CIS 6930:
IoT Security

Lecture 9

Prof. Kaushal Kafle

Spring 2025

Derived from slides by Adwait Nadkarni, William Enck, Micah Sherr and Patrick McDaniel

Class Notes and Clarifications

- Clarifications about the use of Generative AI in this class
  - Follow the syllabus!
  - If syllabus is (and was) not followed, you automatically get a 0 on submissions.
  - If you are unsure, clear with me first!
- Progress on the project..

User Authentication

Three Flavors of Credentials

- … are evidence used to prove identity
- Credentials can be
  1. **Something I am**
  2. **Something I have**
  3. **Something I know**

Web Authentication

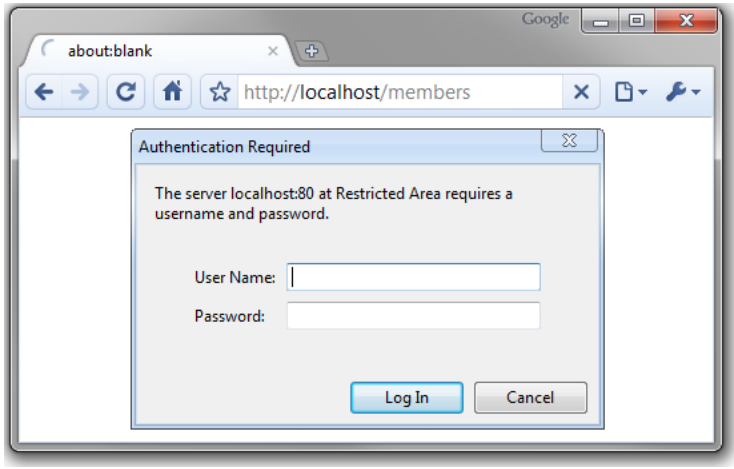
(still based on“something you know”)

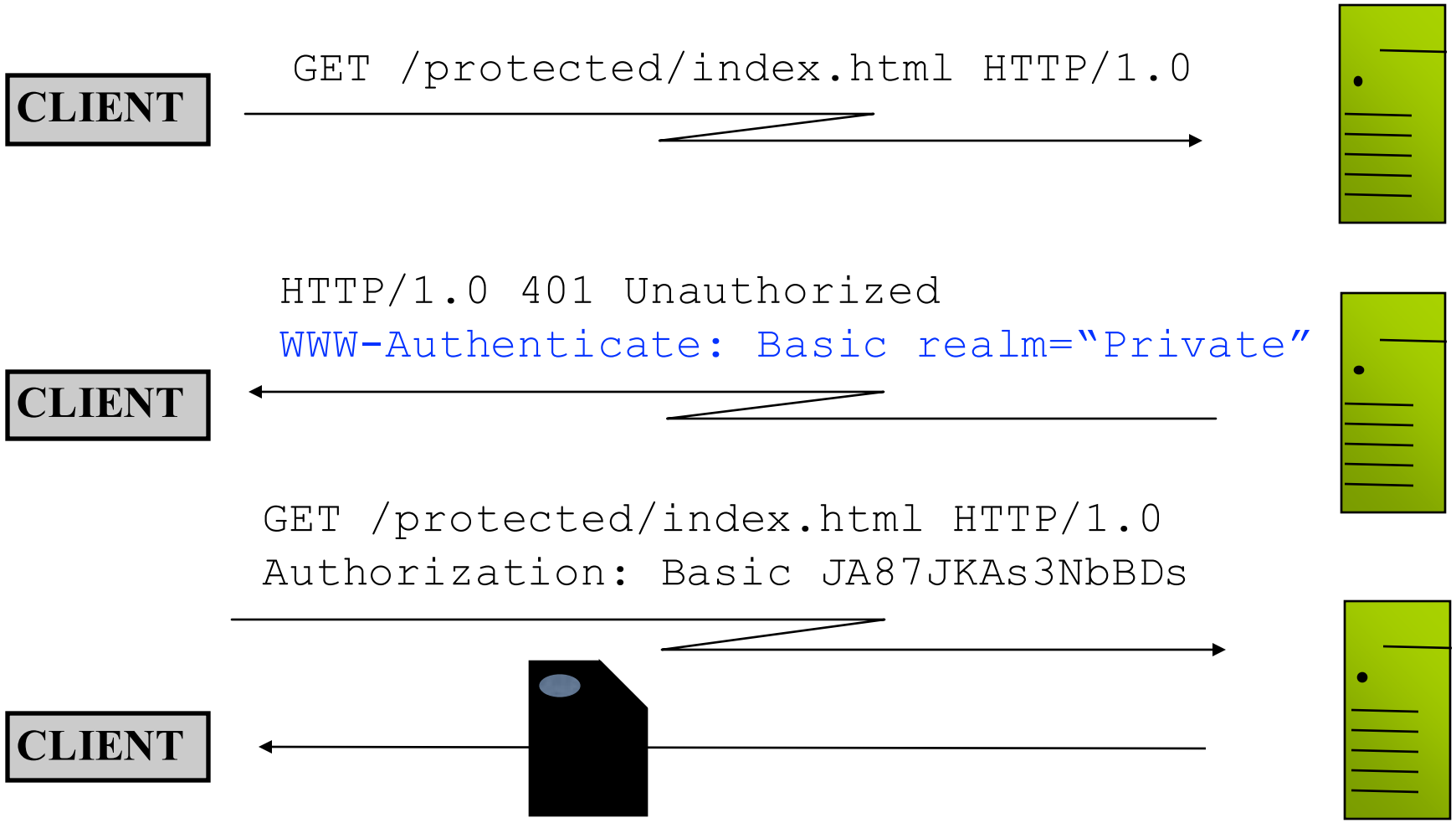
Credentials can be

1. Something I am
2. Something I have
3. **Something I know**

Web Authentication

- Authentication is a bi-directional process
  - Client
  - Server
  - Mutual authentication
- Several standard authentication tools
  - Basic (client)
  - Digest (client)
  - Secure Socket Layer (server, mutual)





Basic Authentication

Basic Authentication -- is this secure?

- Encoded ! = Encrypted
  - Passwords easy to intercept (base-64 encoded; not encrypted)
- Passwords:
  - easy to guess
  - easy to share
- No server authentication - easy to fool client into sending password to malicious server

Digest Authentication

Challenge/Response

- Challenge nonce is a one time random string/value

$$\text{nonce} = H(\text{IPaddress} : \text{timestamp} : \text{server secret})$$

  - more generally, a **nonce** is number or string (often randomly or pseudorandomly chosen) that is **only used once**

- Response: challenge hashed with username and password

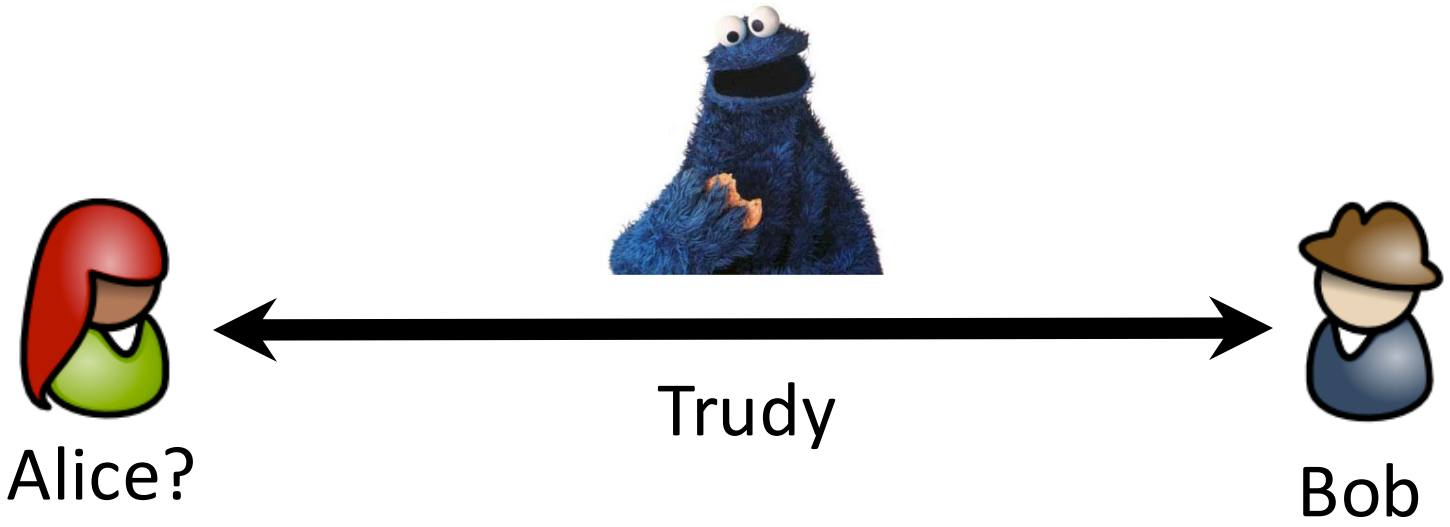
$$response = H(H(name : realm : password) : nonce : H(request))$$

Advantages of Digest over Basic

- Cleartext password never transmitted across network
- Cleartext password never stored on server
- **Replay attacks** difficult
- Intercepted response only valid for a single URL
- Shared disadvantages
  - Vulnerable to man-in-the-middle attacks (no server-side auth)
  - Document itself can be sniffed

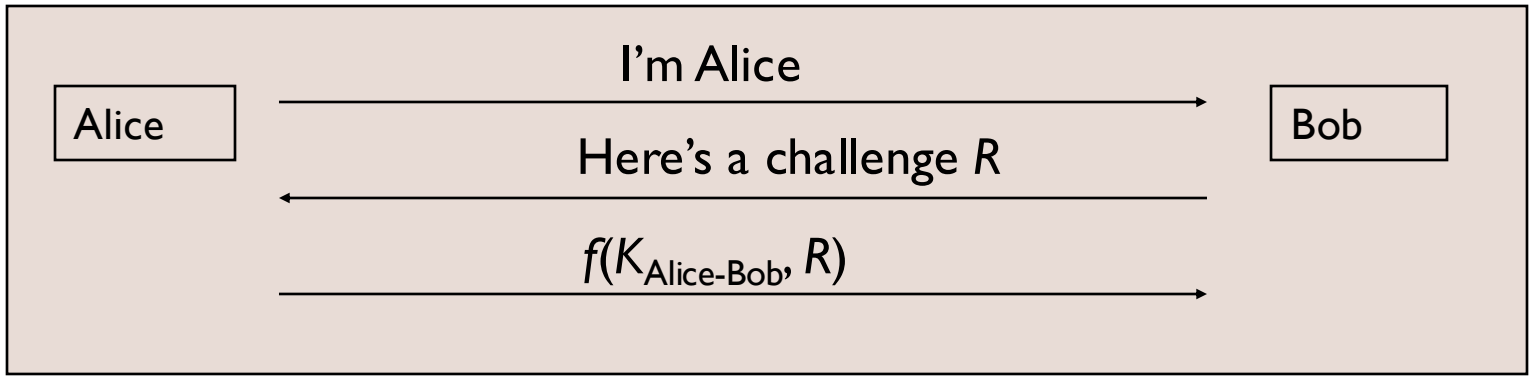
Authentication Handshakes

- Secure communication almost always includes an initial authentication handshake.
  - Authenticate each other
  - Establish session keys
  - *This process is not trivial; flaws in this process undermine secure communication*



Authentication

Authentication with Shared Secret



- Weaknesses
  - Authentication is not mutual; Trudy can convince Alice that she is Bob
  - Trudy can hijack the conversation after the initial exchange
  - If the shared key is derived from a password, Trudy can mount an off-line password guessing attack (*R is known*)
  - Trudy may compromise Bob's database and later impersonate Alice

Authentication with Shared Secret (Cont'd)

- A variation
  - Requires reversible cryptography
  - Other variations are possible
- Weaknesses
  - All the previous weaknesses remain
  - Trudy doesn't have to see R to mount off-line password guessing if R has certain patterns (e.g., concatenated with a timestamp)
    - Trudy sends a message to Bob, pretending to be Alice

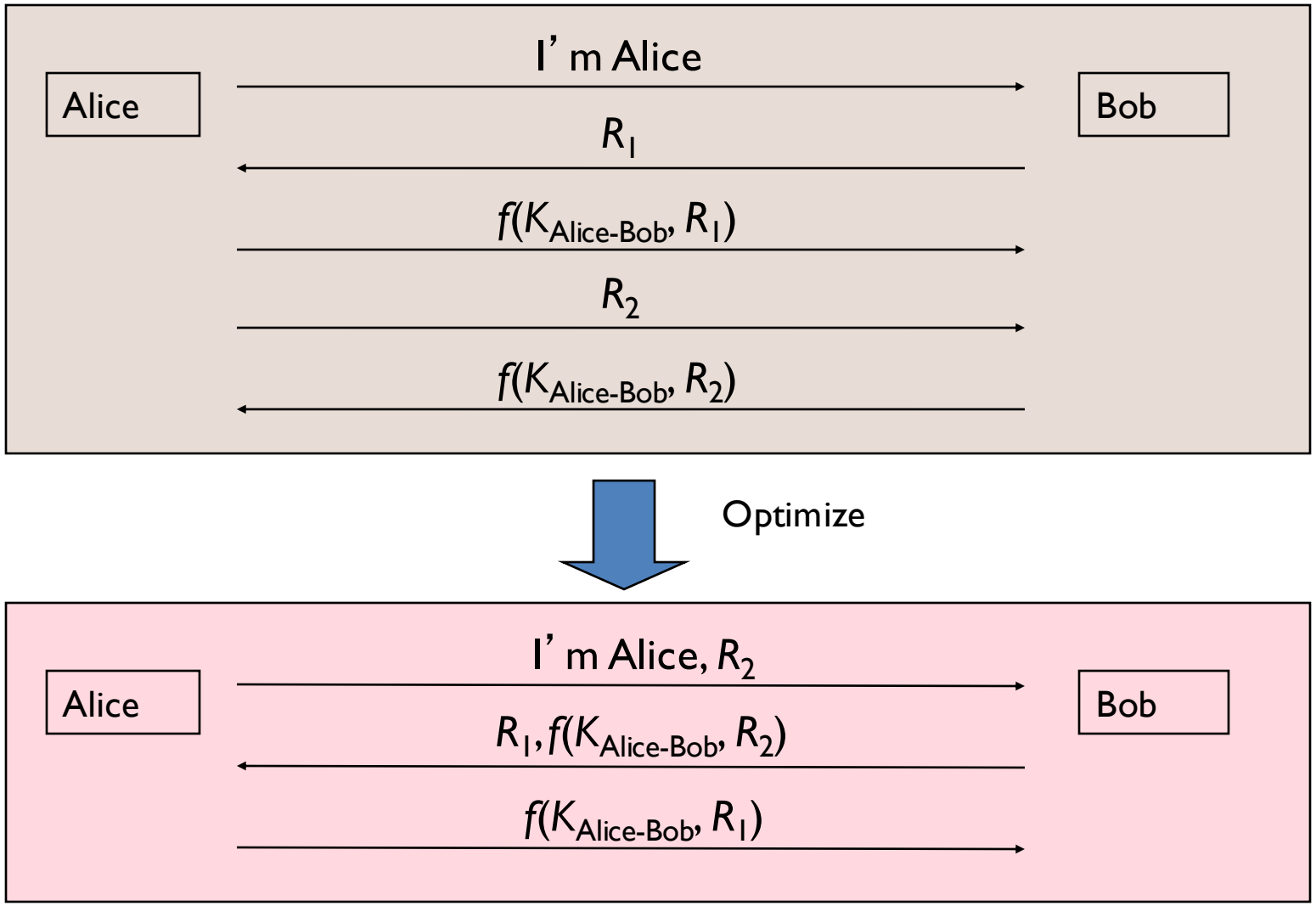
Authentication with Public Key

- Bob's database is less risky
- Weaknesses
  - Authentication is not mutual; Trudy can convince Alice that she is Bob
  - Trudy can hijack the conversation after the initial exchange
  - Trudy can trick Alice into signing something
    - Mitigation: Use different private key for authentication

Authentication with Public Key (Cont'd)

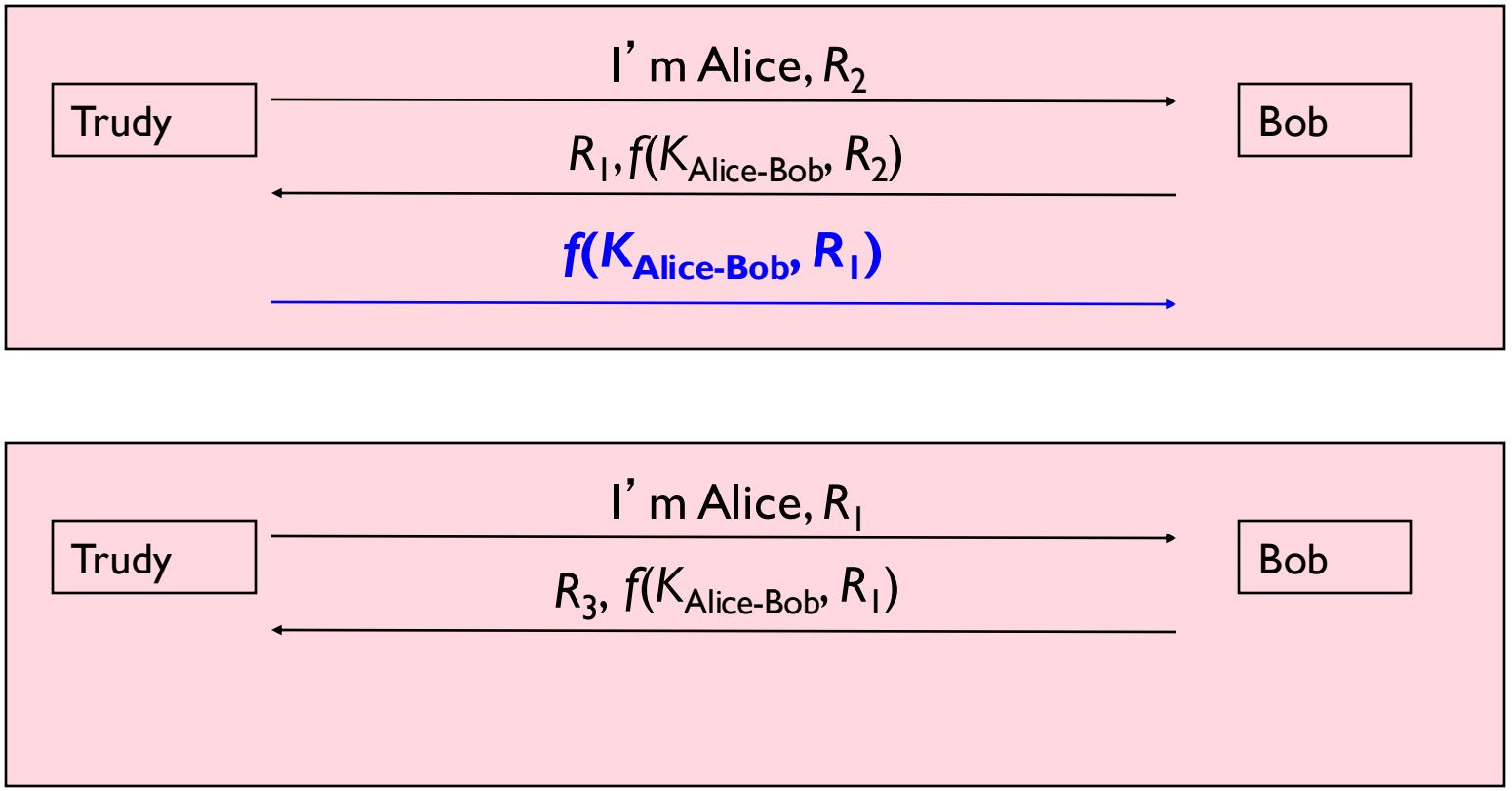
A variation

Mutual Authentication



Mutual Authentication (Cont'd)

- Reflection attack

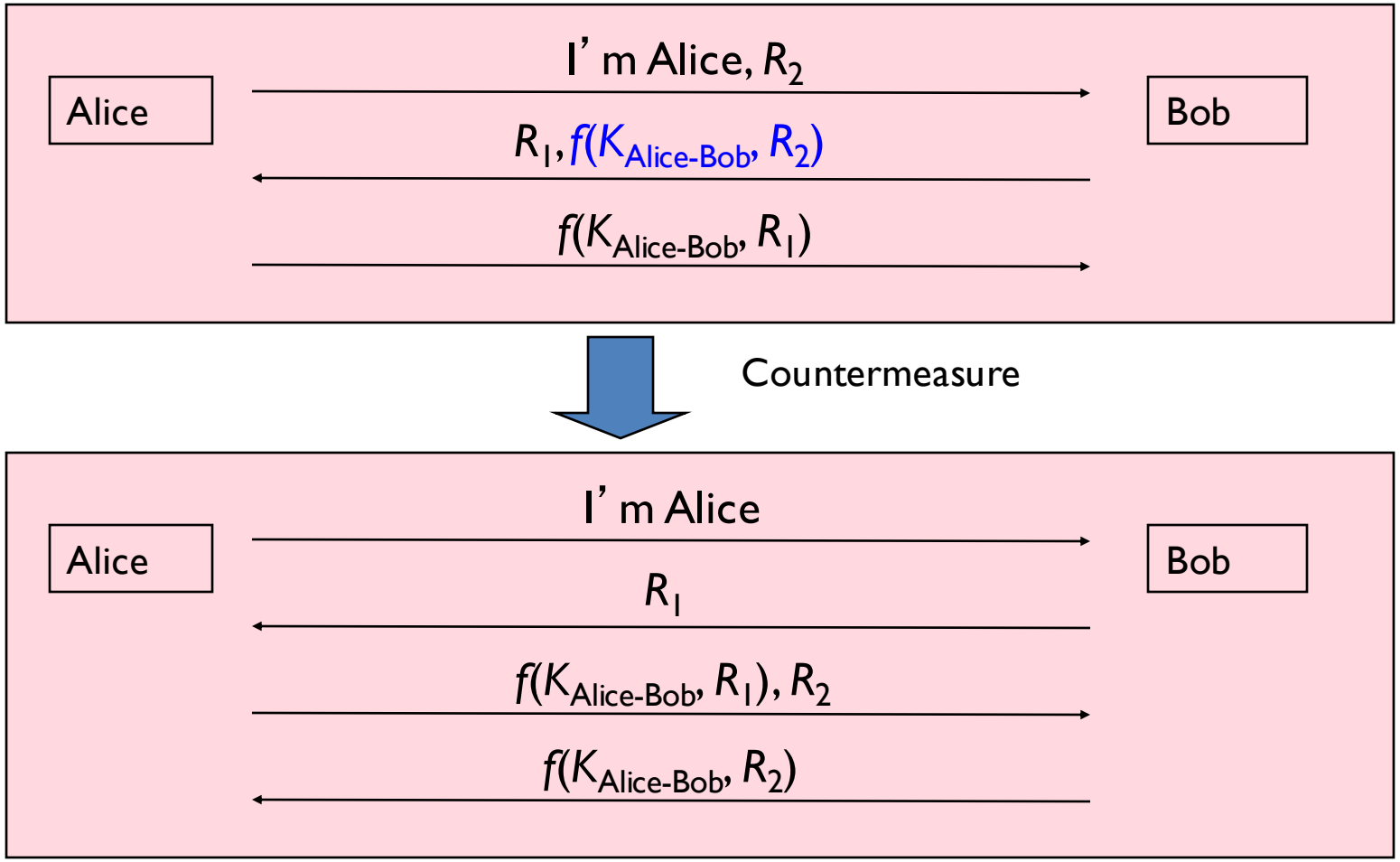


Reflection Attacks (Cont'd)

- Lesson: Don't have Alice and Bob do exactly the same thing
  - Different keys
    - Totally different keys
    - $K_{Alice\text{-}Bob} = K_{Bob\text{-}Alice} + 1$
  - Different Challenges
  - The initiator should be the first to prove its identity
    - Assumption: initiator is more likely to be the bad guy

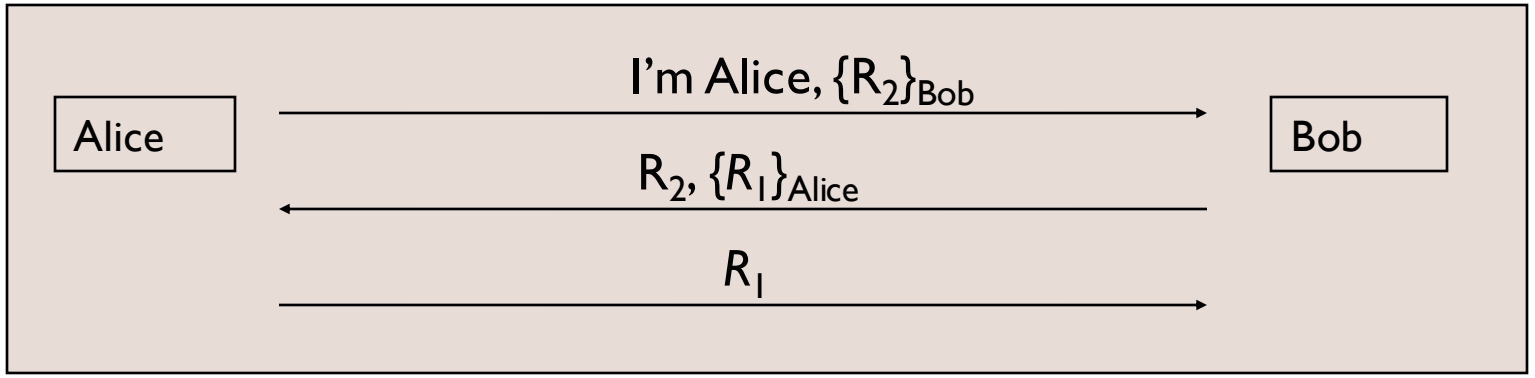
Mutual Authentication (Cont'd)

- Password guessing



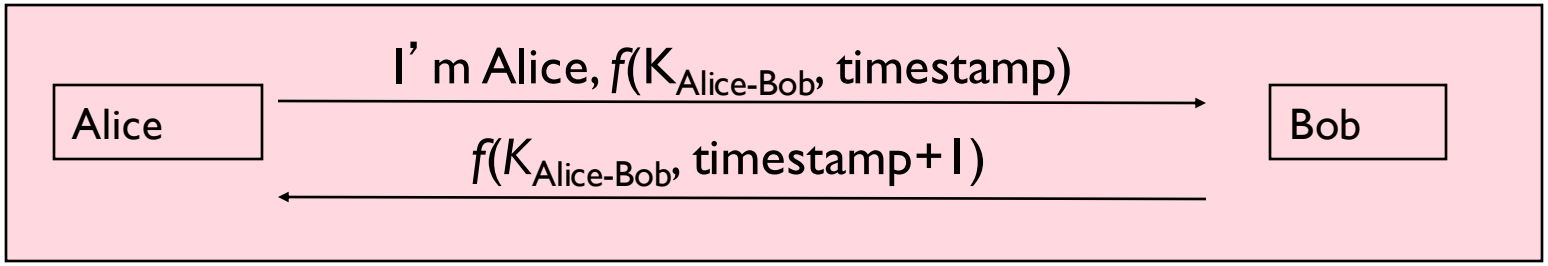
Mutual Authentication (Cont'd)

- Public keys
  - Authentication of public keys is a critical issue



Mutual Authentication (Cont'd)

- Mutual authentication with timestamps
  - Require synchronized clocks
  - Alice and Bob have to encrypt different timestamps

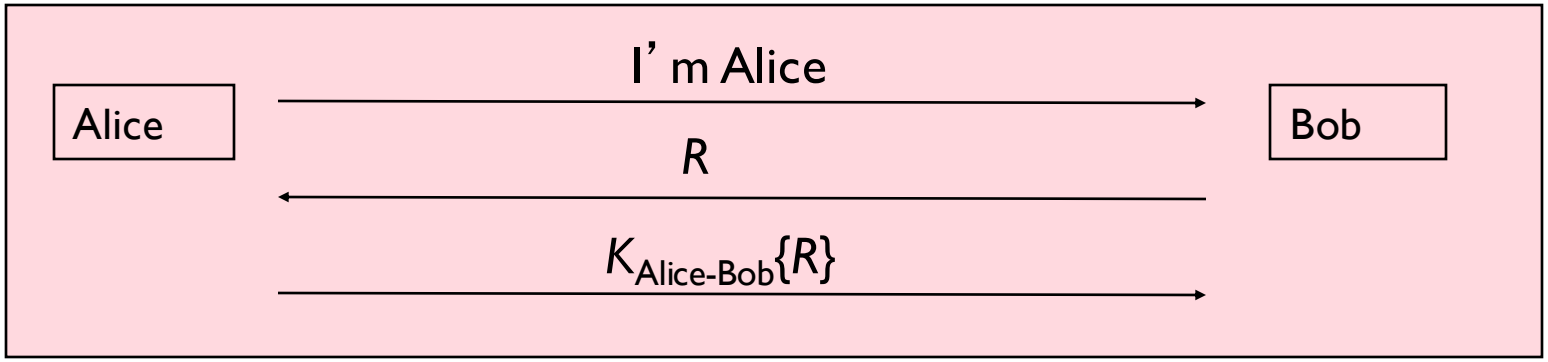


Integrity/Encryption for Data

- Communication after mutual authentication should be cryptographically protected as well
  - Require a *session key* established during mutual authentication

Establishment of Session Keys

- Secret key based authentication
  - Assume the following authentication happened.
  - Can we use $K_{Alice\text{-}Bob}\{R\}$ as the session key?
  - Can we use $K_{Alice\text{-}Bob}\{R+1\}$ as the session key?
  - In general, modify $K_{Alice\text{-}Bob}$ and encrypt R. Use the result as the session key.



Establishment of Session Keys (Cont'd)

- Two-way *public key* based authentication
  1. Alice chooses a random number R, encrypts it with Bob's public key, result used as session key.
     - Trudy may hijack the conversation
  2. Alice encrypts and signs R
     - Trudy may save all the traffic, and decrypt all the encrypted traffic when she is able to compromise Bob
     - Less severe threat

Two-Way Public Key Based Authentication (Cont'd)

- A better approach
  - Alice chooses and encrypts R_1 with Bob's public key
  - Bob chooses and encrypts R_2 with Alice's public key
  - Session key is $R_1 \oplus R_2$
  - Trudy will have to compromise **both** Alice and Bob
- An even better approach
  - Alice and Bob establish the session key with *Diffie-Hellman key exchange*
  - Alice and Bob sign the quantity they send
  - Trudy can't learn anything about the session key even if she compromises both Alice and Bob

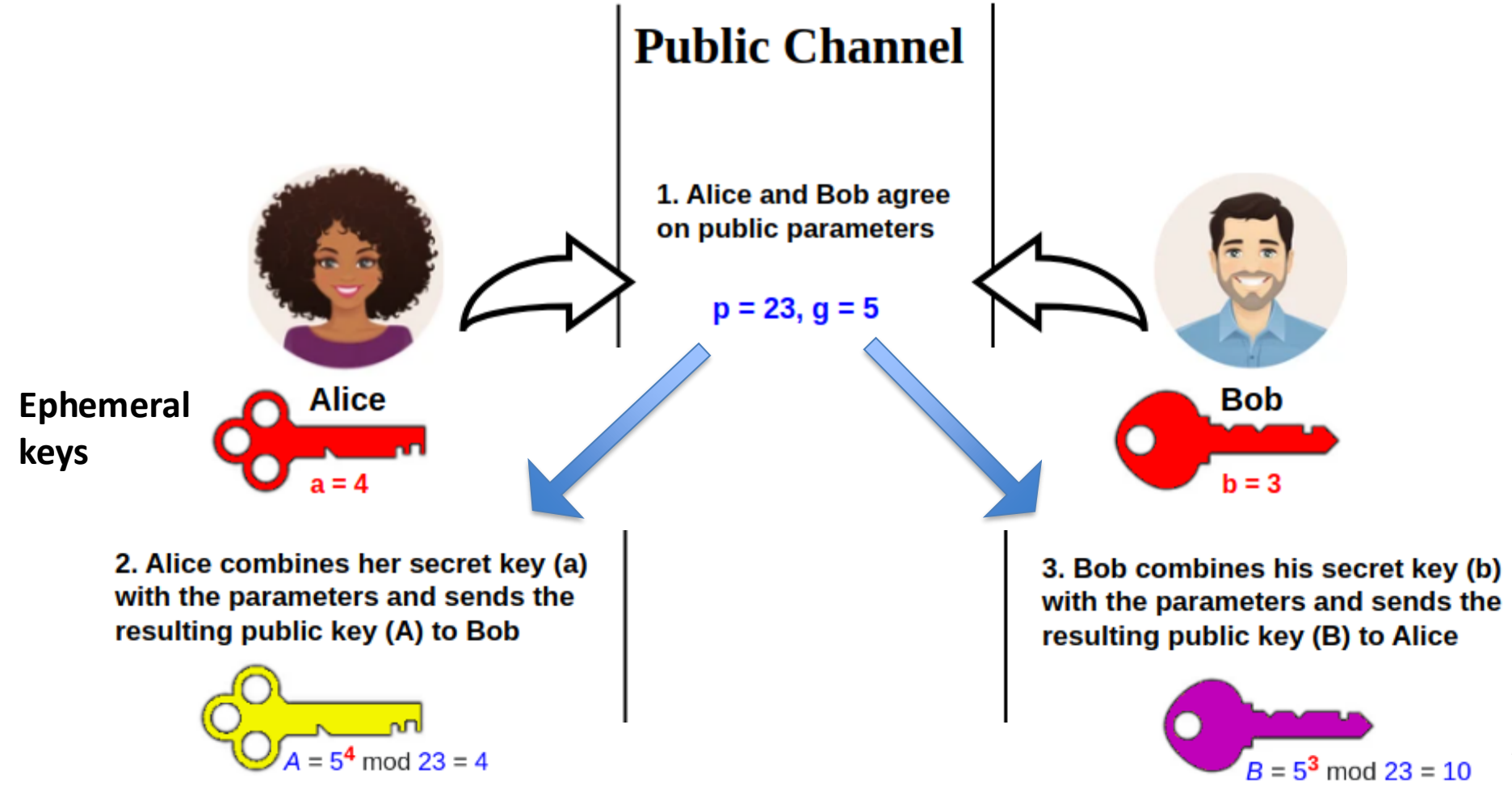
Diffie-Hellman Approach

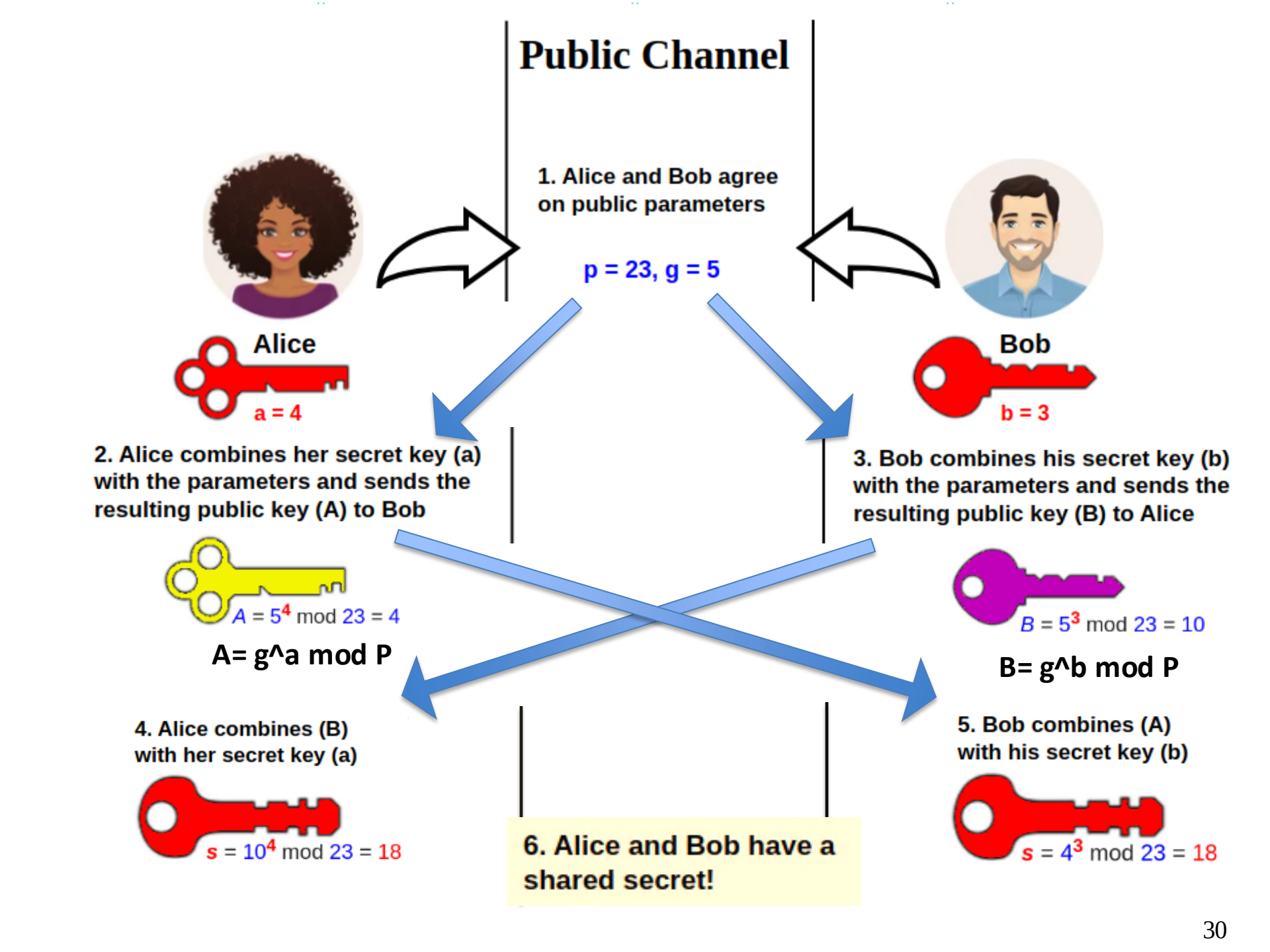
Diffie-Hellman way:

- No pre-exchange of information necessary to arrive at a shared secret!
- Foundation of public key crypto in the modern web

https://www.infoworld.com/article/2334365/understand-diffie-hellman-key-exchange.html

Diffie-Hellman Approach




Public Channel
1. Alice and Bob agree on public parameters
p = 23, g = 5
Alice
a = 4
Bob
b = 3
2. Alice combines her secret key (a) with the parameters and sends the resulting public key (A) to Bob
3. Bob combines his secret key (b) with the parameters and sends the resulting public key (B) to Alice
A = 5^4 mod 23 = 4
A= g^a mod P
B = 5^3 mod 23 = 10
B= g^b mod P
4. Alice combines (B) with her secret key (a)
5. Bob combines (A) with his secret key (b)
s = 10^4 mod 23 = 18
s = 4^3 mod 23 = 18
6. Alice and Bob have a shared secret!

Establishment of Session Keys

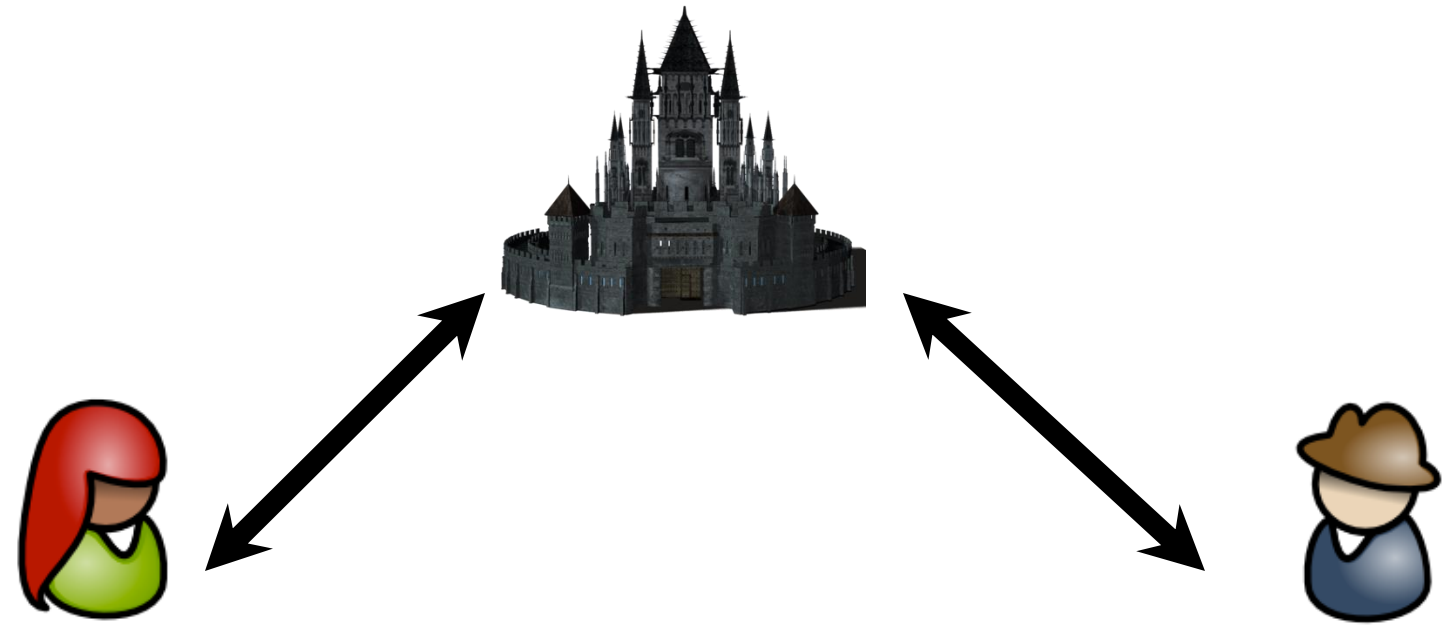
- Ephemeral keys are discarded after the generation of sessions keys
- Different from RSA
  - RSA can encrypt directly, and provides digital identity/signatures.
  - RSA provides *authentication* – mainly used to authenticate servers (via certificates)
  - DH typically faster than RSA

Establishment of Session Keys

- D-H is the primary key-exchange protocol.
  - *Exclusive to key-exchange* i.e., does not provide encryption by itself
- Modern system use RSA to authenticate server, and DH for establishing keys.
  - E.g. DH public parameters signed by server's private key to authenticate server.
- Provides *forward secrecy* (private key compromise does not lead to session key compromise!)
  - Think what happens if a server's private keys are compromised in DH based and RSA based authentication..

Establishment of Session Keys

```
~ openssl ciphers -v
TLS_AES_256_GCM_SHA384          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(256)            Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256    TLSv1.3 Kx=any      Au=any   Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256          TLSv1.3 Kx=any      Au=any   Enc=AESGCM(128)            Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(256)            Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(256)            Mac=AEAD
DHE-RSA-AES256-GCM-SHA384       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(256)            Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305     TLSv1.2 Kx=ECDH     Au=RSA   Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305       TLSv1.2 Kx=DH       Au=RSA   Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256   TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AESGCM(128)            Mac=AEAD
ECDHE-RSA-AES128-GCM-SHA256     TLSv1.2 Kx=ECDH     Au=RSA   Enc=AESGCM(128)            Mac=AEAD
DHE-RSA-AES128-GCM-SHA256       TLSv1.2 Kx=DH       Au=RSA   Enc=AESGCM(128)            Mac=AEAD
ECDHE-ECDSA-AES256-SHA384       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)               Mac=SHA384
ECDHE-RSA-AES256-SHA384         TLSv1.2 Kx=ECDH     Au=RSA   Enc=AES(256)               Mac=SHA384
DHE-RSA-AES256-SHA256           TLSv1.2 Kx=DH       Au=RSA   Enc=AES(256)               Mac=SHA256
ECDHE-ECDSA-AES128-SHA256       TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(128)               Mac=SHA256
ECDHE-RSA-AES128-SHA256         TLSv1.2 Kx=ECDH     Au=RSA   Enc=AES(128)               Mac=SHA256
DHE-RSA-AES128-SHA256           TLSv1.2 Kx=DH       Au=RSA   Enc=AES(128)               Mac=SHA256
```



Mediated Authentication