Secure Messaging

Lecture 23

Messaging

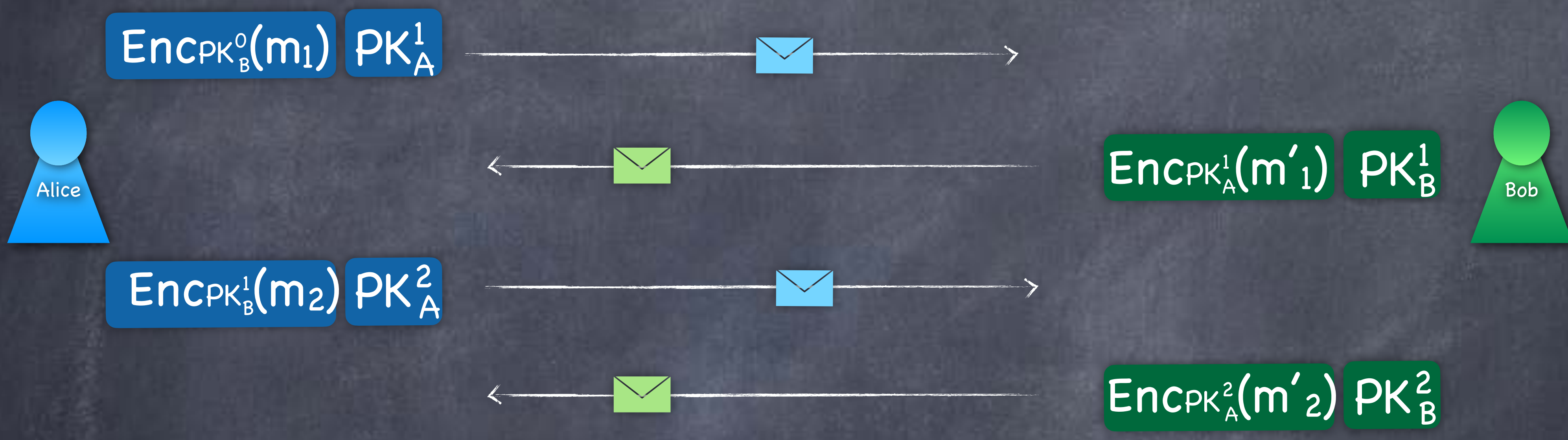
Secure Messaging

- Communication model different from standard setting for TLS
  - Receiver need not be online when Sender sends a message
- Corruption model
  - Server/network is adversarial (trusted identity registration to be enforced separately)
  - Windows of compromise when a party is under adversarial control (or readable to adversary)
    - Messages that are sent/received while a party is corrupt are revealed to the adversary
- Goal: Messages sent/received prior to compromise and after compromise should remain "secure"
  - Forward secrecy (secrecy of prior messages) and "Future secrecy" (secrecy of future messages)
- Protocols rely on secure deletion (of keys and messages)

Secure Messaging

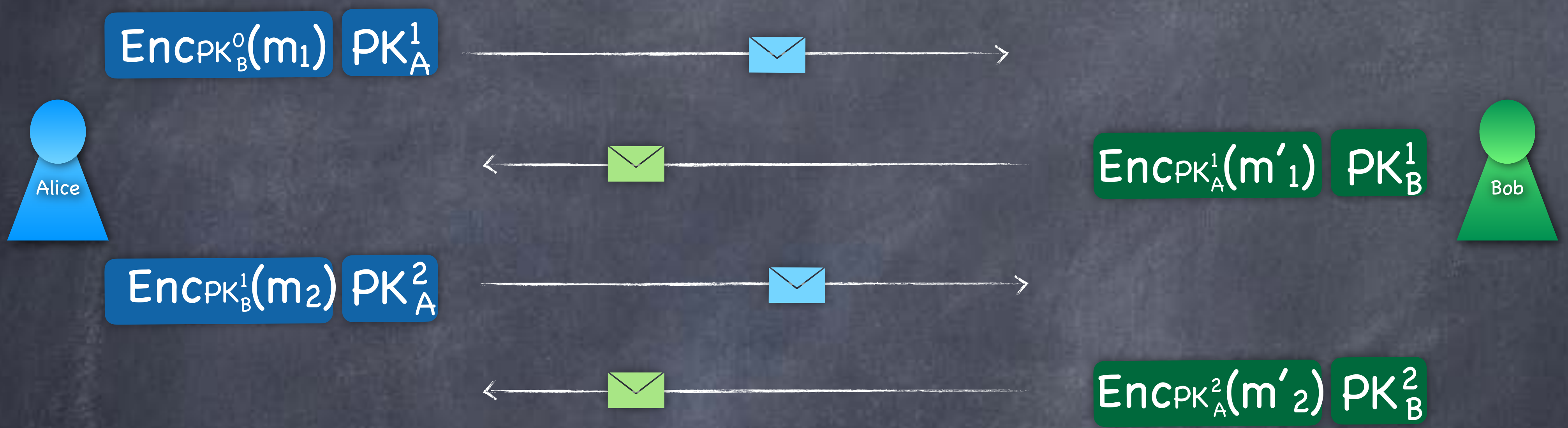
- Many applications/services offering secure chat
  - "Off-The-Record" messaging (2004)
  - Signal protocol (starting 2013)
    - Used in WhatsApp, Google Allo, Facebook Messenger, Skype (optional), etc.
  - More recently, some formal analysis

Synchronous Messaging
A first solution



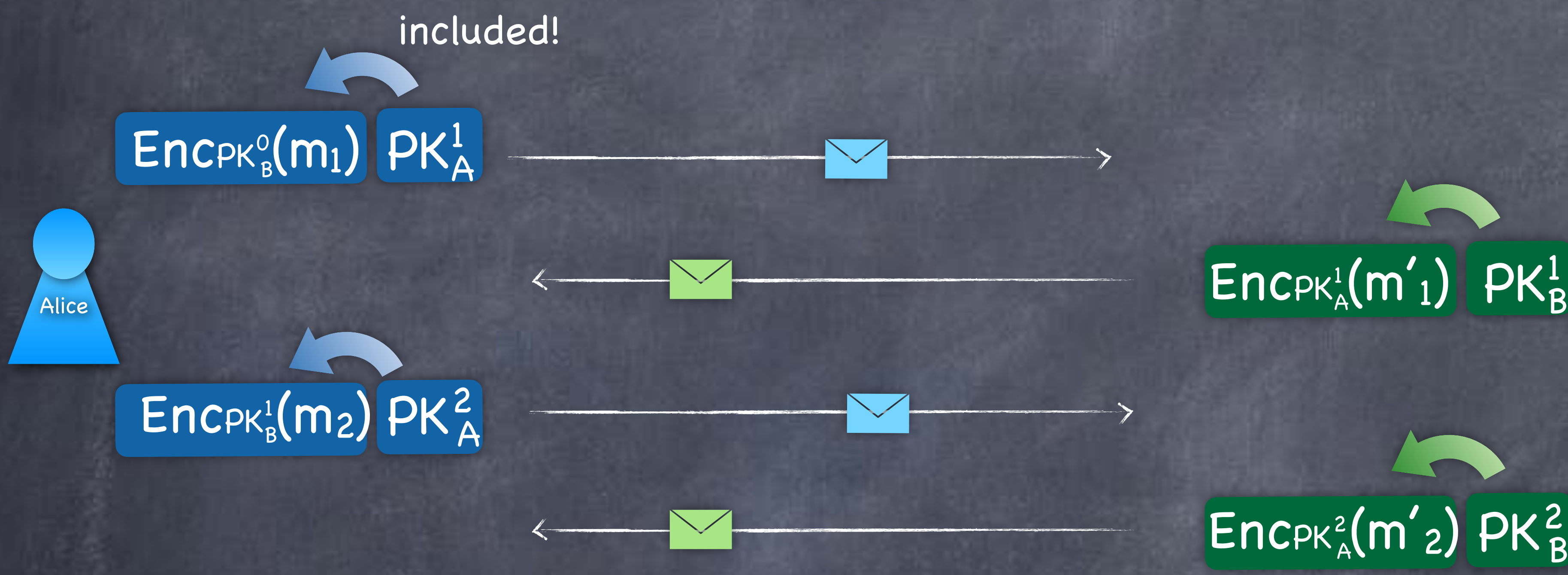
- PK_B^0 should be used only once (over all senders), so that SK_B^0 can be deleted after recovering m_0
  - E.g., Alice may download PK_B^0 from a list of PKs hosted by a server who deletes each PK on download

Synchronous Messaging
A first solution



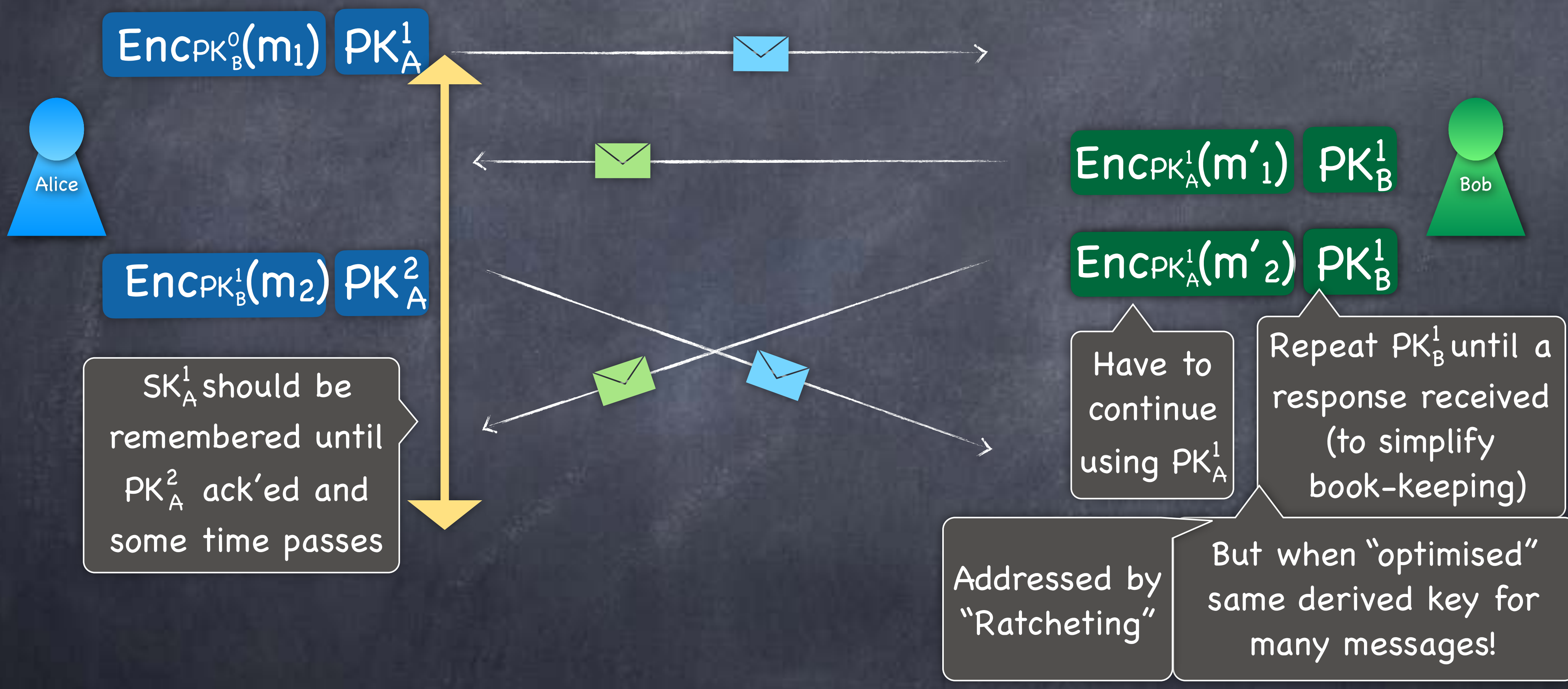
- (SK^i, PK^i) are generated just before sending PK^i and deleted right after using SK^i for decryption (window for compromising SK^i)
- At any point only one SK stored
- Drawback: Assumes strict alternation

An Optimization Suggestion



- Consider using El Gamal encryption: $PK^0_B = g^y$, ciphertext = $(g^x, m+K)$ where K derived from Y^x, and $PK^1_A = g^{x'}$
  - Use $x' = x$?
    - Can be OK when a symmetric key is derived using a random oracle, under stronger assumptions than DDH

Asynchronicity



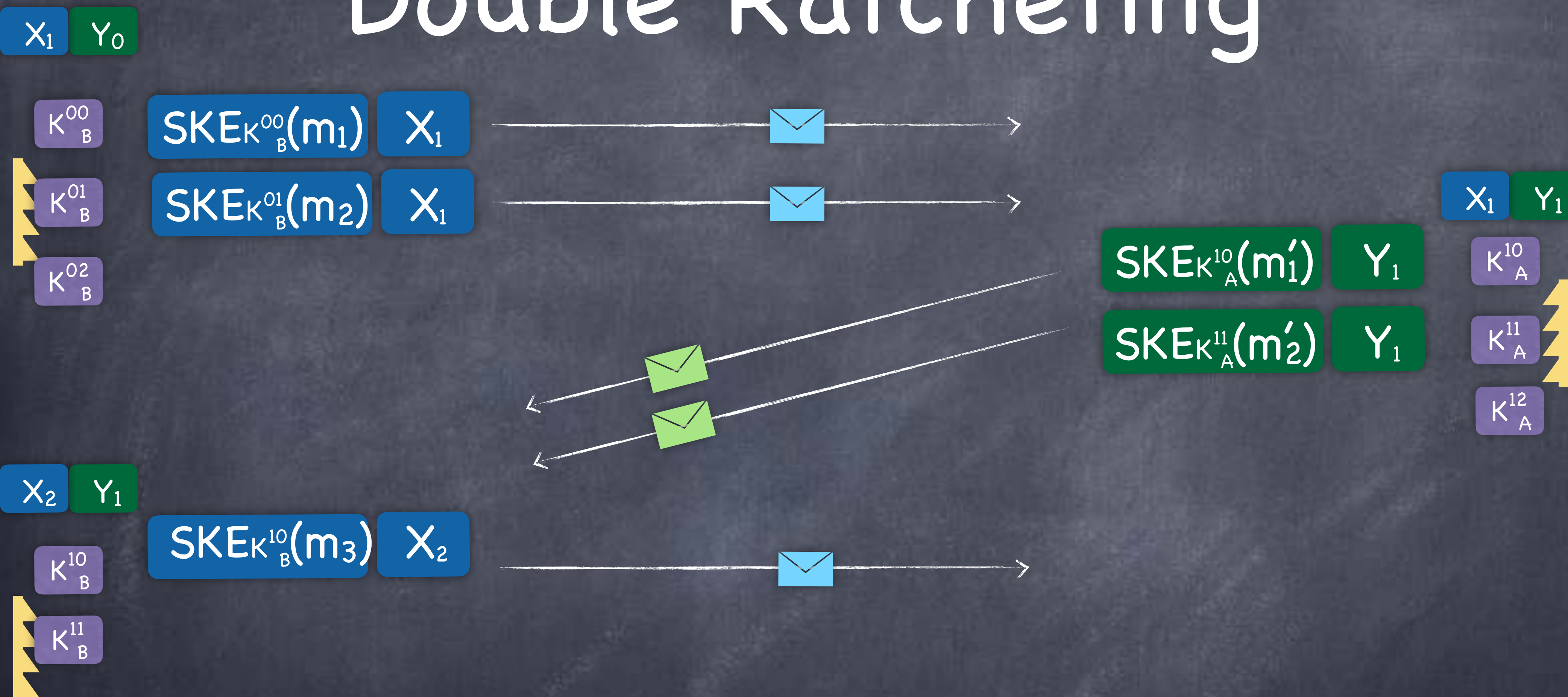
- Ideally, should be able to delete the decryption key right after using it for a single decryption

Ratcheting

- Suppose Alice and Bob have shared a symmetric key
- Want forward secrecy without need for synchronisation
  - E.g., both sending many messages, without receiving any
- Ratcheting

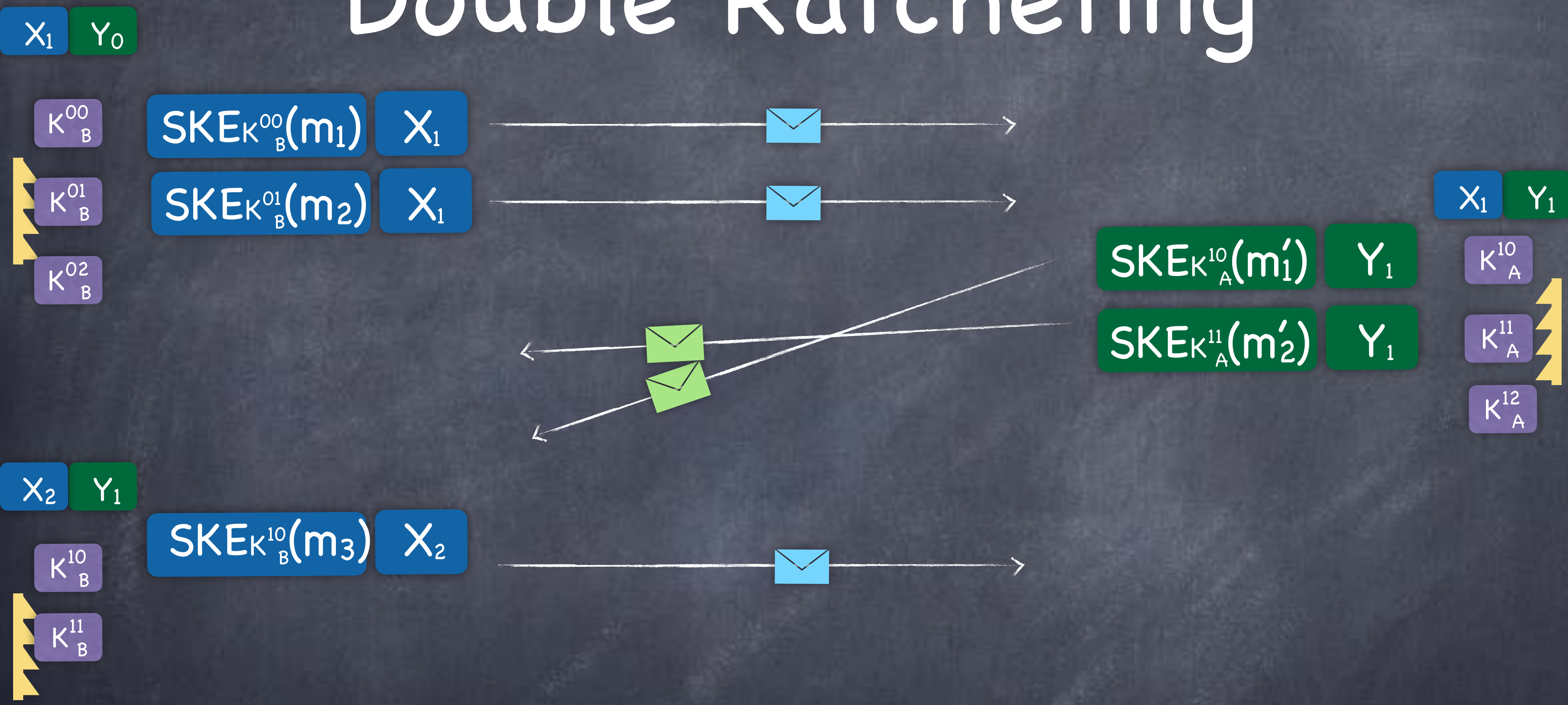
- $K_i \rightarrow K_{i+1}$ using a "forward-secure PRG" s.t. K_i remains pseudorandom even given K_{i+1}
- After using K_i for encryption/decryption, derive K_{i+1} and delete K_i
- Does not help with "future secrecy"

Double Ratcheting



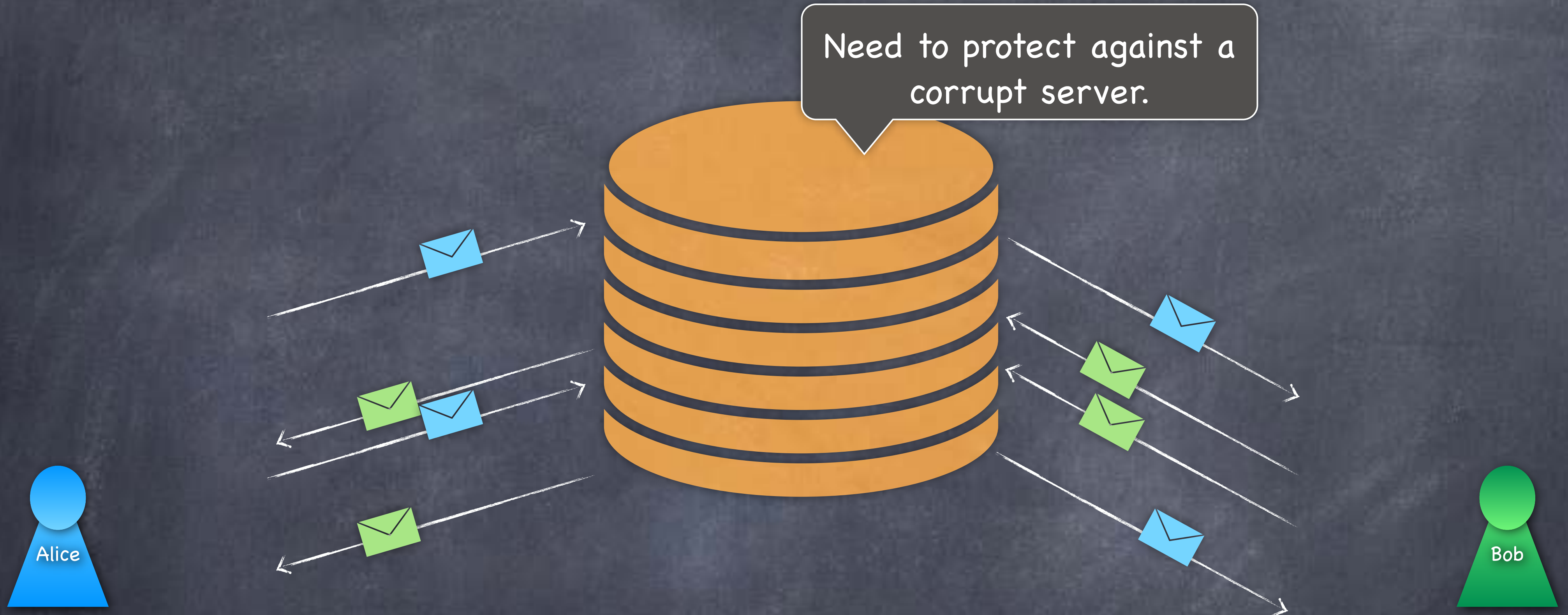
- Update public-keys for every received message, and do symmetric key ratcheting for messages in between
- Can delete an asymmetric secret key after the second symmetric key is derived from it (e.g., above x_1 deleted after K^{10}_A derived)

Double Ratcheting



- If messages received out of order, will need to retain symmetric keys that were ratcheted through

Messaging



- Identity key (i.e., signature verification key) should be obtained via (out-of-band) trusted setup
- Asymmetric key updates are MAC'ed using a key that was derived when the current asymmetric key was in force
- Symmetric keys are used for AEAD (e.g., using encrypt-then-MAC)

Establishing Identity

- Easy to ensure that conversation is with an entity who created a certain "identity key" (signature verification key)
- But in real life, want to ensure it is a certain person
- A malicious server can launch an adversary-in-the-middle attack
- Options (can use a combination):
  - Trust-On-First-Use: problematic assumption, e.g., if server always corrupt.
  - Trusted public-key servers which verify real-life identity! Require "transparency" to deter corrupt key servers.
  - Manual key dissemination, possibly via a web-of-trust
  - Share passwords and use PAKE
  - KeyBase: proves control of social media identities instead of "real-life" identity. Enough to trust at least one service.

Initial encryption PK will be signed with this

Deniability

- Suppose Alice and Bob chat with each other. Later, Bob turns over the transcript to a "judge"
- Can Alice claim that she is not responsible for the transcript?
  - Problem: If the messages are signed by Alice, she can't deny responsibility
    - Caveat: Alice's private key/device could have been stolen
- Alice should not sign the messages, but only MAC them
  - Bob also has the MAC key. So he could have faked the MACs himself

    To be convincing, app should expose this feature to Bob!
  - More complicated if the (encrypted) transcript between Alice and Bob is attested to by trusted intermediaries: Need deniable encryption