Today: Security



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

| Threat Model | Mitigations / Techniques | System Property |
| --- | --- | --- |
| Accidental corruption | * IP header checksum * TCP/UDP has header + payload checksum * Ethernet has header + payload FCS | **Integrity** - data received = data sent |
| Adversarial modification  (Modify dst address / payload and modify the checksum) | * Secure hash with agreed hash value * Message Authentication Code |
| Replay | Idempotence of messages |  |
|  | * AEAD * AKE | **Confidentiality** - only intended recipients can see the message |
|  | **Authenticity** - parties are who the say they are |

Cryptography Tools

* Secure hash algorithm: hash: X: arbitrary-length -> Y: 256 bits
  + hash is a one-way function. In other words, given y, it’s hard to find the x such that 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 hash(x).
  + (But if someone corrupt the message for sending y, and change it to y’, which it get from x’ such that 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)
  + mac(x, key) -> tag
  + verify(x, tag, key) -> bool
  + The adversarial party cannot generate a tag that passes the verify without knowing the key.
  + The key still needs to be sent in a secure way, but this only needs to be done once.
* Authenticated Encryption (AE(AD))
  + box(plain text, key) -> (cipher text, tag)
  + unbox(cipher text, tag, key) -> optional<plain text>
  + It’s hard to generate a pair of (cipher text, tag) to pass the unbox function, and it’s hard to unbox a cipher text without knowing the key.
  + But still , we have the pain of how to establish a shared secret.
* Public-key Cryptography / Authenticated Key Exchange (AKE)
  + Alice: (public key\_1, private key\_1) and   
    Bob: (public key\_2, private key\_2)
  + So Alice know public\_1, private\_1 and public\_2
  + Bob know public\_2, private\_2 and public\_1
  + Alice sends some x\_1 to Bob
  + Bob sends some x\_2 to Alice
  + Adversarial parties can observe public\_1, public\_2, x\_1, x\_2
  + And, we have: AKE(x\_1, x\_2, private\_1, public\_2) = AKE(x\_1, x\_2, private\_2, public\_1) = *key* and this *key* is only known by Alice and Bob.