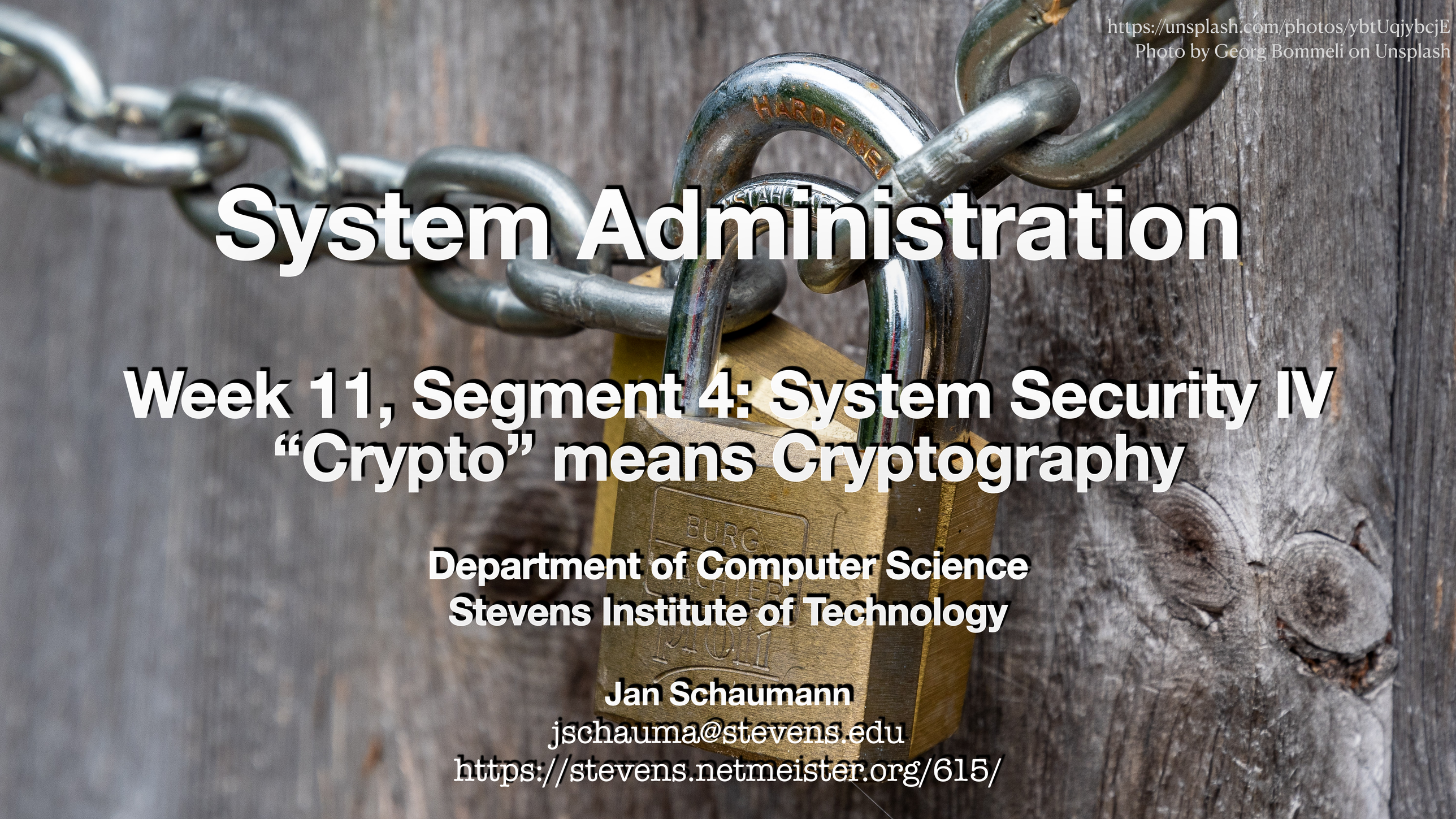


https://unsplash.com/photos/ybtUqjybcjE
Photo by Georg Bommeli on Unsplash

System Administration

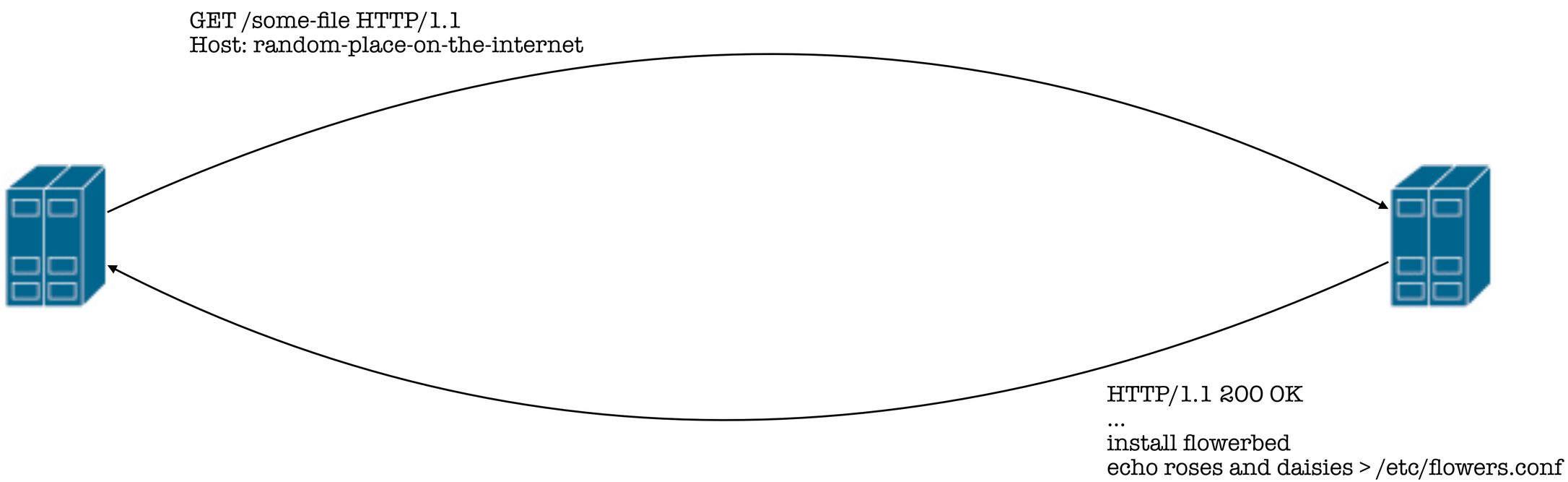
Week 11, Segment 4: System Security IV
"Crypto" means Cryptography

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/615/

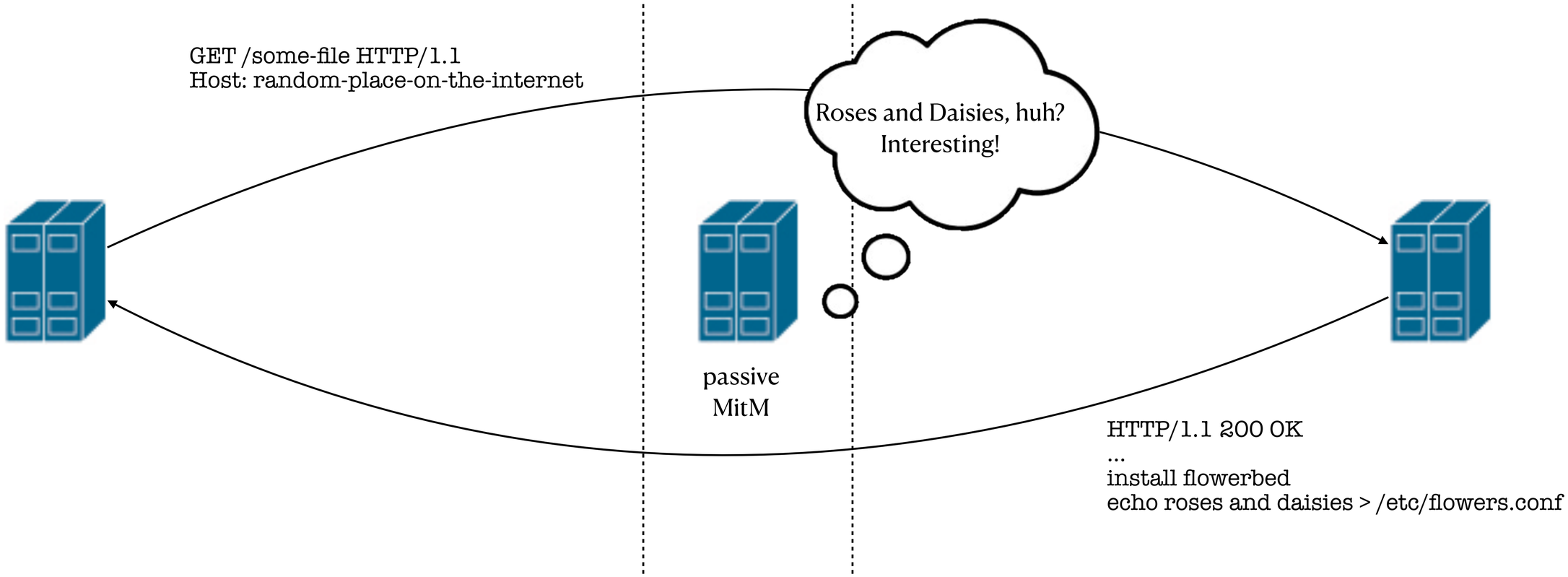
Feeling insecure?

curl http://random-place-on-the-internet/some-file | sudo bash



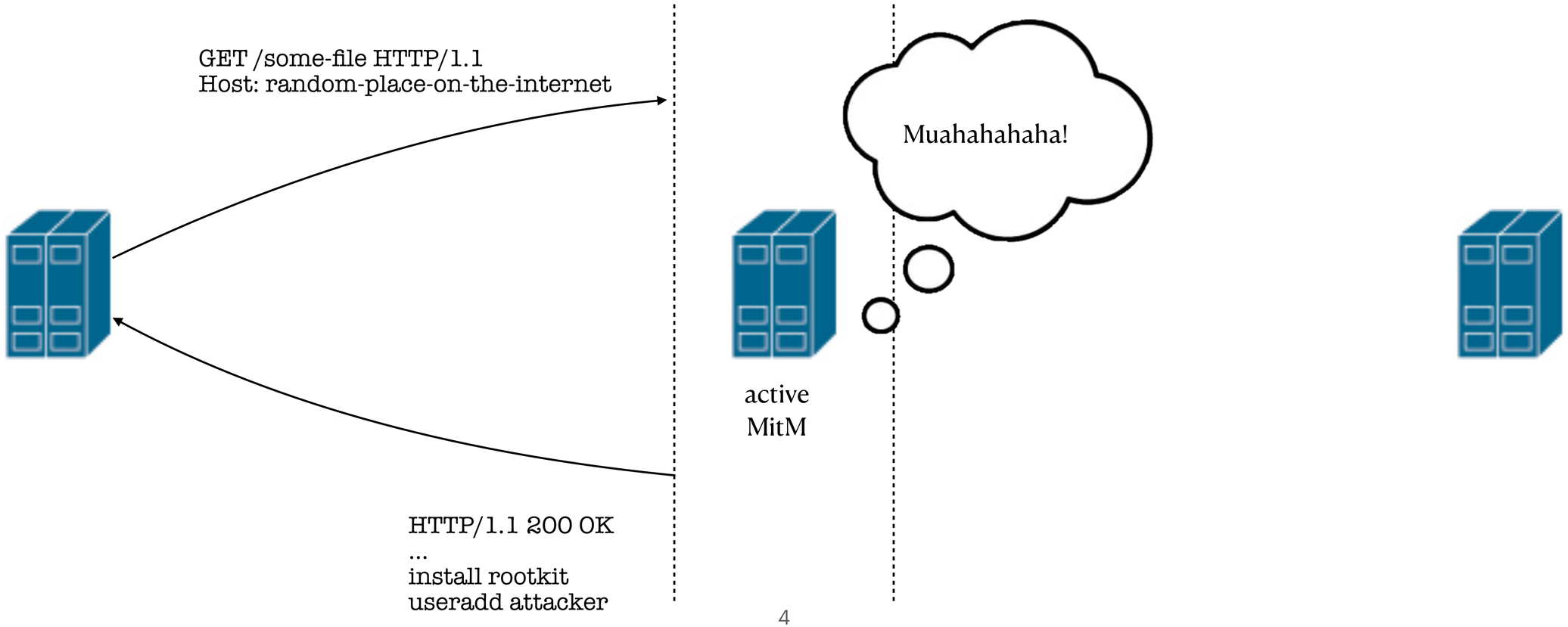
Feeling insecure?

curl http://random-place-on-the-internet/some-file | sudo bash



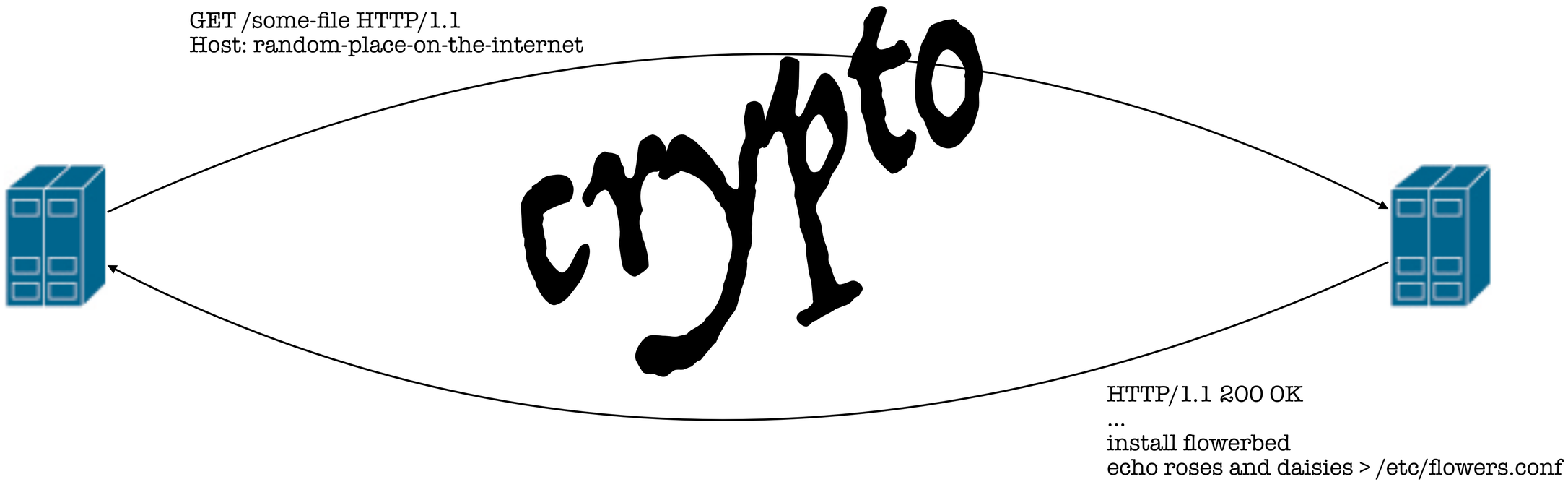
Feeling insecure?

curl http://random-place-on-the-internet/some-file | sudo bash



Feeling insecure?

curl http://random-place-on-the-internet/some-file | sudo bash



Feeling insecure? Rub some crypto on it!

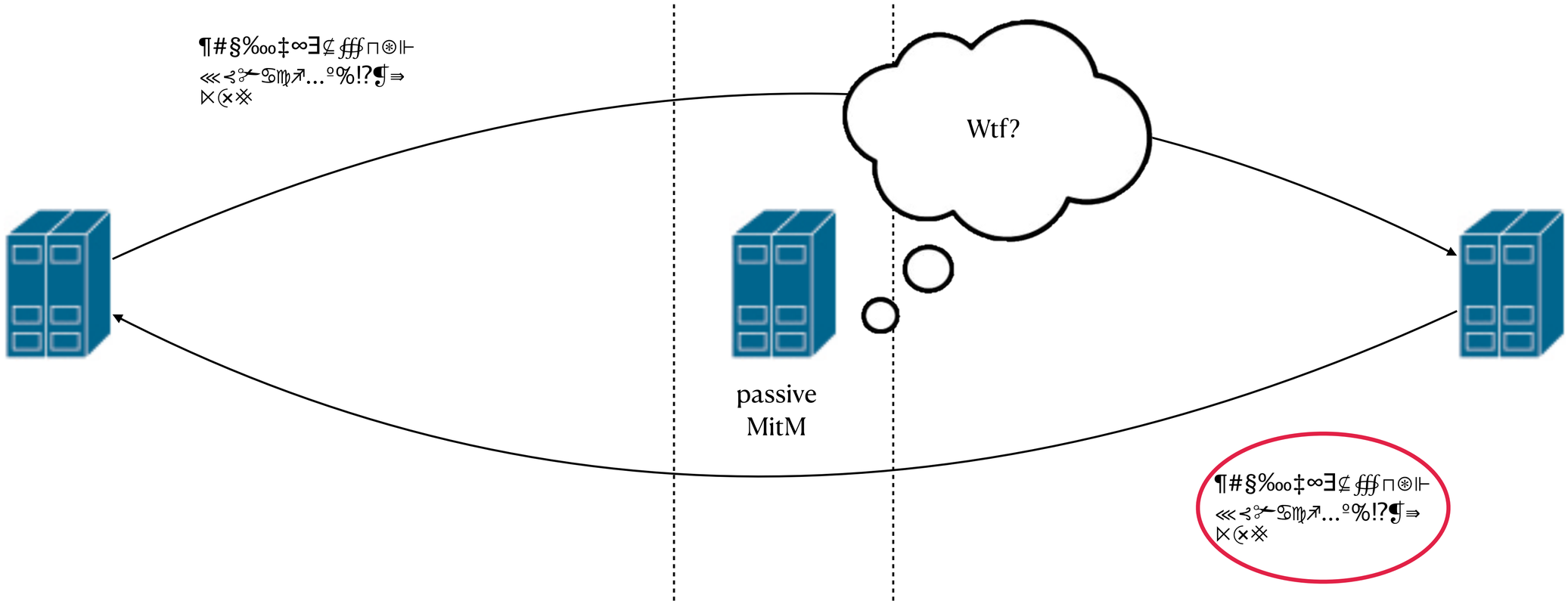
Cryptography can help mitigate some of the risks sometimes.

It may provide security in the areas of:

- Secrecy or Confidentiality
  - *Did/could anybody else see (parts of) the message?*
- Accuracy or Integrity
  - *Was the message (could it have been) modified before I received it?*
- Authenticity
  - *Is the party I'm talking to actually who I think it is / they claim they are?*

Confidentiality

curl https://random-place-on-the-internet/some-file | sudo bash



Threats to Confidentiality

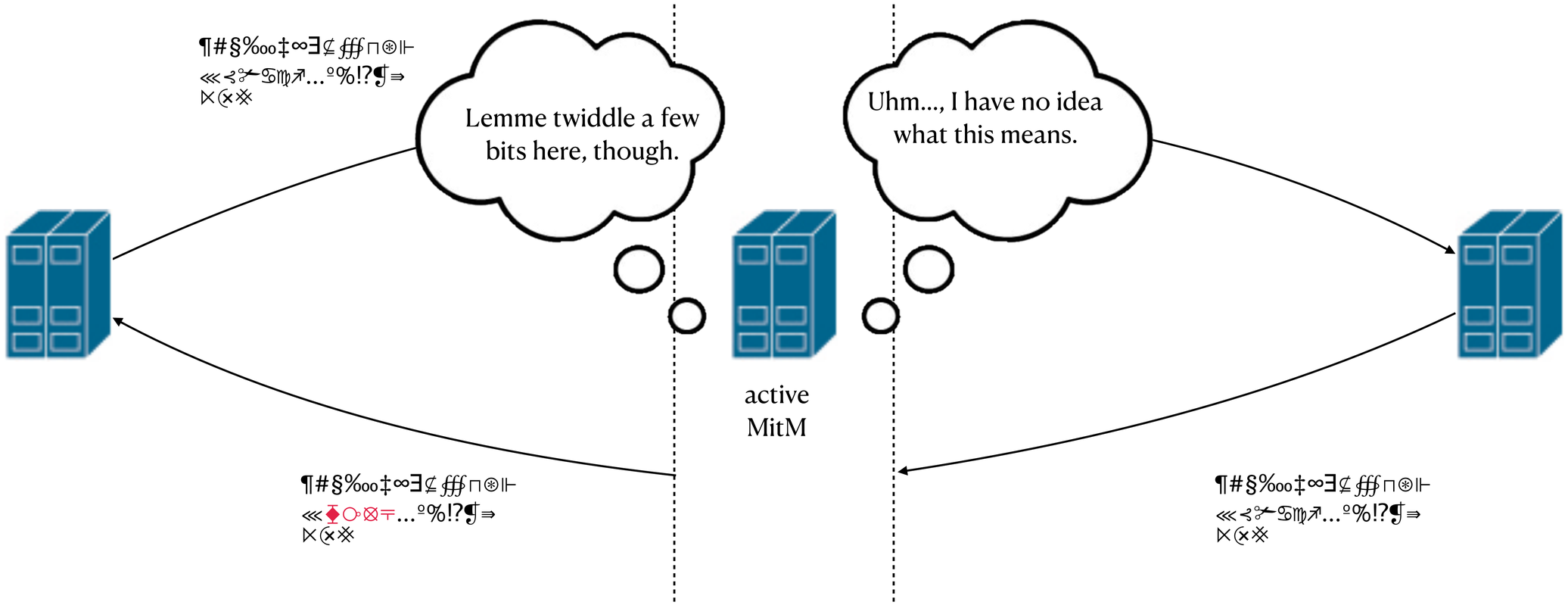
- lack of authenticity
- key exchange
- lack of key rotation
- key disclosure / revocation

Never store secrets in code! Separate code and config, config and secrets.

Always use a key management system.

Integrity?

curl https://random-place-on-the-internet/some-file | sudo bash



Integrity?

```
curl https://random-place-on-the-internet/some-file >file
curl https://random-place-on-the-internet/sha256 >file.sha256
diff <(sha256sum <file) <(file.sha256) && cat file | sudo bash
```

Threats to Integrity

- lack of authenticity
  - Where did I get the checksum?
- lack of integrity
  - Was the checksum tampered to match the (tampered) data?
  - collisions in algorithm
- “verification” with compromised tools

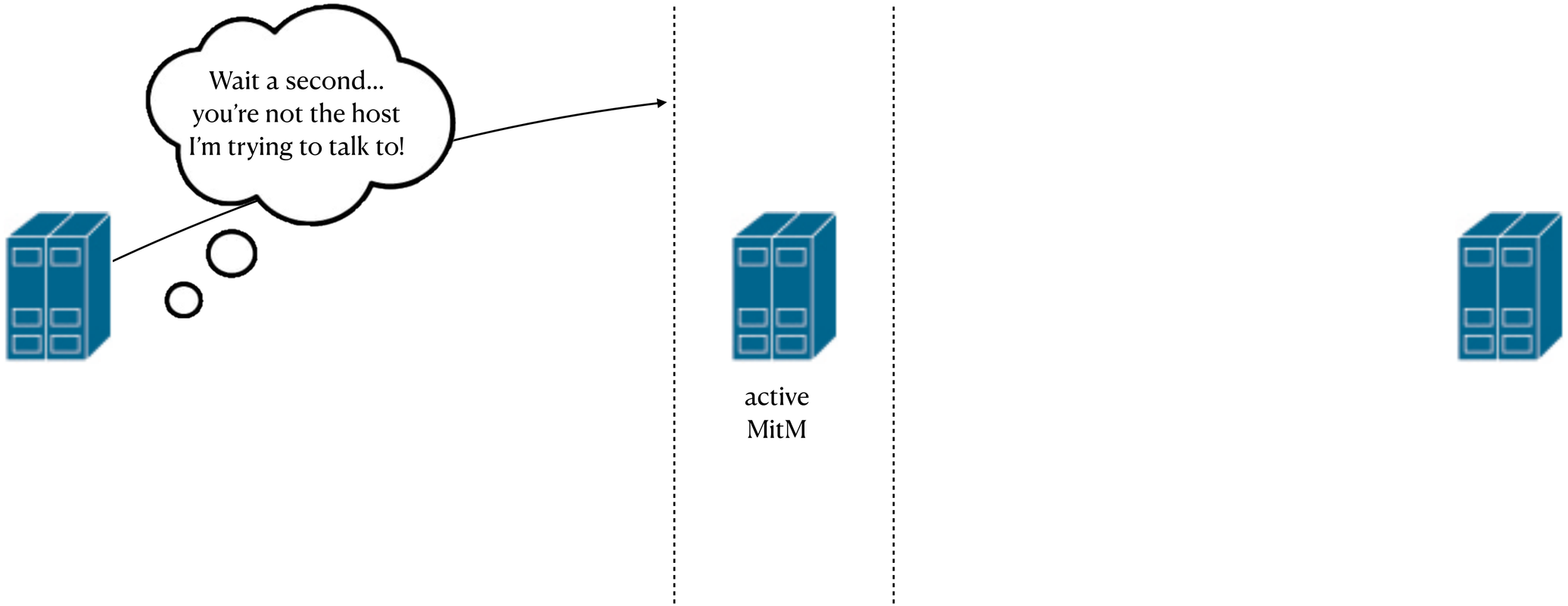
Hashing - not just for integrity

- Integrity checking: fast, collision-resistant, unkeyed, unsalted data
- HMAC: Integrity + Authenticity, shared secret
- Password Hashing: slow, collision-resistant, salted data

- Never confuse *hashing* and *encryption*!
- Never *encrypt* your users' passwords to store them — always *hash* them.
- Always *salt* your hashes.
- Always use adaptive or *key-stretching* functions such as e.g.: `argon2`, `bcrypt`, `PBKDF2`, `scrypt`
- `DK=KDF(key, salt, iterations)`, where *salt* and *iterations* are *public*.

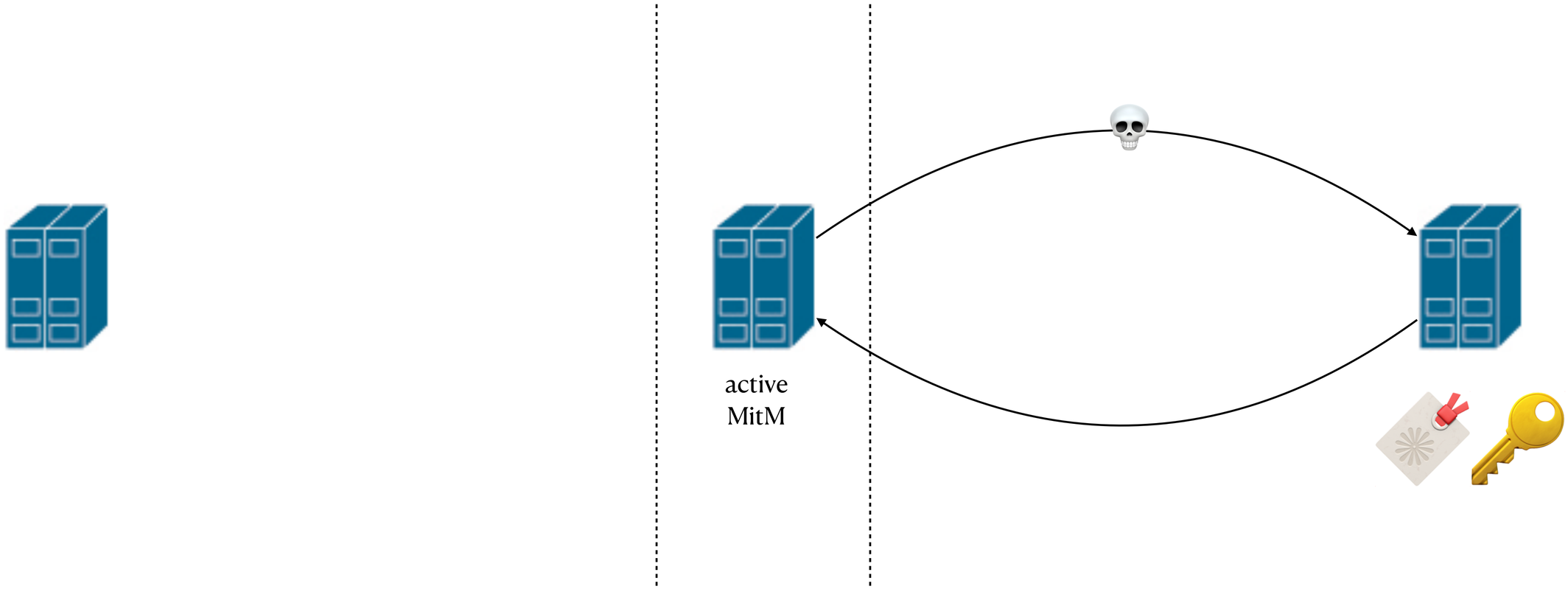
Authenticity?

curl https://random-place-on-the-internet/some-file >file



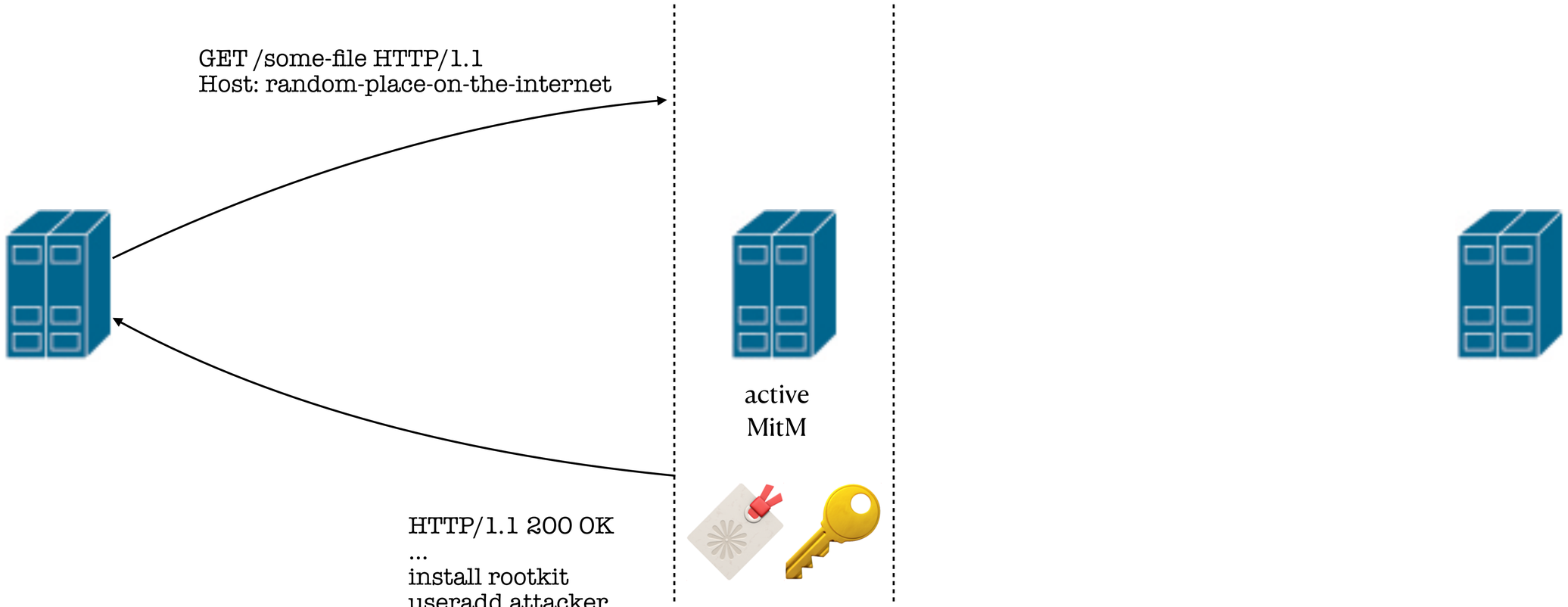
Authenticity?

curl https://random-place-on-the-internet/some-file >file



Authenticity?

curl https://random-place-on-the-internet/some-file >file

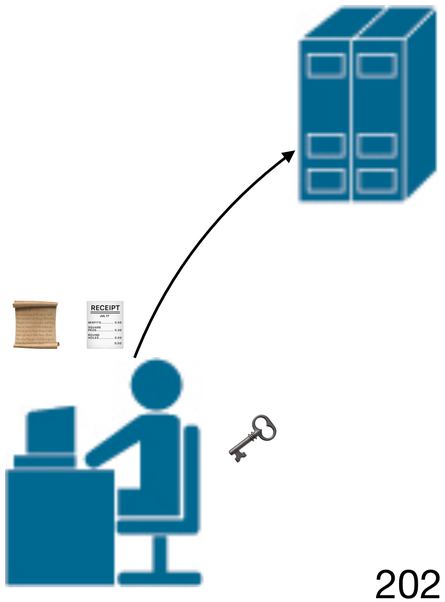


Integrity + Authenticity

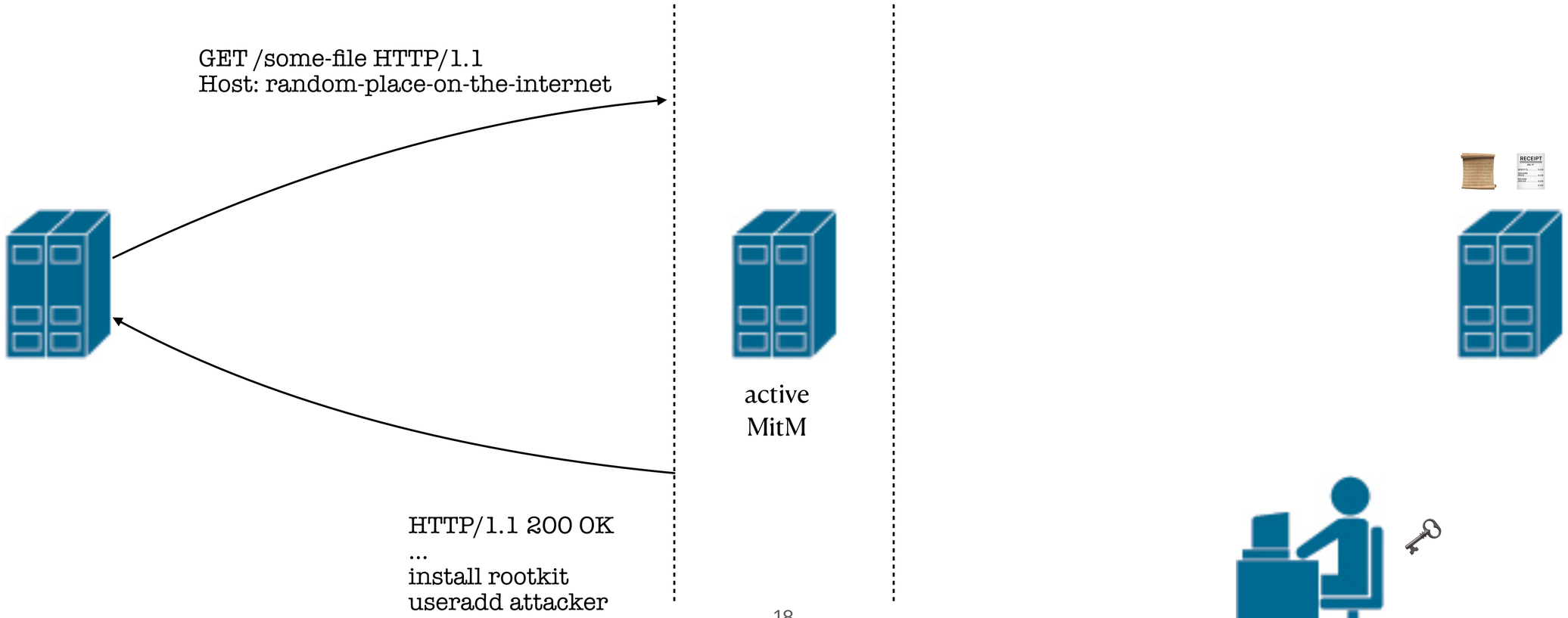
```
curl https://random-place-on-the-internet/some-file >file
curl https://random-place-on-the-internet/signature > sig
gpg --verify sig file && cat file | sudo bash
```

Authenticity?

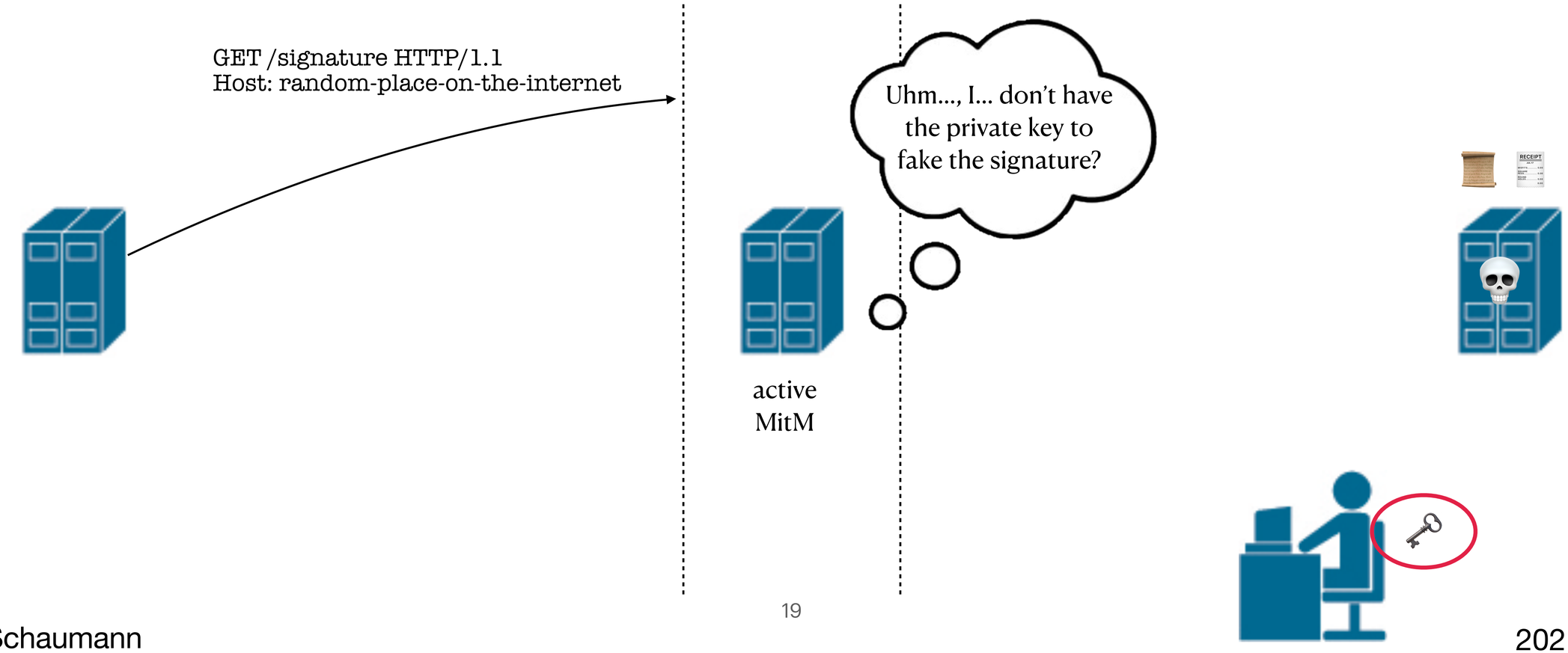
active
MitM



Authenticity?



Authenticity



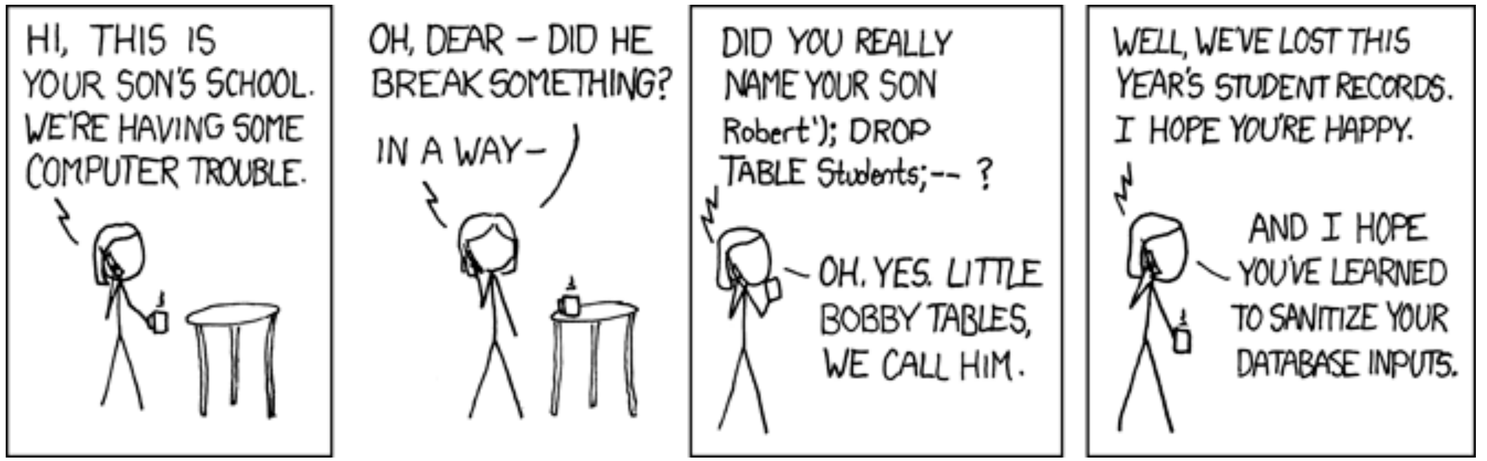
Threats to Authenticity

- lack of *integrity*
- reliance on fragile infrastructure
- unverified or overly broad trust model
- usability
- conflation with *authorization*

(Some) Classes of Vulnerabilities

- memory management:
  - use of uninitialized memory
  - buffer overflow / stack smashing
  - use-after-free / dangling pointer
- input validation:
  - code and command injections
  - format attacks
- race conditions
  - non-atomic TOCTOU
  - symlink attacks

https://www.xkcd.com/327



(Some) Classes of Vulnerabilities

- privilege escalation and confusion:
  - XSS, CSRF
  - setuid with untrusted environment
- social engineering
  - phishing
  - watering hole attacks
- brute-force attacks
  - namespace iteration
  - denial of service
- information disclosure
  - insufficient permissions
  - MitM
  - lack of encryption, authN, authZ

Cryptography

- Never write your own crypto or invent your own protocol.
- Authentication != Authorization
- Cryptography does not handle authorization.
- You usually need all three: confidentiality, integrity, authenticity.
- Cryptography cannot prevent against incorrect use – usability is hard!
- Cryptography often times cannot help you at all.

Know your threat model!