Distributed Storage Systems

Security in Storage Systems

What is a secure system?

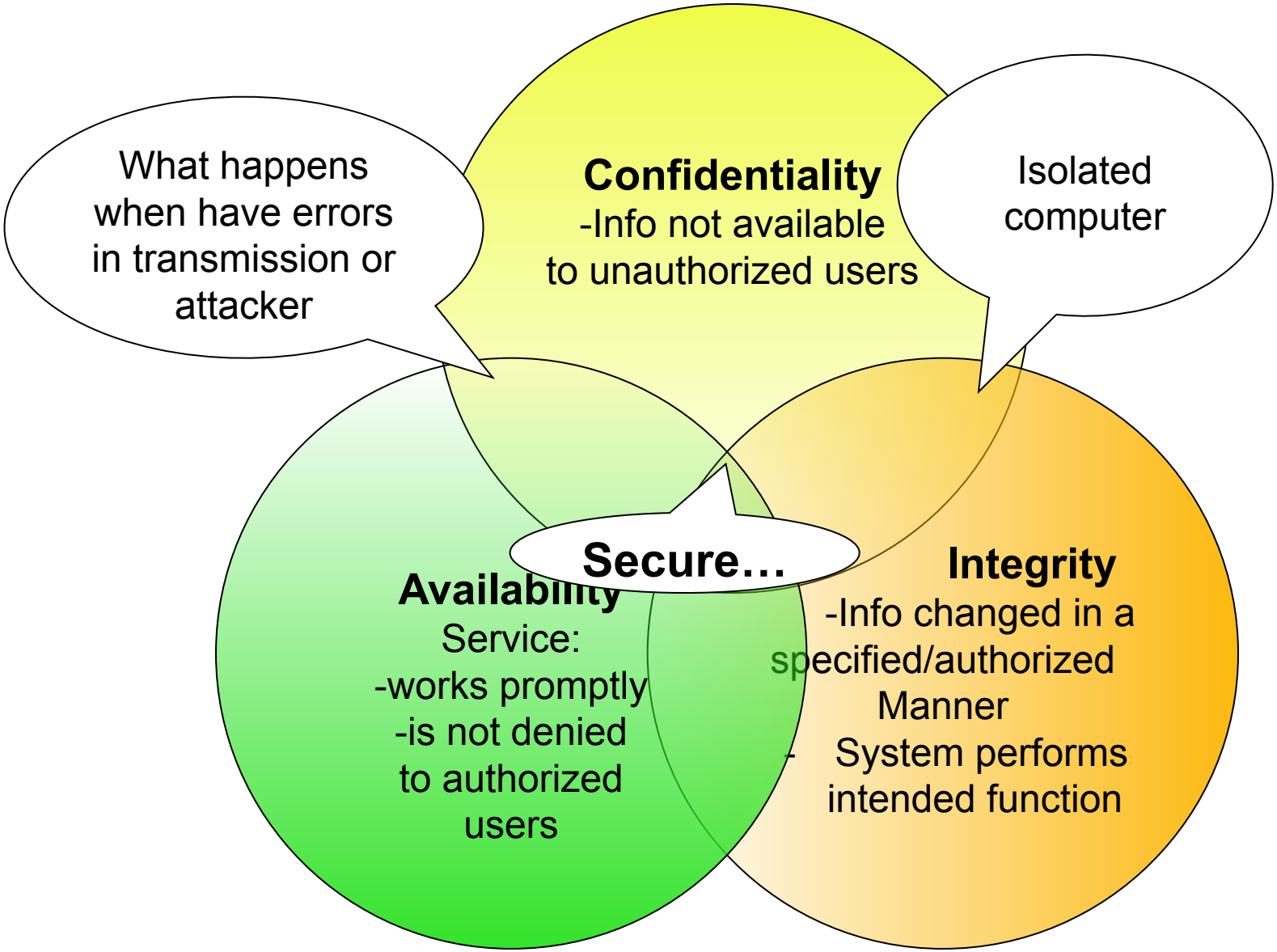
Deciding whether a system is secure or not, depends on your *definition* of “secure”

We must first decide what “secure” means to us, to then identify the threats and breaches to security that we care about.

What would make sense?

What would we like to protect?

CIA… but not the agency



General Security Principles (CIAA)

- Confidentiality
- Integrity
- Availability
- Authentication

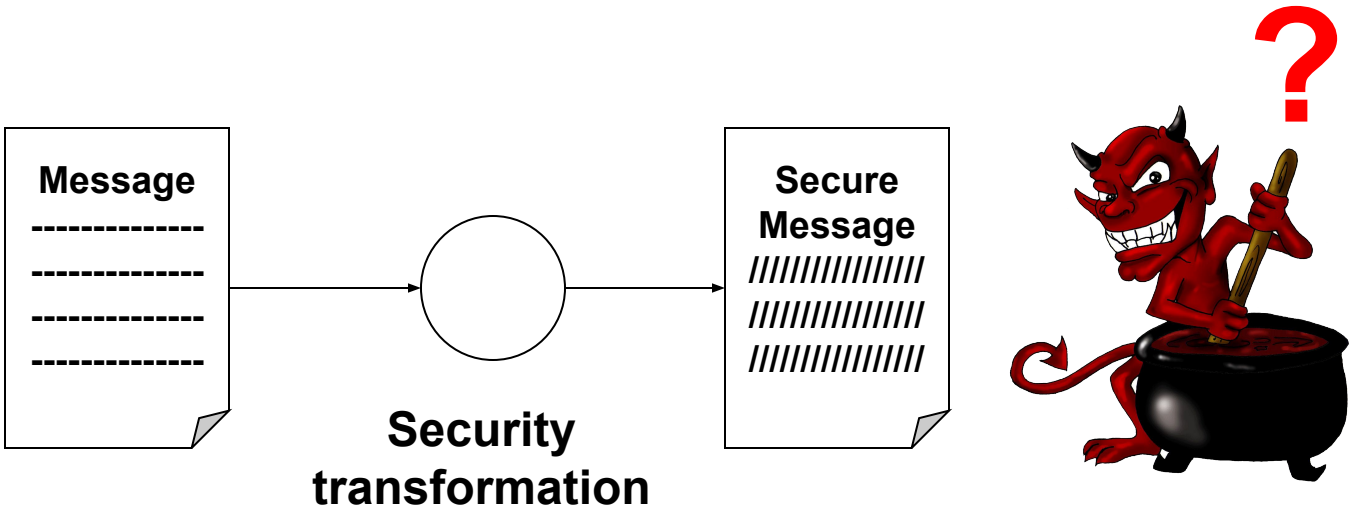
Security Principles in Storage Systems

- Confidentiality
  - Read information without proper authorization
  - The attacker gains admin access or escalates their legitimate user privileges
  - Typically addressed by encryption and authorization
- Integrity
- Availability
- Authentication

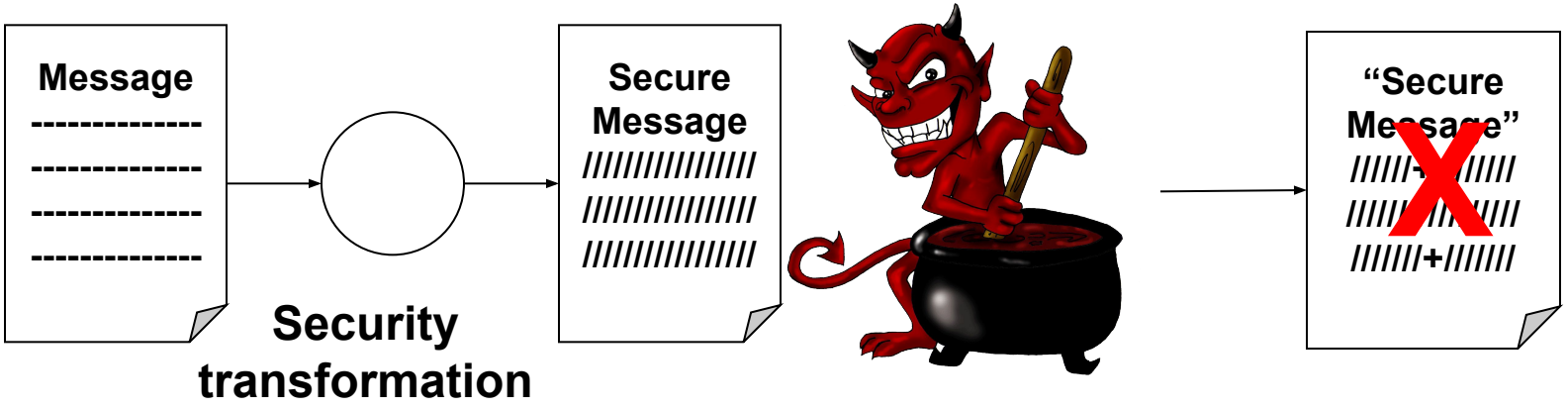
Security Principles in Storage Systems

- Confidentiality
- Integrity
  - Data does not change during storage
  - Hashing to detect and redundant storage to reconstruct compromised data
  - Replication, erasure coding, RAID, backup
- Availability
- Authentication

Confidentiality



Integrity



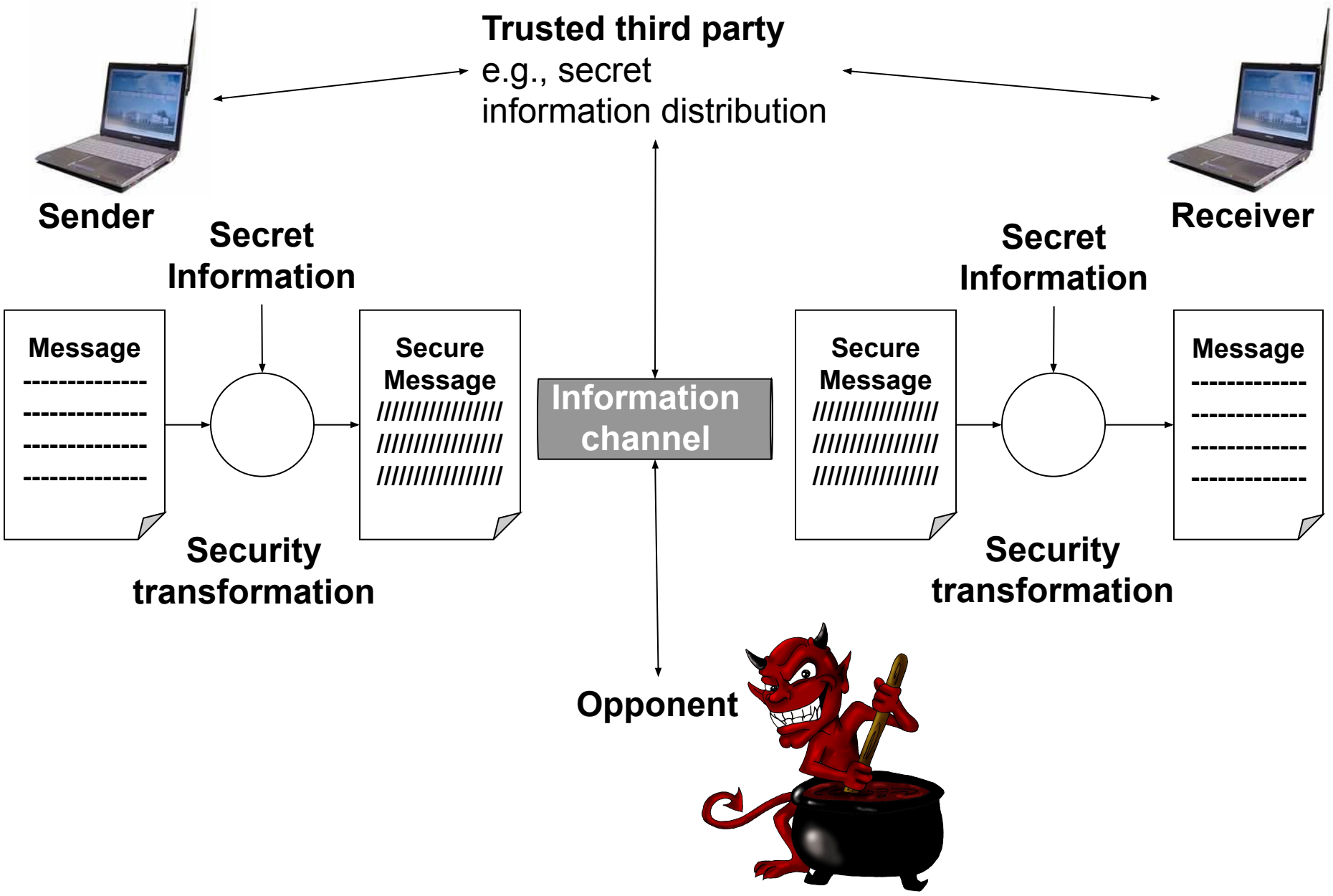
Security Principles in Storage Systems

- Confidentiality
- Integrity
- Availability
  - The stored data is available at any time
  - Redundant storage to account for hardware failure, downtime due to OS update or software bugs, DoS attacks, natural catastrophes, etc
  - Replication, erasure coding, RAID, backup
- Authentication

Security Principles in Storage Systems

- Confidentiality
- Integrity
- Availability
- Authentication
  - Users must prove their identity before being served
  - Internal services of the storage system authenticate among themselves

Security: (Simplified) Model



Security: Simple enough, right?

- Stating security requirements is straight-forward
- Designing mechanisms to meet this requirements: can be very complex
- Challenges:
  - Design of security mechanism: consider all potential attacks
  - Attacker may need only **one weakness**
  - Where to use security mechanisms?
  - Security mechanisms involve more than one algorithm or protocol
  - Need to provide secret information to authentic users

Vulnerabilities, Threats, Controls

- Vulnerability: a weakness in a security system
- Threat: circumstances that have a *potential* to cause harm
- Controls: means and ways to block a threat, which tries to exploit one or more vulnerabilities

Attacks

Exploitation one or more vulnerabilities by a threat

How? Defeat controls

- Attacks may be:
  - Successful
    - Breach of security
  - Unsuccessful
    - Controls block a threat trying to exploit a vulnerability

Types of Attacks

Passive attacks

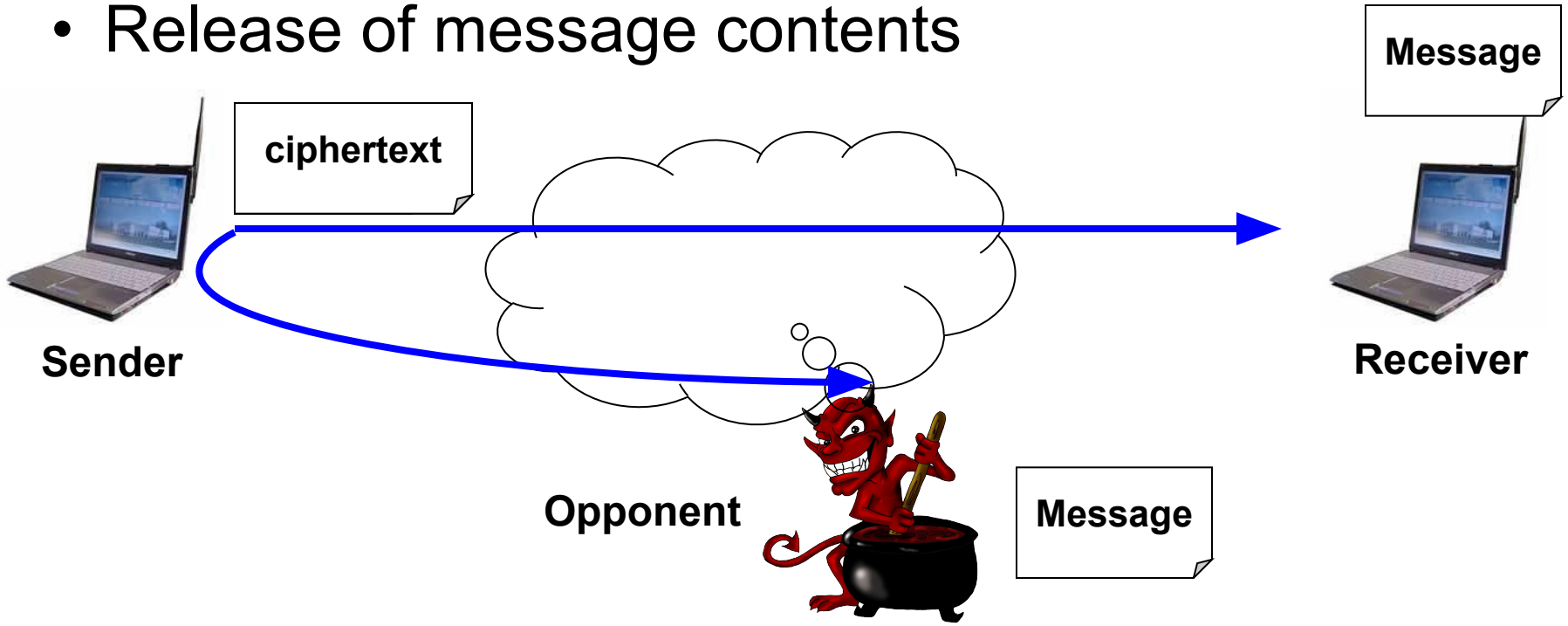
- Release of message contents
- Traffic analysis

Active attacks

- Masquerade
- Replay
- Modification of messages
- Denial of service
- …

Passive Attacks

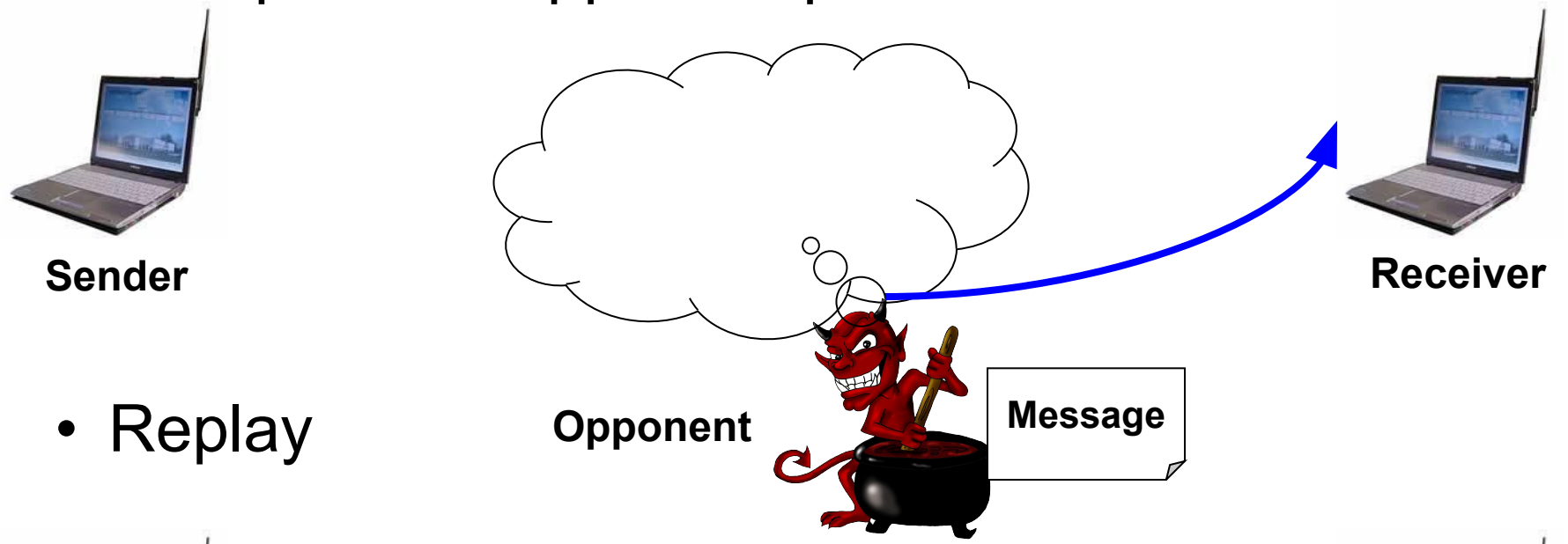
- Release of message contents



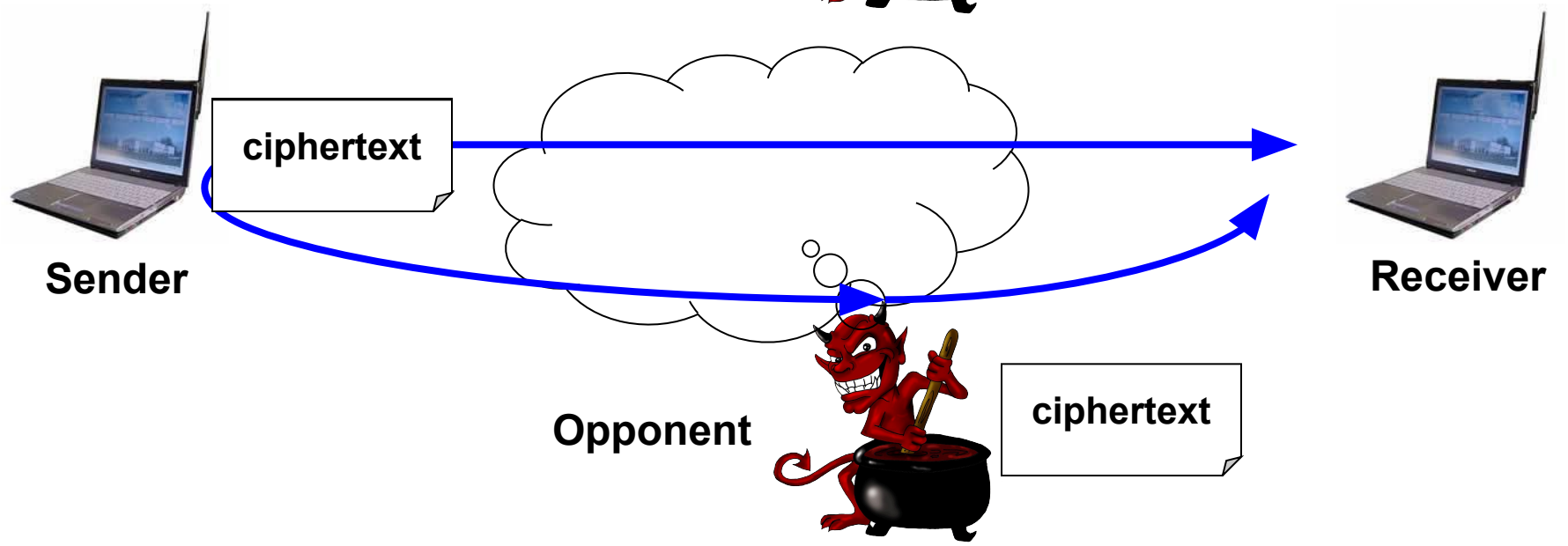
- Traffic analysis
  - Example presented before
  - Opponent observes pattern of messages

Active Attacks

- Masquerade: opponent pretends to be sender

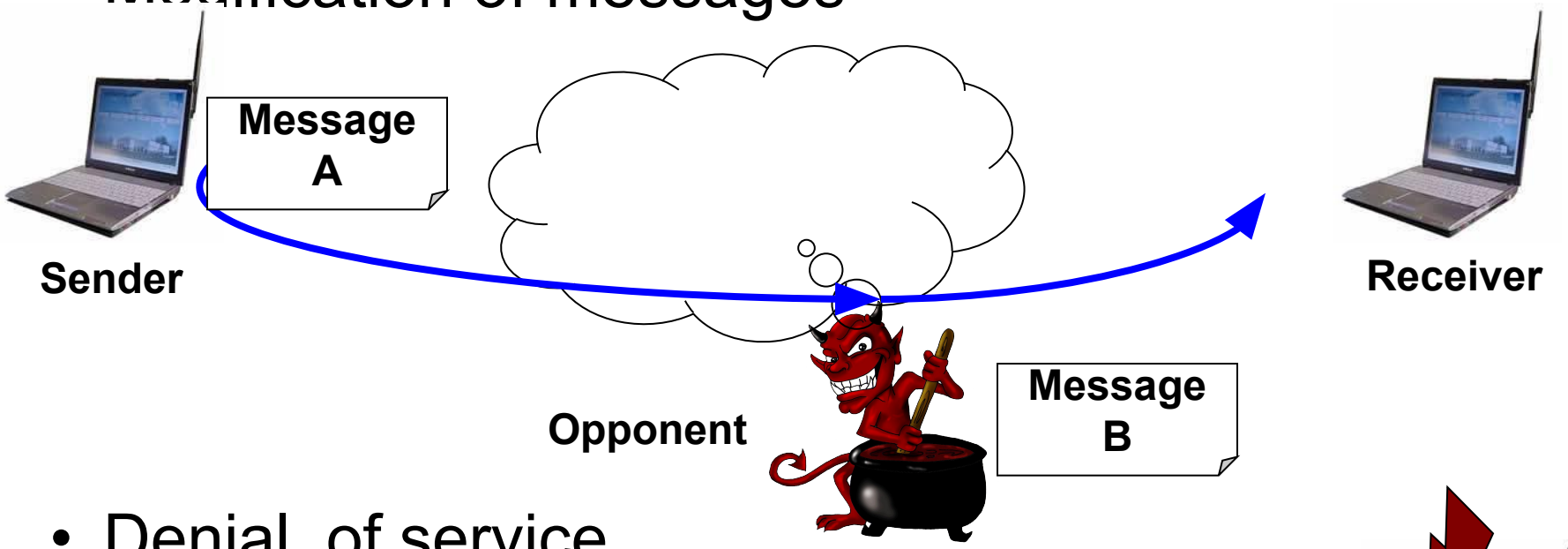


- Replay

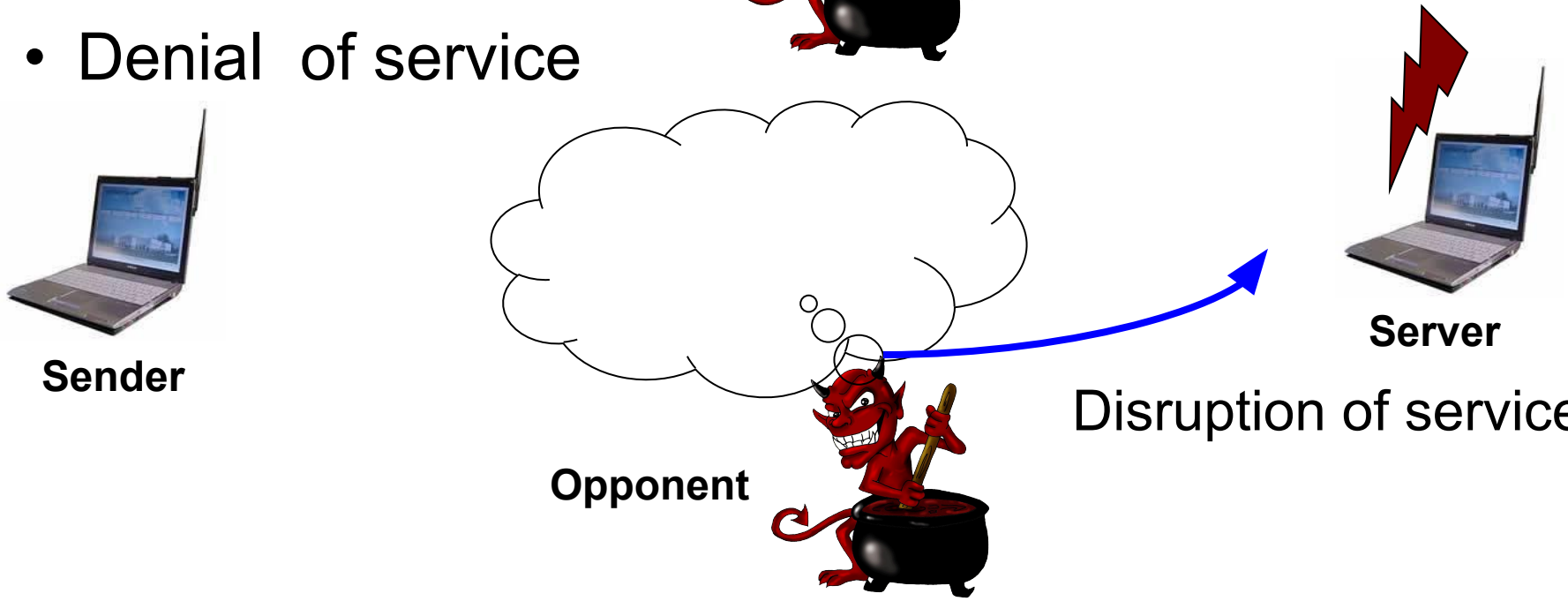


Active Attacks

- Modification of messages

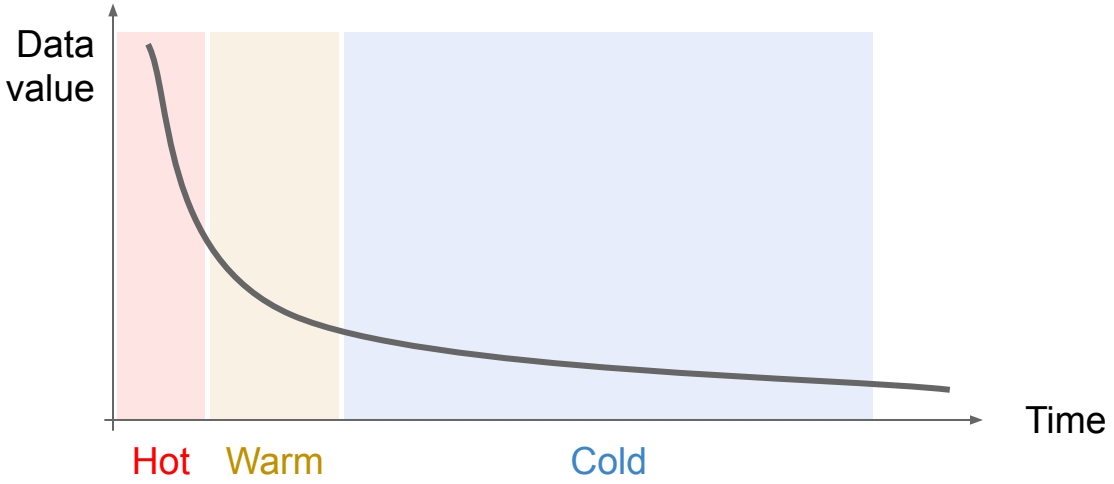


- Denial of service



Storage specialities

- Data typically has a lifecycle and becomes less valuable over time
  - Hot (1 day): data is analyzed heavily, part of all reports
  - Warm (1 day - 1 month): still fresh but only used in latest aggregate reports
  - Cold (1 month - years): used in year-over-year reports, stored because disk and tape are cheap
  - **Result**: same *threat* causes less *risk* as data gets older
  - (Not always the case for very sensitive data)



Storage specialities

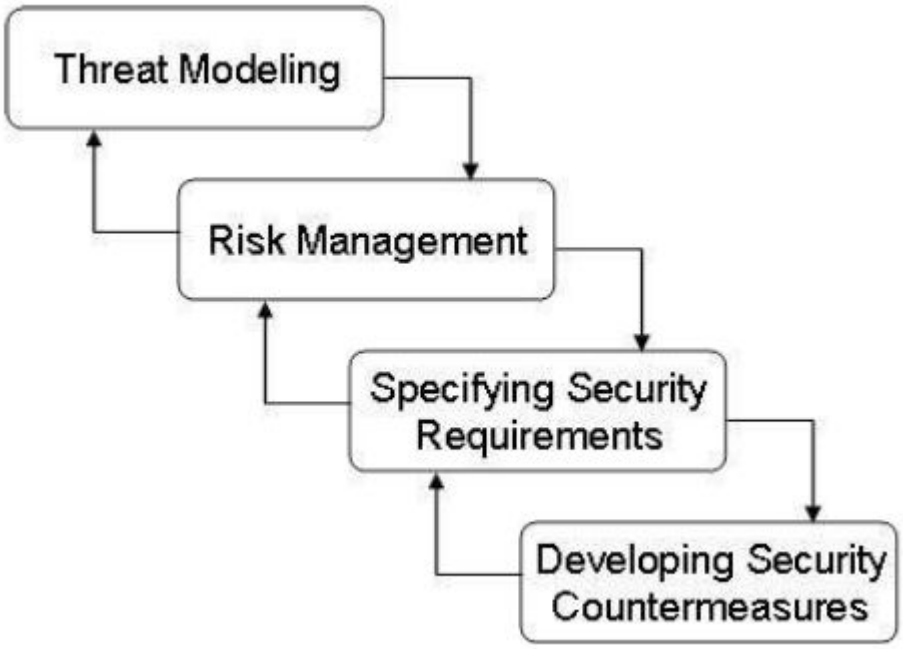
- “Nobody” deletes old data
  - Long term storage is (relatively) cheap
  - Compliance mandates storing for x years
  - New analysis might need old data
  - People are afraid to permanently delete data in general (market advantage)
  - **Result**: any new security technique must be backwards-compatible
    - New encryption scheme: re-encrypt old data or keep the ability to decrypt old scheme
    - Replication → erasure coding: EC old data or implement both strategies for retrieval

Security Engineering

- Solving one issue often affects the others
  - Data encryption solves confidentiality but decreases performance which hurts availability
  - Replication across data centers helps availability but provides a larger attack surface
  - Centralized authentication helps preventing unauthorized access but vulnerable to DoS attacks
- "*Just use every technique together*" is not a good design approach
- Need to understand the threats and solution tradeoffs, and decide what to protect against

Security Engineering

- Each distributed storage system is a unique set of threats, risks and desired features
- The design process has to consider these together



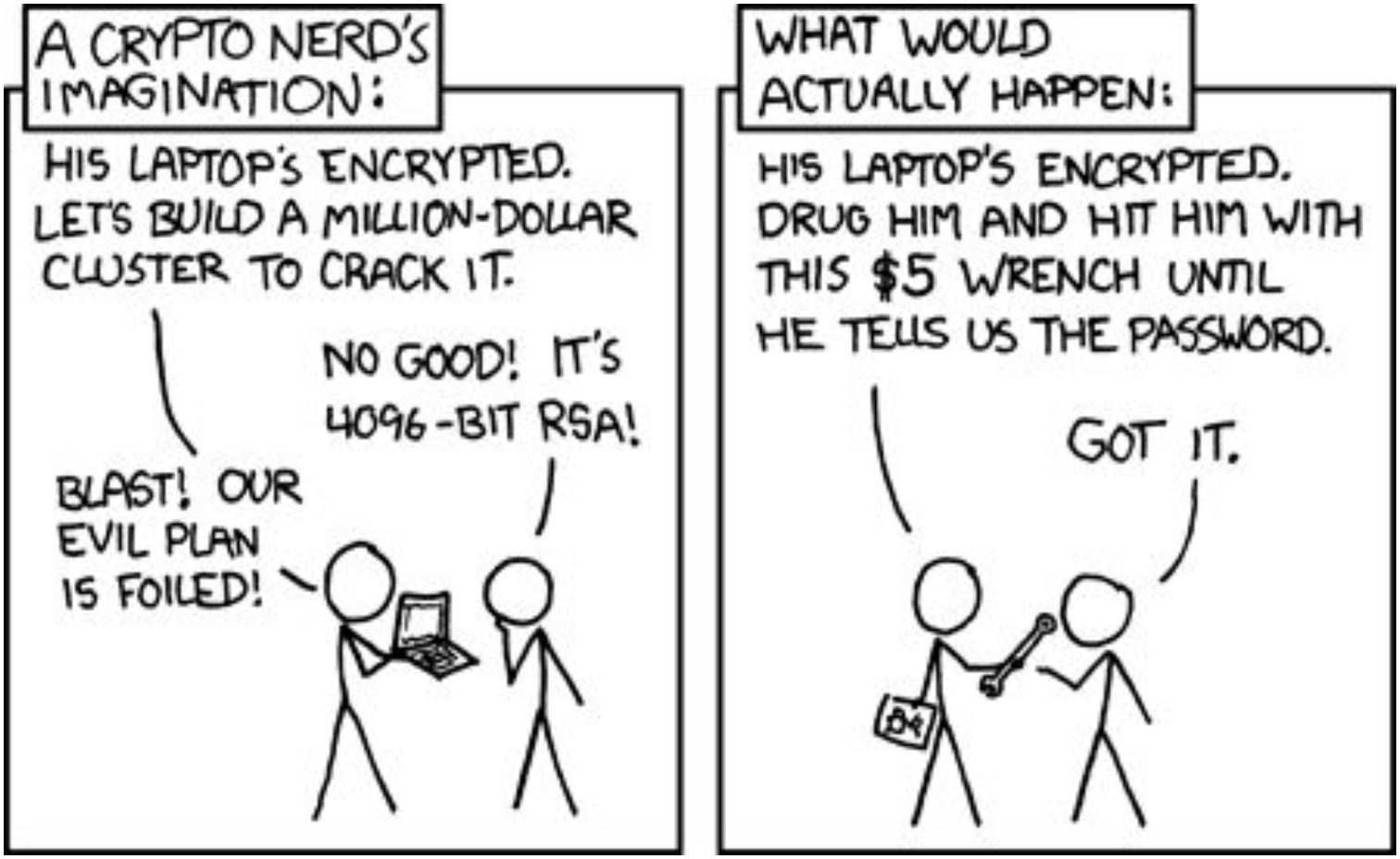
Source: Toward a threat model for storage systems, R. Hasan et al.

Security Engineering

For example:

- Intranet storage system → no need for heavy DoS protection
- All users have same access level → no need for authorization or data encryption
- Temporary data doesn’t need high availability (replicated or erasure coded) storage
- The best authentication system with poor configuration doesn’t help against an employee with bad intentions
- Integrity protection is a high priority when using ordinary spinning disks, but not that important with flash drives

Before we continue…



From http://xkcd.com/

Confidentiality

The attacker tries to read data without authorization

Confidentiality

- Attacker tries to read data
- Typical scenario: stealing information
- Example attacks:
  - Monitor internal traffic of the storage system
  - Monitor buffer cache and deallocated memory of system components or client file systems
  - Try to find deleted storage blocks in the filesystem
  - Profile client file system usage for side-channel information (e.g. infer frequently used files to focus a later attack)

Confidentiality

- Typical solution: encryption
- Data is stored encrypted with a symmetric key like AES
  - Some systems support user-given keys, others always generate the key internally
- Each group, user or storage unit (e.g. bucket) is assigned a different key, authorized users are granted access to the key

Confidentiality

- Encryption makes it very expensive to recover data without the key
  - Weak encryption + an attacker with enough resources = potential threat
- Protection: keys are rotated periodically
  - A new key version is generated every month
  - In file storage the new key is used for new files only, no re-encryption of existing encrypted files
  - If a key is compromised, it only affects a portion of the data

Confidentiality

- Exercise: list confidentiality threats that are protected against with encryption
  -
  -
  -
  -
  -

Confidentiality

- Exercise: list confidentiality threats that are protected against with encryption
  - System admin who has access to storage nodes but not keys
  - Hackers who infiltrate the system and gain access to storage nodes (but not customer keys)
  - Physical attack (internal or external) against the storage facility, where disks are stolen
  - Data recovered from poorly erased disk after decommission

Confidentiality

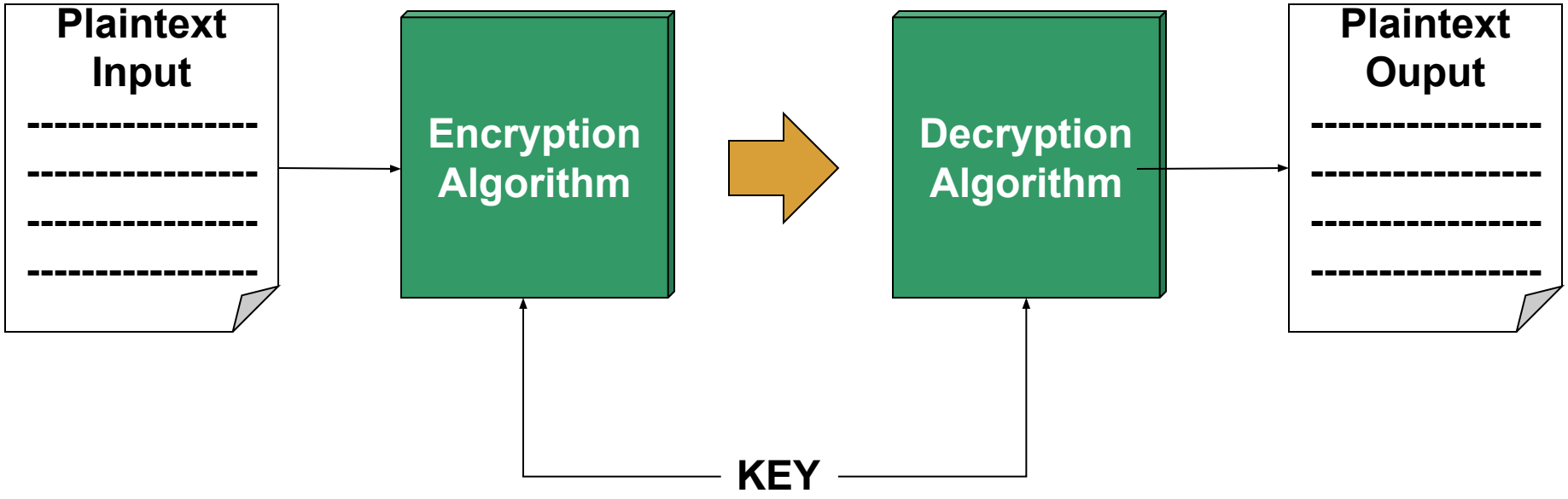
- What confidentiality threats are NOT protected against using encryption?
  -
  -
  -
  -

Confidentiality

- What confidentiality threats are NOT protected against using encryption?
  - Bugs in the authorization module or key manager that exposes keys to wrong users
  - Threatened/blackmailed system admin who has access to customer keys
  - Hackers who gain access to both storage nodes and the keys
  - The storage system itself accessing data for profiling, ad targeting, selling user habits, etc.
    - When using a storage service you have no idea what the provider is doing with your data

Symmetric Ciphers

- Encryption and decryption performed with same key
- Remains the most widely used

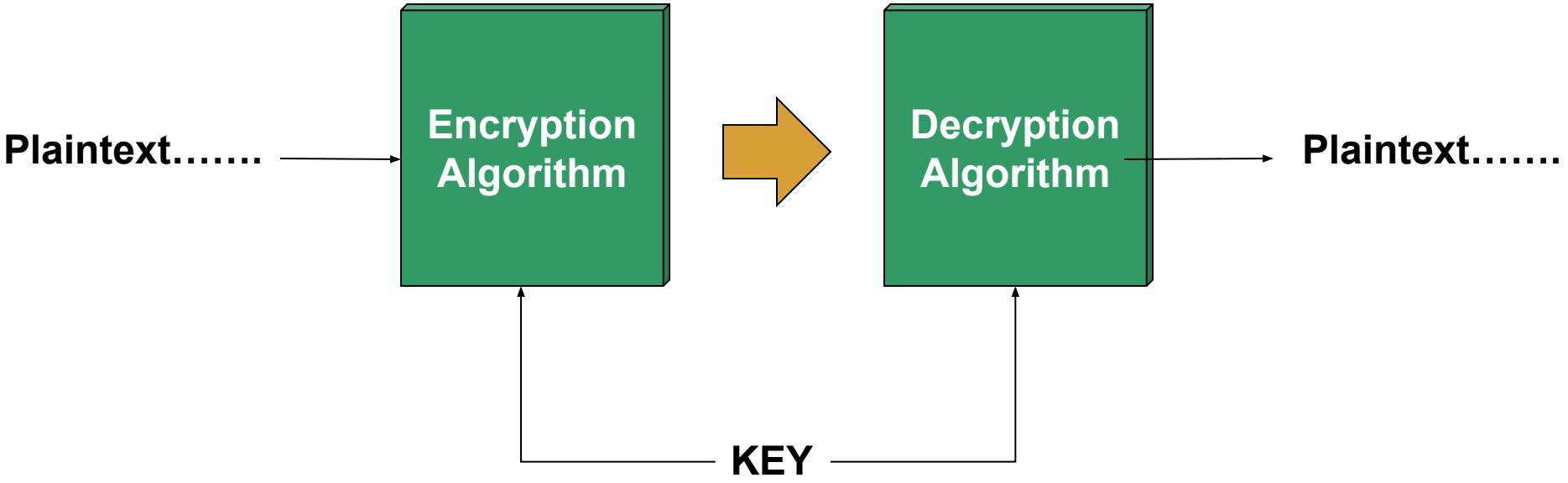


- On the background of our studies: number theory and finite fields.

Symmetric Ciphers

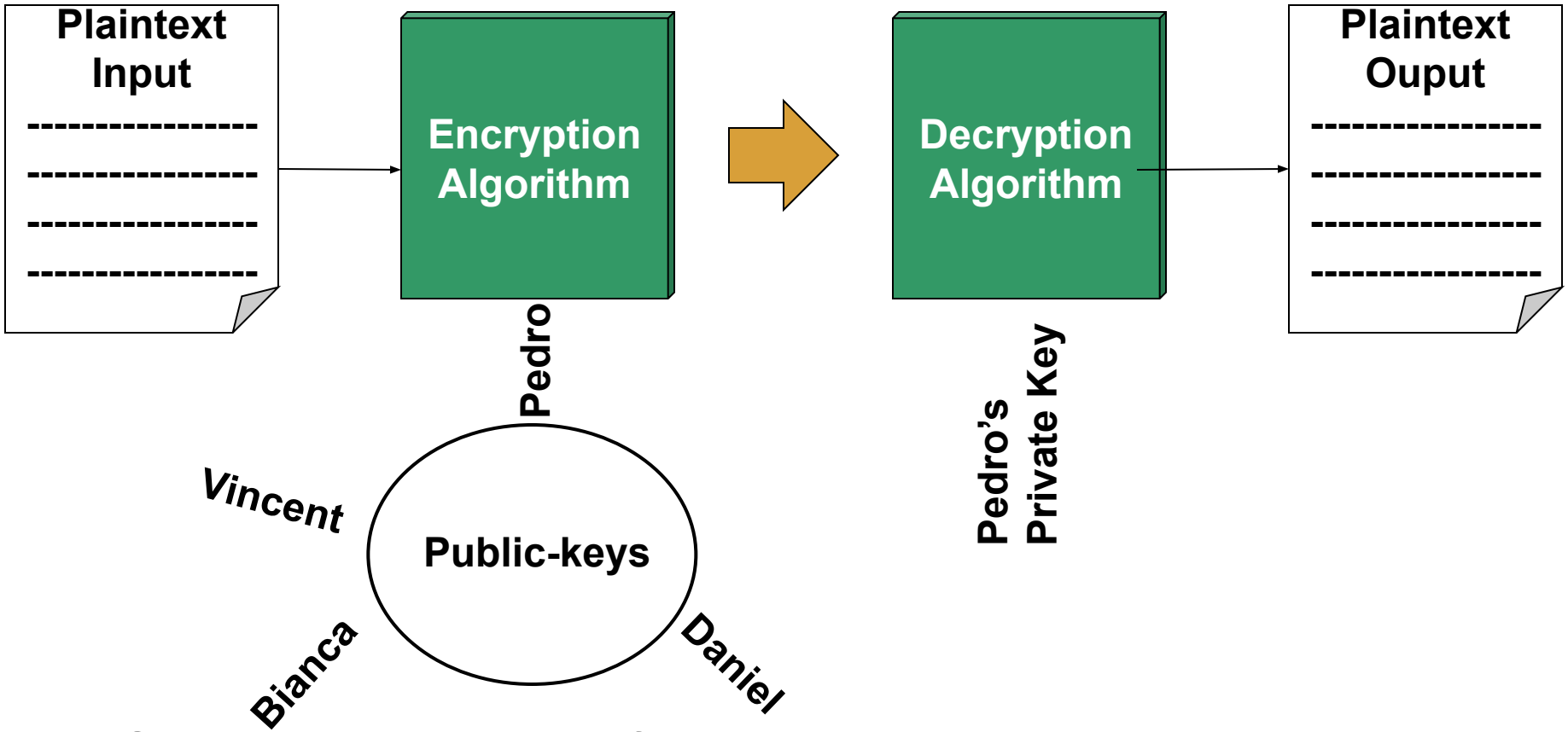
- Block ciphers e.g., DES, 3-DES, AES.

- Stream ciphers
  - Techniques may use block ciphers for generating pseudo-random key
  - On the background: pseudo-random number generation



Public-key Cryptography

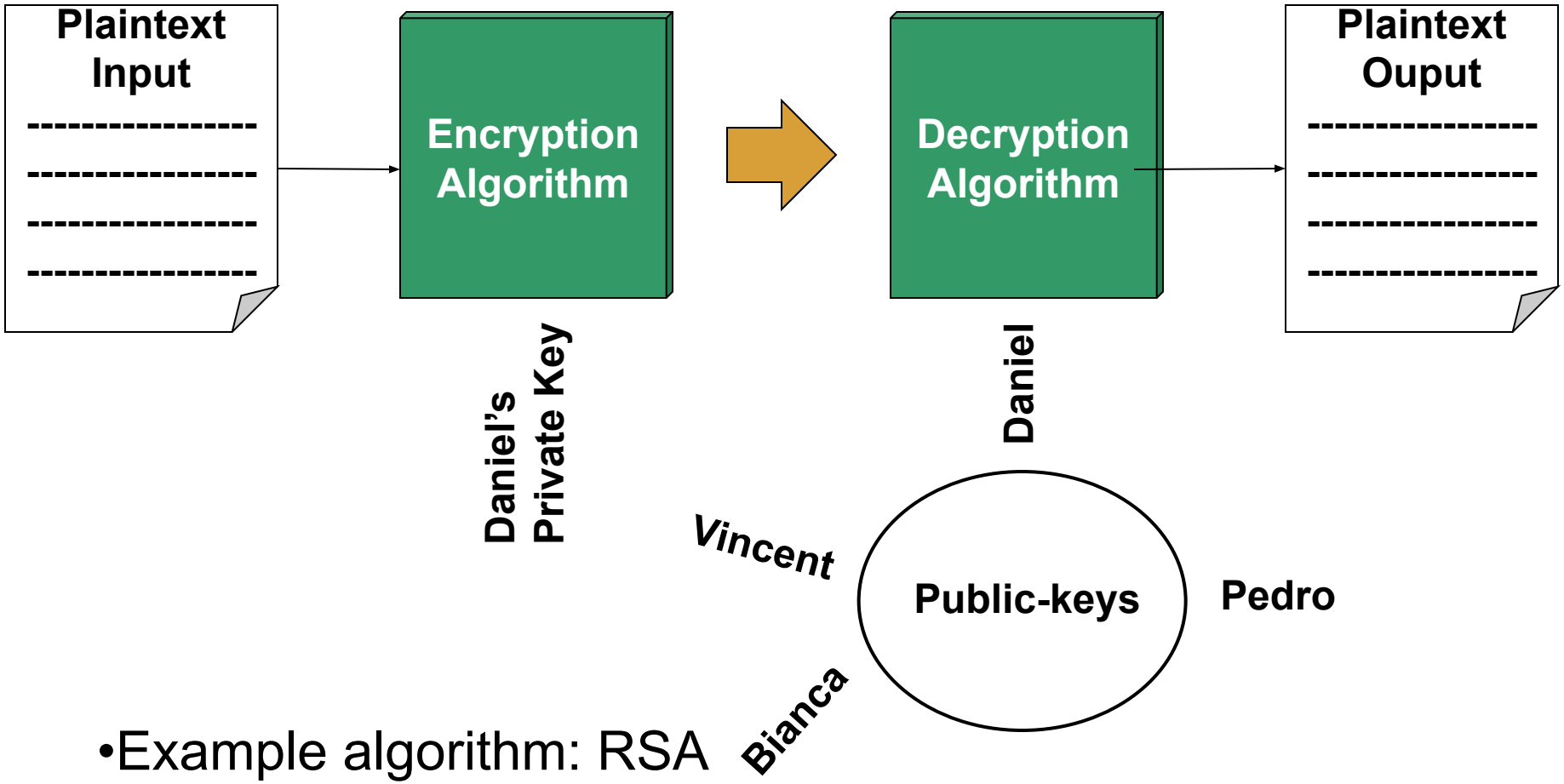
- Encryption and decryption performed with different keys



- On the background of our studies: number theory

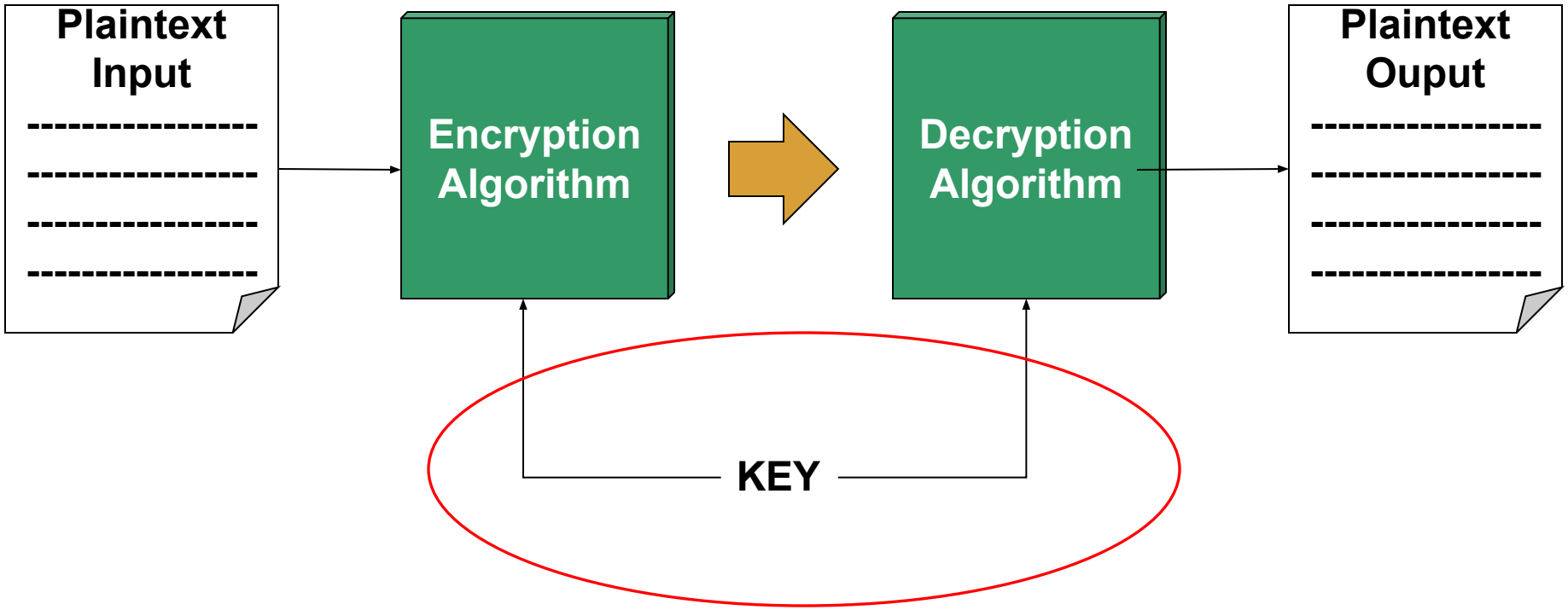
Public-key Cryptography

- Encryption and decryption performed with different keys



- Example algorithm: RSA

Key management and distribution



How do we generate a “strong” key?

How do we deliver to sender and receiver securely?

Integrity

The attacker tries to modify data

Integrity

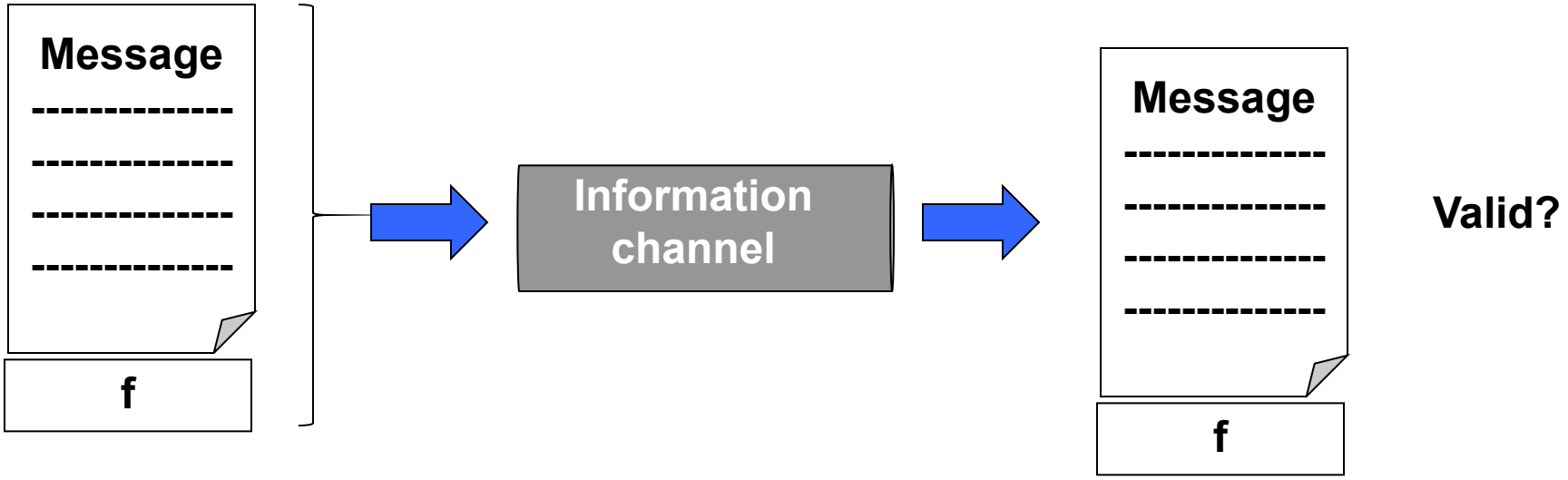
- The attacker tries to change the stored data
- Typical scenarios: ransomware, cause data loss, plant incriminating data, change timestamps or GPS location, delete log entries
- Examples:
  - Corrupt metadata (e.g. internal database) to make files unreadable
  - Overwrite all values with random ones
  - Overwrite files with their encrypted version (ransomware)
  - Temporarily change values to cause disruption
  - After getting fired change your status back to “hired” in the HR system and keep getting salary

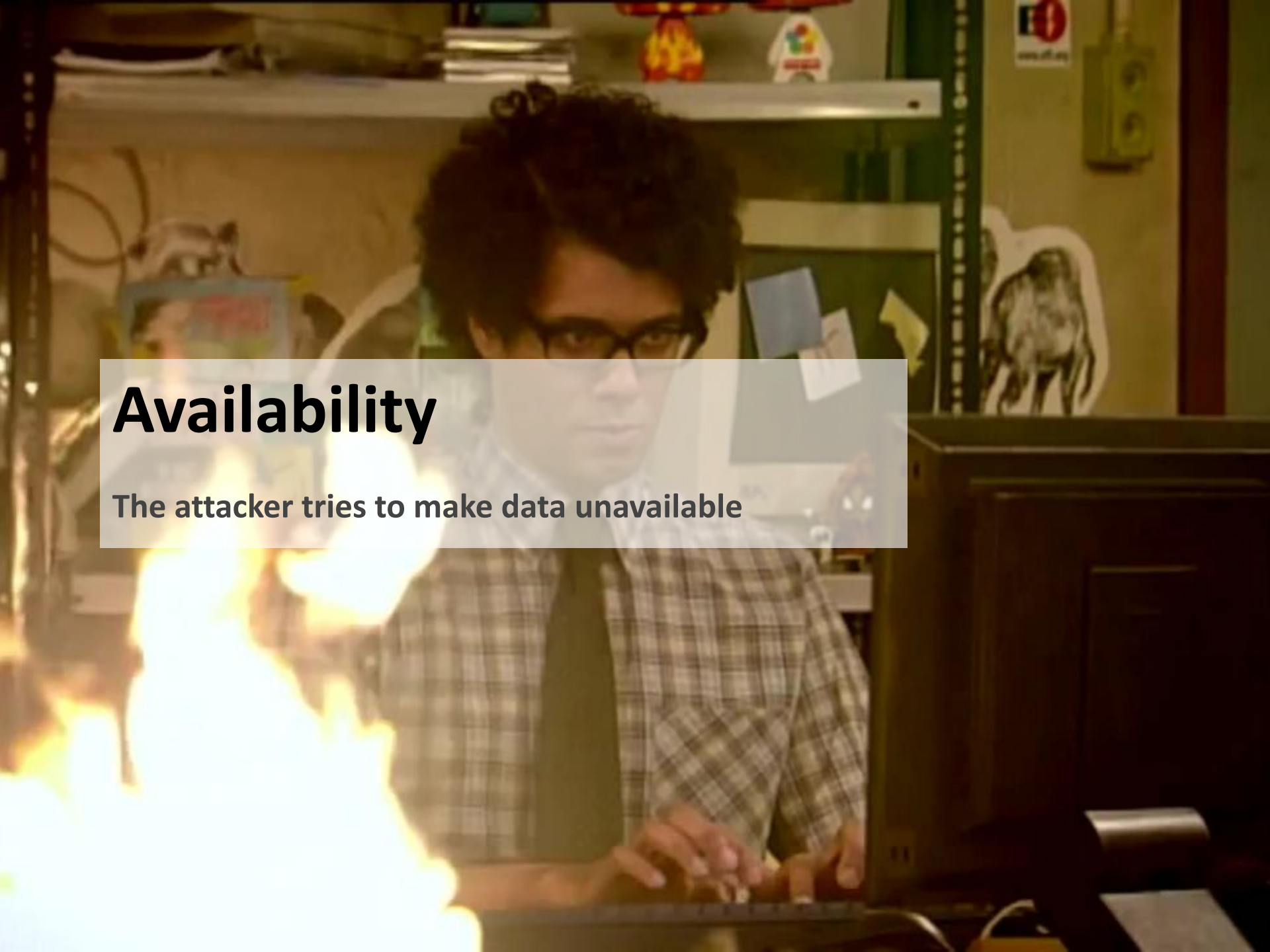
Integrity

- Goal of the security measure are twofold:
  - Detect if integrity is compromised
    - Hash-based fingerprinting
    - Periodically calculate data hash and compare with stored value
    - Expensive operation
    - Different hash types available
  - Recover original data
    - Redundant storage to have enough correct data
    - Erasure coding, replication
    - Online or offline backup (e.g. tape, DVD, USB storage)

Authenticity:

- Data can be verified and trusted
- Confidence in the validity of a message.





Availability

The attacker tries to make data unavailable

Availability

- Make the whole storage system or specific data unavailable
- System-wide attacks typically look for bottlenecks and try to overwhelm them
  - “Denial-of-Service” (DoS) attack
  - “Distributed-DoS” (DDoS): the attacker uses a lot of (previously infected or owned) computers to bomb the target system with false requests

Availability

- Exhaust the system with legitimate requests:
  - Many small requests fill up the log space
  - Many data blocks fill up primary storage
  - Many small files exhaust metadata space (e.g. HDFS namenode memory)
  - Exploit versioned file systems: make a small edit that creates a full size new version
  - Allocate OS file handles in an infinite loop
  - Cause many memory or disk writes to wear out the hardware

Availability

- Possible solutions
  - Try to avoid single points of failure
  - Limit accepted requests per user/group/API key/IP address
  - Monitor storage system hardware (disks, memory) for wear attacks
  - Store data with redundancy and re-route requests from overwhelmed resources
  - Automatic alarms when log or storage space is getting too full

Authentication

The attacker tries to impersonate a user

Authentication

- The attacker tries to blend in as a legitimate user or storage device
- User impersonation:
  - Steal the user's session cookie
  - Social engineering or phishing to trick the user to give their password
  - Brute force user credentials
  - Exploit password recovery mechanism
- Storage device impersonation:
  - Fake virtual disk drive where the user sync their Dropbox folder
  - Peer to peer and decentralized storage networks: join with malicious clients

Authentication

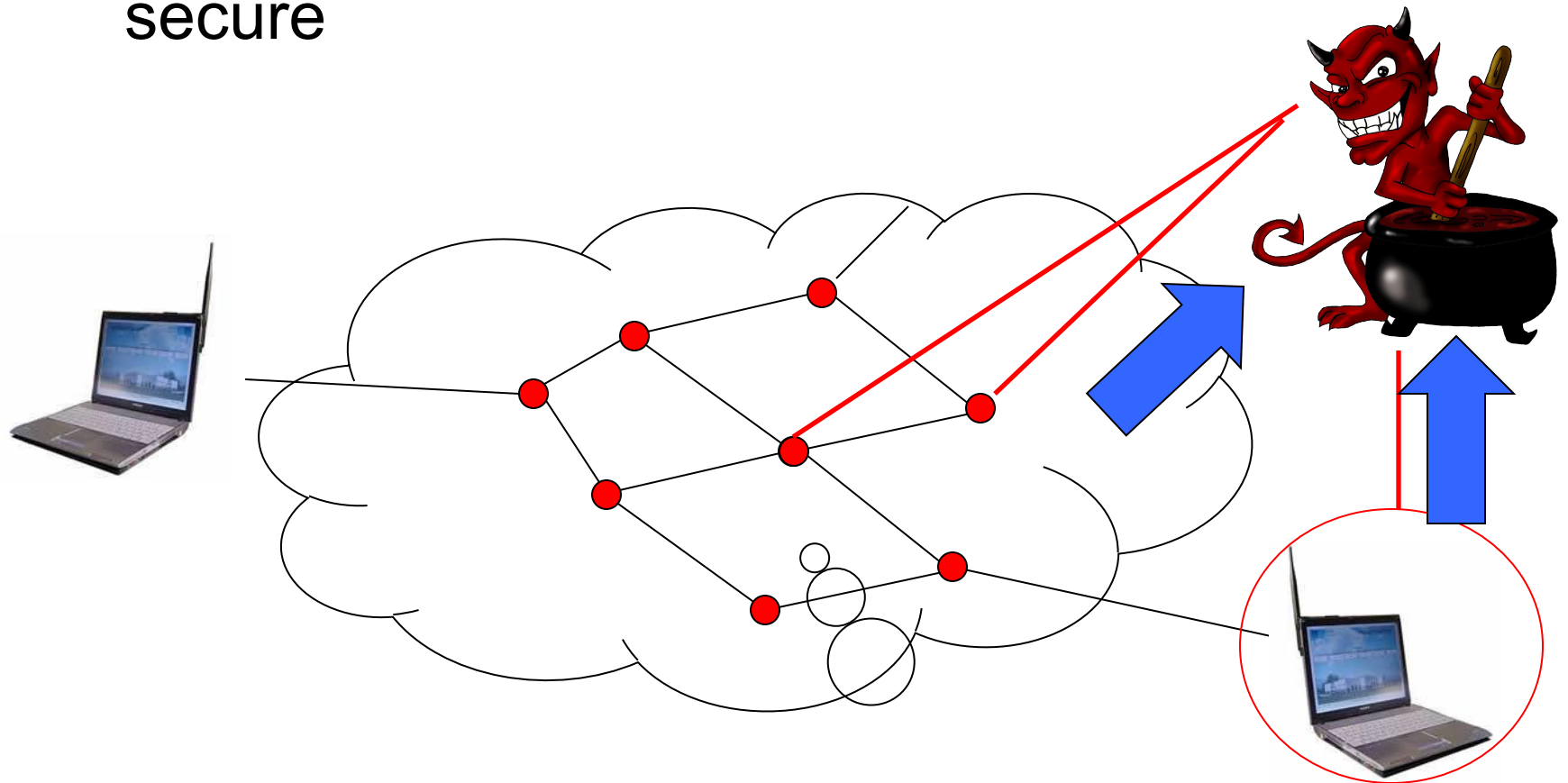
- Solutions:
  - Require strong passwords, never store/send passwords in plaintext
  - Check for known compromised passwords and enforce changing them (haveibeenpwned.com)
  - Two factor authentication with time-based one time password (SMS-based is vulnerable to SIM swap)
  - Authenticate the client software (hard problem)

Summary

- Security has to be tailored to the concrete system
- Identify threats and associated risks before thinking of countermeasures
- Learn and use best practises
- Do not invent new crypto!
- Make sure that different security techniques are not working against each other
- Consider data lifecycle and keep backwards compatibility open

Accountability:

- Actions of an entity can be traced uniquely to that entity.
- Why is it important? Current systems are NOT secure



Anonymity: attackers' perspective

- We would like to maintain sessions and/or sources anonymous to outsiders
- Why? Important nodes may be vulnerable to other attacks if discovered