CS 4173/5173
COMPUTER SECURITY

Hash Functions

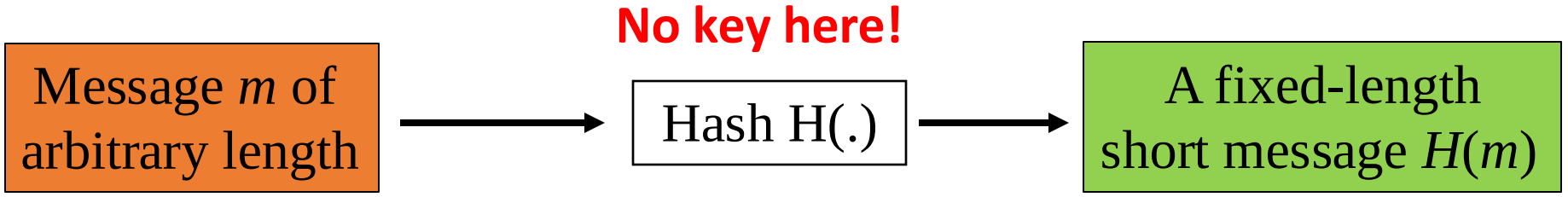
OUTLINE LAST TIME

- Message Authentication
  - MAC-CBC
  - Combination of encryption and authentication
    - Encrypt-then-authenticate
    - Encrypt-and-authenticate
    - Authenticate-then-encrypt

OUTLINE LAST TIME

- Achieving confidentiality does not necessarily mean achieving integrity.
  - Fully encrypting a message does not mean data tampering (integrity violation) can be detected.
- MAC is an effective way to protect integrity.
  - **CBC-MAC** in symmetric cryptography **works but is slow**
  - Integrity protection usually incurs overhead.
- Efficient design is needed to achieve both confidentiality and integrity
  - **Fast MAC** design widely used today based on **hash functions**.
    - Hash function based MAC is called **HMAC**

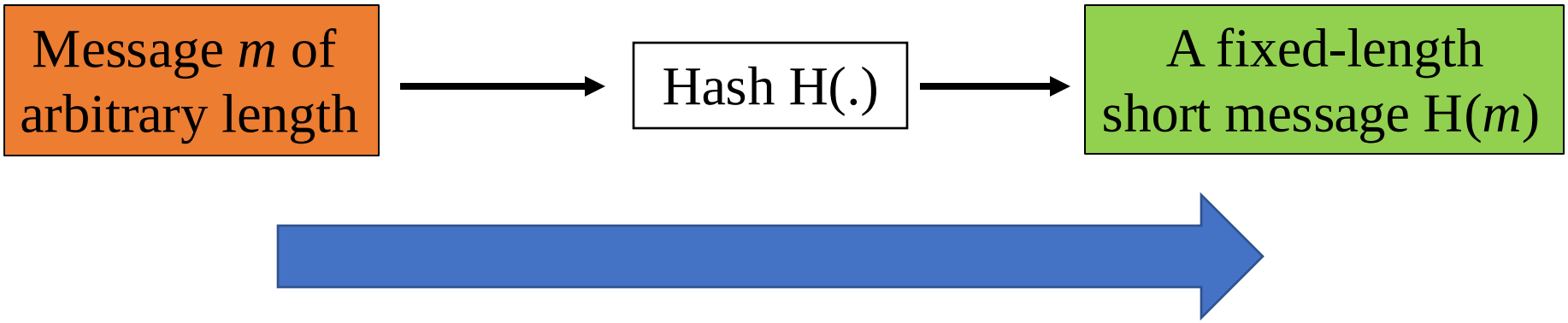
HASH FUNCTION



- H(.) is also known as
  - Message digest
  - One-way transformation
  - One-way function
  - Hash
  - Widely-used hash functions: MD5, SHA1, SHA256.
- Usually fixed lengths: 128, 160, 256 bits, …
- **Note: Hash functions have a broader definition in computer science. But in this class, we only consider cryptographic hash functions.**

DESIRABLE PROPERTY OF HASH FUNCTIONS: 1

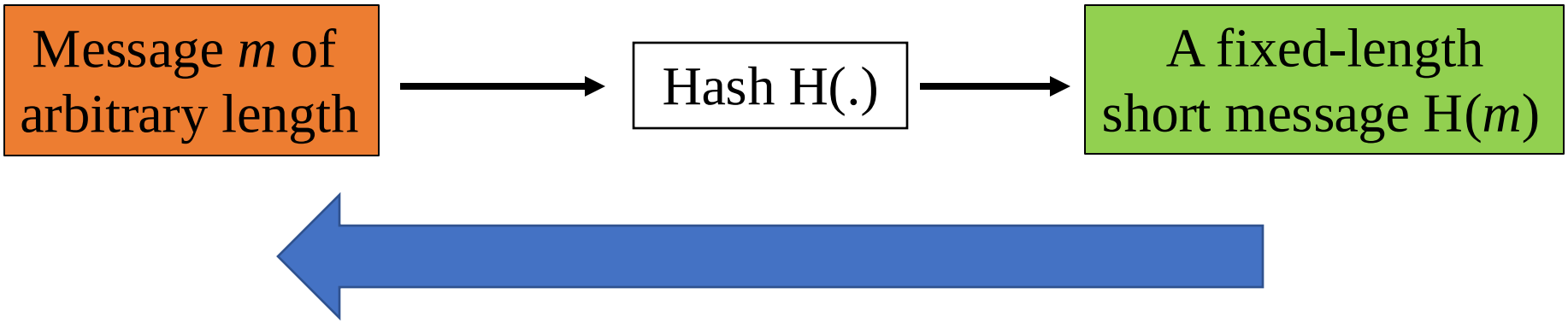
- 1) Performance: Easy to compute H(*m*)



Very fast to compute H(*m*)!

DESIRABLE PROPERTY OF HASH FUNCTIONS: 2

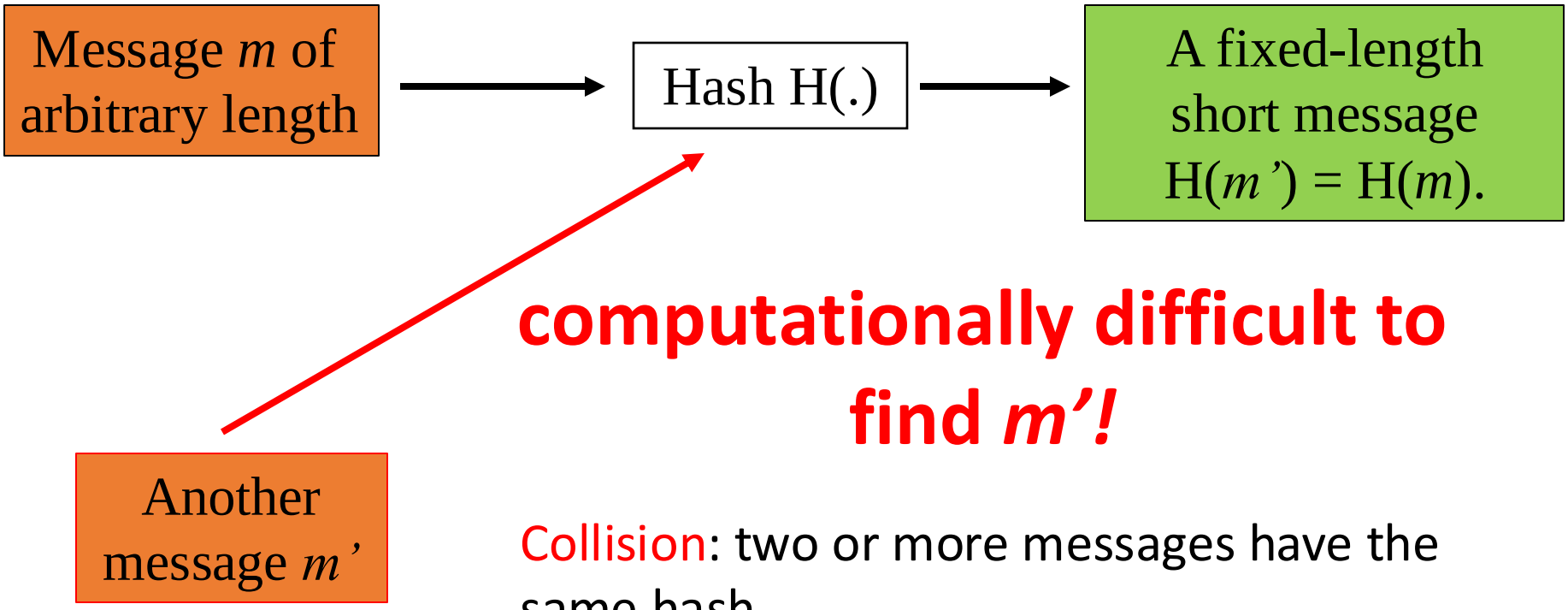
- 2) One-way property: Given H(*m*) but not *m*, it's computationally infeasible to find *m*



computationally difficult to compute get *m* from H(*m*)

DESIRABLE PROPERTY OF HASH FUNCTIONS: 3

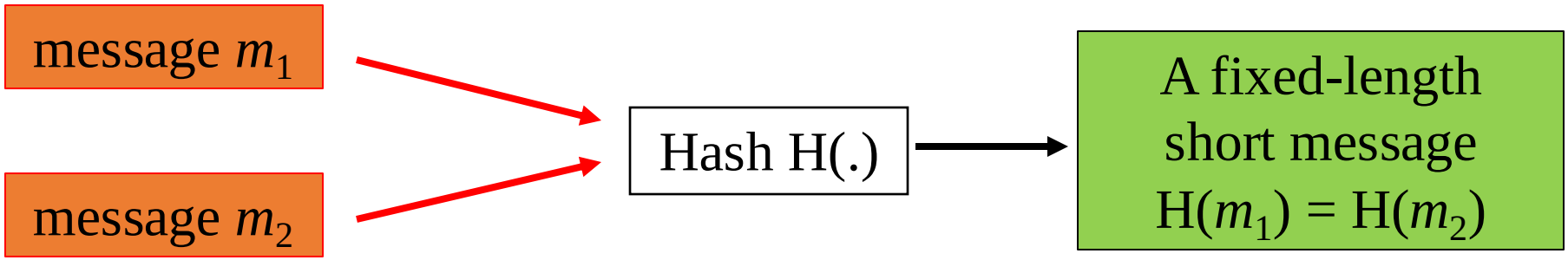
- 3) Weak collision resistance: Given H(*m*), it's computationally infeasible to find *m'* such that H(*m'*) = H(*m*).



Collision: two or more messages have the same hash.

DESIRABLE PROPERTY OF HASH FUNCTIONS: 4

- 4) Strong collision resistance: computationally infeasible to find m_1, m_2 such that $H(m_1) = H(m_2)$



computationally difficult to compute m_1, m_2!

SUMMARY OF PROPERTIES

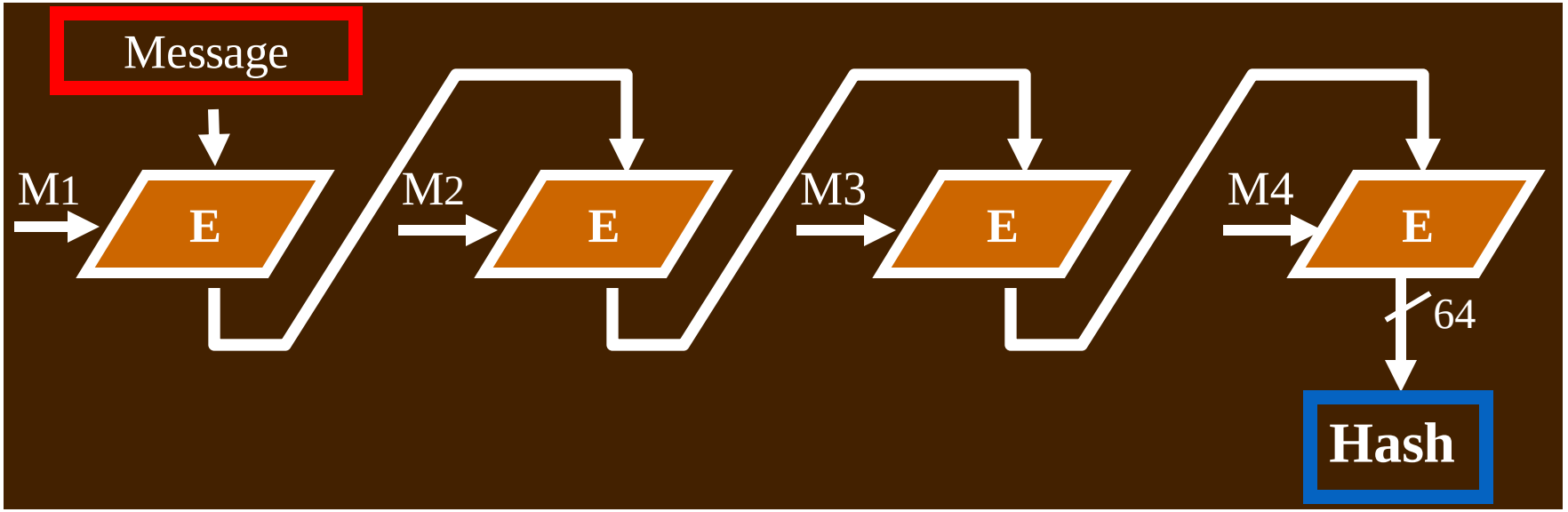
1) Performance: Easy to compute H(m)

2) One-way property: Given H(m) but not m, it's computationally infeasible to find m

3) Weak collision resistance: Given H(m), it's computationally infeasible to find m' such that H(m') = H(m).

4) Strong collision resistance: computationally infeasible to find m_1, m_2 such that H(m_1) = H(m_2)

IS ENCRYPTION A GOOD HASH FUNCTION?



- Computationally efficient?
- One way property?
- Collision resistance?
- Any arbitrary message length?

CS 4173/5173
COMPUTER SECURITY

Hash Function Applications

APPLICATION: FILE AUTHENTICATION

- Want to detect if a file has been changed by someone after it was stored.

- Method
  - Compute the hash H(F) of file F
  - Store H(F) separately from F
  - Can tell at any later time if F has been changed by computing H(F') and comparing to stored H(F)

APPLICATION: PASSWORD STORAGE

- The passwords in the system must be stored in a secure way.

- Method
  - Compute the hash H(P) of password P
  - Store H(P) in a plain-text file (can be known to everyone)
  - Then, how the system verify if the password is correctly entered by a user?
  - Why it is secure?

APPLICATION: PASSWORD STORAGE

- In Ubuntu, the password is stored at /etc/shadow

```
[root@arch01 ~]# cat /etc/shadow | sed 's/michael/test/' | sed 's/mbo/joe/'
root:$6$4GxAA08J$AB7vFkLSCxtVdVMcPav8jZ5u4ZsyG22hy1cqWPdnQgqL84VesJNQYFXSwhfwkhT
UeHNxYwjjUGe8U/sjITBhq/:16672::::::
bin:x:14871::::::
daemon:x:14871::::::
mail:x:14871::::::
ftp:x:14871::::::
http:x:14871::::::
uuidd:x:14871::::::
dbus:x:14871::::::
nobody:x:14871::::::
systemd-journal-gateway:x:14871::::::
systemd-timesync:x:14871::::::
systemd-network:x:14871::::::
systemd-bus-proxy:x:14871::::::
systemd-resolve:x:14871::::::
systemd-journal-upload:!!:16672::::::
systemd-journal-remote:!!:16672::::::
avahi:!:16672::::::
polkitd:!:16672:0:99999:7:::
joe:$6$TA4PslzF$ch961z/ppk1VrmVAqSjSEDf75FIahttselx/bsDdjSXLt8cmsIoX9eAKfVm8epuD
KGvYV1xkohA37aeEvmu8d1:16672:0:99999:7:::
git:!:16683::::::
test:$6$PNkLwU7L$2Hm8YRMGgRoxxt4srAzGBZJFxU7SnlDbaUwb6APg5dyXSiQvQwSxHY1jOi5t2eM
kZ1PwBzY1aHAvZu29wSBpJ0:16735:0:99999:7:::
```


Example of the output of the shadow file

SAVING PASSWORD WITH SALT

- An enhanced way:
  - Create a random number R,
  - Get Hash H(R|P)
    - i.e., hash the combination of random number R and the password P.
  - Save H(R|P) and R in plaintext.
  - R is called the salt.

- How to verify the password?
- Is it more secure?

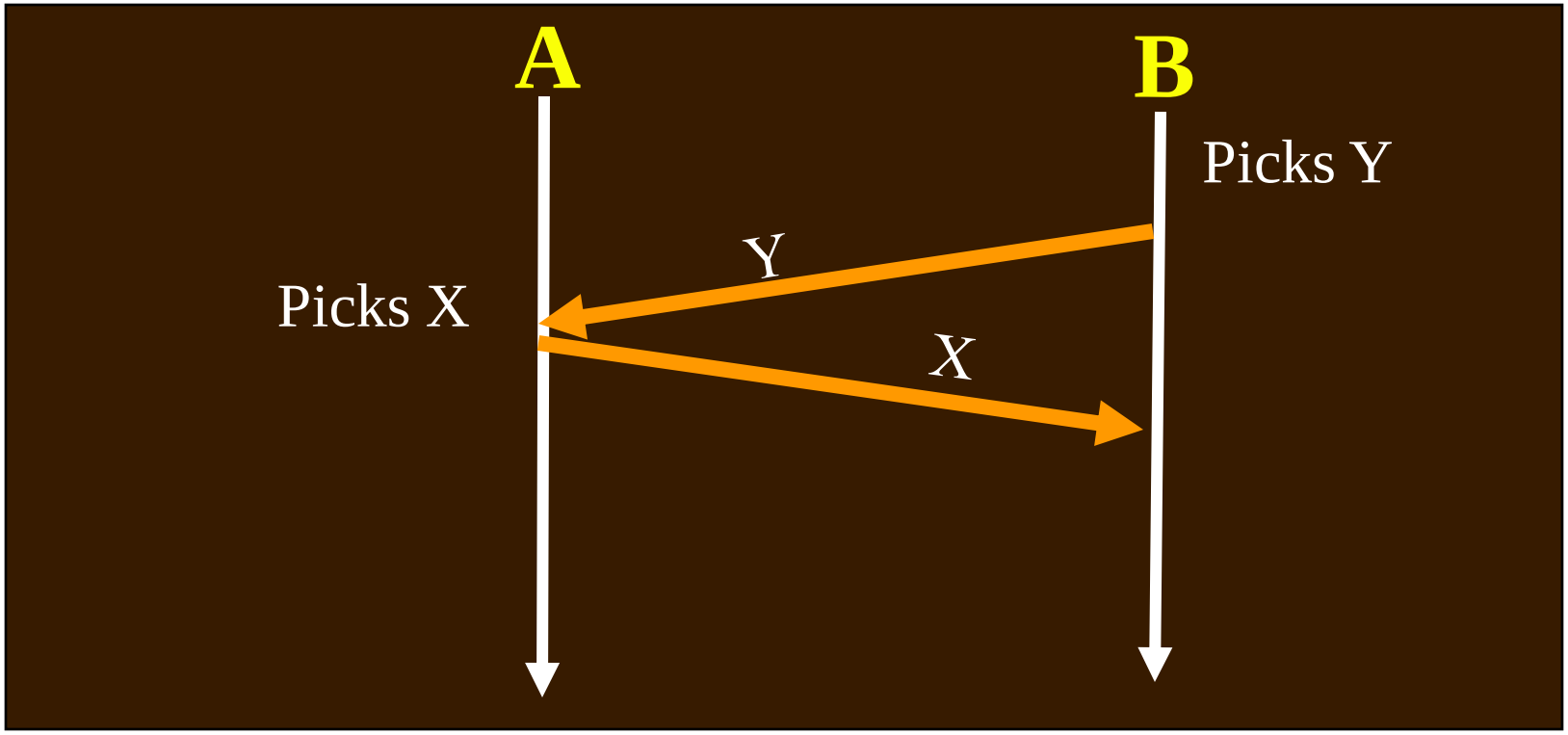
APPLICATION: COMMITMENT PROTOCOLS

- Ensure two parties commit before they communicate.

- Ex.: A and B wish to play the game of "odd or even" over the network
  1. A picks a number X and B picks a number Y
  2. A and B "simultaneously" exchange X and Y
  3. A wins if X+Y is odd, otherwise B wins

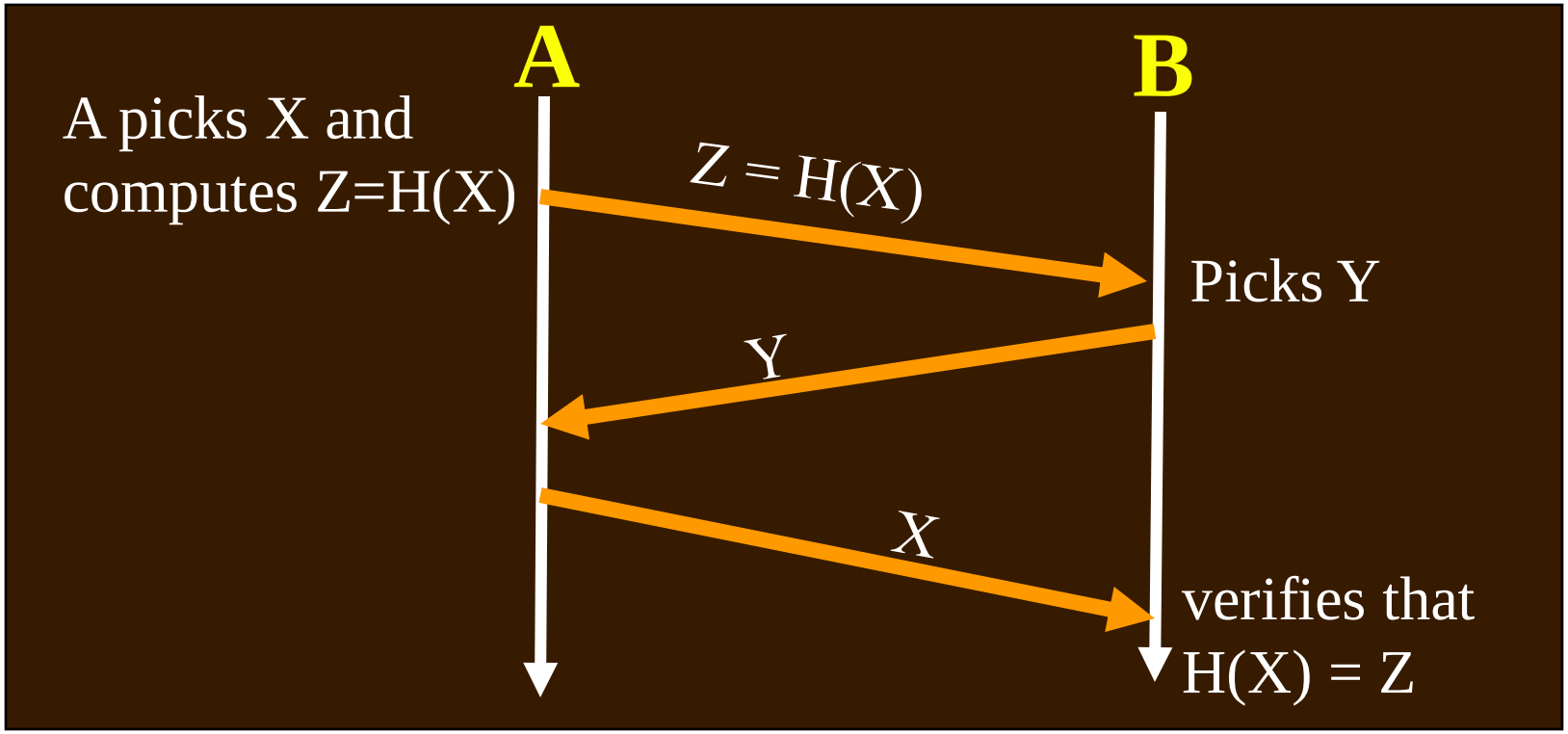
COMMITMENT… (CONT'D)

- If A gets Y before deciding X, A can easily cheat (and vice versa for B)
  - How to prevent this from happening?



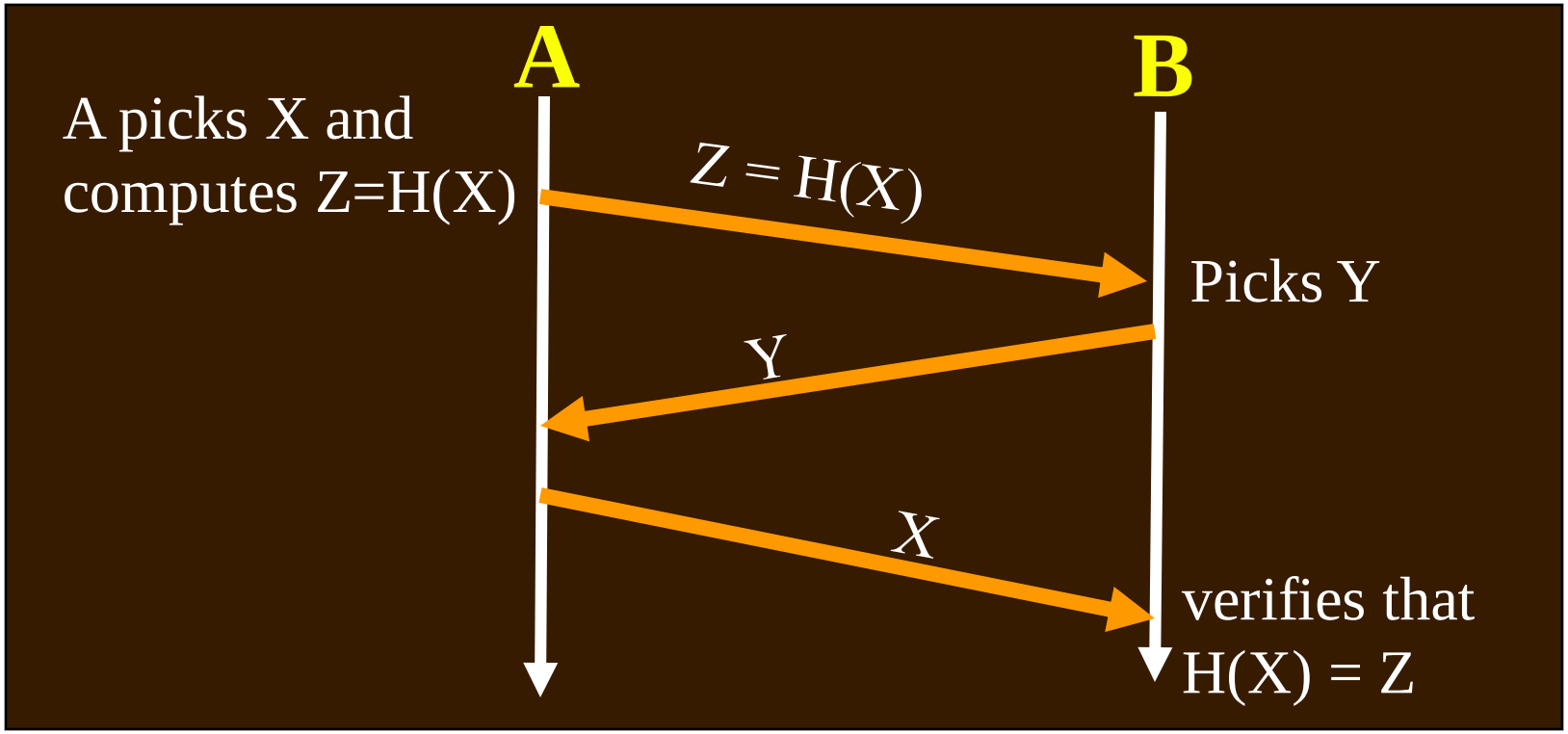
COMMITMENT… (CONT'D)

- Proposal: A must commit to X before B will send Y
- Protocol:



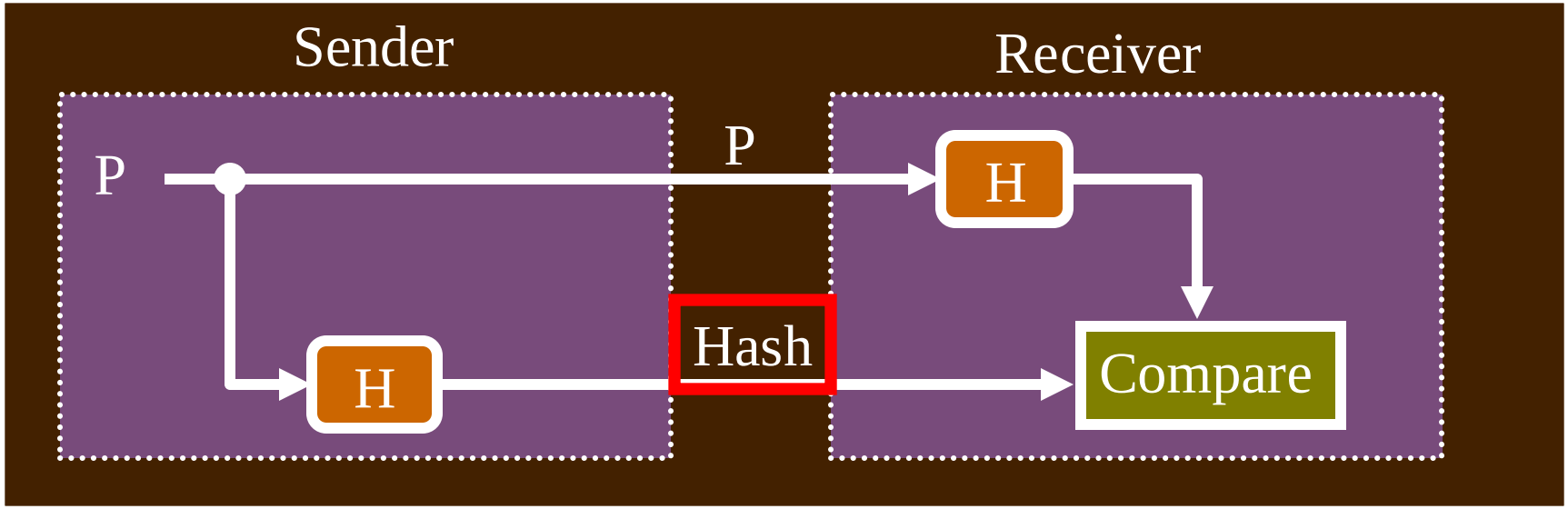
QUESTION

- How should we choose the value space of X in this design?



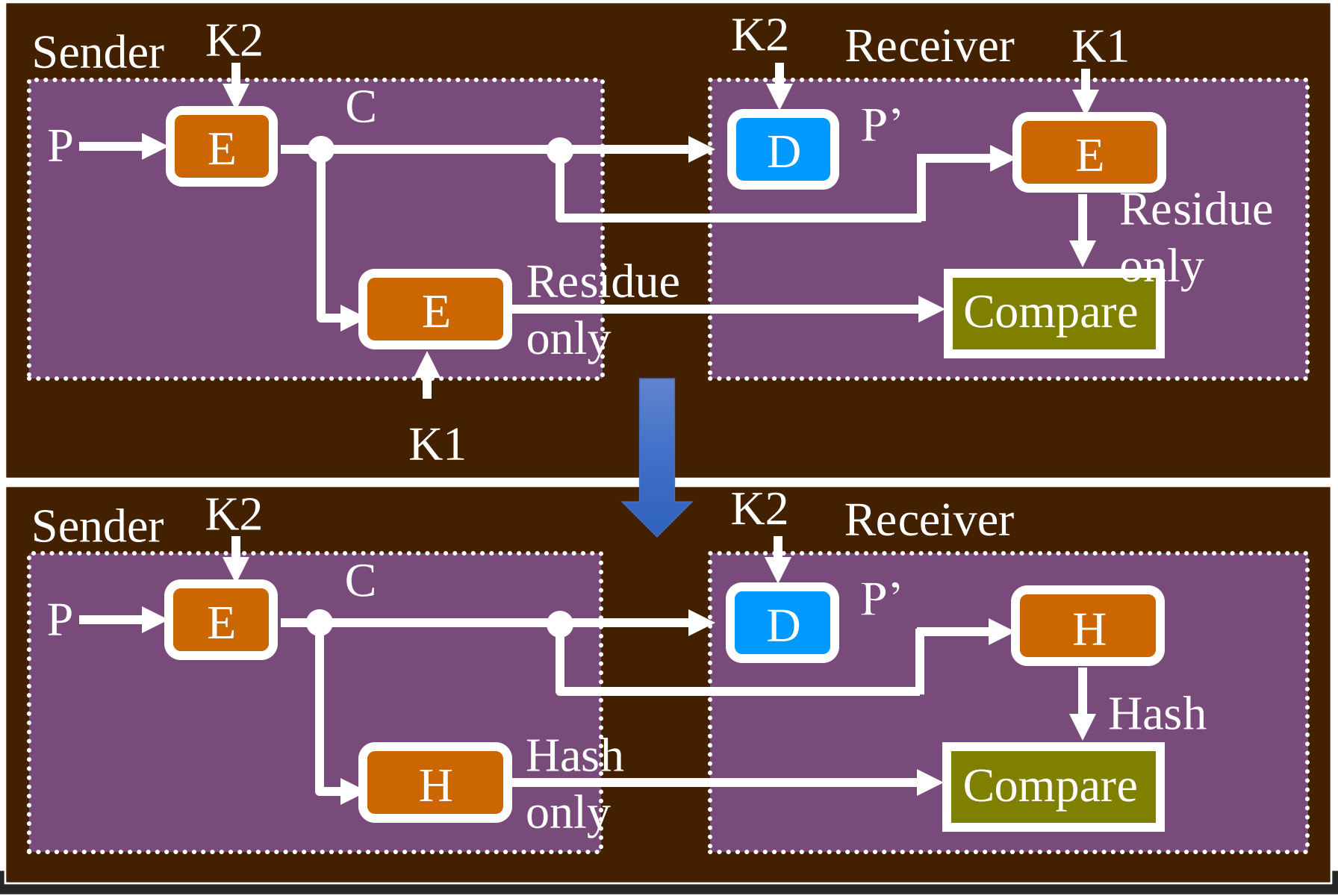
MESSAGE AUTHENTICATION

- For message authentication:



Does this ensure integrity?

ENCRYPT-THEN-AUTHENTICATE

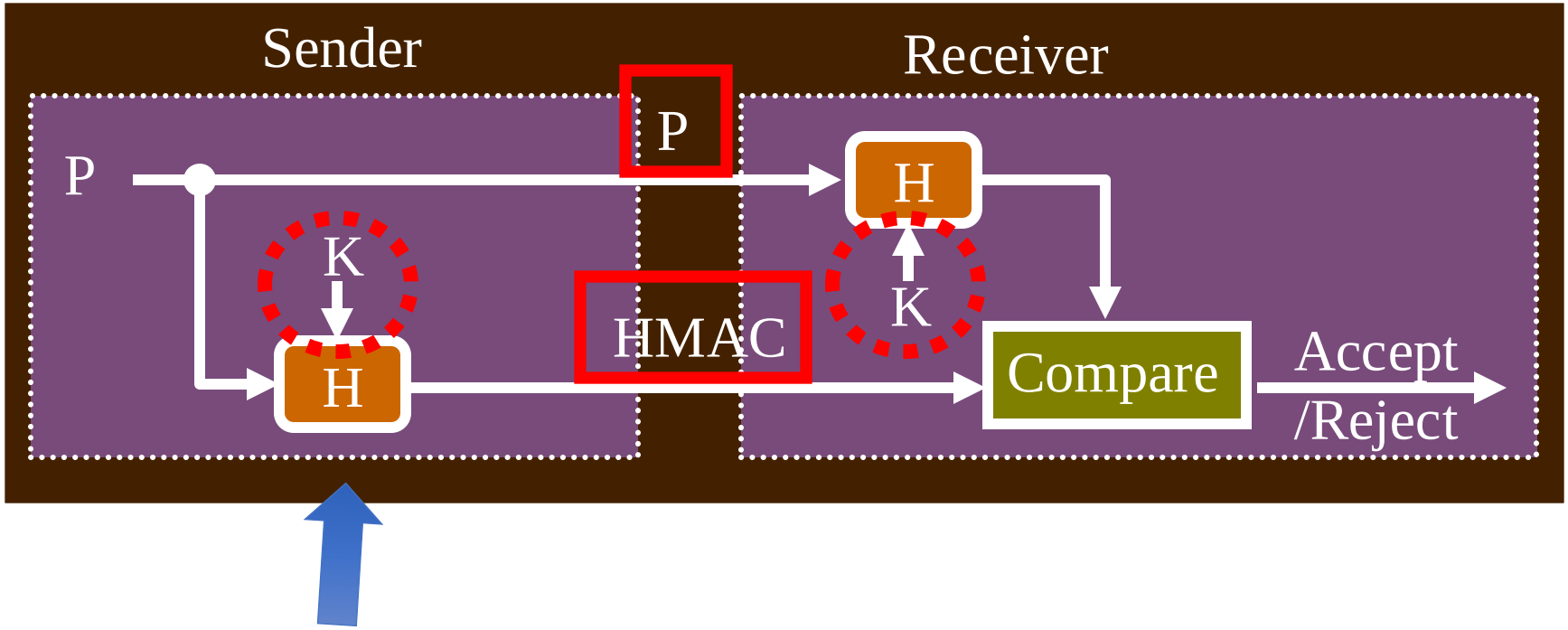


KEY OBSERVATION

- Sender and Receiver should share a common secret K to achieve:
  - Confidentiality
  - Integrity

MESSAGE AUTHENTICATION

- We need to use the key, but how to add the key to hash function?



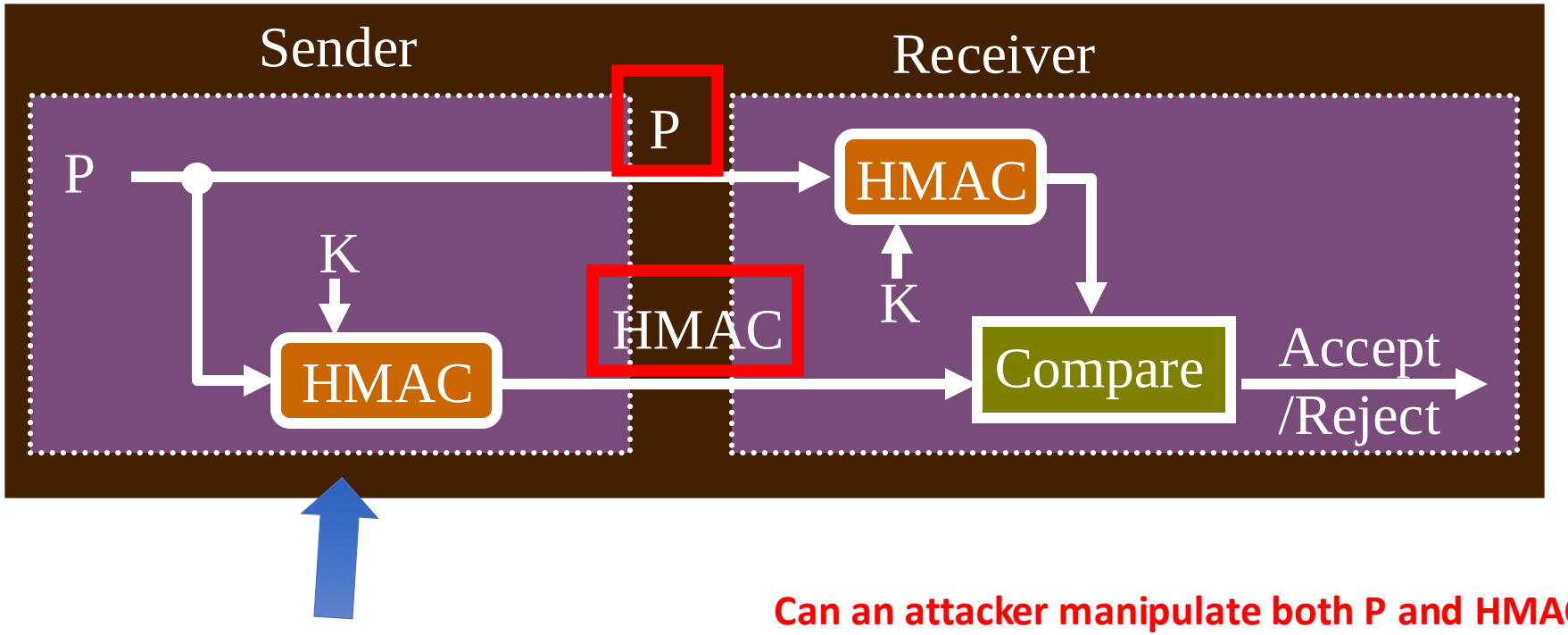
Something like: H(K|P)?

HMAC

- A simple combination H(K|P)
  - Length extension attack
    - Using an existing hash to calculate another hash of a message appended by the attacker's message
      - Take advantage of certain hash function architectures
- HMAC (https://en.wikipedia.org/wiki/HMAC)
  - keyed-hash message authentication code
  - Standard RFC2104
    - Given key K, message M, hash function H( )
    - **HMAC(K, M) = H((K' ⊕ opad) | H(K' ⊕ ipad | M))**
      - **K' = H(K), with padding to K if necessary**
      - **opad: outer padding, consisting of repeated bytes valued 0x5c**
      - **ipad: inner padding, consisting of repeated bytes valued 0x36.**
      - **|: concatenation**
      - **⊕: XOR**

MESSAGE AUTHENTICATION

- We need to use the key, but how to add the key to hash function?

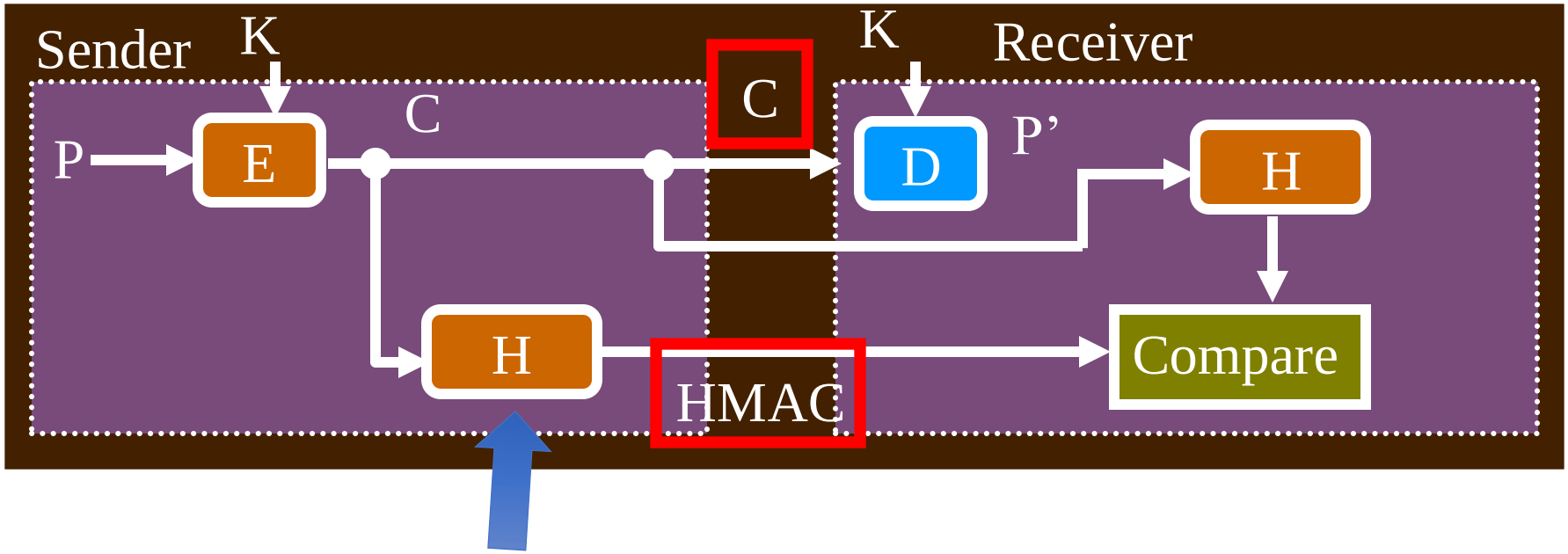


H((K' ⊕ opad) | H(K' ⊕ ipad | P))

Can an attacker manipulate both P and HMAC to pass the authentication verification?

ENCRYPT-THEN-AUTHENTICATE

The way to do encryption and authentication



H((K' ⊕ opad) | H(K' ⊕ ipad | P))

SUMMARY

- Hash functions
  - 4 Important properties
  - Applications
    - Password Storage
    - File Authentication
    - Commitment Protocols
    - HMAC