CCNx Key Exchange

IETF 94 - Yokohama - ICNRG

Christopher A. Wood

November 5, 2015

Motivation and Goals

Motivation

- We need a way to establish session keys between consumers and producers that makes use of CCN properties
    - (D)TLS, QUIC, etc. are a good start

Requirements

- Session keys must be forward secure
    - Compromising long-term secrets does not put session keys at risk
- At most 2 RTTs to establish a session key, with the possibility for session resumption in 0 RTT
- Allow extensions for client authentication in addition to server authentication

TLS and QUIC Overview

- Support 0-, 1-, and 2-RTT forward secure key derivation
    - Long-term public key shares enable faster handshakes
- Different keys are used to encrypt (and MAC) different parts of the protocol
    - A short-term ephemeral key is used for exchanging random key shares to derive a master key
- Server is authenticated to the client
- Prevents address spoofing (via SYN cookies) and replay attacks (via QUIC Source Address Tokens and TLS nonce)

CCNx Key Exchange (CCNx-KE)

Assumptions

- Consumers know the prefix of the target producer, e.g., /prefix/
- Consumers possess the appropriate trust anchors to authenticate the server
- … that's it

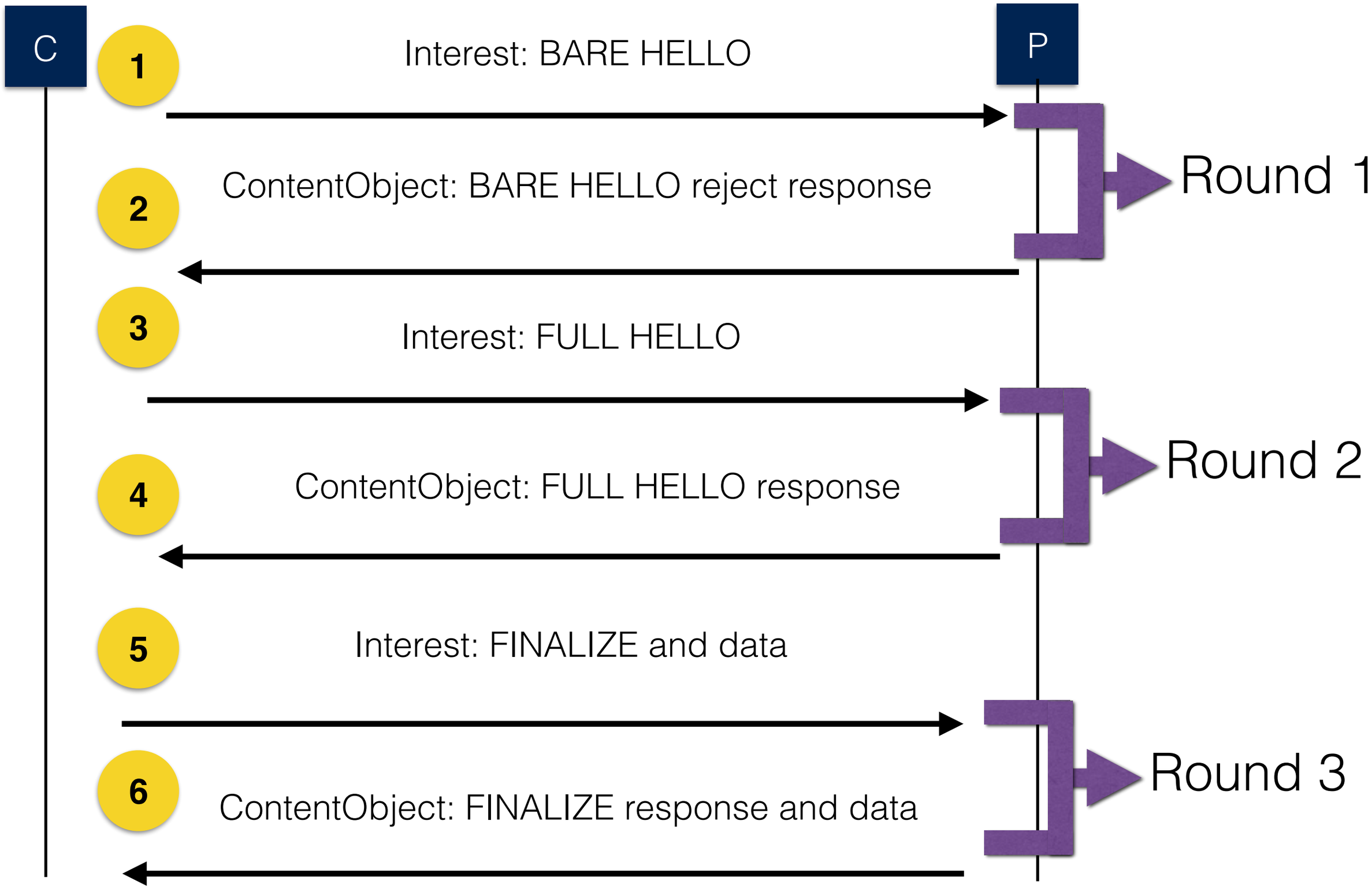
Protocol Overview

- Round 1: Obtain the server config (if not available or it has expired)

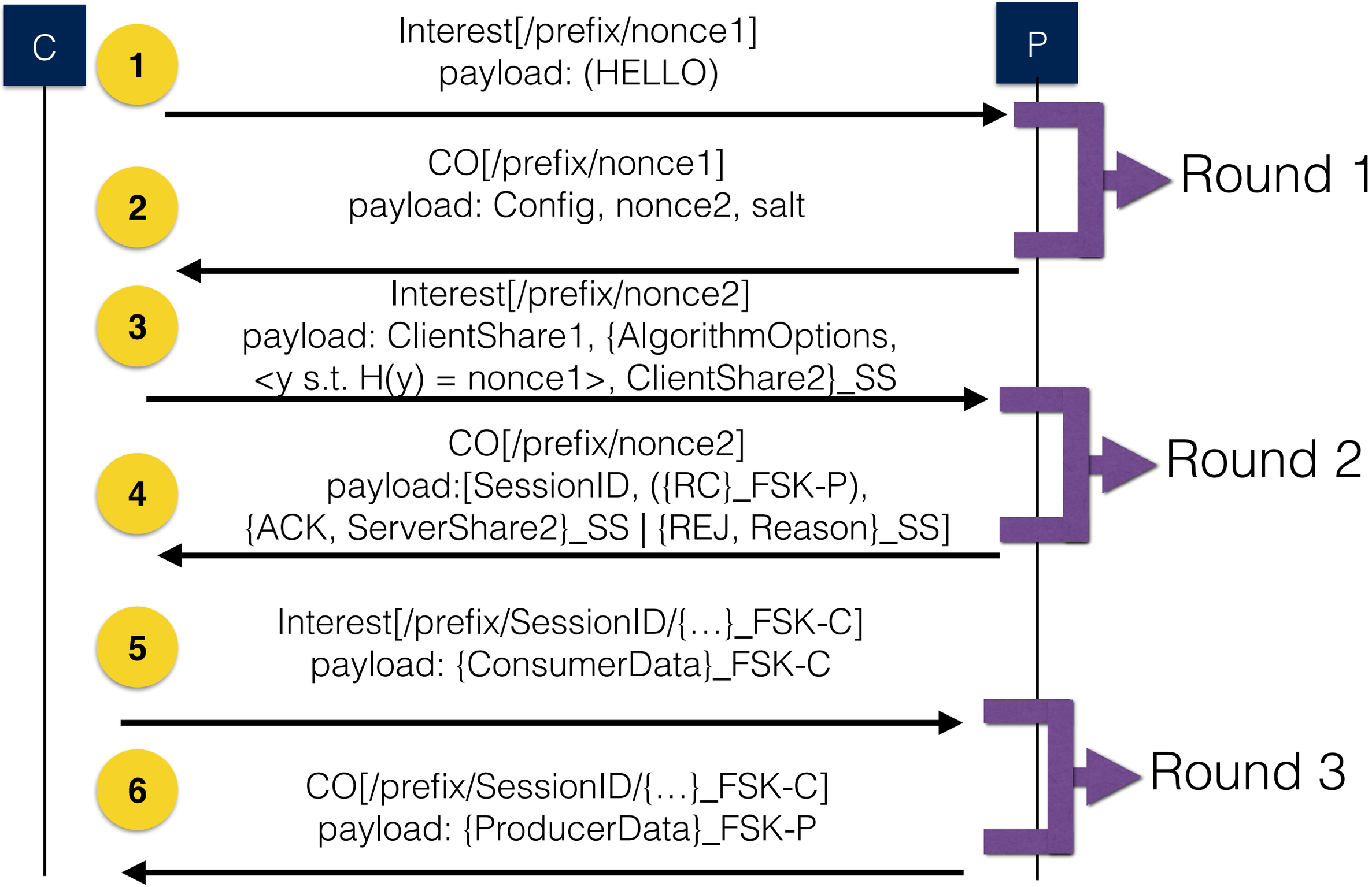
- Round 2: FULL HELLO handshake and establish ephemeral keys

- Round 3: Final exchange to derive forward-secure secrets for all subsequent communication

Sketch of the Full Protocol



Sketch of the Full Protocol



SS Derivation

$$\mathsf{SS} = \mathsf{HKDF}(\mathsf{Salt}, \mathsf{IKM})$$

$$\mathsf{Salt} = \texttt{CSALT1}||\texttt{PSALT1}||\text{“ss generation”}$$

$$\mathsf{IKM} = \text{32-byte key-exchange output}$$

FSK-C/P Derivation

Second key exchange uses the ServerShare2 and ClientShare2 inputs

$$\mathsf{FSK} = \mathsf{HKDF}(\mathsf{Salt}, \mathsf{IKM})$$

$$\mathsf{Salt} = \texttt{CSALT2}||\texttt{PSALT2}||\text{“fsk generation”}$$

$$\mathsf{IKM} = \text{Second 32-byte key-exchange output}$$

FSP-C/P and IVs are pumped from FSK in the following order:

1. FSK-C
2. FSK-P
3. FSK-CIV (client IV)
4. FSK-PIV (producer IV)

SessionID and RC Properties

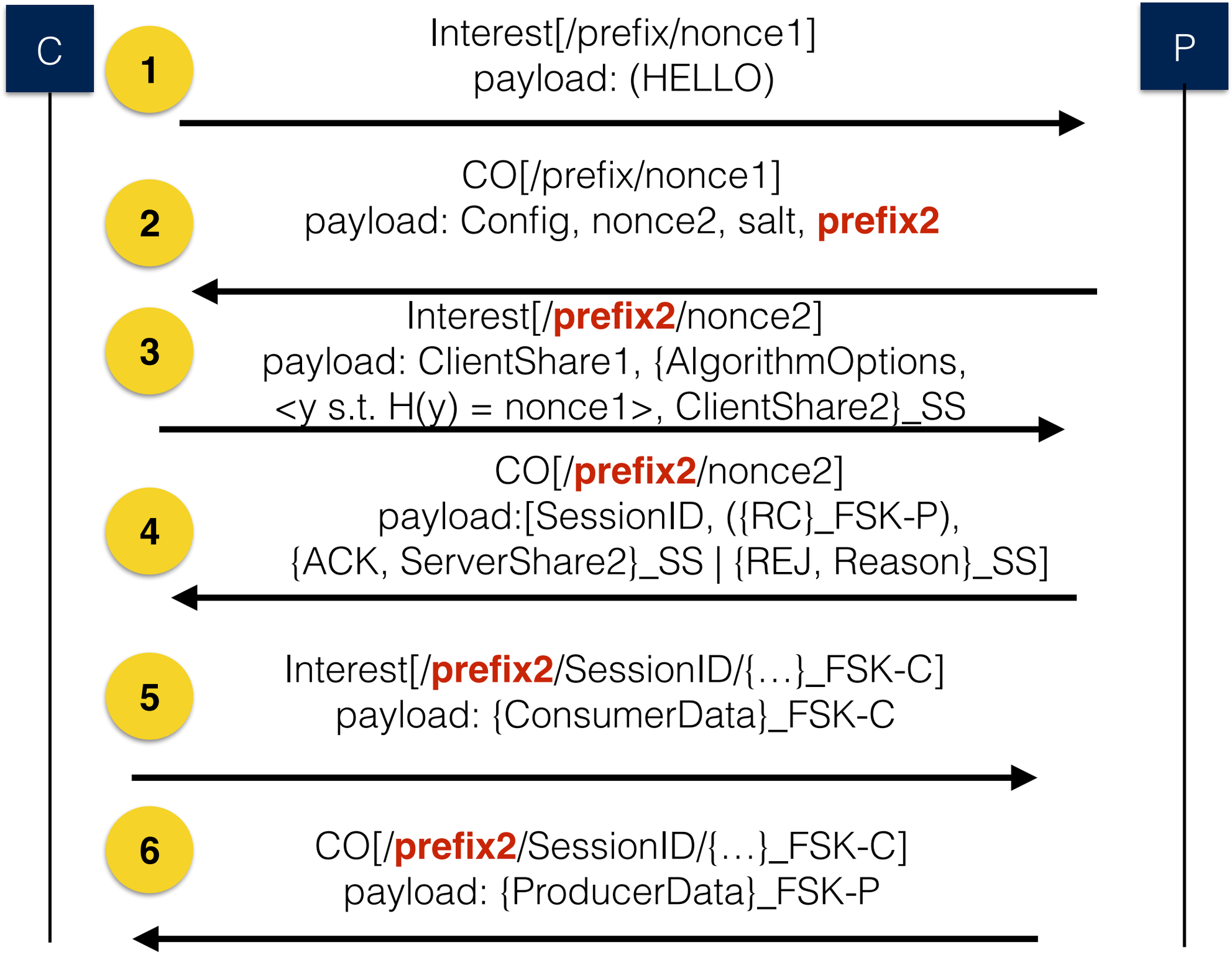
SessionID

- Used to uniquely identifies a single session
- … a random string/number suffices

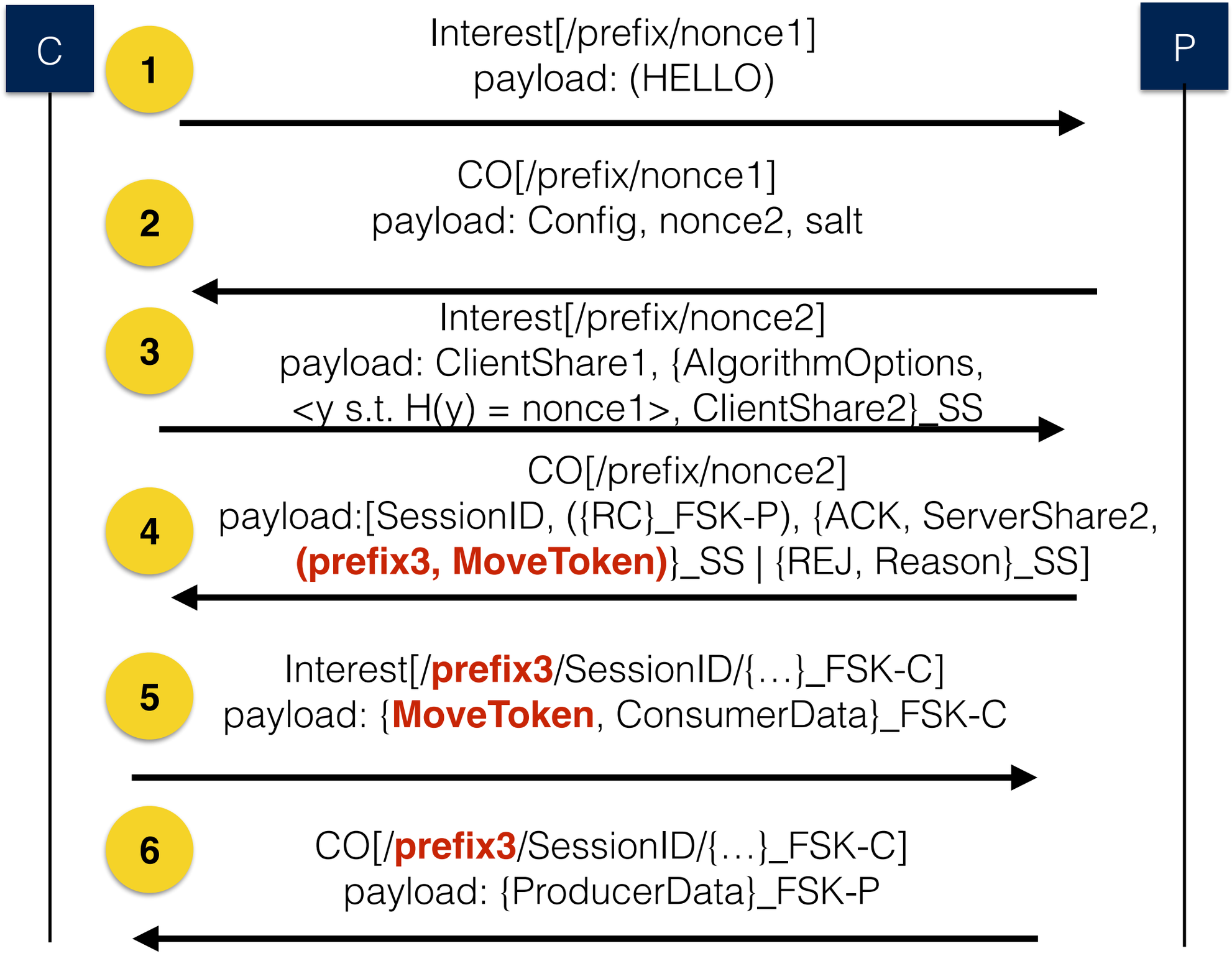
RC

- Used to recover SS and FSK for a given session

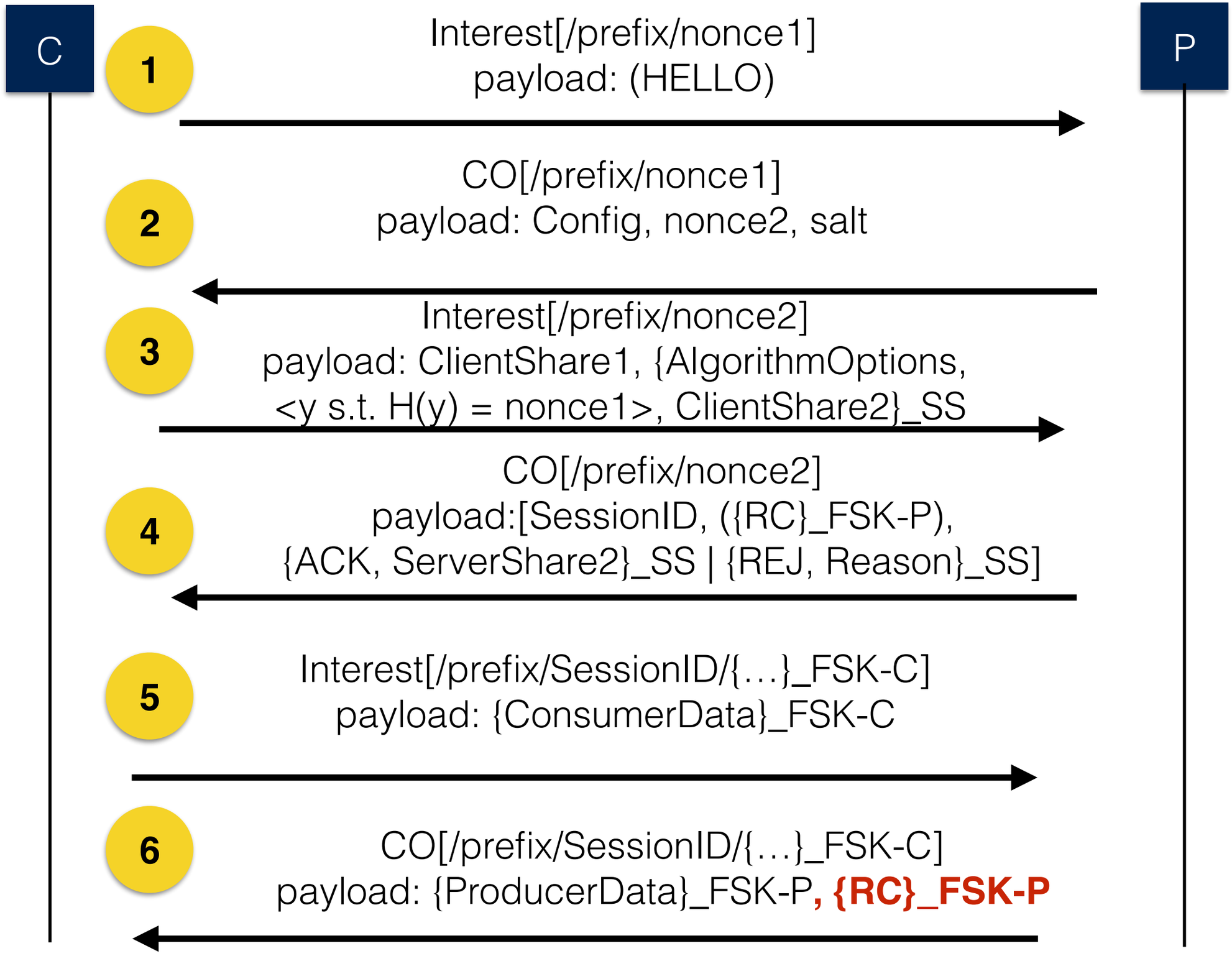
Option #1: HELLO prefix redirection



Option #2: Final prefix redirection



Option #3: Resumption Cookie Echo



CCNx-KE Properties

- Minimal deviation from TLS and QUIC.
- Forward-secure session keys derived similar to TLS and QUIC.
- Server-to-client authentication.
    - Client-to-server authentication is future work.
- Clients are securely bound to the protocol execution (via the hash-based tokens).
- Session state can be securely migrated from the producer to a trusted party.

Backup

SessionID*

Structure: Generated as encryption of the hash digest of a sever secret, FSK, and optional prefix (e.g., Prefix3). Encryption happens with a long-term, private key held by the server.

$$\mathsf{SessionID} = \mathsf{Enc}(k, \mathsf{H}(secret||\mathsf{FSK}||\,(\mathsf{Prefix3}|\,\perp)))$$

Usage: Append to service prefix (in the name) to indicate what key is used for encrypting payload data

*** This is only one way to create the SessionID

Resumption Cookie (RC)*

Structure: Encryption of H(server secret), SS, FSK, and the (Prefix3,MoveToken) tuple (if provided), with a producer secret key that is also known to the service operating under Prefix3 (if provided)

$$\mathsf{RC} = \mathsf{Enc}(k, \mathsf{SS}||\mathsf{FSK}||\left((\mathsf{Prefix3}||\mathsf{MoveToken})|\perp\right))$$

Usage: The SessionID and RC are needed to resume a session (i.e., recompute SessionID and check for equality):

$$(\mathsf{SS}||\mathsf{FSK}||\left((\mathsf{Prefix3}||\mathsf{MoveToken})|\perp\right)) = \mathsf{Dec}(\mathsf{RC})$$

$$\mathsf{SessionID} =?\mathsf{Enc}(k, \mathsf{H}(secret||\mathsf{FSK}||\left(\mathsf{Prefix3}|\perp\right)))$$

*** This is only one way to create the RC

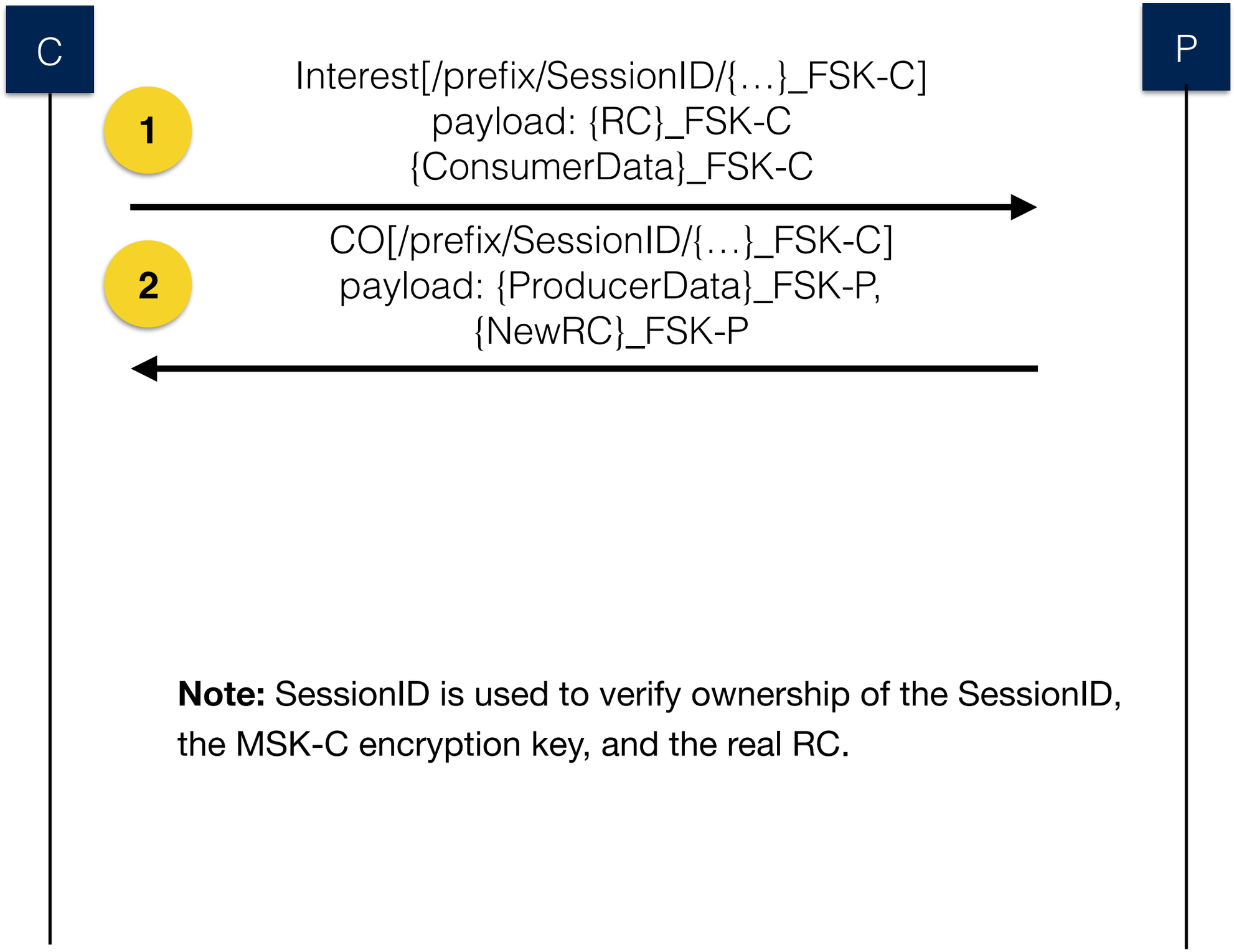
Session Resumption

- Approach 0: If client has nothing, start with HELLO **[2 RTT delay]**

- Approach 1: If the client already has the config, start at the second step **[1 RTT delay]**

- Approach 2: If the client already has the SessionID and the ResumptionCookie, provide both to resume sessions after long periods of inactivity (requires producer state) **[0 RTT delay]**

Session Resumption (cont'd)



Note: SessionID is used to verify ownership of the SessionID, the MSK-C encryption key, and the real RC.