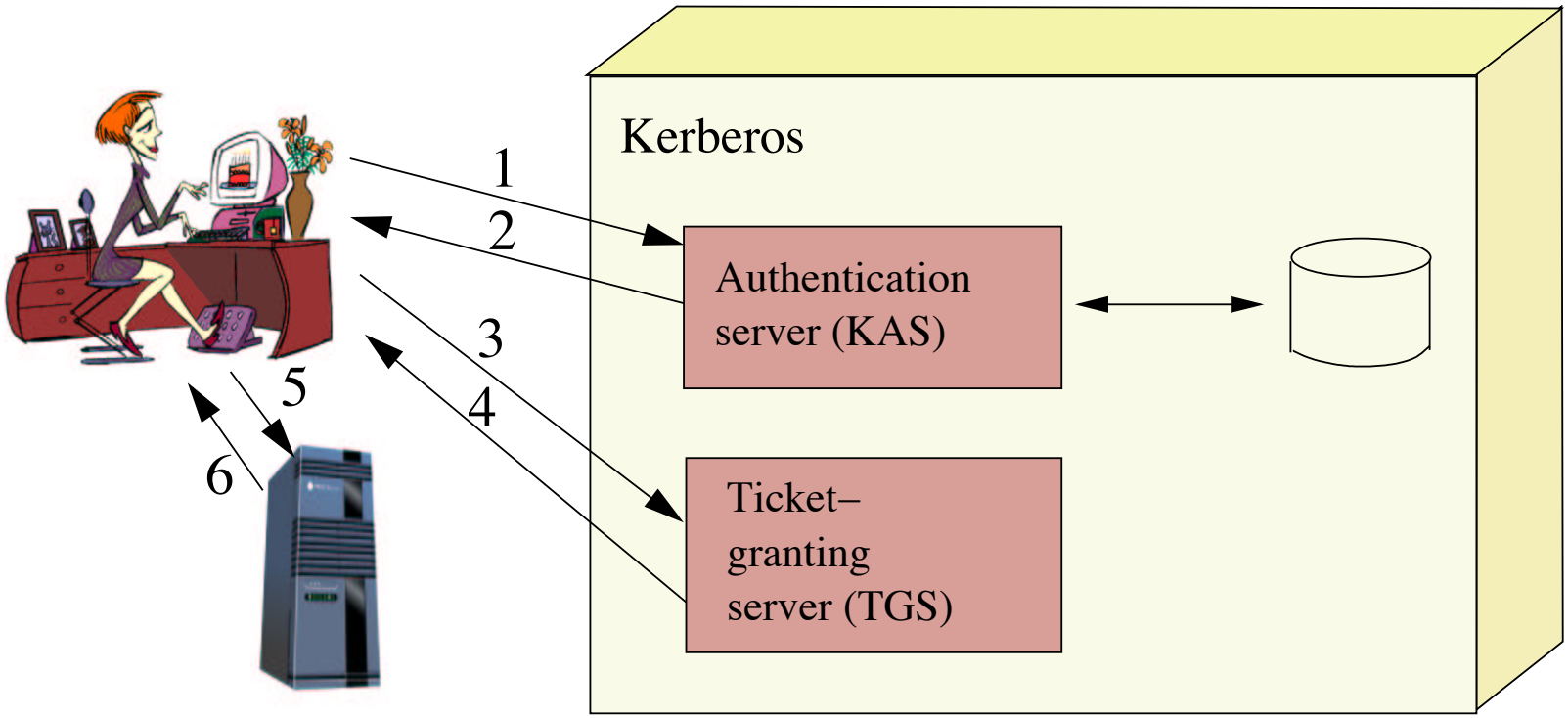
Authorization using TGS, the Ticket Granting Server.

Access Control where servers check TGS tickets.

Kerberos version 4: operation



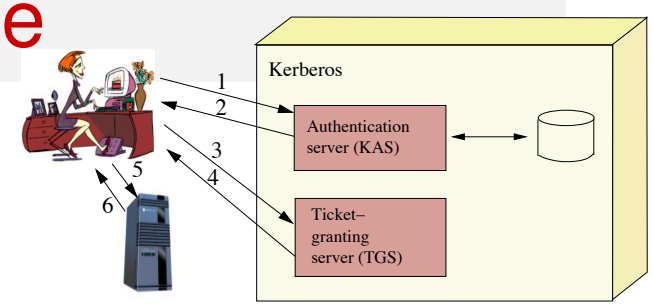
Kerberos version 4: protocol



Authentication	messages 1 and 2.	Once per user login session.
Authorization	messages 3 and 4.	Once per type of service.
Service	messages 5 and 6.	Once per service session.

We present the three parts below (slightly simplified).

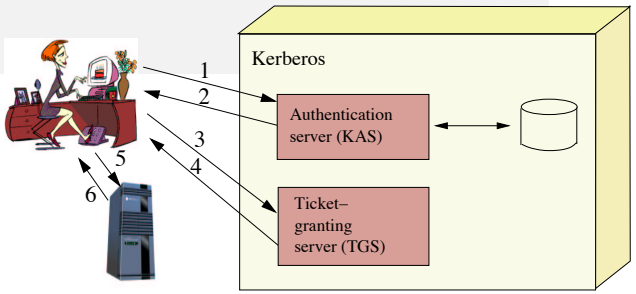
Kerberos version 4: authentication phase



1. A $\rightarrow$ KAS : A, TGS
2. KAS $\rightarrow$ A : $\{K_{A,TGS}, \text{TGS}, \mathcal{T}_1, \underbrace{\{\text{A}, \text{TGS}, K_{A,TGS}, \mathcal{T}_1\}_{K_{KAS,TGS}}}_{AuthTicket}\}_{K_{AS}}$

- *A* logs onto workstation and requests network resources.
- KAS accesses database and sends *A* a session key $K_{A,TGS}$ and an encrypted ticket AuthTicket.
- $K_{A,TGS}$ has lifetime of several hours (depending on application). K_{AS} derived from user's password.
  Both user and server keys must be registered in database.
- *A* types password on workstation to decrypt results, which are stored for session. *A* is logged out when $K_{A,TGS}$ expires.

Kerberos version 4: authorization phase



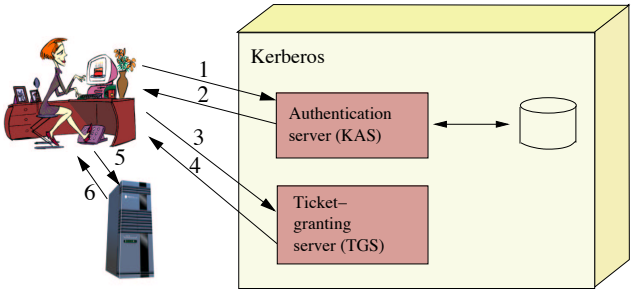
3. $A \rightarrow TGS: \underbrace{\{A, TGS, K_{A,TGS}, T_1\}_{K_{KAS,TGS}}}_{AuthTicket}, \underbrace{\{A, T_2\}_{K_{A,TGS}}}_{authenticator}, B$

4. $TGS \rightarrow A: \{K_{AB}, B, T_3, \underbrace{\{A, B, K_{AB}, T_3\}_{K_{B,TGS}}}_{ServTicket}\}_{K_{A,TGS}}$

Before *A*'s first access of network resource *B*:

- *A* presents AuthTicket from message 2 to TGS together with a new *authenticator*, with short (seconds) lifetime.
  - Role of authenticator? Short validity prevent replay attacks.
  - Servers store recent authenticators to prevent immediate replay.
- TGS issues *A* a new session key K_{AB} (lifetime of few minutes) and a new ticket ServTicket. $K_{B,TGS}$ is key shared between TGS and network resource.

Kerberos version 4: service phase



5. $A \rightarrow B : \underbrace{\{A,\ B,\ K_{AB},\ \mathcal{T}_3\}_{K_{B,TGS}}}_{ServTicket}, \underbrace{\{A, \mathcal{T}_4\}_{K_{AB}}}_{authenticator}$
6. $B \rightarrow A : \{\mathcal{T}_4 + 1\}_{K_{AB}}$

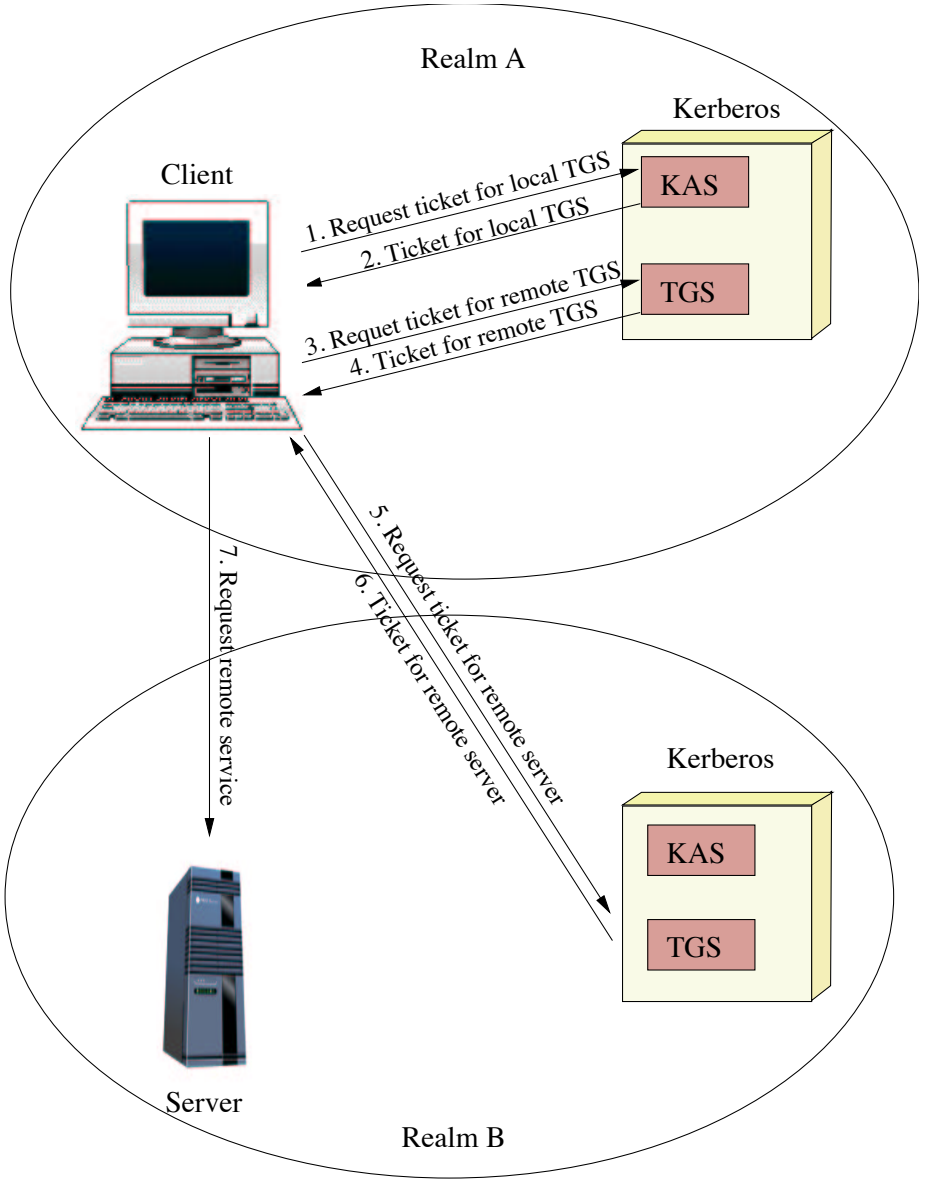
For *A* to access network resource *B*:

- *A* presents ticket from 4 to *B* along with new *authenticator*. In practice, other information for server might be sent too.
- *B* replies (option), authenticating service.

Scalability — Multiple Realms/Kerberi

- A realm is defined by a Kerberos server.
  Server stores user and application server passwords for realm.
- Large networks may be divided into administrative realms.
- Kerberos supports inter-realm protocols.
  - Servers register with each other.
  - For A to access B in another realm, the TGS in A's realm receives request and grants ticket to access TGS in B's realm.
- Protocol extension simple: two additional steps.
  But for n realms, $O(n^2)$ key distribution problem (in Version 4).

Inter-realm communication



Limitations of Kerberos 4

- M1: encryption is not needed, but attacker can flood KAS

$$A \rightarrow KAS: \ A, \ TGS$$

- M2: double encryption is redundant; eliminated in Kerberos 5.

$$\mathrm{KAS} \rightarrow \mathrm{A}: \ \{K_{\mathrm{A,TGS}}, \ TGS, \ \mathcal{T}_1, \ \{\mathrm{A}, \ \mathrm{TGS}, \ K_{\mathrm{A,TGS}}, \ \mathcal{T}_1\}_{K_{\mathrm{KAS,TGS}}}\}_{K_{\mathrm{AS}}}$$

- Relies on (roughly) synchronized and uncompromised clocks.
  If the host is compromised, then the clock can be compromised and replay is easy.

Some of these limitations also apply to Kerberos 5.

Bibliography

- MIT: `http://web.mit.edu/kerberos/`.
- William Stallings. *Cryptography and Network Security. Principles and Practice*, 7th ed., Prentice Hall, 2016.
- Matt Bishop. *Computer Security (Art and Science)*. Pearson, 2003.
- Kaufman, Perlman, Speciner. *Network Security: Private Communication in a Public World*, Prentice Hall, 2002.
- Bruce Schneier. *Applied Cryptography*, John Wiley & Sons, 1996 (and 20th anniversary edition in 2016).
- John Clark and Jeremy Jacob: A survey of authentication protocol literature, 1997.
- Catherine Meadows: Open Issues in Formal Methods for Cryptographic Protocol Analysis. Proceedings of DISCEX'00, IEEE Computer Society Press, 2000.
- Martín Abadi and Roger Needham: Prudent Engineering Practice for Cryptographic Protocols. IEEE Transactions on Software Engineering, 22(1):2-15, 1996.
- David Basin, Sebastian Mödersheim, Luca Viganò: OFMC: A symbolic model checker for security protocols, International Journal of Information Security 4(3):181–208, 2005.

Cryptography (and Information Security)
6CCS3CIS / 7CCSMCIS

Prof. Luca Viganò

Department of Informatics
King's College London, UK

First term 2019/20

Lecture 5

Roadmap

The sections labeled with (*) contain additional material that will contibute to your general knowledge but will not be subject of exam questions.

Outline

1. Passwords
2. Zero-knowledge protocols
3. Social engineering
4. Malware, viruses, worms, DoS, DDoS, buffer overflows ... (*)
5. Firewalls (*)

Table of contents I

1 Passwords

2 Zero-knowledge protocols

3 Social engineering

4 Malware, viruses, worms, DoS, DDoS, buffer overflows ... (*)

5 Firewalls (*)

Authentication

Data or services available only to authorized identities

- Authentication is verification of identity of a person or system.
- Some form of authentication is a pre-requisite if we wish to allow access to services or data to some people but deny access to others, using an access control system.
- Methods for authentication are often characterised as:
  - **something you have**, e.g. an entrycard,
  - **something you know**, e.g. a password or secret key, or
  - **something you are**, e.g. a fingerprint, signature, biometric.
- Also, where you are may be implicitly or explicitly checked. Several methods can be combined for extra security.
- Examples of violation: using cryptanalysis to break a cryptographic algorithm and learn a secret key; purporting to be somebody else (identity theft) by faking email, IP spoofing, or stealing a private key and signing documents.

Strong authentication

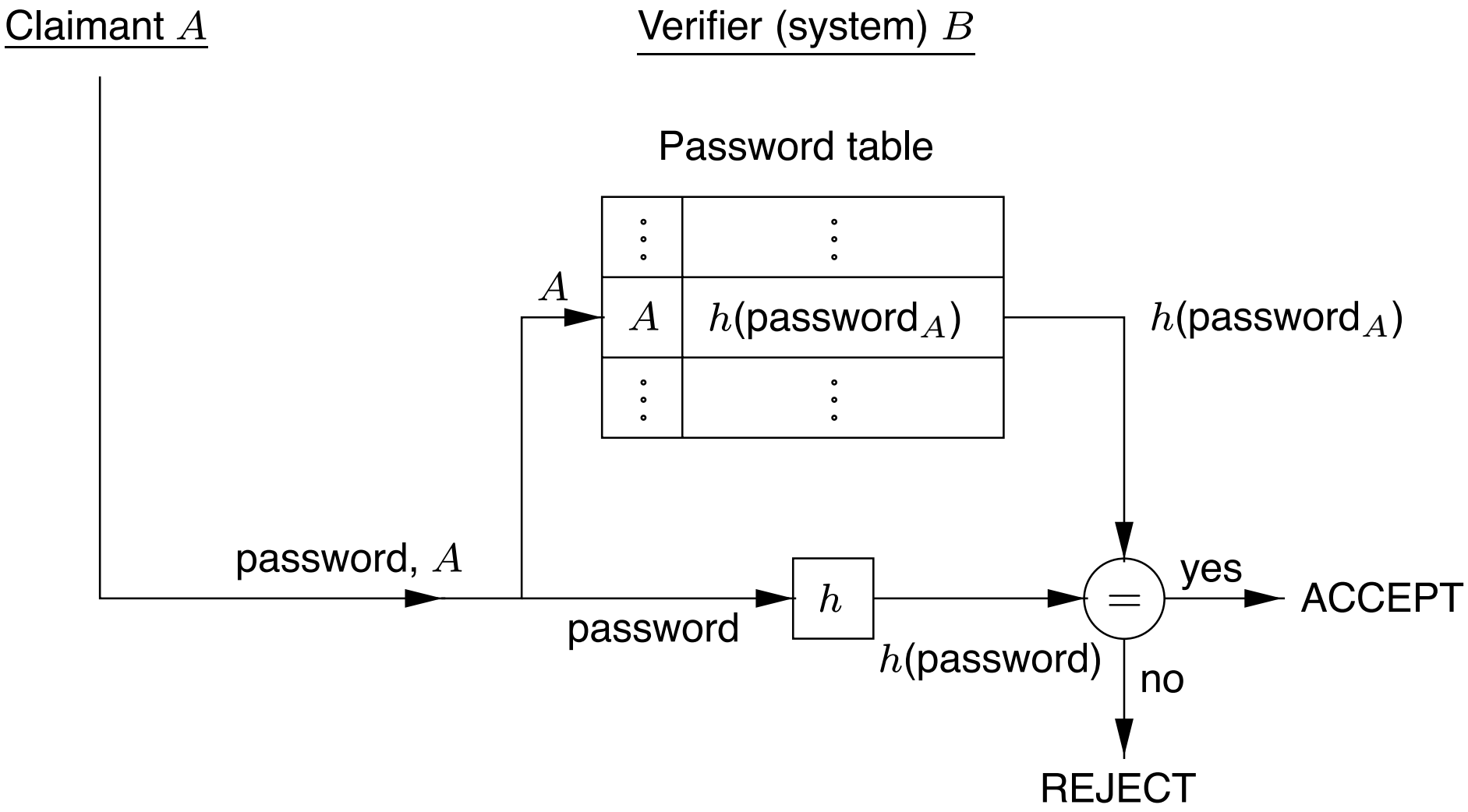
- **Cryptographic challenge-response protocols**: one entity (the claimant) “proves” its identity to another entity (the verifier) by demonstrating knowledge of a secret known to be associated with that entity, without revealing the secret itself to the verifier during the protocol.
  - This is done by providing a response to a time-variant challenge, where the response depends on both the entity’s secret and the challenge.
  - The challenge is typically a number chosen by one entity (randomly and secretly) at the outset of the protocol.
  - We have seen examples of this based on symmetric and public-key cryptography, using nonces and timestamps.
  - Later we will see examples based on zero-knowledge proofs.

Weak authentication

- Conventional password schemes involve time-invariant passwords, which provide so-called **weak authentication**.
- Basic idea:
  - A password (typically a string of 6 to 10 or more characters) serves as a shared secret between the user and system.
  - User enters a (userid, password) pair, where
    - userid is a claim of identity, and
    - password is the evidence supporting the claim.
  - System checks that the password matches corresponding data it holds for that userid.
  - Demonstration of knowledge of this secret (by revealing the password itself) is accepted by the system as corroboration of the entity's identity.

Weak authentication

For instance, using hashes:



Password rules

But we are not very good at following these rules

- Typical net user needs at least nine passwords
- Men store passwords on paper or PC
- Women use familiar names
- 30% never change passwords
- 29% less than once a year
- 35% of people use the same password for multiple applications
- 60% of people cycle two passwords across all applications
- 70% have forgotten a password at least once

Password rules

- Passwords found in any on-line or available list of words may be uncovered by an adversary who tries all words in this list, using a so-called **dictionary attack**.
- Since dictionary attacks are successful against predictable passwords, some systems impose **password rules** to discourage or prevent users from using “weak” passwords.

Password rules: examples

- Typical password rules include
  - lower bound on the password length (e.g., 8 or 12 characters),
  - at least one character from each of a set of categories (e.g., uppercase, numeric, non-alphanumeric),
  - checks that candidate passwords are not found in on-line or available dictionaries, and are not composed of account-related information such as userids or substrings thereof.

Password rules: entropy

- Objective of password rules is to increase the **entropy** (rather than just length) of user passwords beyond the reach of dictionary and exhaustive search attacks.
  - Entropy here refers to the uncertainty in a password.
  - If all passwords are equally probable, then the entropy is maximal and equals the base-2 logarithm of the number of possible passwords.
- While ideally arbitrary strings of n characters would be equiprobable as user-selected passwords, most (unrestricted) users select passwords from a small subset of the full password space (e.g., short passwords; dictionary words; proper names; lowercase strings).
  - Such weak passwords with low entropy are easily **guessed**.
  - Studies indicate that a large fraction of user-selected passwords are found in typical (intermediate) dictionaries of only 150000 words, while even a large dictionary of 250000 words represents only a tiny fraction of all possible n-character passwords.

Password rules: entropy example

- Suppose passwords consist of strings of 7-bit ASCII characters.
- Each has a numeric value in the range 0-127.
- ASCII codes 0-31 are reserved for control characters; 32 is a space character; 33-126 are keyboard-accessible printable characters; and 127 is a special character.
- Number of distinct *n*-character passwords composed of typical combinations of characters, indicating an upper bound on the security of such password spaces.
- Bitsize of password space for various character combinations:

$\rightarrow c$ $\downarrow n$	26 (lowercase)	36 (lowercase alphanumeric)	62 (mixed case alphanumeric)	95 (keyboard characters)
5	23.5	25.9	29.8	32.9
6	28.2	31.0	35.7	39.4
7	32.9	36.2	41.7	46.0
8	37.6	41.4	47.6	52.6
9	42.3	46.5	53.6	59.1
10	47.0	51.7	59.5	65.7

  - Number of *n*-character passwords, given *c* choices per character, is c^n.
  - Table gives the base-2 logarithm of this number of possible passwords.

Password: ageing and passphrases

- Another procedural technique intended to improve password security is **password ageing**.
  - A time period is defined limiting the lifetime of each particular password (e.g., 30, 90 or 180 days).
  - This requires that passwords be changed periodically.

- To allow greater entropy without stepping beyond the memory capacity of human users, **passphrases** may be used.
  - Passphrase is hashed down to a fixed-size value, which plays the same role as a password.
  - Idea is that users can remember phrases easier than random character sequences.
  - But it requires more typing.

Personal identification numbers (PINs)

Personal identification numbers (PINs) fall under the category of fixed (time-invariant) passwords.

- PINs are most often used in conjunction with “something possessed”, typically a physical token such as a plastic banking card with a magnetic stripe, or a chipcard.
- To prove one’s identity as the authorized user of the token, and gain access to the privileges associated therewith, entry of the correct PIN is required when the token is used.
- This provides a second level of security if the token is lost or stolen.
- PINs may also serve as the second level of security for entry to buildings which have an independent first level of security (e.g., a security guard or video camera).

Multi-factor authentication (MFA)

Multi-factor authentication (MFA) is an authentication method in which a user has to successfully present authentication factors from at least two of the three categories:

- knowledge factors ("things only the user knows"), such as passwords,
- possession factors ("things only the user has"), such as ATM cards or smartphones,
- inherence factors ("things only the user is"), such as biometrics.

Requiring more than one independent factor increases the difficulty of providing false credentials.

One-time passwords

- A major security concern of fixed password schemes is eavesdropping and subsequent replay of the password.
- A partial solution is **one-time passwords**: each password is used only once.
- Such schemes are safe from passive adversaries who eavesdrop and later attempt impersonation.
- Variations include:
  - **Shared lists of one-time passwords**: user and system use a sequence or set of t secret passwords, (each valid for a single authentication), distributed as a pre-shared list.
  - **Sequentially updated one-time passwords**: initially only a single secret password is shared, and then, during authentication using password i, user creates and transmits to system a new password (password $i + 1$) encrypted under a key derived from password i.
  - **One-time password sequences based on a one-way function**.

Table of contents I

1 Passwords

2 Zero-knowledge protocols

3 Social engineering

4 Malware, viruses, worms, DoS, DDoS, buffer overflows ... (*)

5 Firewalls (*)

Idea

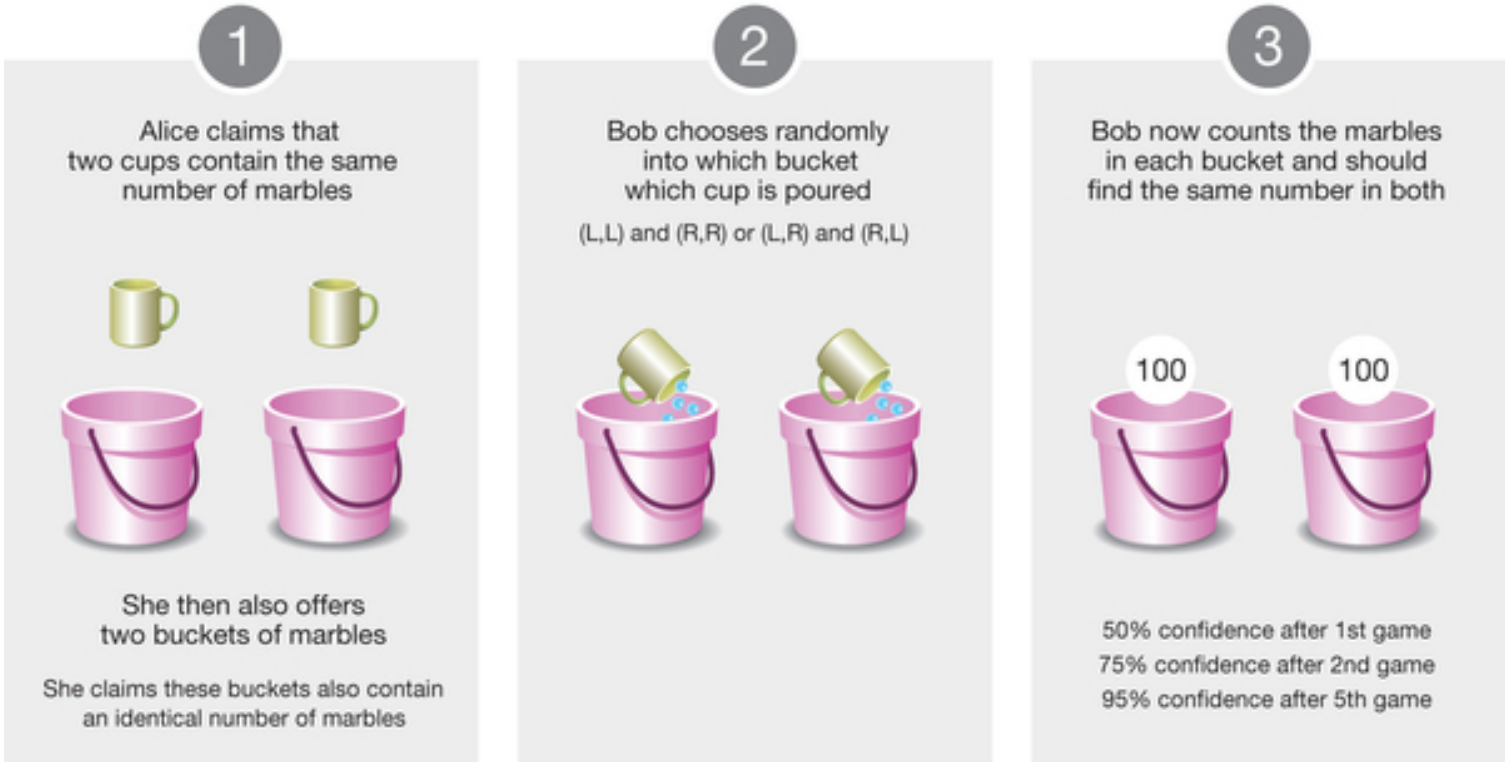
What do weapons of mass destruction, a drink and Ali Baba's cave all have in common?

Zero-knowledge proofs

In zero-knowledge proofs we can usually specify a statement that is being proved.

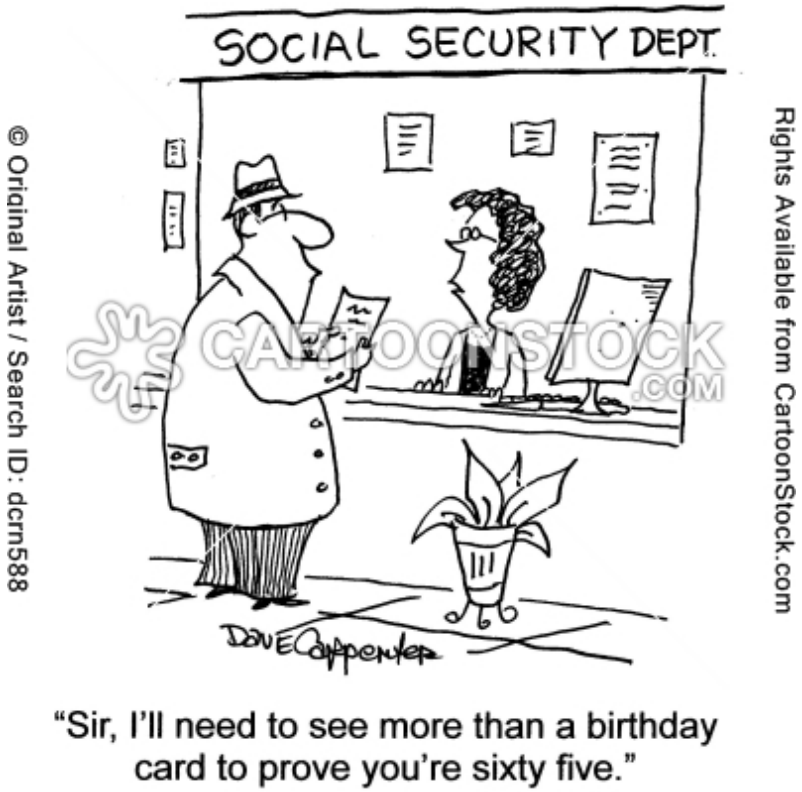
- Definitely, that statement is revealed to the verifier
- The verifier (or others) should not learn anything else
- Everybody can draw conclusions from everything they learned

Zero-knowledge proofs: nuclear warhead verification

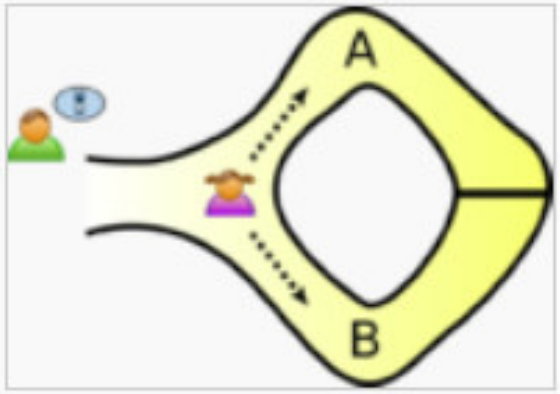


- Show that two or more putative warheads have identical neutron transmission and emission counts under irradiation by high-energy neutrons.
- Reliably verify that the inspected warhead is authentic while avoiding disclosure of information about its design.

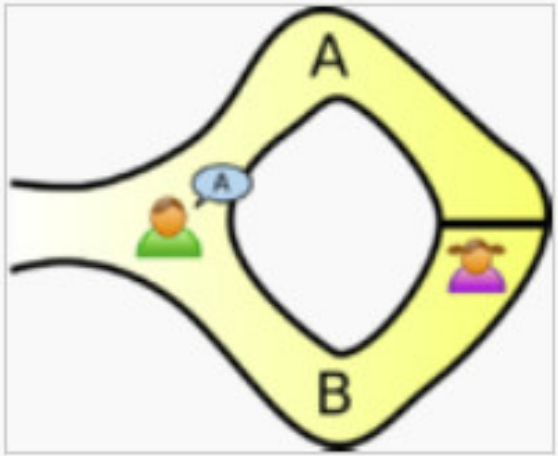
Zero-knowledge proofs: proof of age



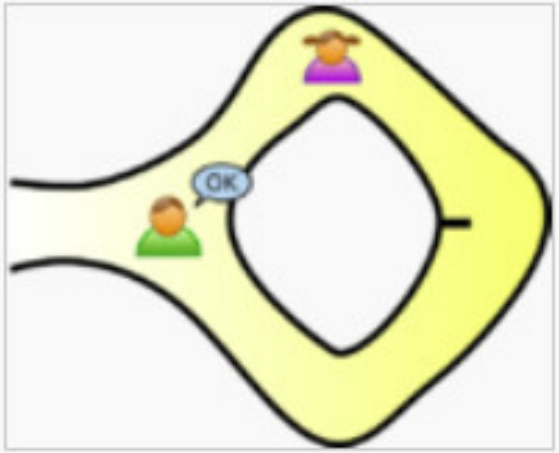
Zero-knowledge proofs: Ali Baba's cave



Peggy randomly takes either path A or B, while Victor waits outside

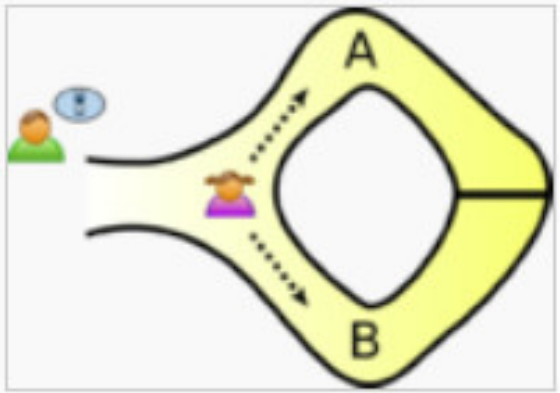


Victor chooses an exit path

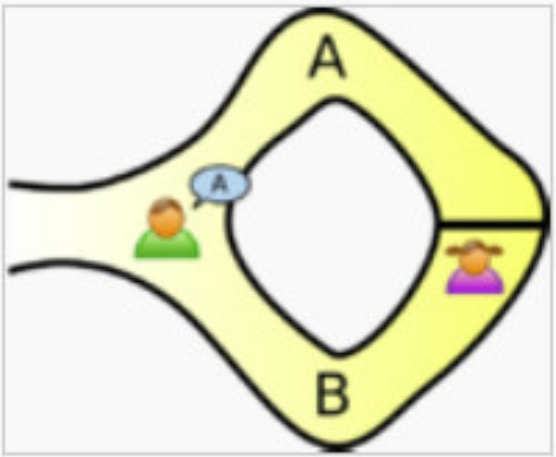


Peggy reliably appears at the exit Victor names

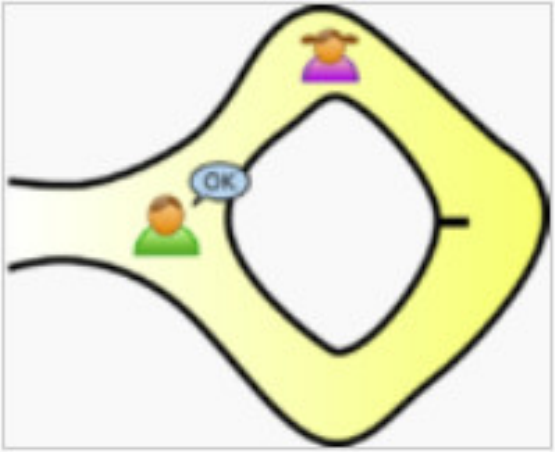
- A cave has a door that opens only when a secret word is spoken.
- Peggy (the Prover) wants to convince Victor (the Verifier) that she knows the secret word, but without reveling it!
- If they walk to the door together, Peggy will be able to open it but then Victor will learn the secret word.
- So, they carry out a zero-knowledge protocol.



Peggy randomly takes either path A or B, while Victor waits outside

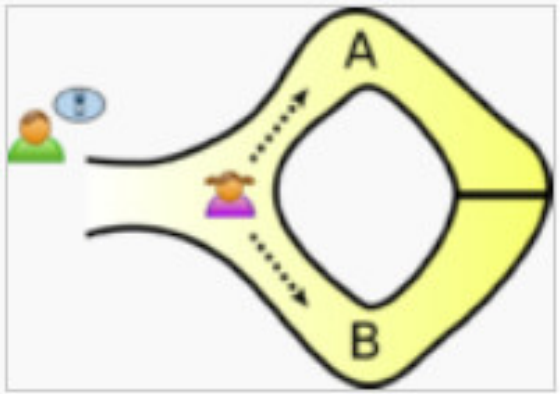


Victor chooses an exit path

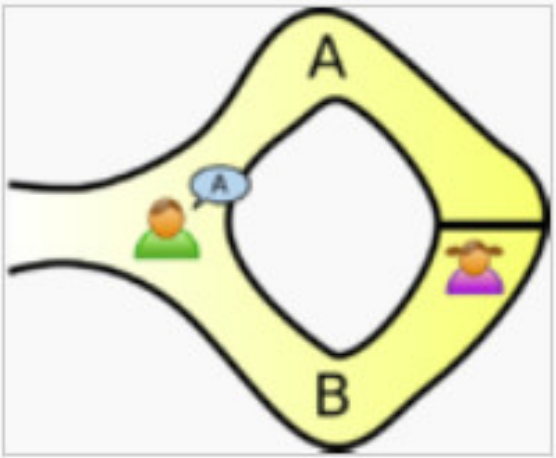


Peggy reliably appears at the exit Victor names

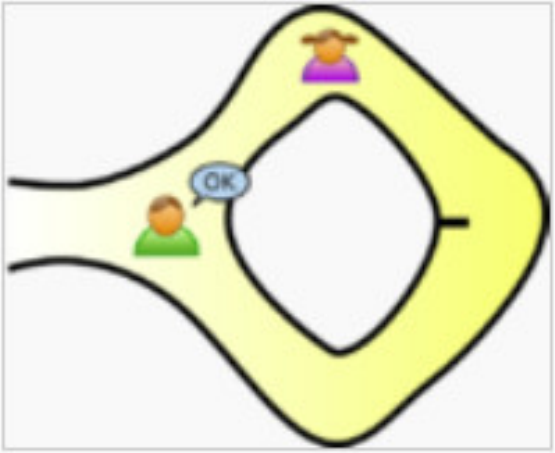
1. Victor stands at the cave's entrance, while Peggy walks to the door.
2. Victor walks to the bifurcation of the cave's paths and shouts to Peggy either to
   - come out of the left path A or
   - come out of the right path B.
3. Peggy complies, using the secret word to open the door, if needed.
4. Peggy and Victor repeat the experiment (steps 1-3) *n* times.



Peggy randomly takes either path A or B, while Victor waits outside



Victor chooses an exit path



Peggy reliably appears at the exit Victor names

- Now assume that Peggy doesn't actually know the secret word.
- Then she can only come out the way she went in.
  - After 1 round, she has only 1 chance out of 2 of fooling Victor.
  - After n rounds, she has only 1 chance out of 2^n of fooling Victor.
- So, after a while, Victor will be convinced that Peggy knows the secret.
- In other words: Peggy wins if she passes the test all of the time.
  - The probability that Peggy wins is very low if she does not know the secret word: after n rounds, it is $(1/2)^n = \frac{1}{2^n}$.

Zero-knowledge proofs: the idea

- In a challenge-response protocol, the Prover proves that she knows a secret.
  - If a symmetric cryptosystem is used, then the Verifier also knows the secret.
  - If a public-key signature system is used, then Verifier does not know the secret.
- An example of a zero-knowledge protocol is the Fiat-Shamir Identification Protocol.

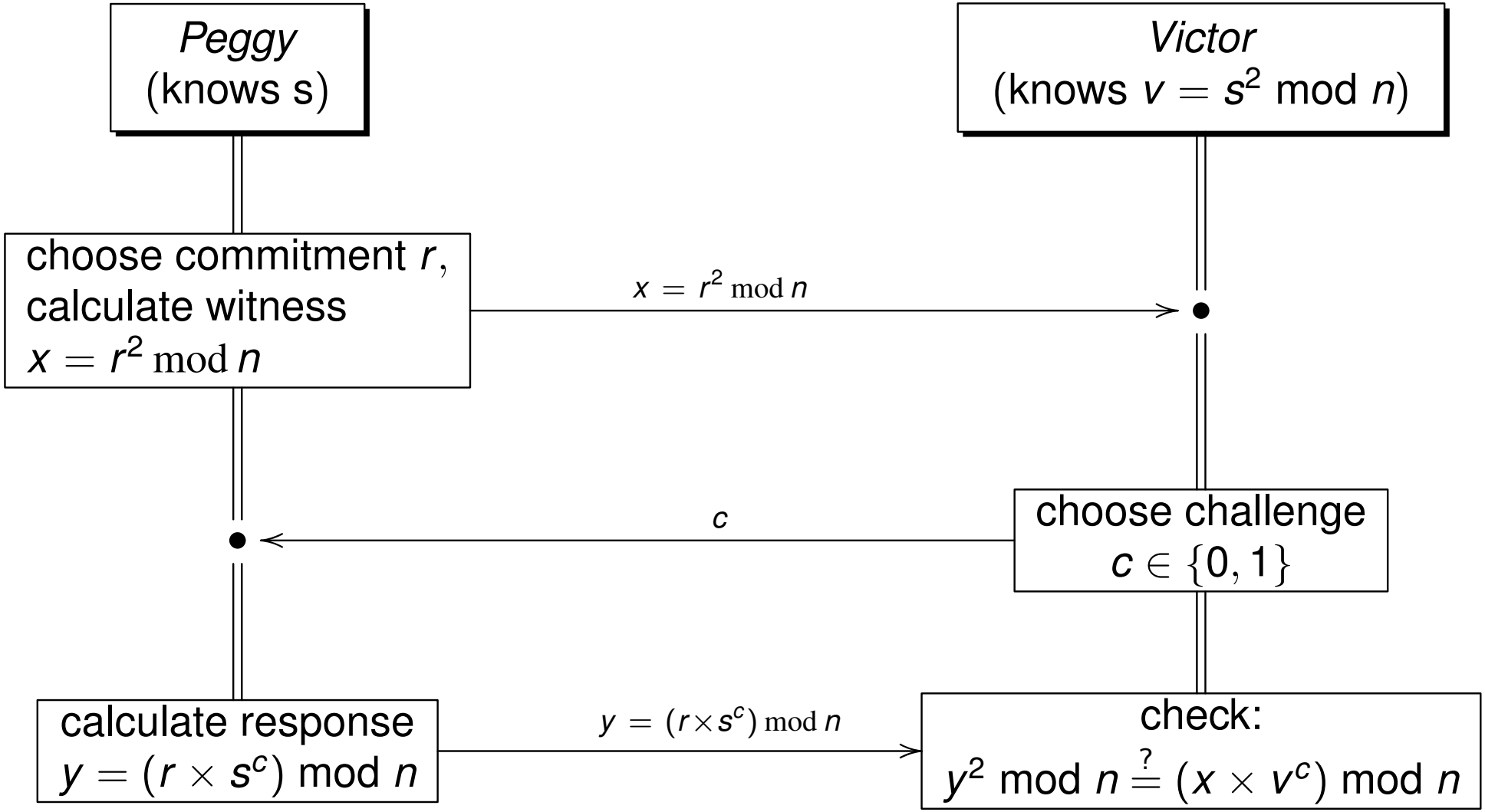
Example: Fiat-Shamir Identification Protocol

- Three principals:
  - **Prover Peggy**,
  - **Verifier Victor** and
  - **Trusted Third Party Trent**.
- Setup:
  - Trent chooses two large prime numbers p and q to calculate $n = p \times q$.
  - n is announced to the public, whereas p and q are kept secret.
  - Peggy chooses a **secret number** s between 1 and $n - 1$, and calculates $v = s^2 \bmod n$.
    Peggy keeps s as her **private key** and registers v as her **public key** with the third party.
- Victor knows $v = s^2 \bmod n$, but does not know s.
- Squaring modulo n is easy to compute but square root modulo n is probably not (we believe...).
- **Goal**: Peggy wants to convince Victor that she knows the secret s but Victor should not learn s!

- Verification of Peggy by Victor then proceeds in 4 steps:
  1. Peggy chooses a random number r between 0 and $n-1$.
     r is called the **commitment**.
     Peggy then calculates the **witness** $x = r^2 \bmod n$ and sends it to Victor.
  2. Victor sends the **challenge** c to Peggy, where c is either 0 or 1.
  3. Peggy calculates the **response** $y = (r \times s^c) \bmod n$ and sends it to Victor to show that she knows her private key s modulo n.
     She claims to be Peggy.
  4. Victor calculates $y^2 \bmod n$ and $(x \times v^c) \bmod n$. If these values are congruent, then Peggy either knows the value of s (she is honest) or she has calculated the value of y in some either ways (dishonest) because in modulo n arithmetic we actually have that

$$y^2 =_n (r \times s^c)^2 =_n r^2 \times s^{2c} =_n r^2 \times (s^2)^c =_n x \times v^c$$

- The 4 steps constitute a **round**.
- The verification is repeated several times with the value of c equal to 0 or 1, chosen randomly.
- Peggy must pass the test in each round to be verified: if she fails one single round, the process is aborted.



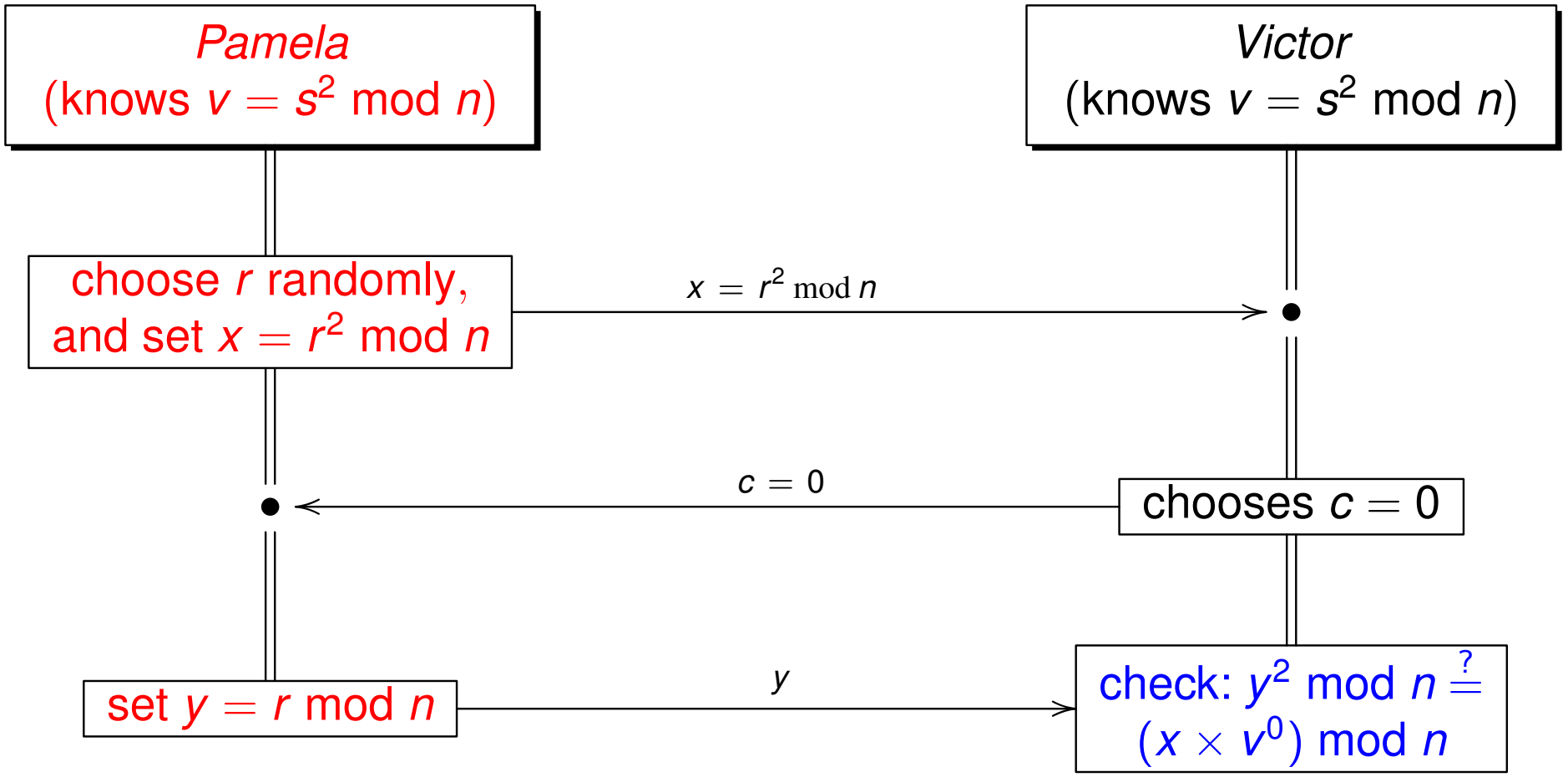
If the check

$$y^2 =_n (r \times s^c)^2 =_n r^2 \times s^{2c} =_n r^2 \times (s^2)^c =_n x \times v^c$$

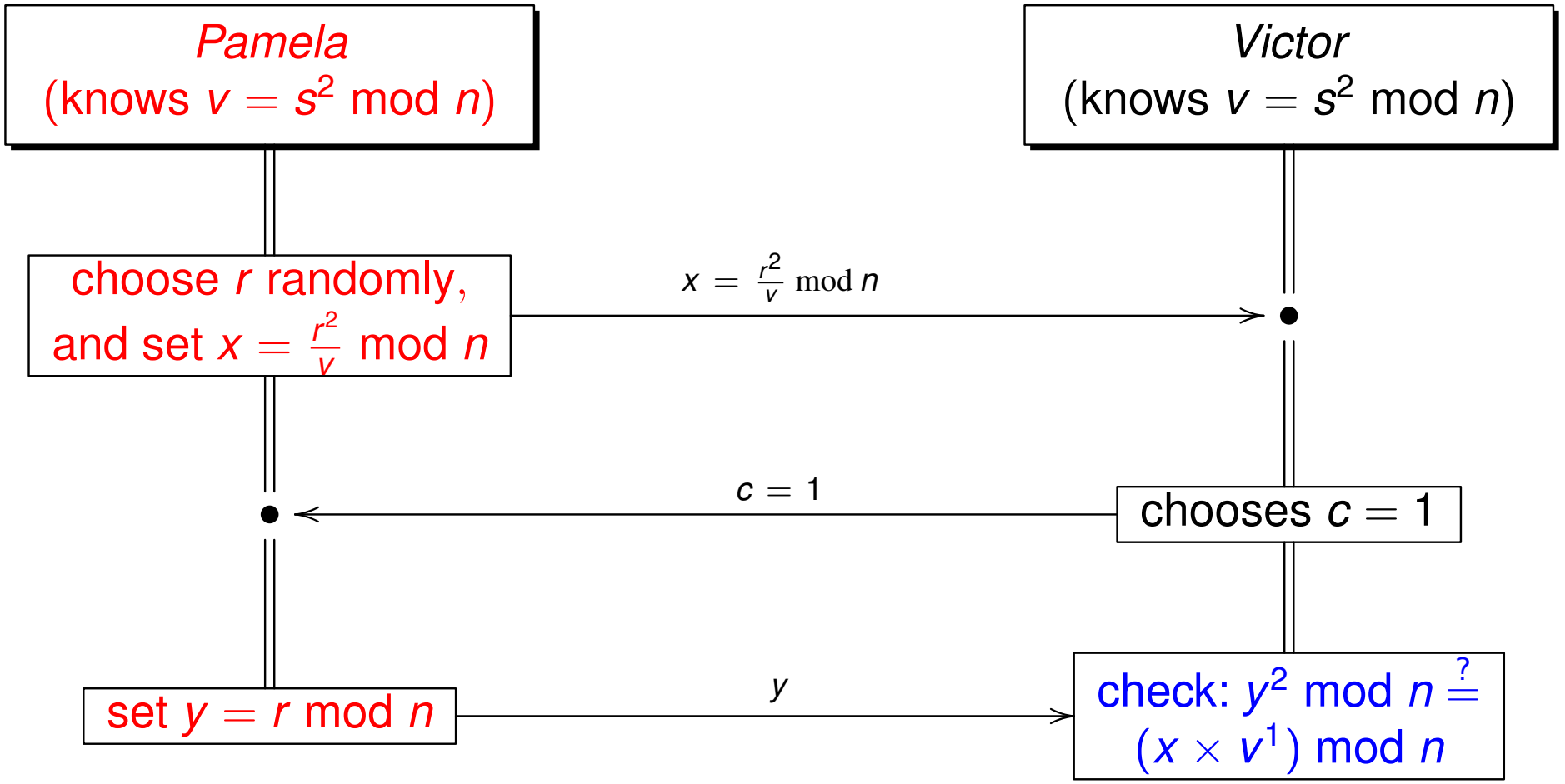
returns a yes, then verification is probable; otherwise, the process is aborted.

Fiat-Shamir: Trying to Cheat

Pamela does not know the secret s, but tries to prove its knowledge.
Pamela guesses that Victor is going to choose $c = 0$ (if she guesses wrong, then she loses).



Pamela guesses that Victor is going to choose $c = 1$ (if she guesses wrong, then she loses).



So, Pamela must find numbers x and y such that $x \times v =_n (x \times s^2)$.
Choose y randomly and then set $\frac{r^2}{v} \bmod n$ (division modulo n is also easy!).

Fiat-Shamir: Trying to Cheat

- Pamela has a strategy to cheat if $c = 0$:

Choose r randomly, set $x = r^2 \bmod n$.

- Pamela has a strategy to cheat if $c = 1$:

Choose r randomly, set $x = \frac{r^2}{v} \bmod n$.

- If $c \in \{0, 1\}$ is randomly chosen, Pamela has thus a chance of $\frac{1}{2}$ to cheat.
- If Victor accepts only after n successful rounds, the chance to cheat is only $\frac{1}{2^n}$.
- We can conclude that

Pamela has no strategy to cheat for unpredictable c.

Fiat-Shamir: Curious Victor

Victor would like to learn the secret s...
but we can conclude that

Zero-Knowledge Property

Victor learns nothing except the proved statement.

Other zero-knowledge protocols

- Feige-Fiat-Shamir Protocol:
  - similar to the Fiat-Shamir Protocol except that it is uses
    - a vector of private keys $[s_1, s_2, \ldots, s_k]$,
    - a vector of public keys $[v_1, v_2, \ldots, v_k]$,
    - a vector of challenges $[c_1, c_2, \ldots, c_k]$.
- Guillou-Quisquater Protocol:
  - an extension of the Fiat-Shamir Protocol in which fewer number of rounds can be used to prove the identity of the claimant.

Table of contents I

1 Passwords

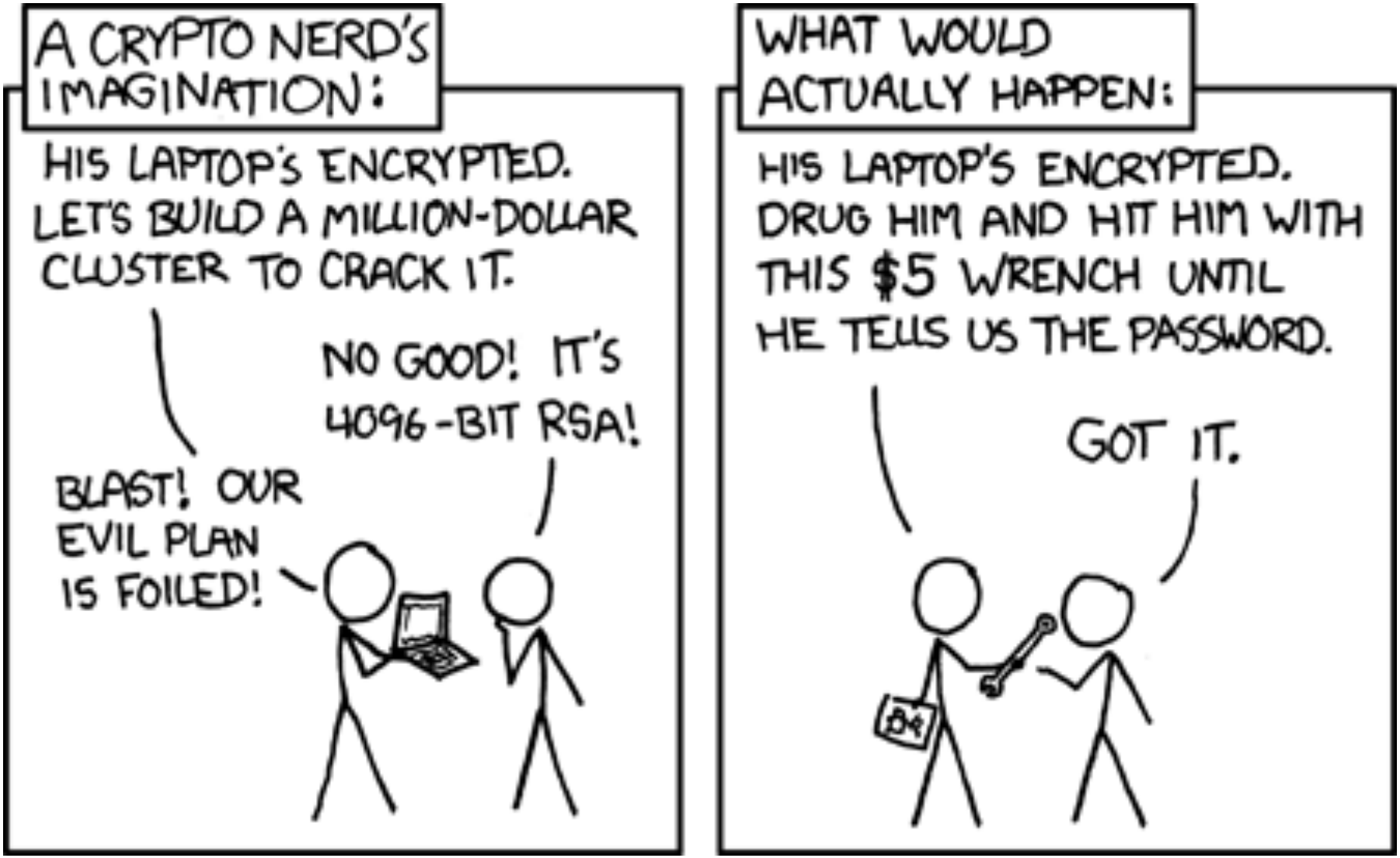
2 Zero-knowledge protocols

3 Social engineering

4 Malware, viruses, worms, DoS, DDoS, buffer overflows ... (*)

5 Firewalls (*)

What would actually often happen



Social engineering

Social engineering

- The acquisition of security-sensitive information or unauthorized access privileges by an outside attacker, based upon the abuse of a trust relationship (resulting possibly in the building of an inappropriate trust relationship with insiders).
- In other words, it is the psychological manipulation of people into performing actions or into providing valuable information or access to confidential information.

- A type of confidence trick for the purpose of information gathering, fraud, or system access, it differs from a traditional “con” in that it is often one of many steps in a more complex fraud scheme.
- Why try to hack through someone’s security system when you can get a user to open the door for you?

Social sciences and information security

- The term “social engineering” as an act of psychological manipulation is also associated with the **social sciences**.
- Famous social engineers:
  - **Kevin Mitnick**, author of “The Art of deception”, “The Art of intrusion” and “Ghost in the wires”. See `https://www.mitnicksecurity.com`.
    It is much easier to trick someone into giving a password for a system than to spend the effort to crack into the system.
  - **Christopher Hadnagy**, author of “Social Engineering: The Art of Human Hacking” and “Unmasking the Social Engineer: The Human Element of Security”. See `http://www.social-engineer.org`.
  - **Frank Abagnale**, author of “The Art of the Steal” and “Catch Me If You Can” (see also Steven Spielberg’s movie).
  - ...
  - See also the TV show **The real hustle** for many social engineering scams.

Human-based social engineering: impersonation

Help desks are the most frequent targets of social engineering attacks.

- Social engineer calls the help desk, which most of the time is helpful.
  - Social engineer will often know names of employees (e.g., through www) and the lingo of the company.
- **Important User**: a common ploy is to pretend not to be only an employee, but an important user (e.g., a vice president, or even better: his/her secretary).
  - Help desk is less likely to turn down a request coming from a high-level official, or from a fellow employee under power pressure.
  - Social engineer may threaten to report the employee to their supervisor.
- **Third-party Authorization**: social engineer may have obtained the name of someone in the organization who has authority to grant access to information.

  "Before he went on vacation, Mr. Big said I should call you to get this information."
- **Tech Support**: social engineer pretends to be someone from the infrastructure-support groups.
  - The system is having a problem.
  - We need to log on to test the connection.

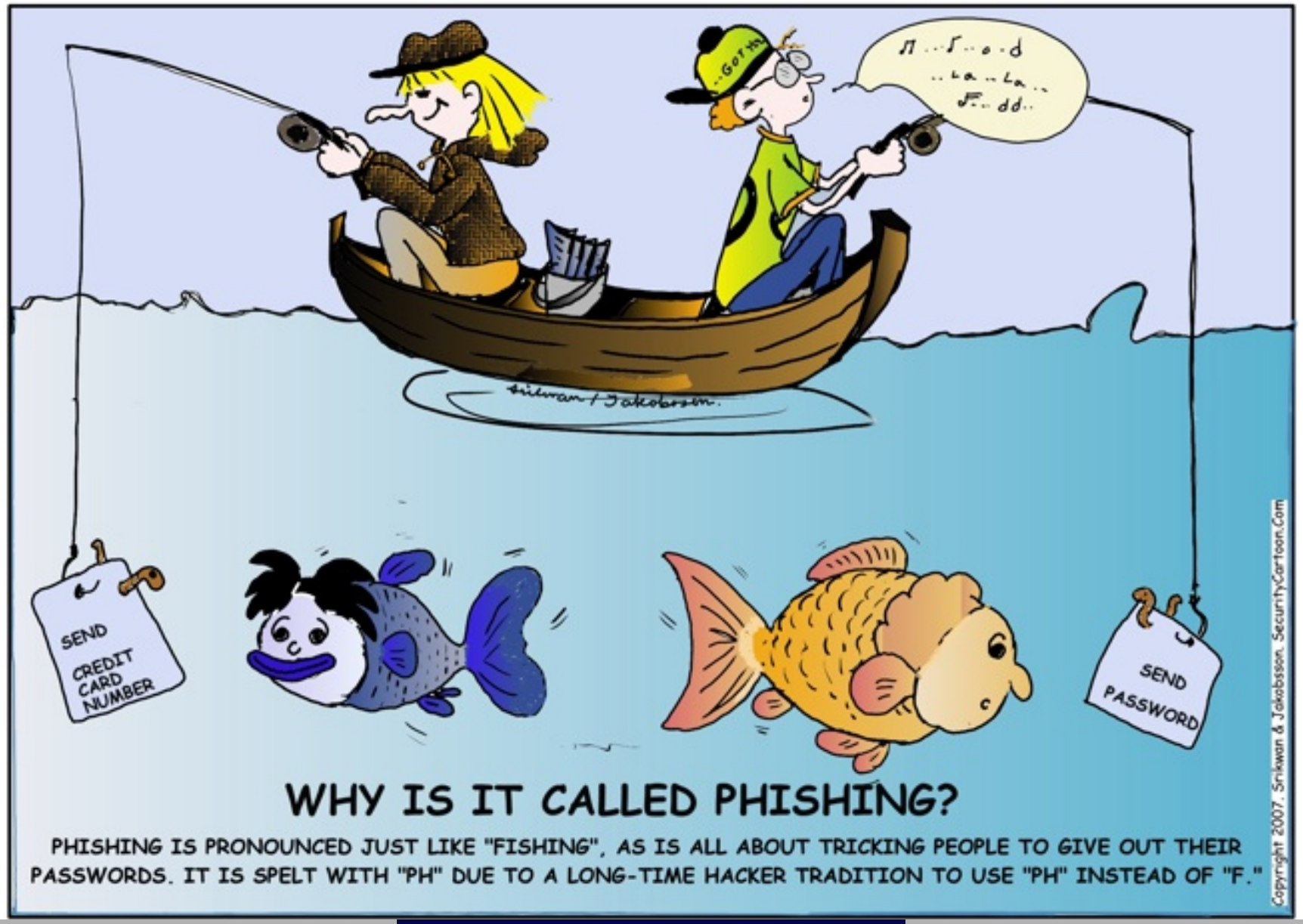
Computer-based social engineering: types

- **Mail attachments**:
  - Malware programs hidden in e-mail attachments (viruses, worms, “I love you”).
- **Websites**:
  - A common ploy is to offer something free or a chance to win a lottery on a website.
  - To win requires an e-mail address and password, assuming, as is often the case, that victims will reuse their “standard” password.
- **Popup Windows**:
  - A window appears on the screen telling the victim he has lost his network connection and needs to reenter user name and password.
  - A program will then e-mail the social engineer with the information.
- **Fake wireless network**:
  - Social engineer sits in a hotel lobby, sets up a fake wireless network with an evocative name (e.g., using a minor variant of the hotel’s name).
  - Victim logs in, assuming that it must be the real hotel’s wireless.
- **...**

Computer-based social engineering: simple examples

- Social engineer sends to the victim a usb-key/CD/file labeled “Security patches to be installed” with appropriate company logo.
  - The victim might be a normal user or even a superuser (e.g., somebody in the helpdesk or computing support).
  - Once the malware has been installed, social engineer gains access to the remote computer.
    He might even send some acknowledgment (“Security patches have been successfully installed”) to the victim.
- There was a famous company that bragged about its security because none of its internal networks was connected to the Internet... but curiosity killed the cat!
  - As he could not penetrate via the net, social engineer loaded a virus on a usb key and threw the key over the perimeter wall of the company, so that it landed in the parking lot. Then he waited...
  - Some employee saw the key in the parking lot (and thought “it is within the company’s perimeter, so it must have been lost by some colleague”), picked it up and plugged into his/her computer to see to whom it belonged, thereby infecting the whole company.

Phishing, SMiShing, Vishing



Phishing, SMiShing, Vishing

Phishing

Attackers use Web-based services to launch attacks on those devices connected to the Web to acquire information such as usernames, passwords, and credit card details and other private data of the device owner by the intruder masquerading as a trustworthy friend in an electronic communication like e-mail and text.

SMiShing (= SMS + phishing)

A social engineering crime like phishing in that it uses the mobile devices and texts as baits to pull in the mobile device owner to divulge private and sometimes personal information.

Vishing (= voice + phishing)

It mostly uses the mobile device phone features facilitated by Voice over IP (VoIP) to gain access to private personal and financial information for the purpose of financial rewards.