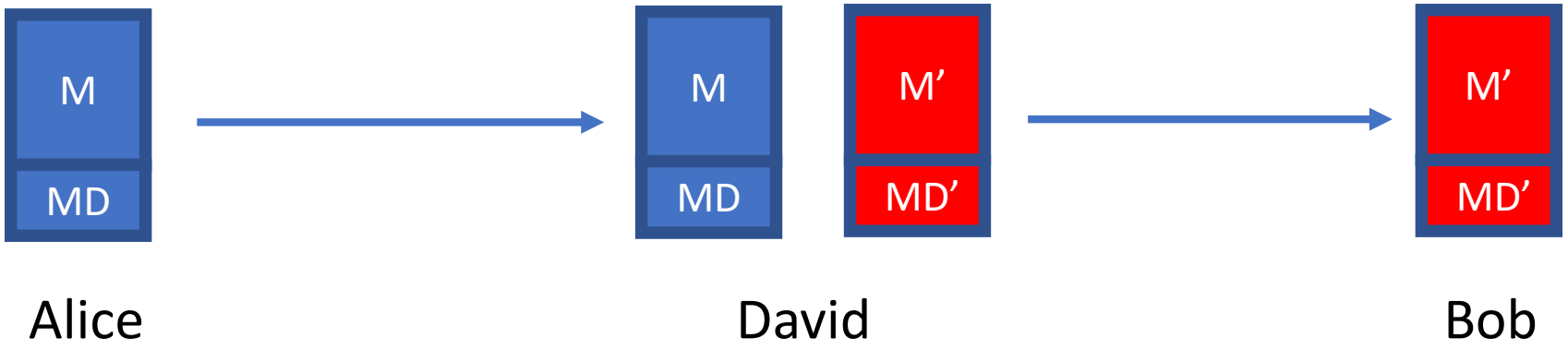
Does hashes provide integrity?

- It depends on your threat model
- Scenario
  - Mozilla publishes a new version of Firefox on some download servers
  - Alice downloads the program binary
  - How can she be sure that nobody tampered with the program?
- Idea: use cryptographic hashes
  - Mozilla hashes the program binary and publishes the hash on its website
  - Alice hashes the binary she downloaded and checks that it matches the hash on the website
  - If Alice downloaded a malicious program, the hash would not match (tampering detected!)
  - An attacker can’t create a malicious program with the same hash (collision resistance)
- Threat model: We assume the attacker cannot modify the hash on the website
  - We have integrity, as long as we can communicate the hash securely

Do hashes provide integrity?

- It depends on your threat model
- Scenario
  - Alice and Bob want to communicate over an insecure channel
  - David might tamper with messages
- Idea: Use cryptographic hashes
  - Alice sends her message with a cryptographic hash over the channel
  - Bob receives the message and computes a hash on the message
  - Bob checks that the hash he computed matches the hash sent by Alice
- Threat model: David can modify the message *and the hash*
  - No integrity!

Man-in-the-middle attack



Do hashes provide integrity?

- It depends on your threat model
- If the attacker can modify the hash, hashes don’t provide integrity
- Main issue: Hashes are *unkeyed* functions
  - There is no secret key being used as input, so any attacker can compute a hash on any value

Solutions

- A message digest created using a secret symmetric key is known as a Message Authentication Code (MAC), because it can provide assurance that the message has not been modified
- The sender can also generate a message digest and then encrypt the digest using the private key of an asymmetric key pair, forming a digital signature. The signature must then be verified by the receiver through comparing it with a locally generated digest

Hashes: Summary

- Map arbitrary-length input to fixed-length output
- Output is deterministic
- Security properties
  - One way: Given an output y, it is infeasible to find any input x such that $H(x) = y$.
  - Second preimage resistant: Given an input x, it is infeasible to find another input $x' \neq x$ such that $H(x) = H(x')$.
  - Collision resistant: It is infeasible to find another any pair of inputs $x' \neq x$ such that $H(x) = H(x')$.
- Some hashes are vulnerable to length extension attacks
- Hashes don’t provide integrity (unless you can publish the hash securely)