Messengers

Network Security

Messengers

- One of the primary ways of communication nowadays is through instant messengers.
- However, many protocols are proprietary, and security cannot easily be verified.
- We will look at two protocols:
  - iMessage (on a higher level, proprietary)
  - Signal Protocol (open protocol, used by Signal and WhatsApp)

Naive Attempt

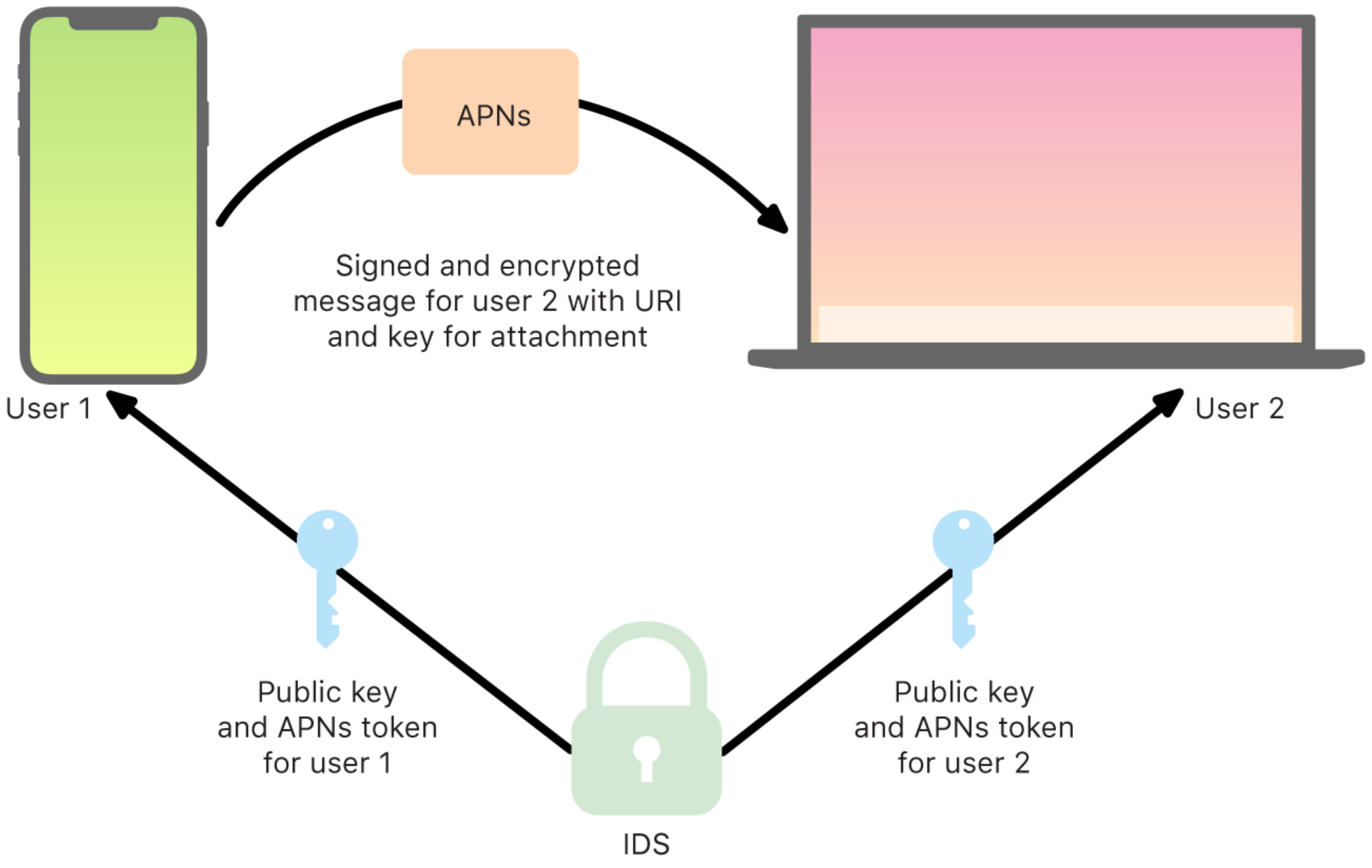
- Let’s design our own instant messaging service.
- What properties we might wish for?
  - **Confidentiality (end-to-end):** Only the communicating parties should be able to read messages.
  - **Data integrity:** It should not be possible to modify messages in transit.
  - **Authenticity:** Communicating parties should be sure of their identities.
  - **Forward Secrecy:** Ideally, if an attacker compromises a key, he should not be able to read past messages.

Naive Attempt

- Use **TLS** (provides all of the properties)
  - Relies on certificates, which makes it very cumbersome.
  - Even more problematic: Requires both parties to be online!
- Use **S/MIME**, **PGP**
  - S/MIME is too much focused on email format.
  - Key distribution of PGP seems unpractical for the masses.
  - Both do not provide forward secrecy.

iMessage

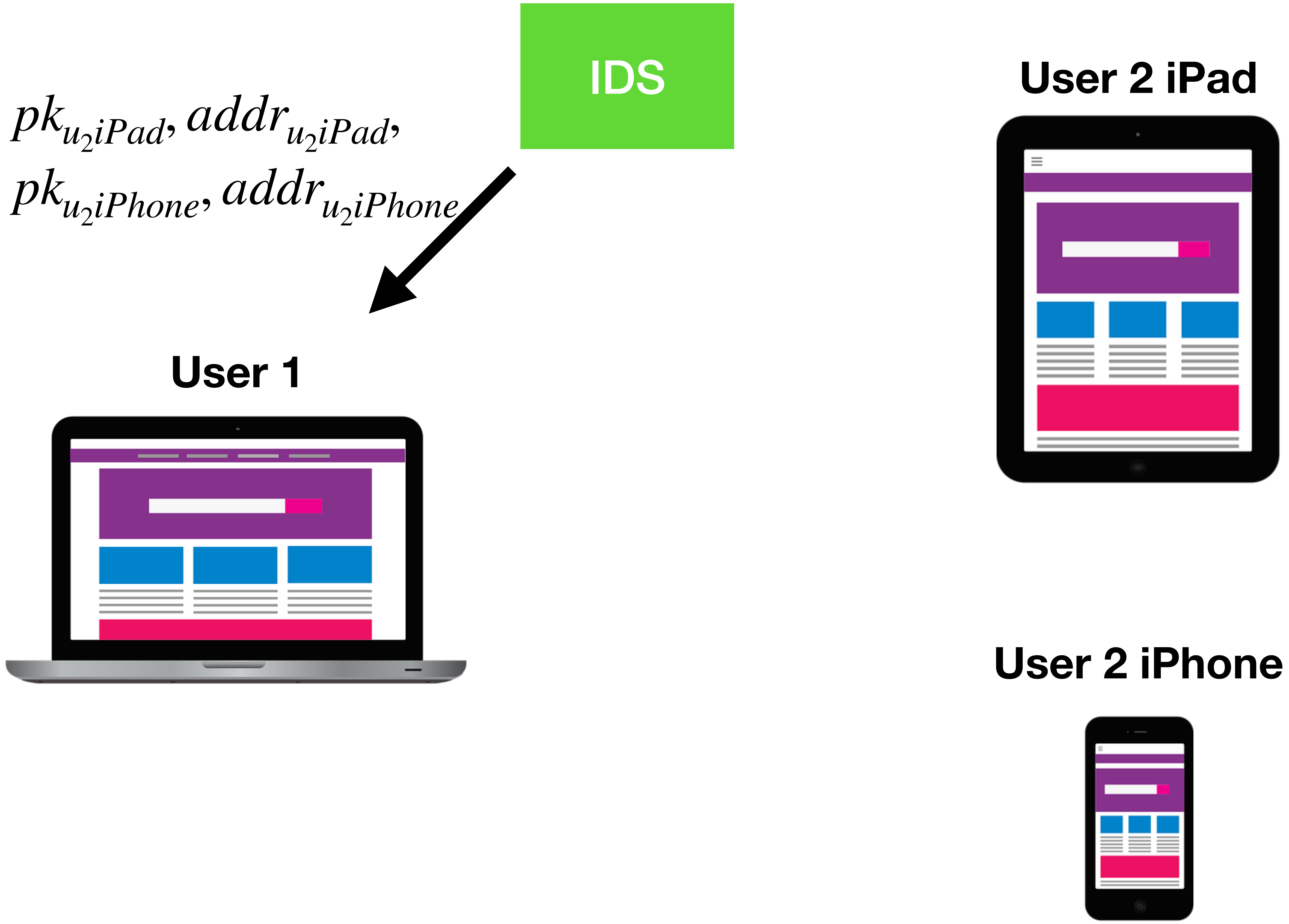
- Only works on Apple devices and requires an iCloud account.
- Apple runs the **Apple IDS** (Identity Service), which given a phone number or email address returns the corresponding user's public keys and APN addresses (one for each of their devices).
- APN is Apples Push Notification service and APN addresses are used to notify users, e.g., about incoming messages.



Source: *https://support.apple.com/en-gb/guide/security/sec70e68c949/web*

iMessage

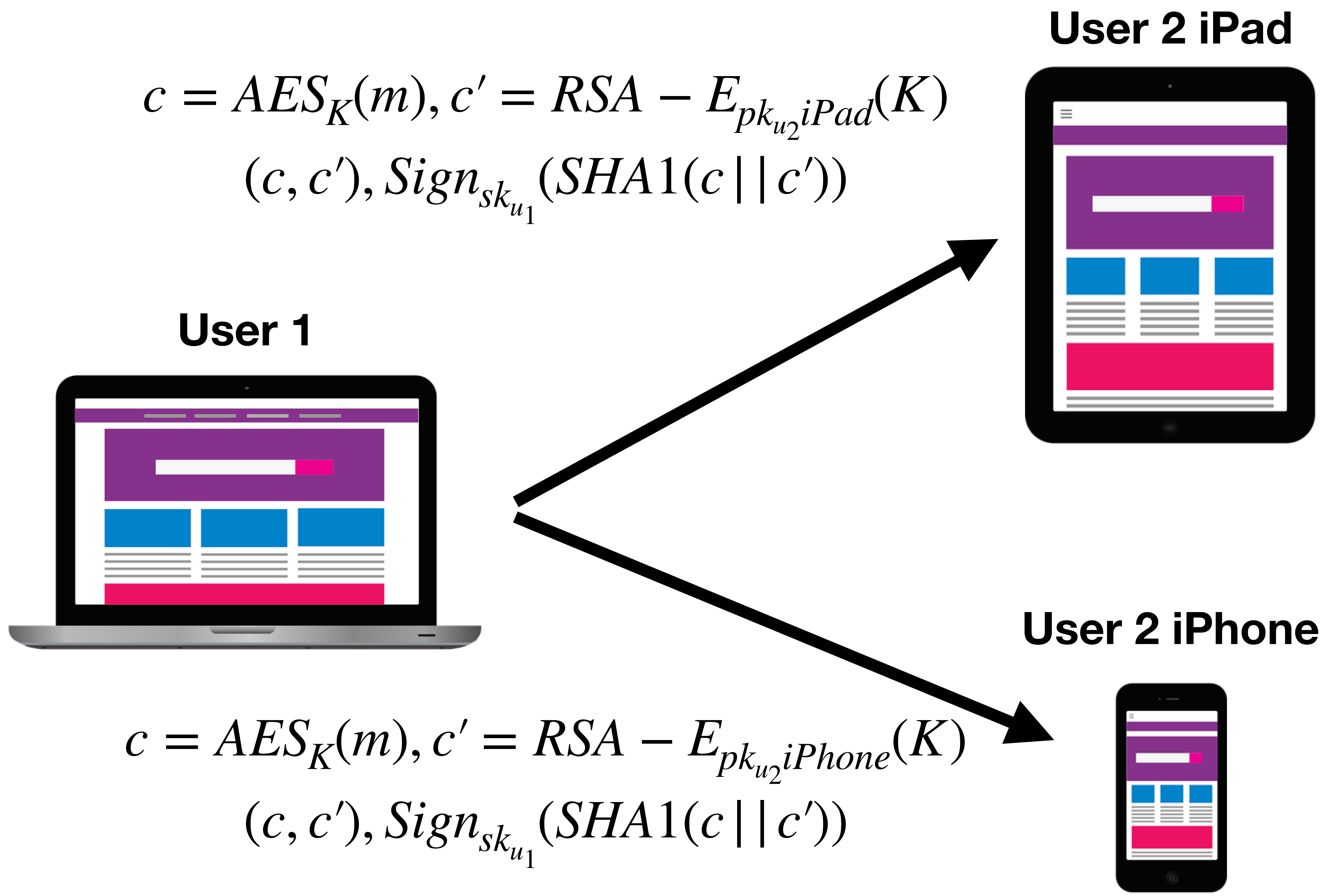
- The outgoing message is encrypted for each of the user's devices individually (as every device has its own public key).
- For each receiving device, the sender:
  - Generates a random 88-bit value,
  - Uses it as a HMAC-SHA256 key to construct a 40-bit value derived from the sender and receiver public key and the plaintext.
  - The concatenation of these is the key for a AES-128 CTR encryption of the message.



Source: https://support.apple.com/en-gb/guide/security/sec70e68c949/web

iMessage

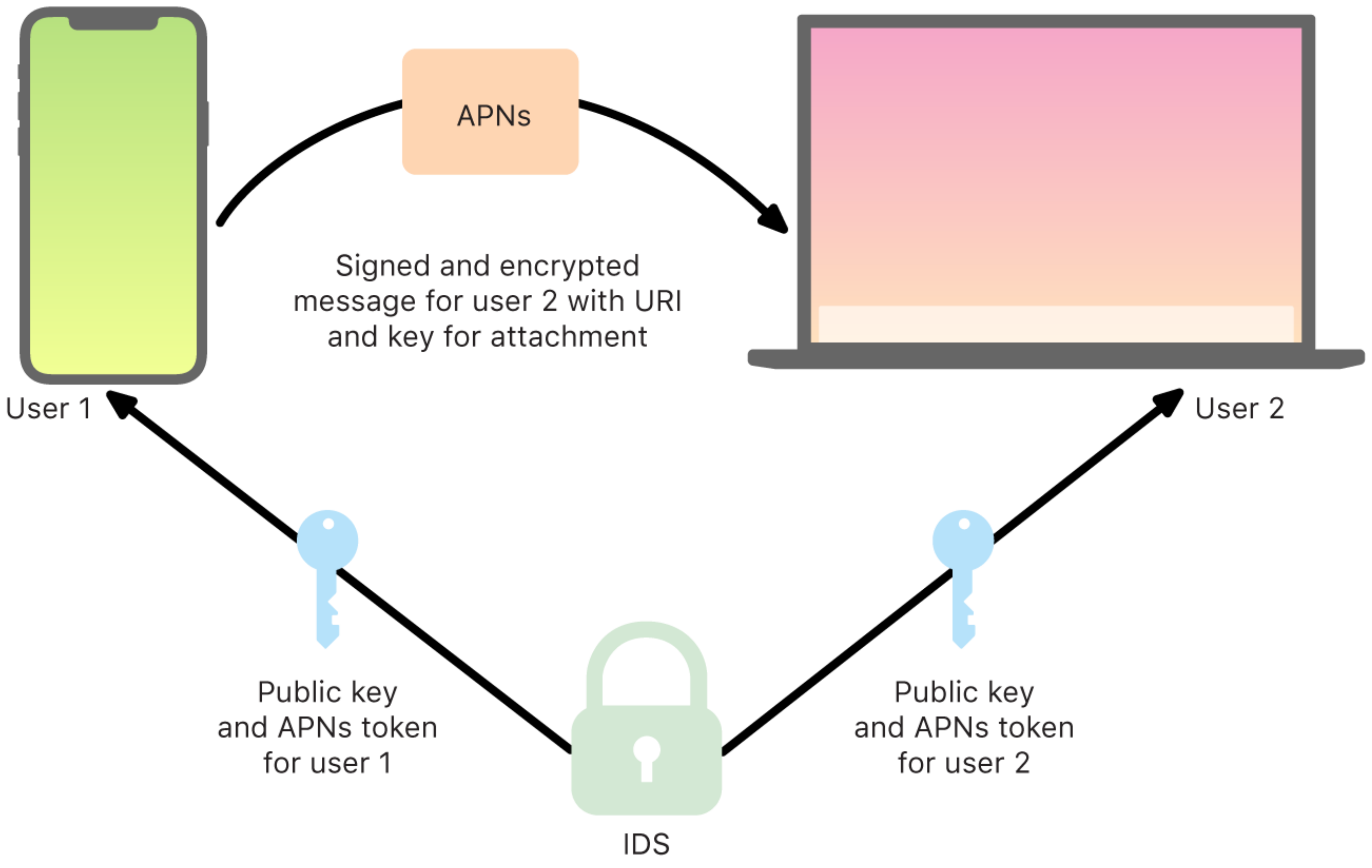
- The 40 bit part of the key is used by the receiver to validate the **integrity** of the decrypted plaintext.
- The full 128 bit encryption key is encrypted using a variant of RSA (RSA-OAEP) using the public key of the receiver.
    - Newer versions of Apple devices use the ECIES Hybrid Encryption Scheme instead of RSA.
- The combination of the encrypted message and the encrypted key is hashed with SHA-1 and signed by the sender using ECDSA (an elliptic curve digital signature algorithm).



Source: https://support.apple.com/en-gb/guide/security/sec70e68c949/web

iMessage

- The encrypted message, encrypted key, and digital signature are then forwarded via push notifications to the devices (using a forward-secret TLS channel).
- Metadata (timestamp, etc.) is not part of the encryption and only secured in transit via TLS.
- In group conversations, the whole process is repeated for every member of the group.



Source: https://support.apple.com/en-gb/guide/security/sec70e68c949/web

iMessage Issues

- iMessage does not provide forward-secrecy.
- Users have to rely on the Apple IDS, and Apple could theoretically add their own public keys to eavesdrop a communication.
- In earlier versions, replay of messages was possible.
- Many more discussed in a paper from 2016:
  e.g., one exploiting the compression
  https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_garman.pdf
  - Some issues have been fixed, but due to the closed source, it is hard to tell.

iMessage Properties

- iMessage aims at achieving: **confidentiality**, **data integrity**, **authenticity**, **accountability** (non-repudiation of origin)

- Due to the non-standard, proprietary design, vulnerabilities may be present and some properties might not hold.

Signal Protocol

- Similarly to iMessage, the Signal Protocol is used to provide **end-to-end encryption** for instant messaging conversations (but also voice calls).
- It provides forward secrecy and a property they call “future secrecy”.
- The Signal protocol can be divided into three stages:
  - The initial key exchange using **X3DH** (Extended Triple Diffie-Hellman). X3DH establishes a shared secret key and provides **mutual authentication** based on public keys.
  - An **asymmetric ratchet stage**.
  - A **symmetric ratchet stage**.
- The two ratchet stages form the **Double Ratchet** algorithm, which is used to derive new keys for every message so that earlier keys cannot be calculated from later ones.

Signal Protocol

- Basic setup:
  - Each party has a long-term private/public key pair (**identity key**).
  - Standard DH would not work as both communicating parties need to be online.
  - **Solution:** potential recipients pre-share batches of ephemeral public keys, which are buffered on a server. A sender can retrieve one of these and perform a key exchange protocol with the other party being offline.
  - Message keys depend on all previous exchanges between two parties and are derived using the ratcheting mechanism to form chains of keys.
  - Messages are then authenticated and encrypted using an encrypt-then-mac scheme with AES-256 in CBC mode and HMAC-SHA256.

Source: https://eprint.iacr.org/2016/1013.pdf

Signal Protocol

- Stages of the protocol:
  - **Registration:** "At installation (and periodically afterwards), both Alice and Bob independently register their identity with a key distribution server and upload some long-term, medium-term, and ephemeral public keys."
  - **Session Setup:** "Alice requests and receives a set of Bob's public keys from the key distribution server and use them to setup a long-lived messaging session and establish initial symmetric encryption keys. This is called the TripleDH handshake or X3DH."

Source: https://eprint.iacr.org/2016/1013.pdf

Signal Protocol

- Stages of the protocol:
  - **Synchronous messaging (a.k.a. asymmetric ratchet updates):** "When Alice wants to send a message to Bob (or vice versa) and **has just received a message from Bob**, she exchanges Diffie–Hellman values with Bob, generating new shared secrets and uses them to begin new chains of message keys. Each DH operation is a stage of the "asymmetric ratchet" (and strictly occurs in a ping-pong fashion)."

Source: https://eprint.iacr.org/2016/1013.pdf

Signal Protocol

- Stages of the protocol:
  - **Asynchronous messaging (a.k.a. symmetric ratchet):**
    "When Alice wants to send a message to Bob (or vice versa) but **has not received a message from Bob since her last sent message to Bob**, she derives a new symmetric encryption key from her previous state using a pseudo-random function (PRF). Each PRF application is a stage of the "symmetric ratchet"."

Source: https://eprint.iacr.org/2016/1013.pdf

Stages of the Protocol

Responder–initiator symmetric ratchets
(initiator's receiving chain, responder's sending chain)

not present

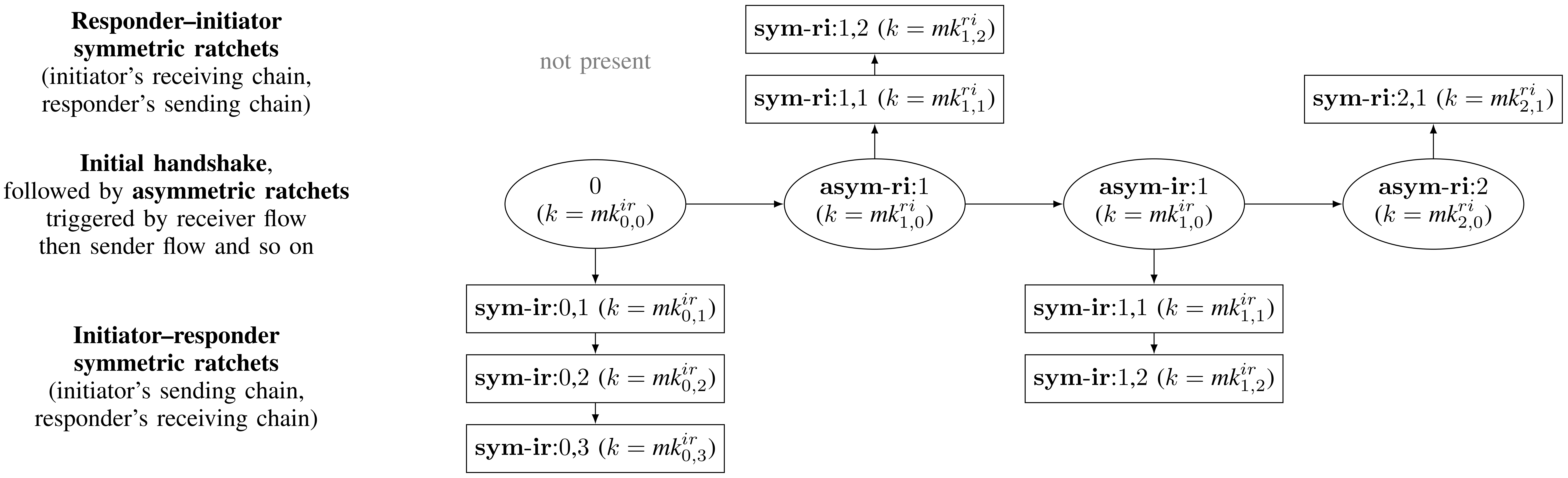
Initial handshake, followed by **asymmetric ratchets** triggered by receiver flow then sender flow and so on

0
$(k = mk_{0,0}^{ir})$

Initiator–responder symmetric ratchets
(initiator's sending chain, responder's receiving chain)

Source: https://eprint.iacr.org/2016/1013.pdf

Stages of the Protocol



Source: https://eprint.iacr.org/2016/1013.pdf

Signal Protocol Properties

- The Signal Protocol achieves: **confidentiality**, **data integrity**, **authenticity**
- Additionally, it provides: **forward secrecy** and **identity hiding**, i.e., a passive adversary should not be able to learn the identities communicating in a session
- Some of these properties have been **formally verified**!
- However, it relies on out-of-band verification of identities

Summary

- Challenges
  - In contrast to many other networking scenarios, the two parties might not be online at the same time.
  - Key/certificate management should be easy for everyone!
- iMessage
  - Keys are distributed via an Apple provided service.
  - Generates one-time keys for messages, but does not provide forward secrecy.
- Signal Protocol
  - Combines several types of keys (long-term, medium-term, one-time) via X3DH, which even works with the other party being offline (by pre-publishing values).
  - Uses two forms of KDF chains to generate one-time keys for messages while providing forward secrecy.
  - Limitation: Identities have to be verified out-of-band.