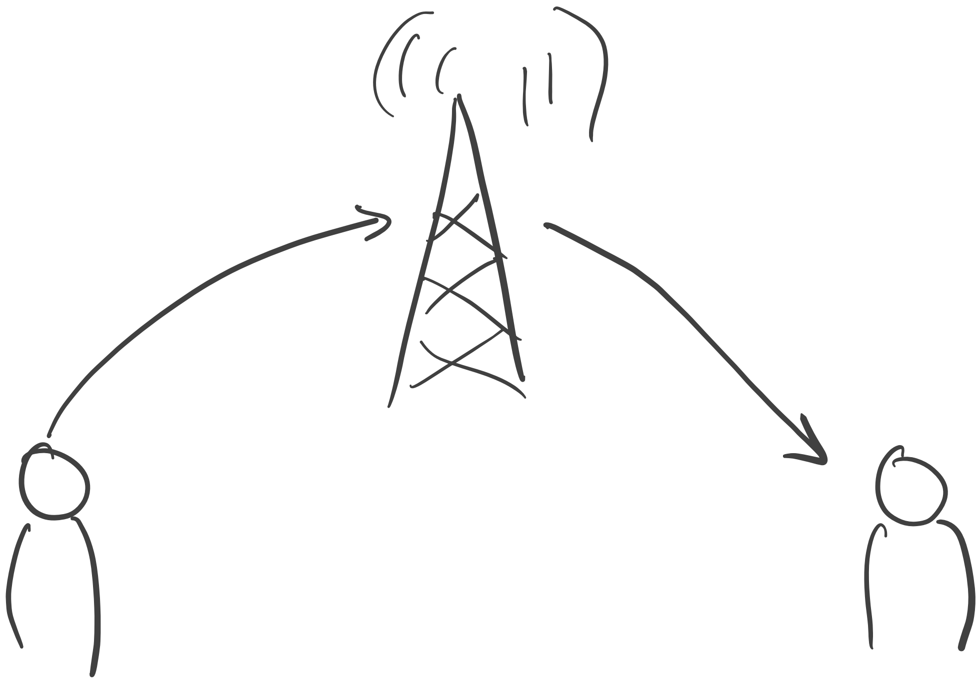
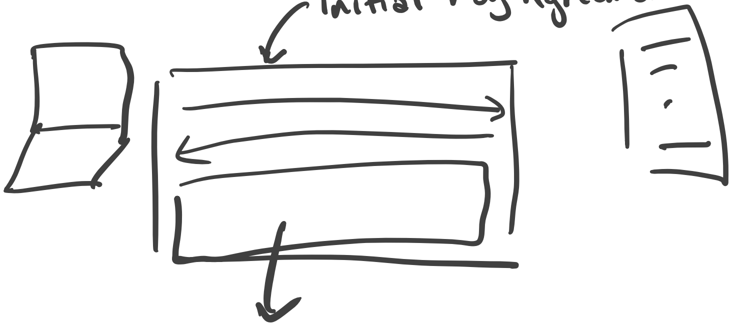
End To End Encryption

SMS / MMS

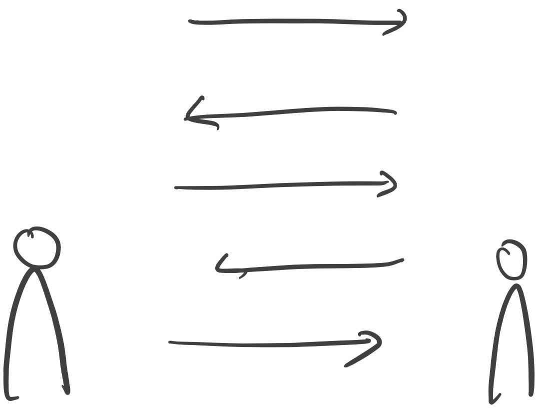


Initial key Agreement



encrypted conversation

- short communication
- both parties online
- recovery is not a concern



- long communication
- asynchronous

corruption model

full network control

corruption of a party's state

Encrypted Email

k

assymetric
$enc_{pk_B}(k) \to c_k$

symmetric
$enc_k(m) \to c_m$

Public Information

pk_B , pk_A

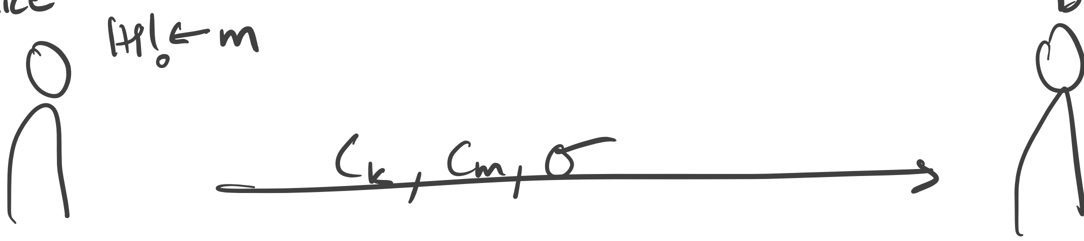
keys are longterm

loses all security on break in

Alice

"Hi!" $\leftarrow m$

Bob



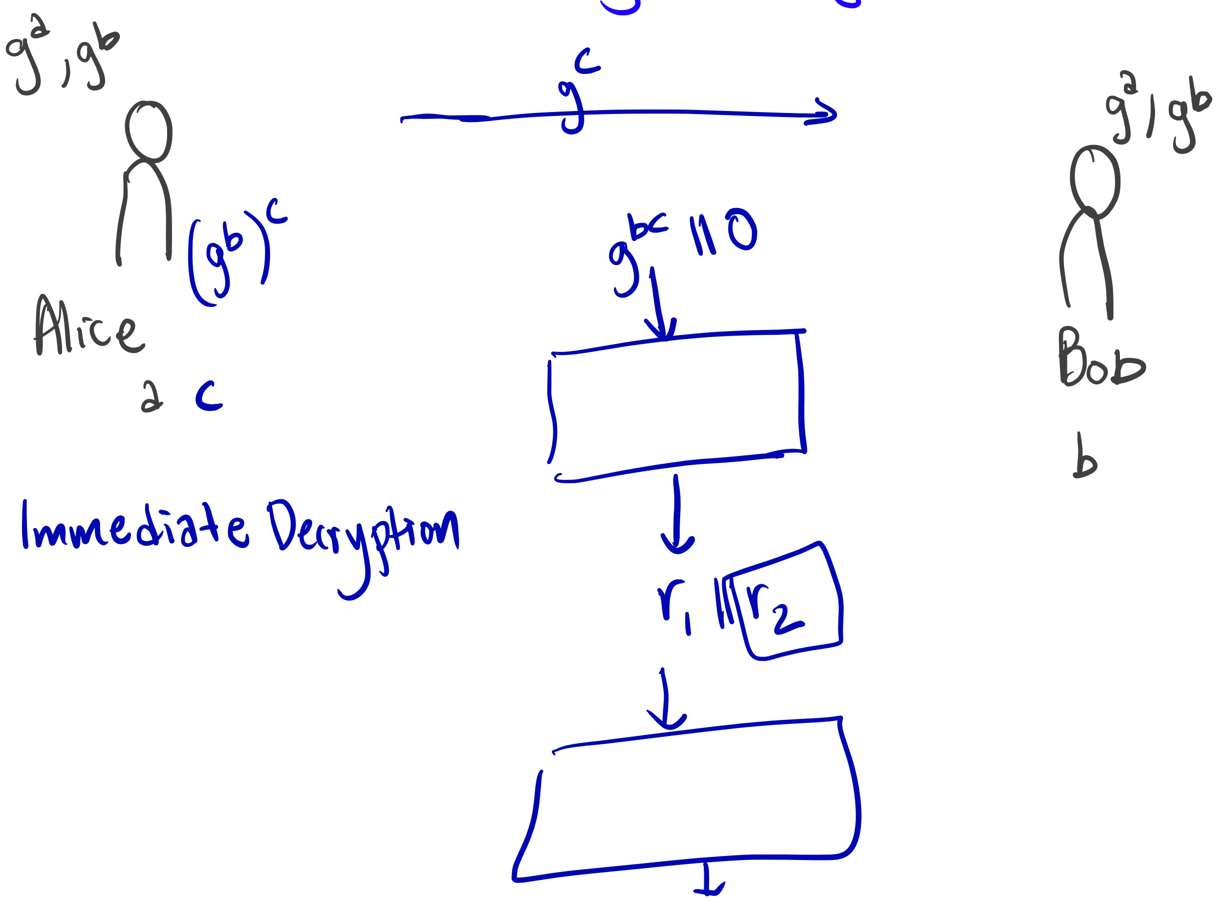
forward secrecy

makes deletion meaningful

post-compromise security

recovery after state compromise

Force periodic key exchange



Immediate Decryption

This page was added after lecture. These are useful concrete descriptions of the properties.

Post-Compromise Security. The notion of *post-compromise security*, also known as *backward secrecy*, *future secrecy*, and *channel healing*, informally says that a party A has a security guarantee about communication with another party B, even after B's secrets have been compromised [6]. Note that post-compromise security does not say that A will have a security guarantee *immediately* after the compromise; in actual protocols that achieve post-compromise security, A has a security guarantee once some kind of *healing* has occurred after the compromise of B's secrets.

Even though this might seem unintuitive at first, there are in fact protocols that achieve forms of post-compromise security. As with forward secrecy, *how* post-compromise security is realized in practice depends on the precise security guarantees a protocol aims to achieve, and in particular, on the actual secrets that are compromised. Two typical cases are *session-key compromise* and *full local-state compromise.*

In the case of session-key compromise, only ephemeral cryptographic material is leaked to the attacker. Here, post-compromise security can be provided by a *key-evolving scheme*, i.e., a mechanism that computes the session key using some secret information from the previous session. For instance, given a key-derivation function *KDF* (i.e., a one-way function that derives one or more secret keys from its input), the i-th session key sk_i could be computed from the previous session key sk_{i-1} and a token t_i. Here, the token t_i could, for example, be a shared secret established by the two honest parties via a Diffie-Hellman key exchange. If the session key is evolved in this way, the attacker cannot compute a future session key with only the knowledge of the current session key.

Forward Secrecy. The notion of *forward secrecy*, also known as *perfect forward secrecy*, informally says that ciphertexts that were ~~sent or~~ received *prior* to the compromise of a party remain secure after the compromise [3, 15]. For instance, imagine A and B are exchanging encrypted messages until some timepoint t at which an attacker compromises A by stealing A's secret keys—forward secrecy guarantees that the attacker will not be able to decrypt any of the messages ~~exchanged between A and B *before* timepoint *t*.~~ The exact mechanism used to provide forward secrecy depends on the level of granularity at which a protocol aims to achieve forward secrecy. Typical levels of granularity are *per message* or *per session.*

sent by A or received by A before time t.

instant messaging

TLS

In the per-message case, forward secrecy can be achieved by evolving the encryption keys for every sent message. This is commonly achieved by deriving new keys from old keys via a one-way function. If keys are regularly updated, messages that were encrypted with previous keys stay secure even if the current key is compromised. ~~Note that other mechanisms that achieve forward secrecy have been proposed in the literature, such as *puncturable encryption* [8, 14, 16] and *time-based methods* [9].~~

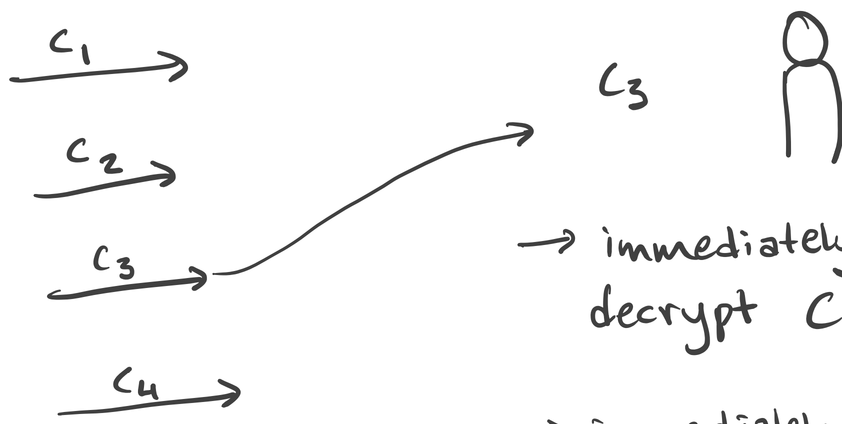
In the per-session case, the communicating partners exchange *ephemeral* session keys in typical protocols. Using public-key cryptography, a fresh session key is generated independently of the previous session key, and thus cannot be computed given future session keys. ~~To reduce the expensive public-key operations, parties can derive an initial session key and, similar to the per-message case, *evolve* it throughout the conversation, e.g., by using a key-~~ derivation function.

Forward Secrecy

Post Compromise Security

Immediate Decryption

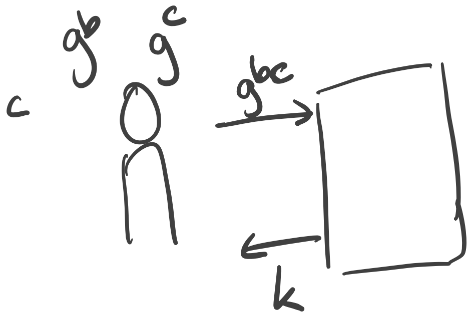
Usability under Asynchrony



→ immediately decrypt c_3

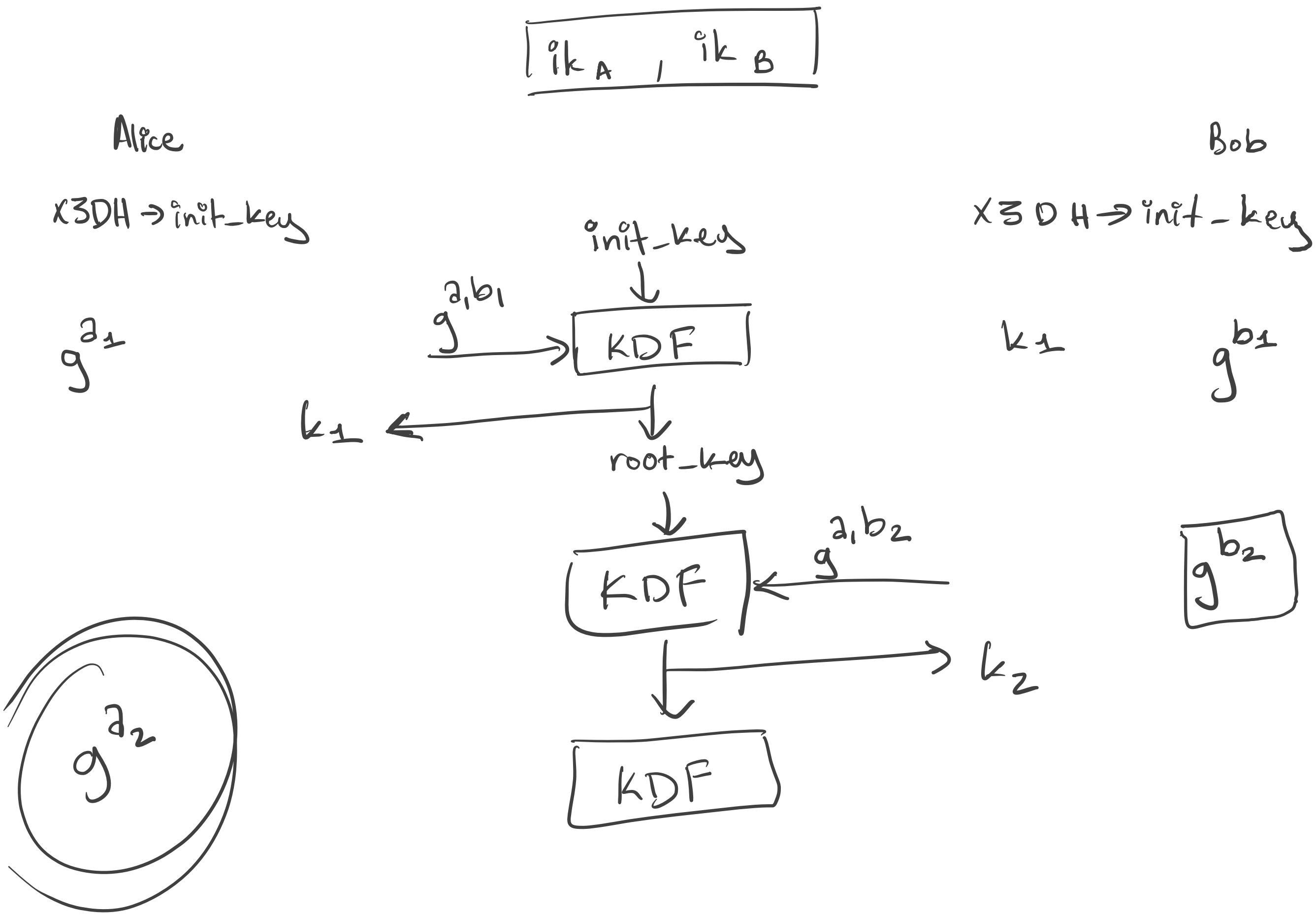
→ immediately know <u>where</u> in the conversation c_3 is

Key Derivation Function

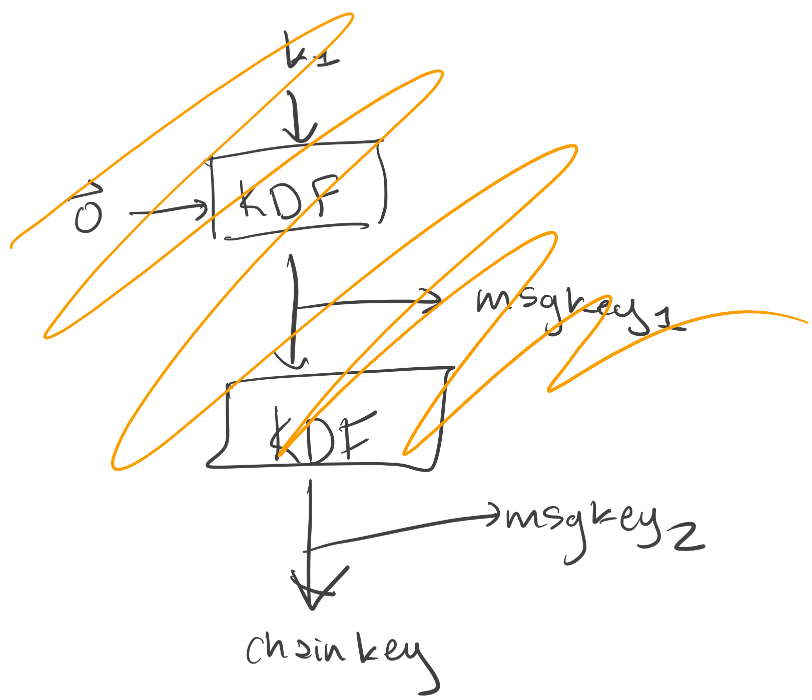


$$AEAD_k(m, g^c)$$

Authenticated Encryption w/ Associated Data   AEAD



ik A , ik B
Alice
Bob
X3DH → init_key
X3DH → init_key
init_key
g^{a1b1}
KDF
k1
g^{a1}
k1
g^{b1}
root_key
g^{a1b2}
KDF
g^{b2}
k2
g^{a2}
KDF



k1
0
KDF
msgkey1
KDF
msgkey2
chain key