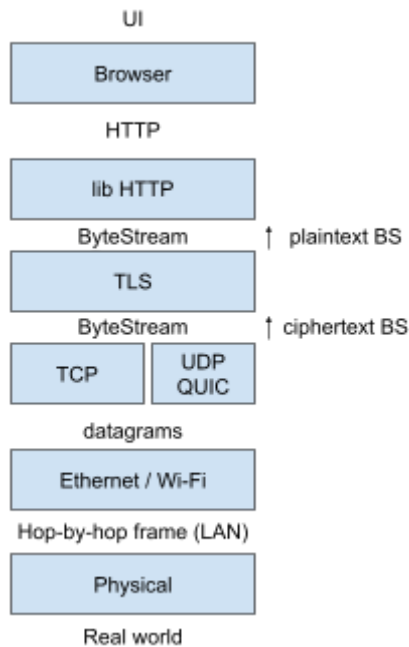


## Today: Security



- Before discussing the system property, a common understanding of the threat model is necessary.

<u>Threat Model</u>	Mitigations / Techniques	<u>System Property</u>
Accidental corruption	<ul style="list-style-type: none"> <li>- IP header checksum</li> <li>- TCP/UDP has header + payload checksum</li> <li>- Ethernet has header + payload FCS</li> </ul>	<b>Integrity</b> - data received = data sent
Adversarial modification (Modify dst address / payload and modify the checksum)	<ul style="list-style-type: none"> <li>- Secure hash with agreed hash value</li> <li>- Message Authentication Code</li> </ul>	
Replay	Idempotence of messages	
	<ul style="list-style-type: none"> <li>- AEAD</li> <li>- AKE</li> </ul>	<b>Confidentiality</b> - only intended recipients can see the message
		<b>Authenticity</b> - parties are who they say they are

## Cryptography Tools

- Secure hash algorithm:  $\text{hash}: X: \text{arbitrary-length} \rightarrow Y: 256 \text{ bits}$ 
  - $\text{hash}$  is a one-way function. In other words, given  $y$ , it's hard to find the  $x$  such that  $\text{hash}(x) = y$ .

- If two parties agree on the  $y$  before-hands, then the receiving party can verify whether the  $x$  is not corrupted by calculating  $\text{hash}(x)$ .
- (But if someone corrupt the message for sending  $y$ , and change it to  $y'$ , which it get from  $x'$  such that  $\text{hash}(x') = y'$ , this may still be insecure, so that the process of sending  $y$  needs to be done in a 100% secure way: e.g. hand a physical piece of paper in person. And this needs to happen for every  $x$ ).
- Trust On First Use (TOFU)
- Message Authentication Code (keyed hash)
  - $\text{mac}(x, \text{key}) \rightarrow \text{tag}$
  - $\text{verify}(x, \text{tag}, \text{key}) \rightarrow \text{bool}$
  - The adversarial party cannot generate a  $\text{tag}$  that passes the  $\text{verify}$  without knowing the  $\text{key}$ .
  - The key still needs to be sent in a secure way, but this only needs to be done once.
- Authenticated Encryption (AE(AD))
  - $\text{box}(\text{plain text}, \text{key}) \rightarrow (\text{cipher text}, \text{tag})$
  - $\text{unbox}(\text{cipher text}, \text{tag}, \text{key}) \rightarrow \text{optional}\langle \text{plain text} \rangle$
  - It's hard to generate a pair of  $(\text{cipher text}, \text{tag})$  to pass the  $\text{unbox}$  function, and it's hard to  $\text{unbox}$  a cipher text without knowing the  $\text{key}$ .
  - But still , we have the pain of how to establish a shared secret.
- Public-key Cryptography / Authenticated Key Exchange (AKE)
  - Alice: (public key<sub>1</sub>, private key<sub>1</sub>) and Bob: (public key<sub>2</sub>, private key<sub>2</sub>)
  - So Alice know public<sub>1</sub>, private<sub>1</sub> and public<sub>2</sub>
  - Bob know public<sub>2</sub>, private<sub>2</sub> and public<sub>1</sub>
  - Alice sends some  $x_1$  to Bob
  - Bob sends some  $x_2$  to Alice
  - Adversarial parties can observe public<sub>1</sub>, public<sub>2</sub>,  $x_1$ ,  $x_2$
  - And, we have:  $\text{AKE}(x_1, x_2, \text{private}_1, \text{public}_2) = \text{AKE}(x_1, x_2, \text{private}_2, \text{public}_1) = \text{key}$  and this  $\text{key}$  is only known by Alice and Bob.