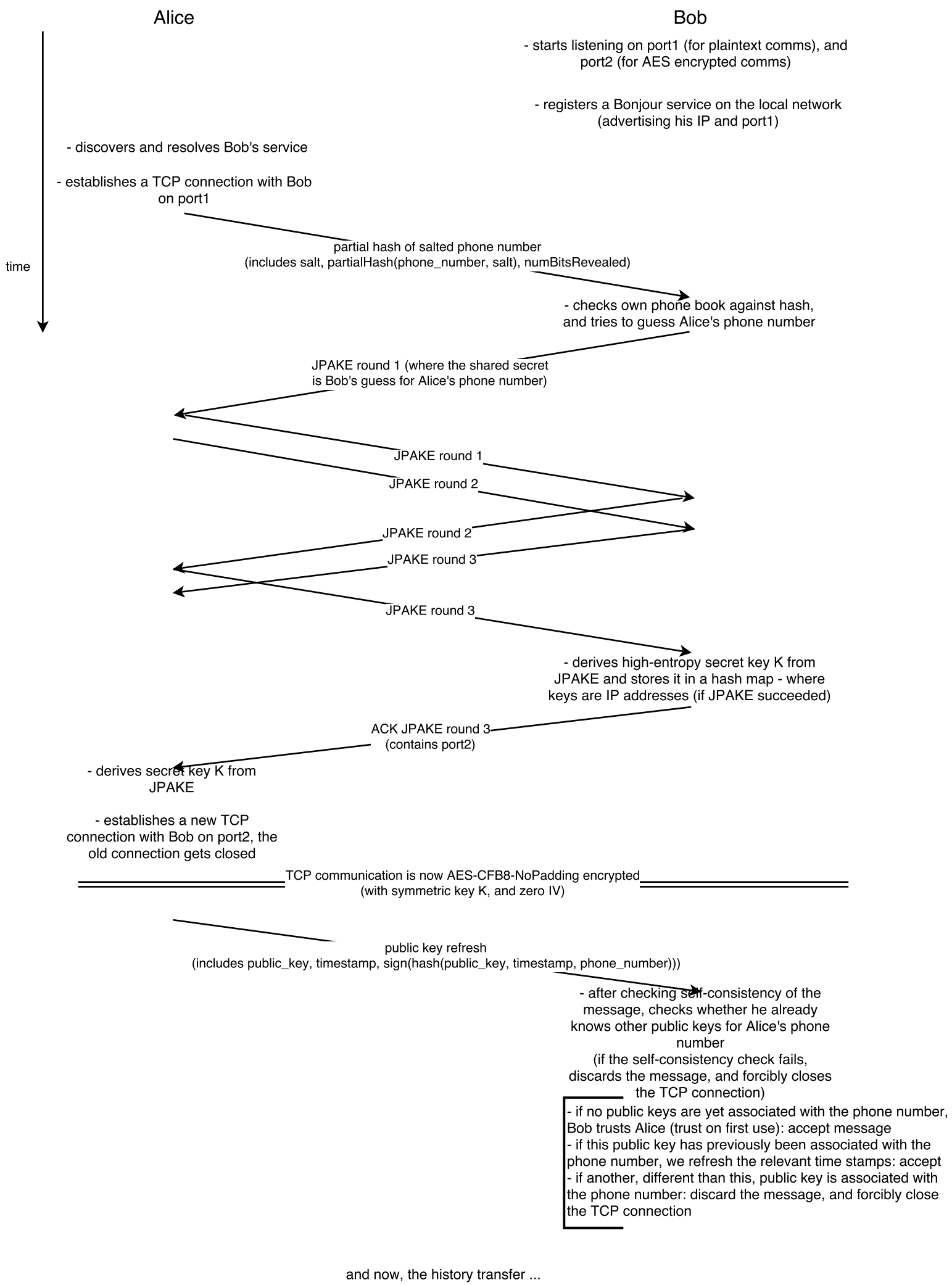
Alice
Bob
- starts listening on port1 (for plaintext comms), and port2 (for AES encrypted comms)
- registers a Bonjour service on the local network (advertising his IP and port1)
- discovers and resolves Bob's service
- establishes a TCP connection with Bob on port1
time
partial hash of salted phone number
(includes salt, partialHash(phone_number, salt), numBitsRevealed)
- checks own phone book against hash, and tries to guess Alice's phone number
JPAKE round 1 (where the shared secret is Bob's guess for Alice's phone number)
JPAKE round 1
JPAKE round 2
JPAKE round 2
JPAKE round 3
JPAKE round 3
- derives high-entropy secret key K from JPAKE and stores it in a hash map - where keys are IP addresses (if JPAKE succeeded)
ACK JPAKE round 3
(contains port2)
- derives secret key K from JPAKE
- establishes a new TCP connection with Bob on port2, the old connection gets closed
TCP communication is now AES-CFB8-NoPadding encrypted
(with symmetric key K, and zero IV)
public key refresh
(includes public_key, timestamp, sign(hash(public_key, timestamp, phone_number)))
- after checking self-consistency of the message, checks whether he already knows other public keys for Alice's phone number
(if the self-consistency check fails, discards the message, and forcibly closes the TCP connection)
- if no public keys are yet associated with the phone number, Bob trusts Alice (trust on first use): accept message
- if this public key has previously been associated with the phone number, we refresh the relevant time stamps: accept
- if another, different than this, public key is associated with the phone number: discard the message, and forcibly close the TCP connection
and now, the history transfer ...