Noise Protocol Framework

Trevor Perrin

Cryptography

- Public Key (DH, RSA, ECC etc)

- Symmetric Key (AES etc)

AKE

- **A**uthenticated **K**ey **E**xchange (or Agreement)
- Goals
  - Shared symmetric key
  - Authentication
  - Forward secrecy

AKE in practice

- TLS, QUIC, SSH, IPsec
- Tor (Ntor, obfsproxy)
- CurveCP, MinimaLT
- OTR, Pond, ZRTP, PGP, Signal
- Noise

AKE variations

- One-sided or mutual auth
- Pre-specified or post-specified peers
- Identity hiding
- Early encryption (0-RTT)
- Mix in pre-shared keys
- Key confirmation, deniability, efficiency, …

AKE patterns

- Framework of AKE “patterns”
- Easy to <u>analyze</u> their properties
- Easy to <u>select</u> based on requirements
- Easy to <u>instantiate</u> and <u>implement</u>

AKE

→ g^x

← g^y

shared_key = g^{xy}

Server Auth

→ g^x

← g^y, **[certificates,] signature**

shared_key = g^{xy}

Mutual Auth

→ g^x

← g^y, [certificates,] signature

→ [certificates,] signature

shared_key = g^{xy}

Details

→ g^x **(can x be reused?)**

← g^y, [certificates,] signature

→ [certificates,] signature

shared_key = g^{xy}

Details

→ g^x **(can x be reused?)**

← g^y, [certificates,] signature**(?)**

→ [certificates,] signature**(?)**

shared_key = g^{xy}

Details

→ g^x **(can x be reused?)**

← g^y, [certificates,] signature**(?)**

→ [certificates,] signature**(?)**

shared_key = **H(**g^{xy}**)**

Details

→ g^x **(one-time ephemeral)**

← g^y, [certificates,] signature**(?)**

→ [certificates,] signature**(?)**

shared_key = $\mathbf{H}(g^{xy})$

Details

→ g^x **(one-time ephemeral)**

← g^y, [certificates,] signature**(h)**

→ [certificates,] signature**(h)**

shared_key = **H(**g^{xy}**)**

h = hash of handshake so far

Details

→ g^x **(one-time ephemeral)**

← g^y, [certificates,] signature**(h)**

→ [certificates,] signature**(h)**

shared_key = **Hash(**g^{xy}**)**

h = hash of handshake so far

Details

→ g^x **(one-time ephemeral)**

← g^y, [certificates,] signature**(h)**

→ [certificates,] signature**(h)**

shared_key = **Hash(**g^{xy} **|| h)**

h = hash of handshake so far

Identity hiding

→ g^x **(one-time ephemeral)**

← g^y,**<DH>,** [certificates,] signature**(h)**

→ [certificates,] signature**(h)**

shared_key = **Hash(**g^{xy} || **h)**

h = hash of handshake so far

Simplifying

→ g^x **(one-time ephemeral)**

← g^y, <DH>, [certificates,] signature**(h)**

→ [certificates,] signature**(h)**

Simplifying

→ **e**

← **e**, <DH>, [certificates,] signature**(h)**

→ [certificates,] signature**(h)**

Simplifying

→ e

← e, <DH>, [certificates,] **signature**

→ [certificates,] **signature**

Simplifying

→ e

← e, <DH>, **s,** signature

→ **s,** signature

Simplifying

→ e, **[payload]**

← e, <DH>, s, signature, **[payload]**

→ s, signature, **[payload]**

Simplifying

→ e

← e, <DH>, s, signature

→ s, signature

Noise patterns

→ e

← e, **dhee,** s, **dhse**

→ s, dhse

$$\begin{tikzcd} s_i & s_r \arrow[dl] \\ e_i & e_r \arrow[l] \end{tikzcd}$$

Noise patterns

→ e

← e, dhee, s, dhse

→ s**, dhse**

$$\begin{tikzcd} s_i \arrow[dr, red] & s_r \arrow[dl] \\ e_i & e_r \arrow[l] \end{tikzcd}$$

Noise patterns

→ e

← e, **dhee**, s, **dhse**

→ s, **dhse**

e: Send ephemeral public key

s: Send static public key (encrypt if shared key exists)

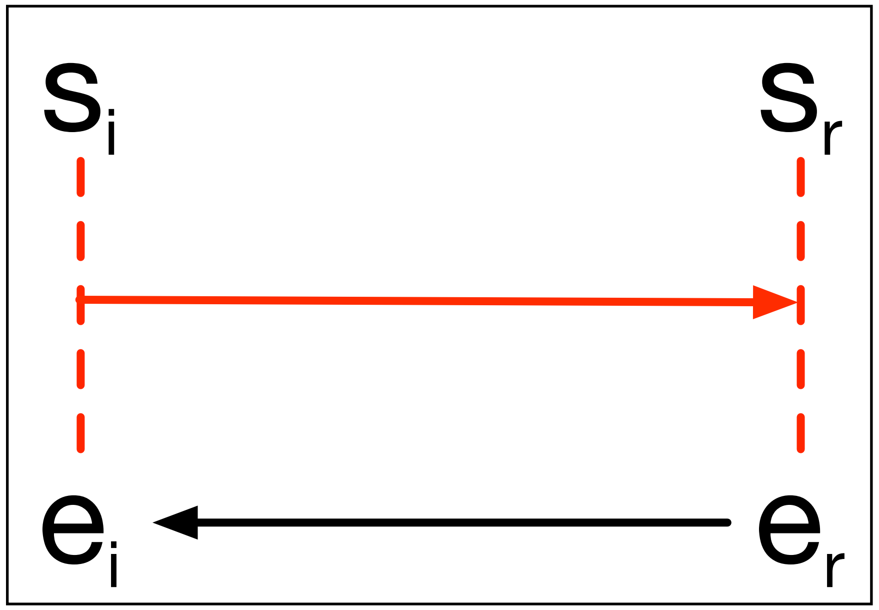
dh[send][recv]: Mix the specified DH into the shared key

MQV?

→ e

← e, dhee, s

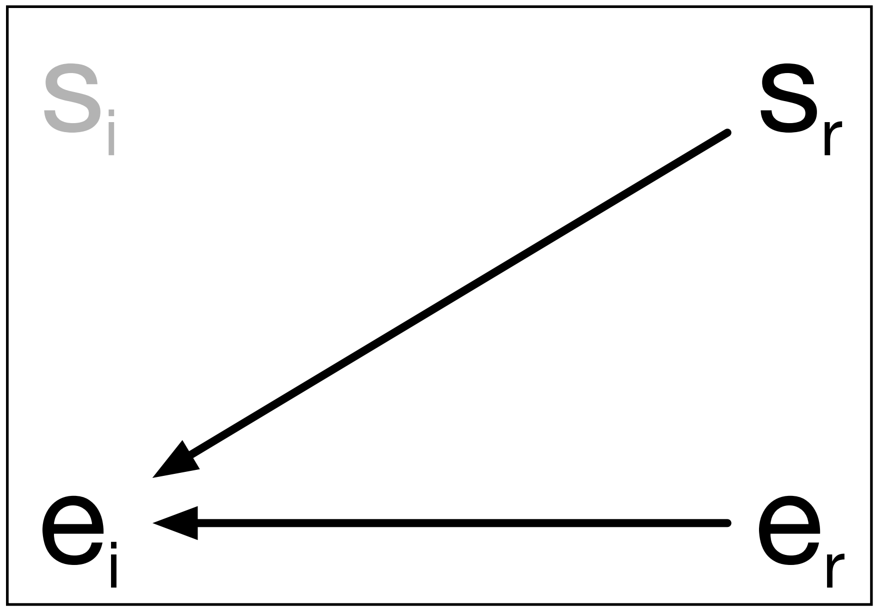
→ s, **MQV**



Noise patterns (Ntor?)

→ e

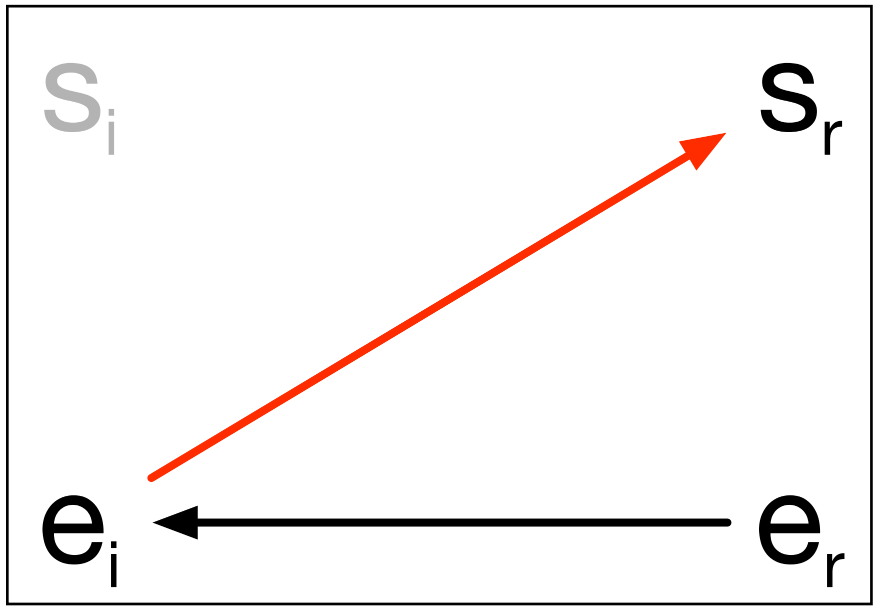
← e, dhee, dhse


$$\begin{tikzcd}
s_i & & s_r \arrow[dll] \\
e_i & & e_r \arrow[ll]
\end{tikzcd}$$

0-RTT encryption

→ e, **dhes**

← e, dhee



0-RTT + client auth

→ e, dhes**, s, dhss**

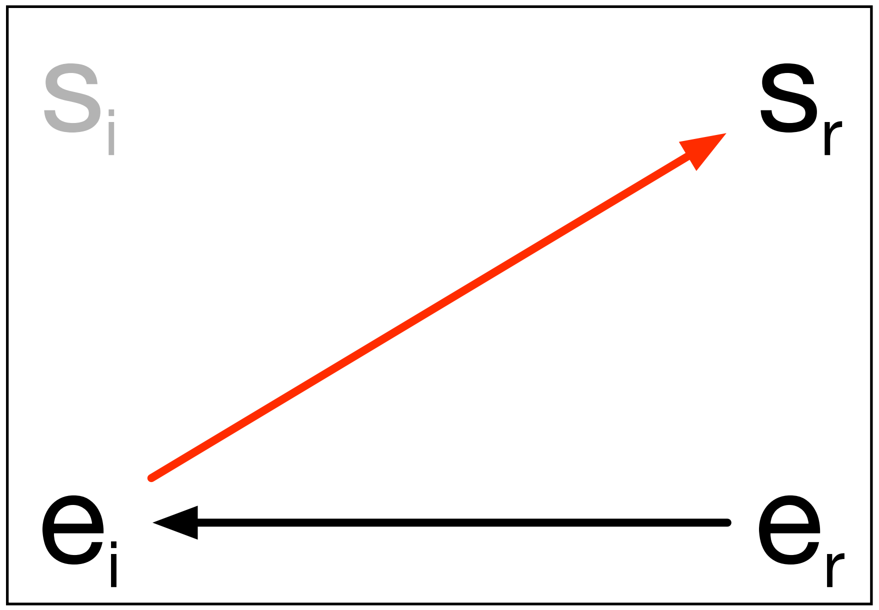
← e, dhee**, dhes**

$$\begin{tikzcd}
s_i \arrow[r] & s_r \\
e_i \arrow[ur] & e_r \arrow[l] \arrow[ul]
\end{tikzcd}$$

0-RTT encryption

→ e, **dhes**

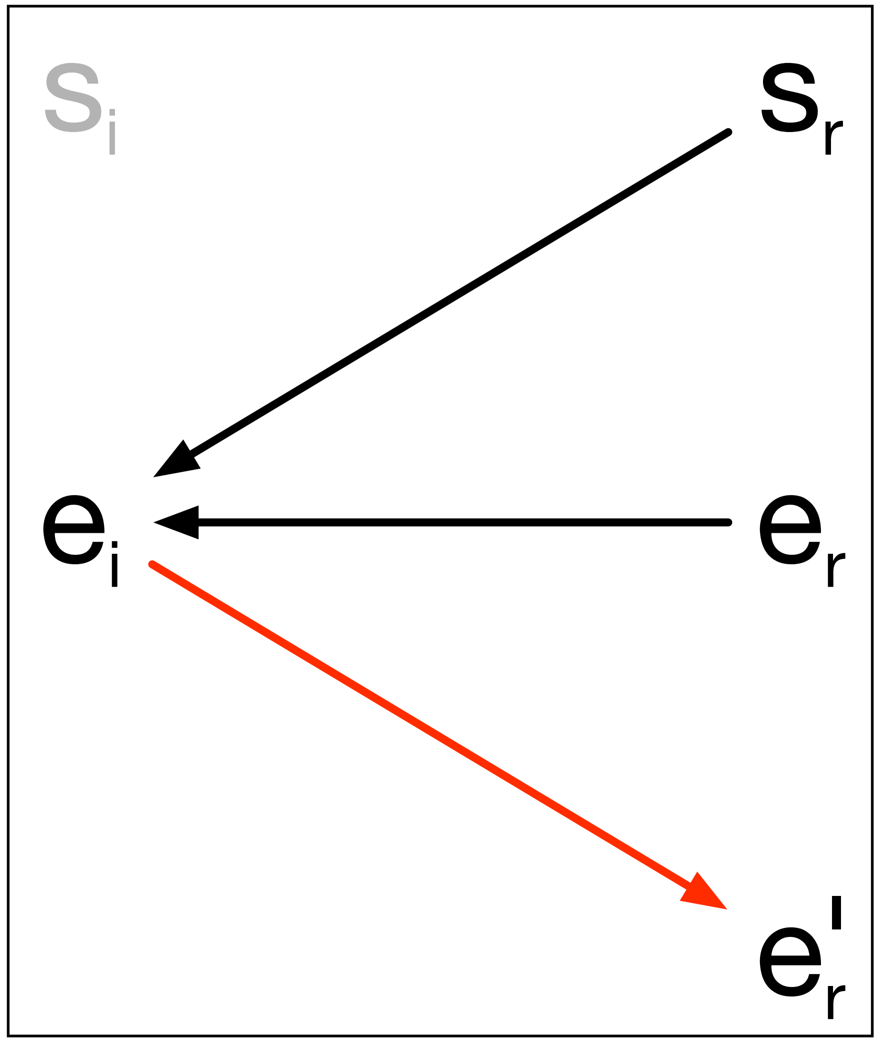
← e, dhee



0-RTT (QUIC?)

→ e, **dhee**

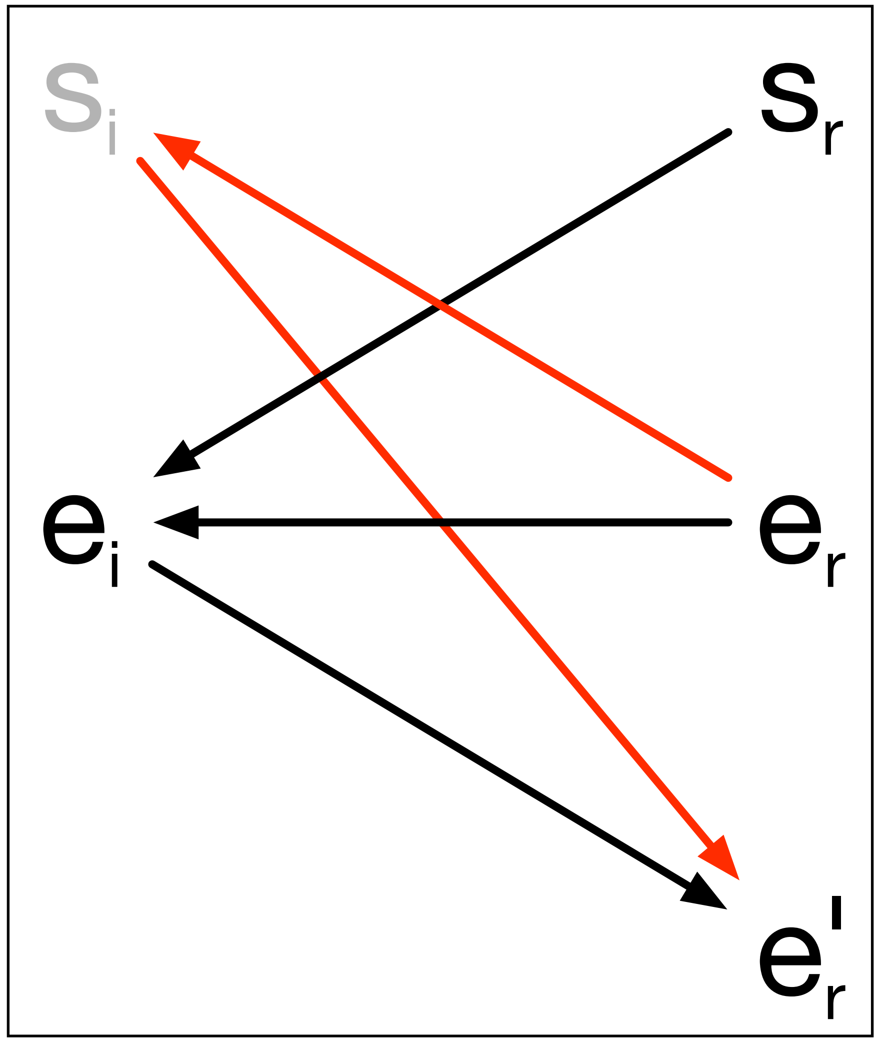
← e, dhee, dhse



0-RTT + client auth

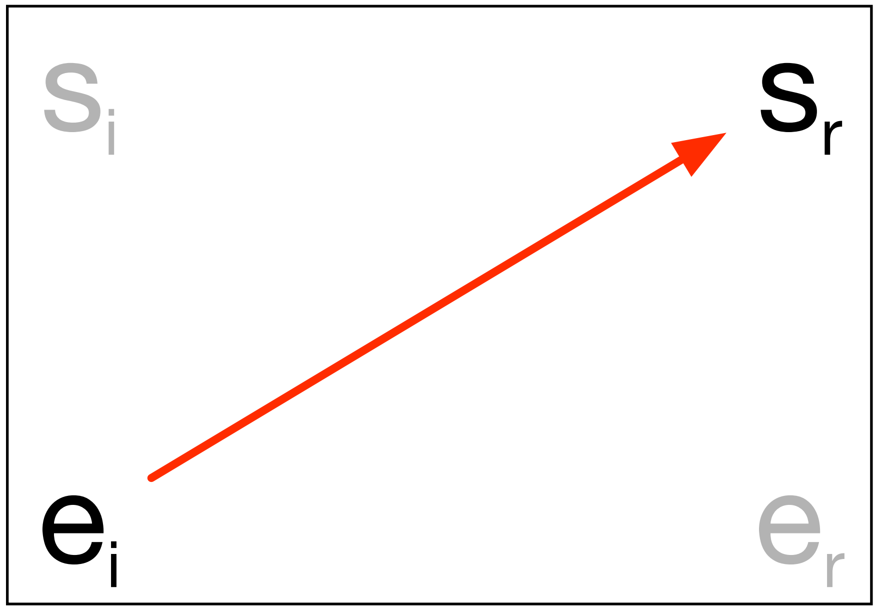
→ e, dhee, s, dhse

← e, dhee, dhse, **dhes**


$$\begin{tikzcd}
s_i \arrow[drr] & & s_r \arrow[dll] \\
e_i \arrow[drr] & & e_r \arrow[ll] \arrow[ull] \\
& & e'_r
\end{tikzcd}$$

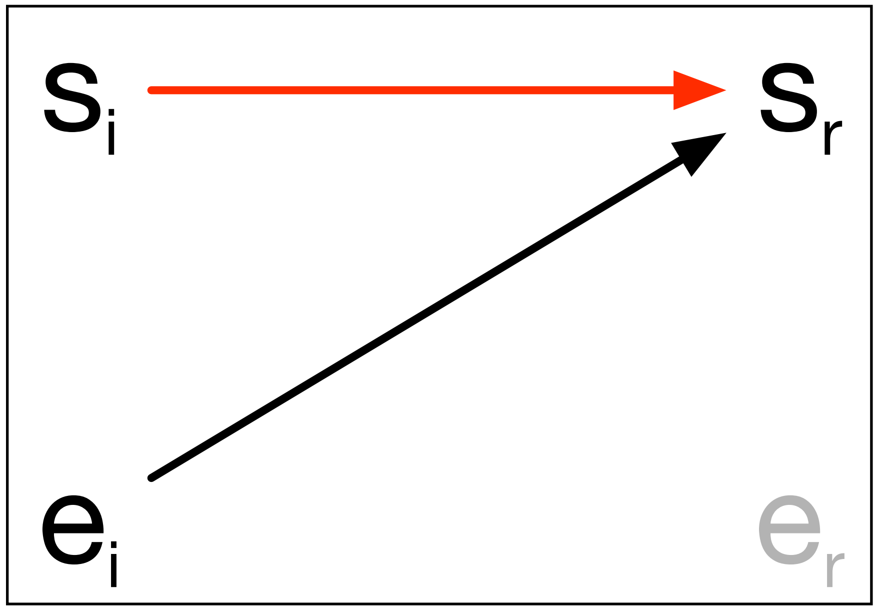
Public-key encryption

→ e, **dhes**



Authenticated encryption

→ e, dhes, **dhss**



Protocol names

- Noise_*pattern_dh_cipher_hash*

- Noise_XX_25519_AESGCM_SHA256
- Noise_IK_448_ChaChaPoly_BLAKE2b
- Noise_NX_25519_AESGCM_BLAKE2s

Symmetric crypto

- Key derivation (KDF)
- Bind keys to context

Strategy

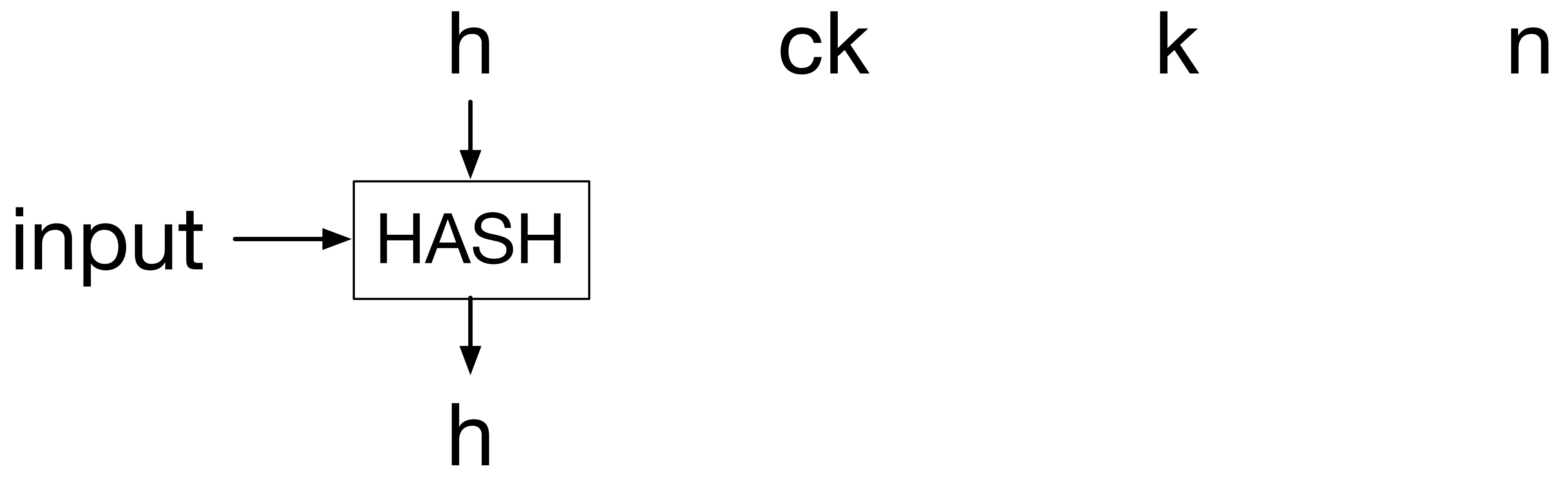
- Mix all secrets into a **chaining key** and **encryption key**
- Encrypt everything except ephemerals, once an encryption key is present
- Hash all transcript into **handshake hash**
- Authenticate handshake hash in all handshake encryptions

Symmetric state

h	ck	k	n

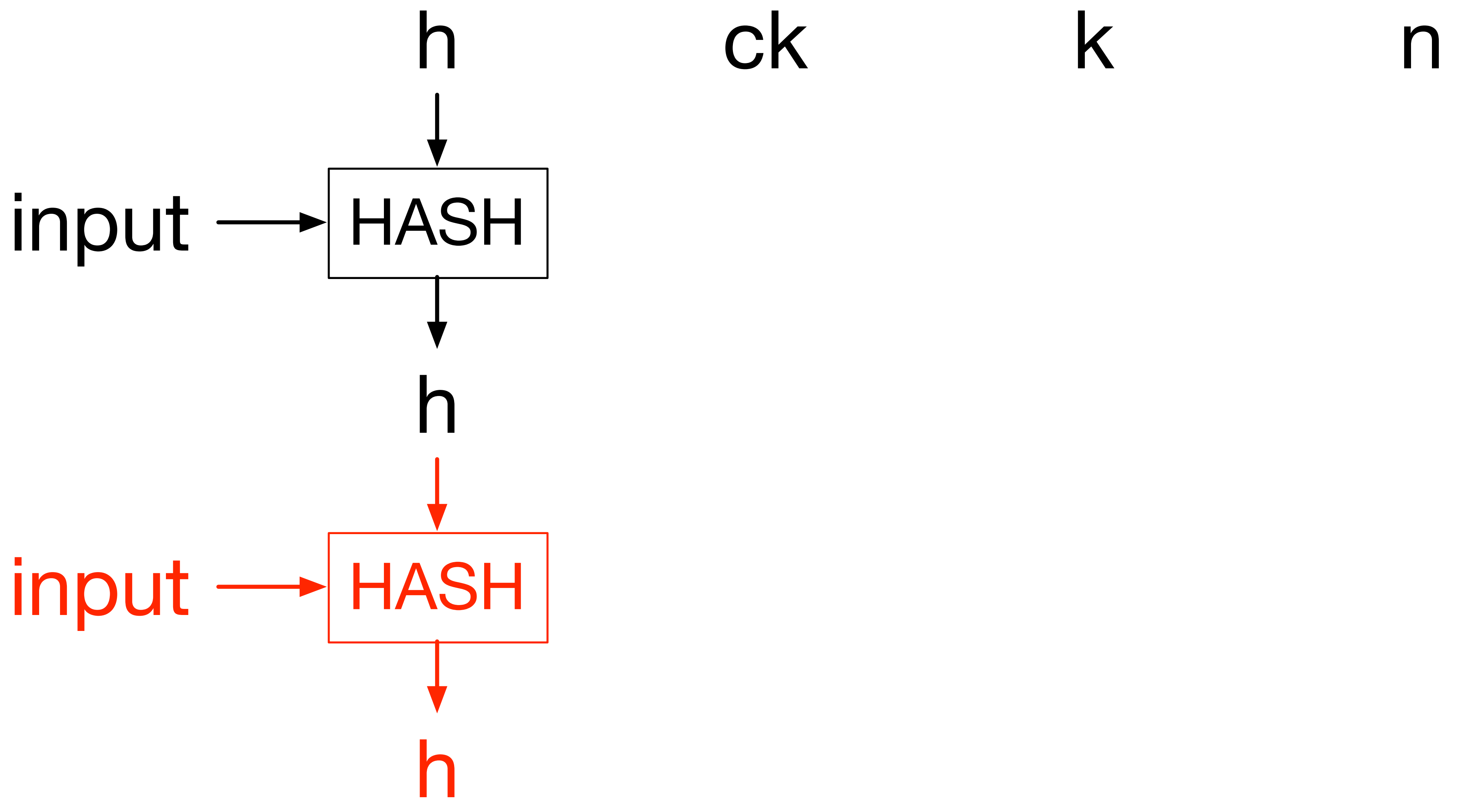
- **h** = handshake hash
- **ck** = chaining key
- **k** = encryption key
- **n** = encryption nonce (counter)

Hashing inputs

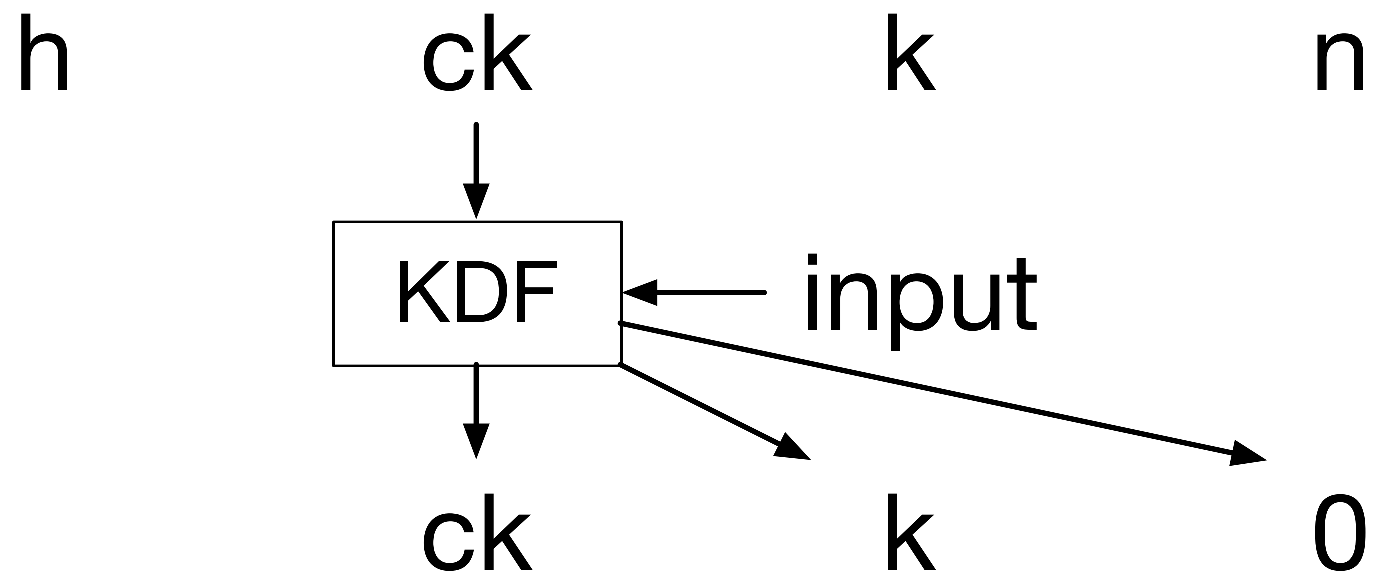


$$\boxed{h = \text{HASH}(h \mathbin{||} \text{input})}$$

Hashing inputs



KDF



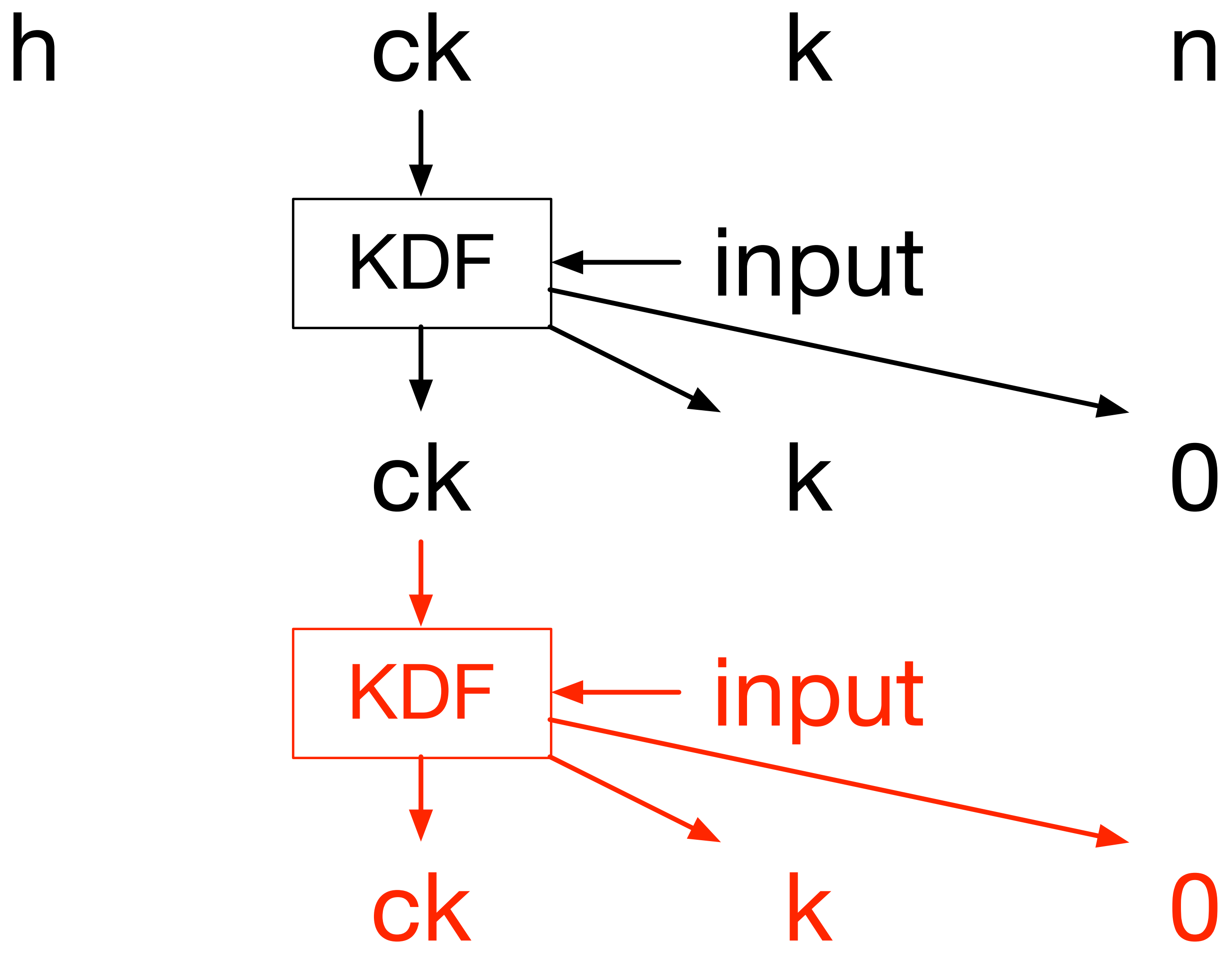
ck, k = HKDF(salt=ck, input)

KDF

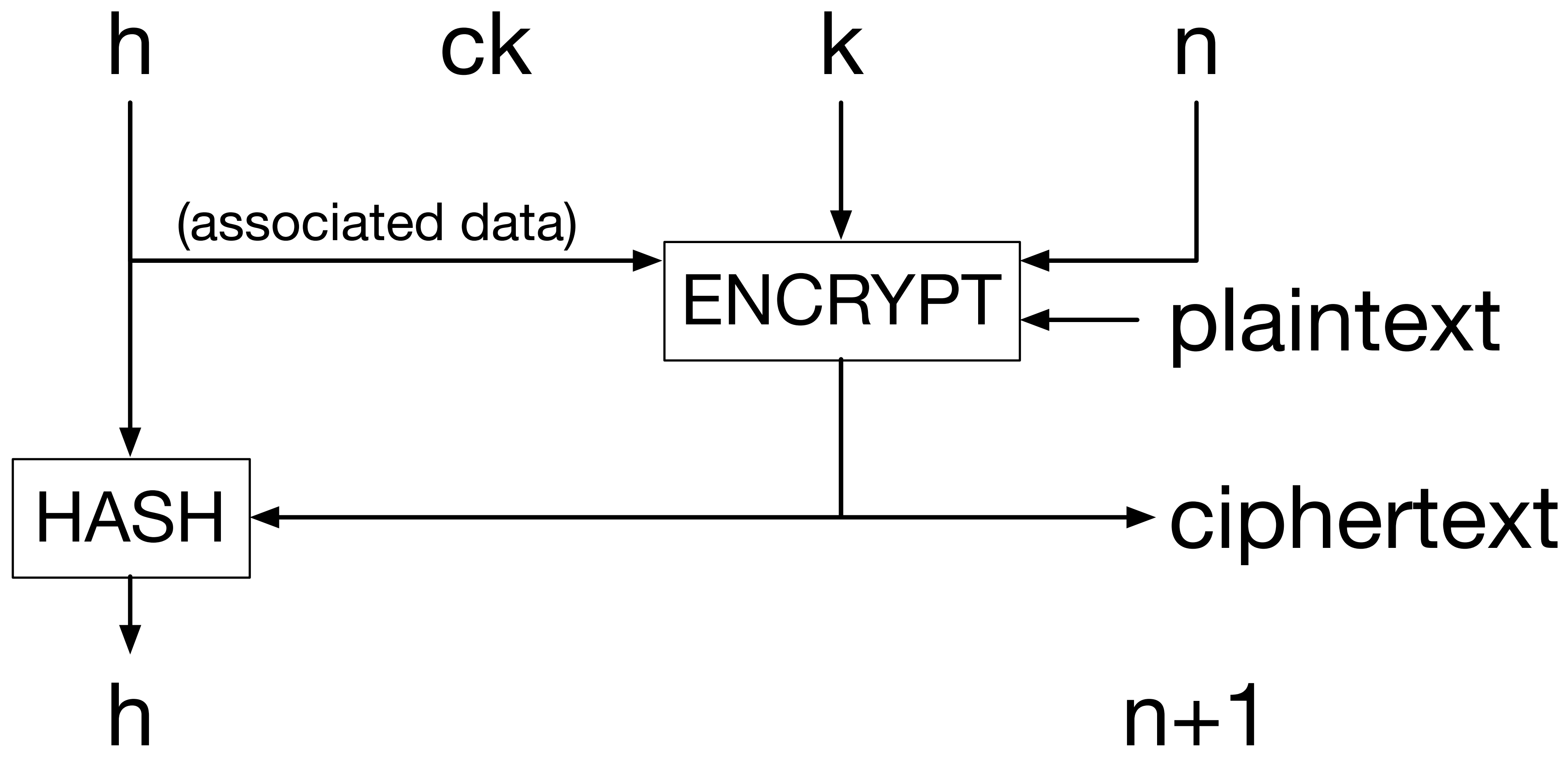
$$\begin{tikzcd}
h & ck \arrow[d] & k & n \\
 & \boxed{\text{KDF}} \arrow[d] \arrow[dr] \arrow[drr] & \text{input} \arrow[l] & \\
 & ck & k & 0
\end{tikzcd}$$

```
temp = HMAC(ck, input)
ck = HMAC(temp, 0x01)
k = HMAC(temp, ck || 0x02)
```

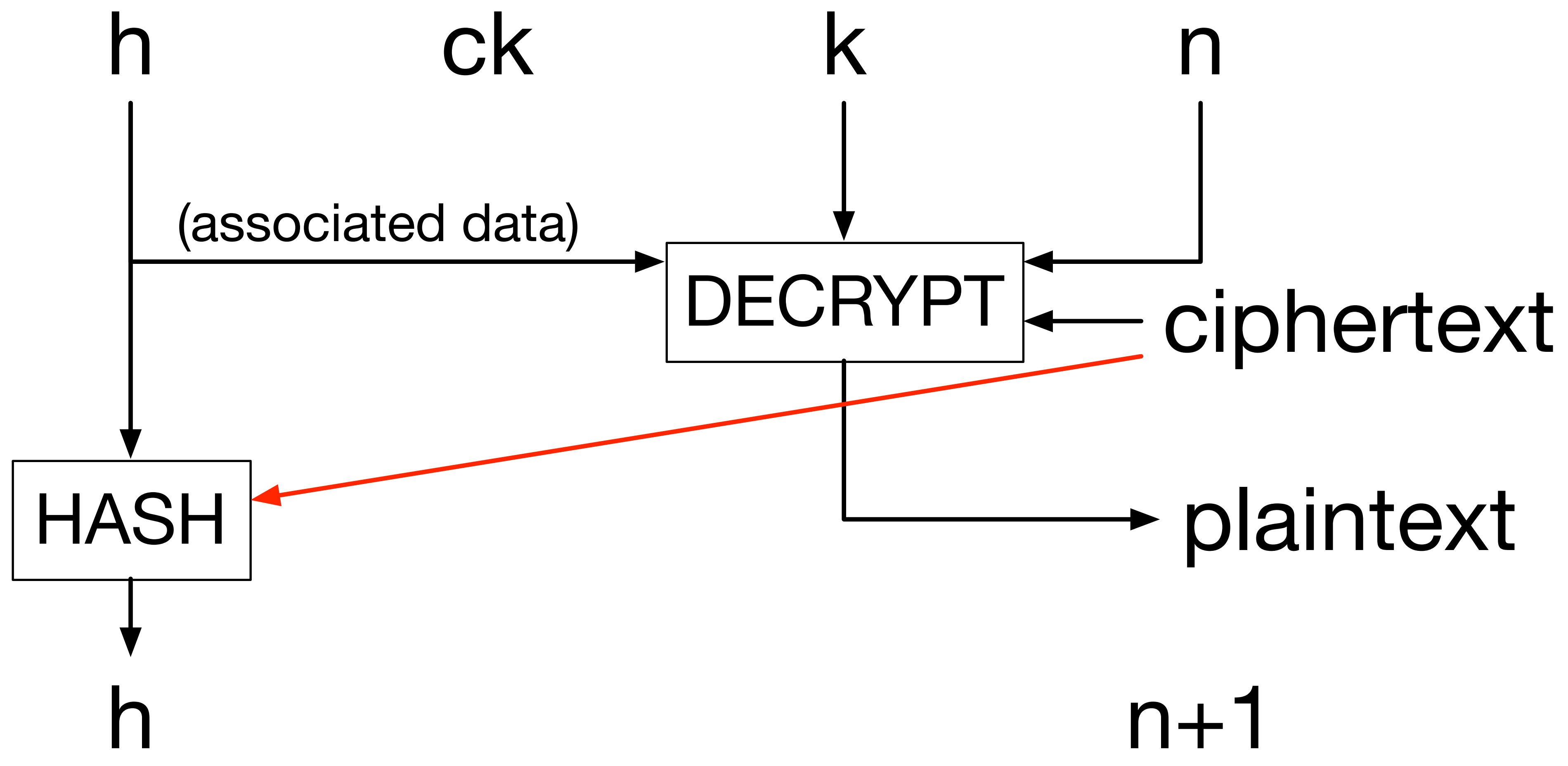
KDF

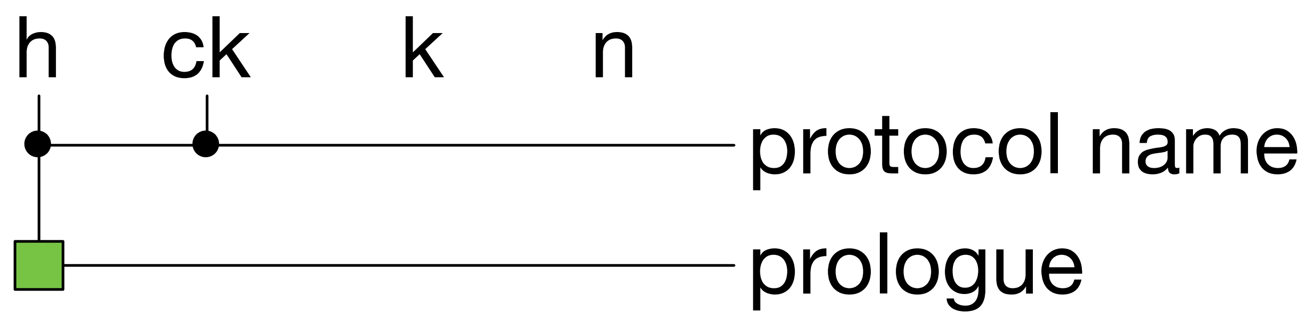


Handshake encryption



Handshake decryption





→ e

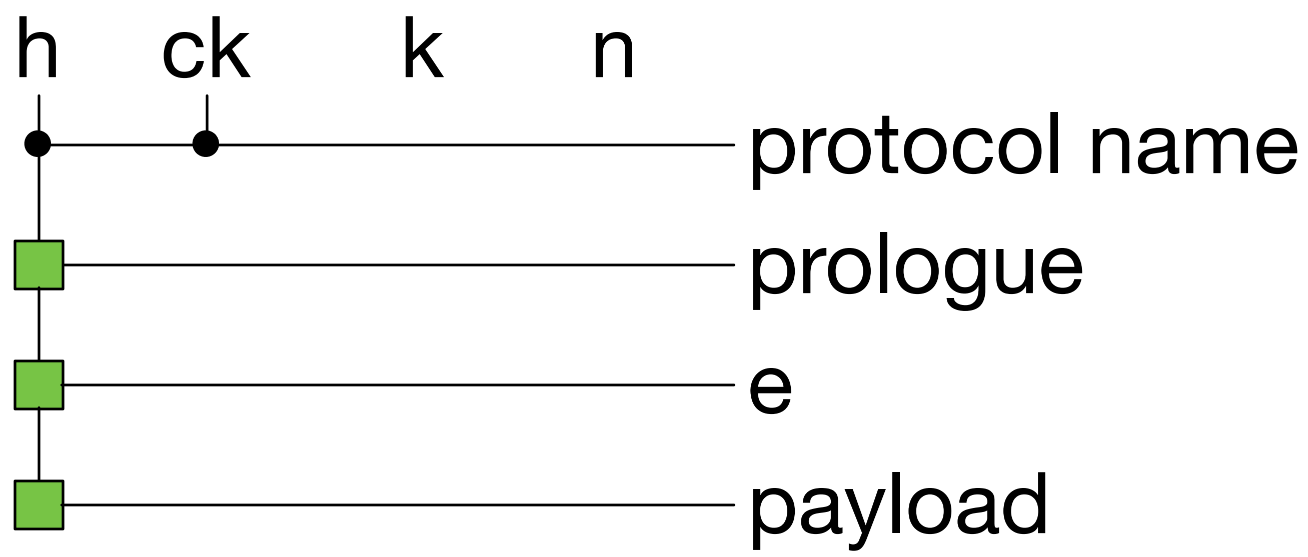
← e, dhee, s, dhse

→ s, dhse

→ **e**

← e, dhee, s, dhse

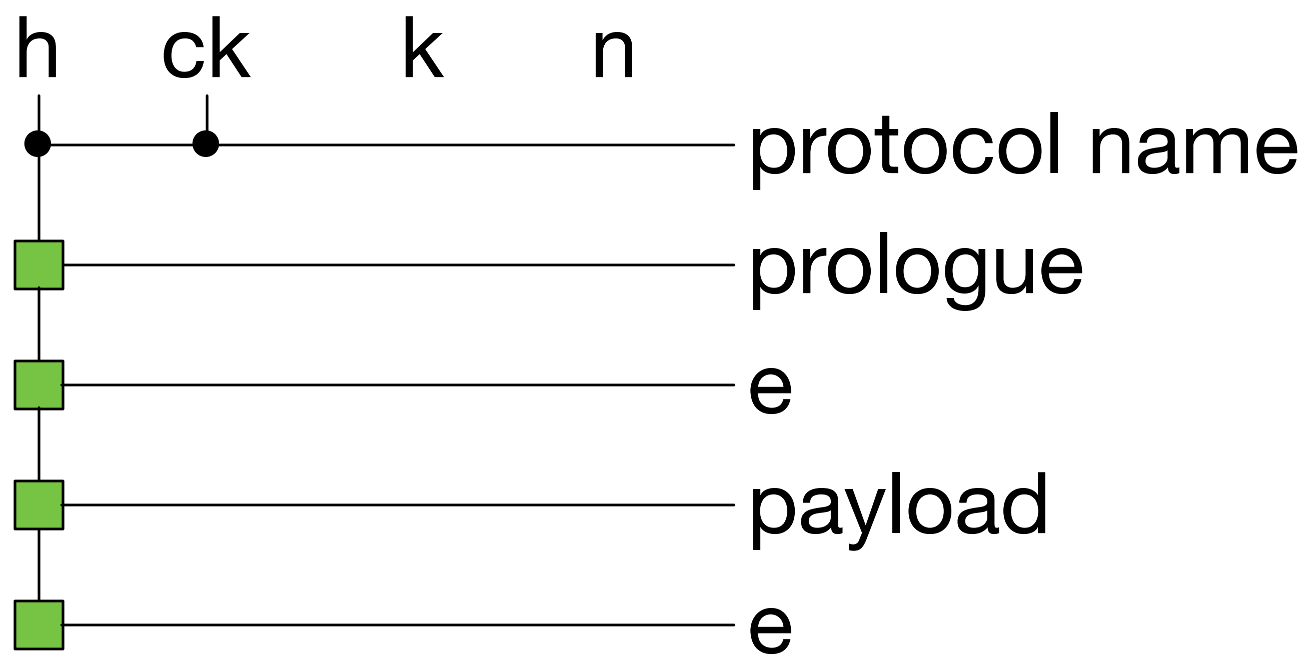
→ s, dhse



→ e

← **e,** dhee, s, dhse

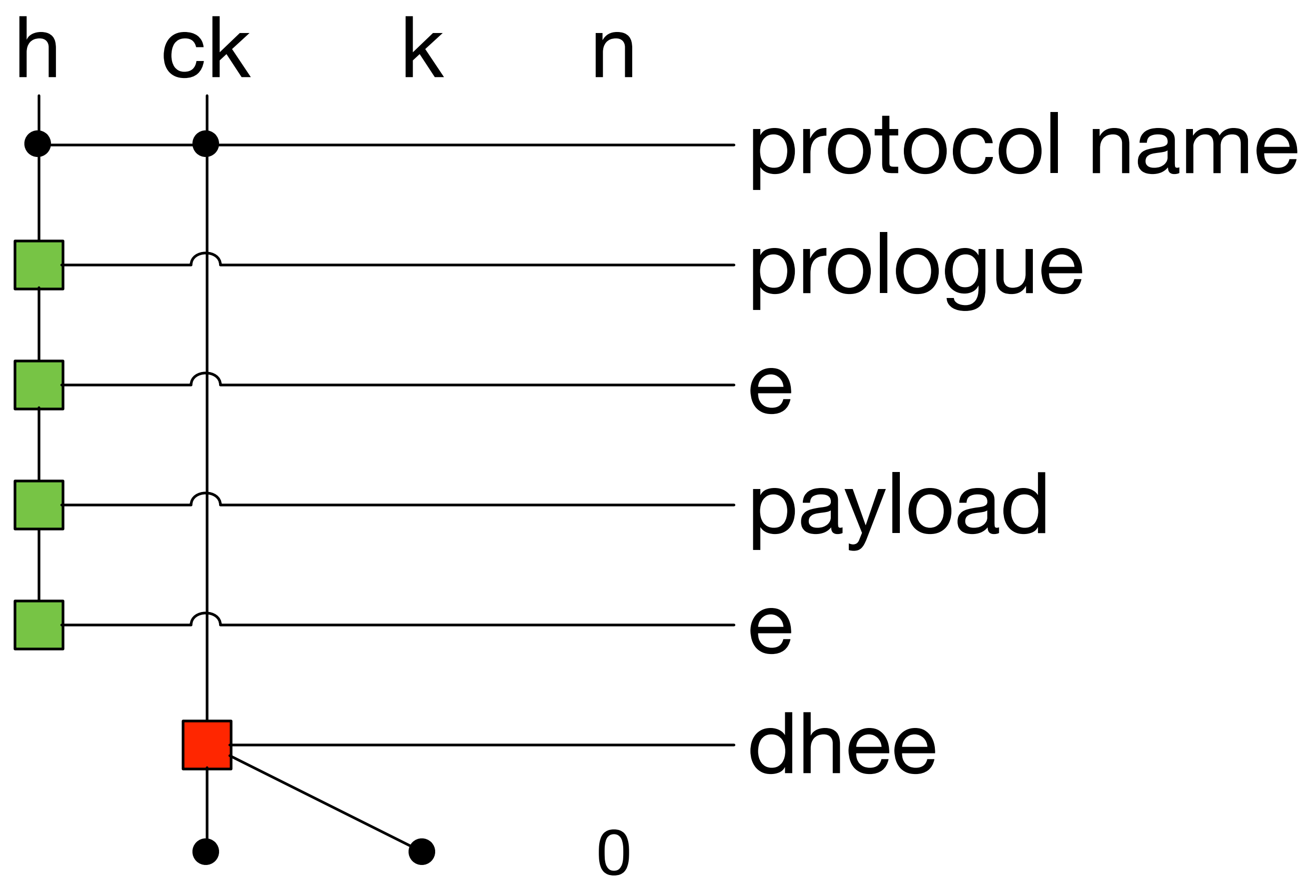
→ s, dhse



→ e

← e, **dhee,** s, dhse

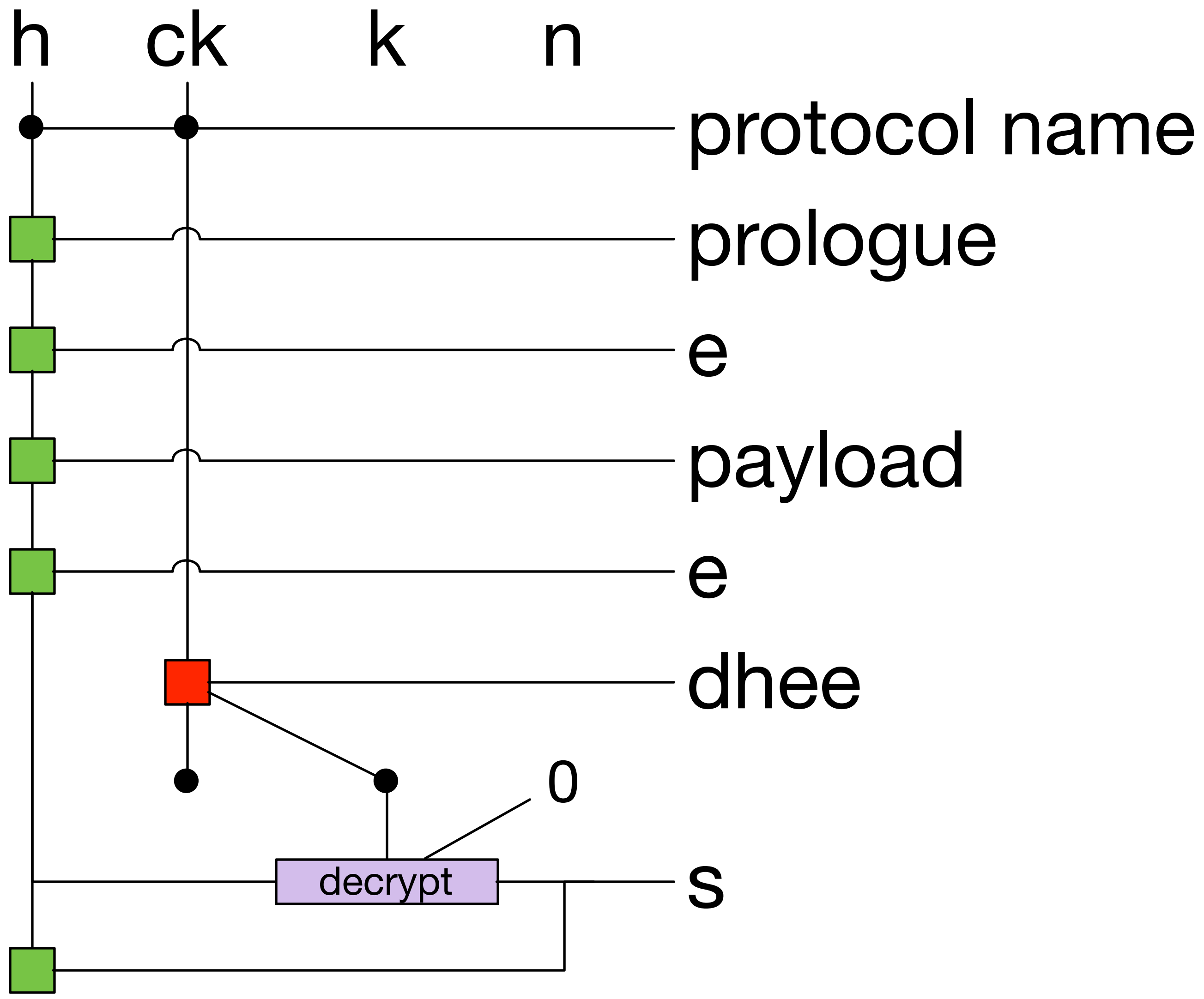
→ s, dhse



→ e

← e, dhee, **s,** dhse

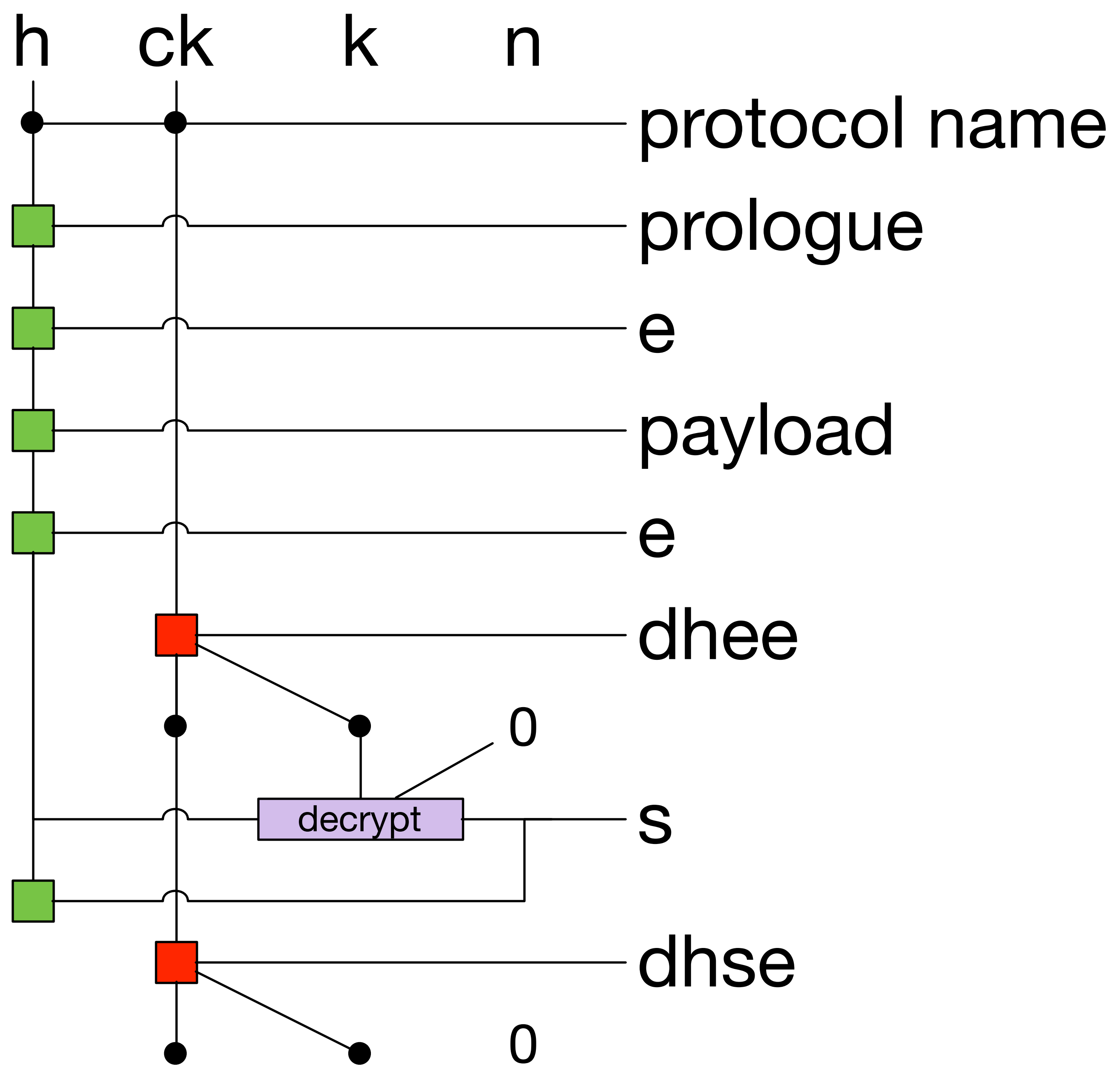
→ s, dhse



→ e

← e, dhee, s, **dhse**

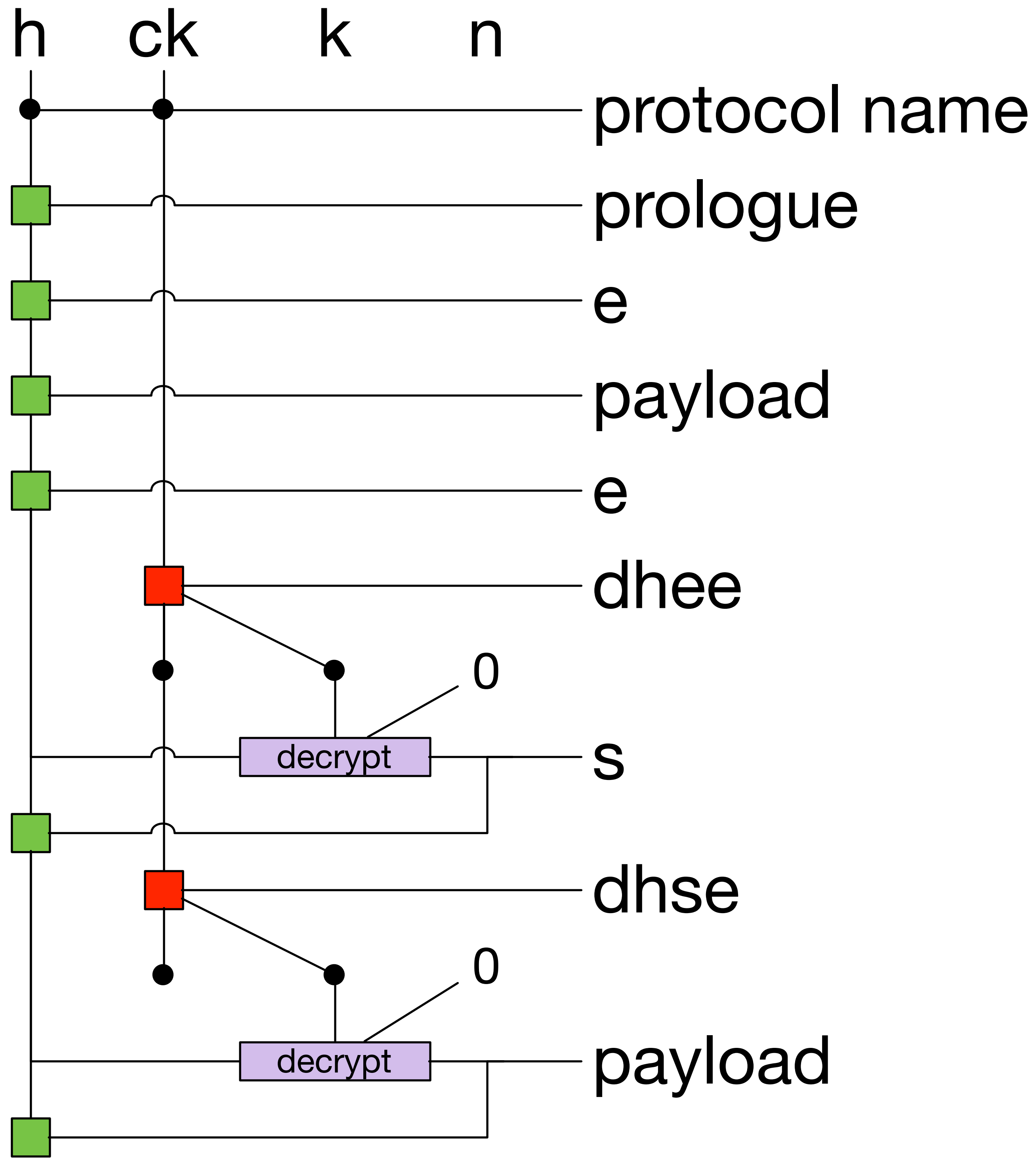
→ s, dhse



→ e

← e, dhee, s, **dhse**

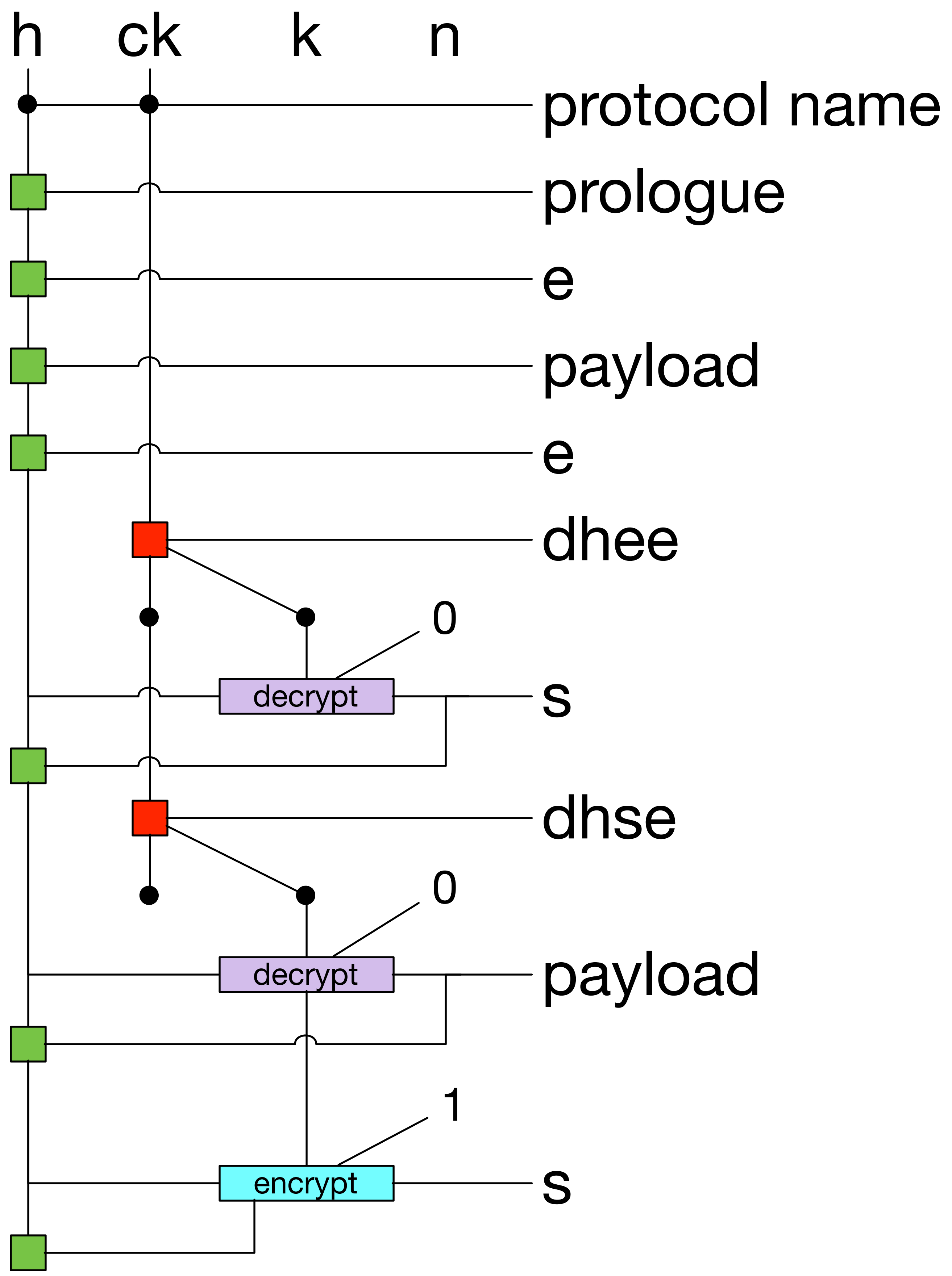
→ s, dhse



→ e

← e, dhee, s, dhse

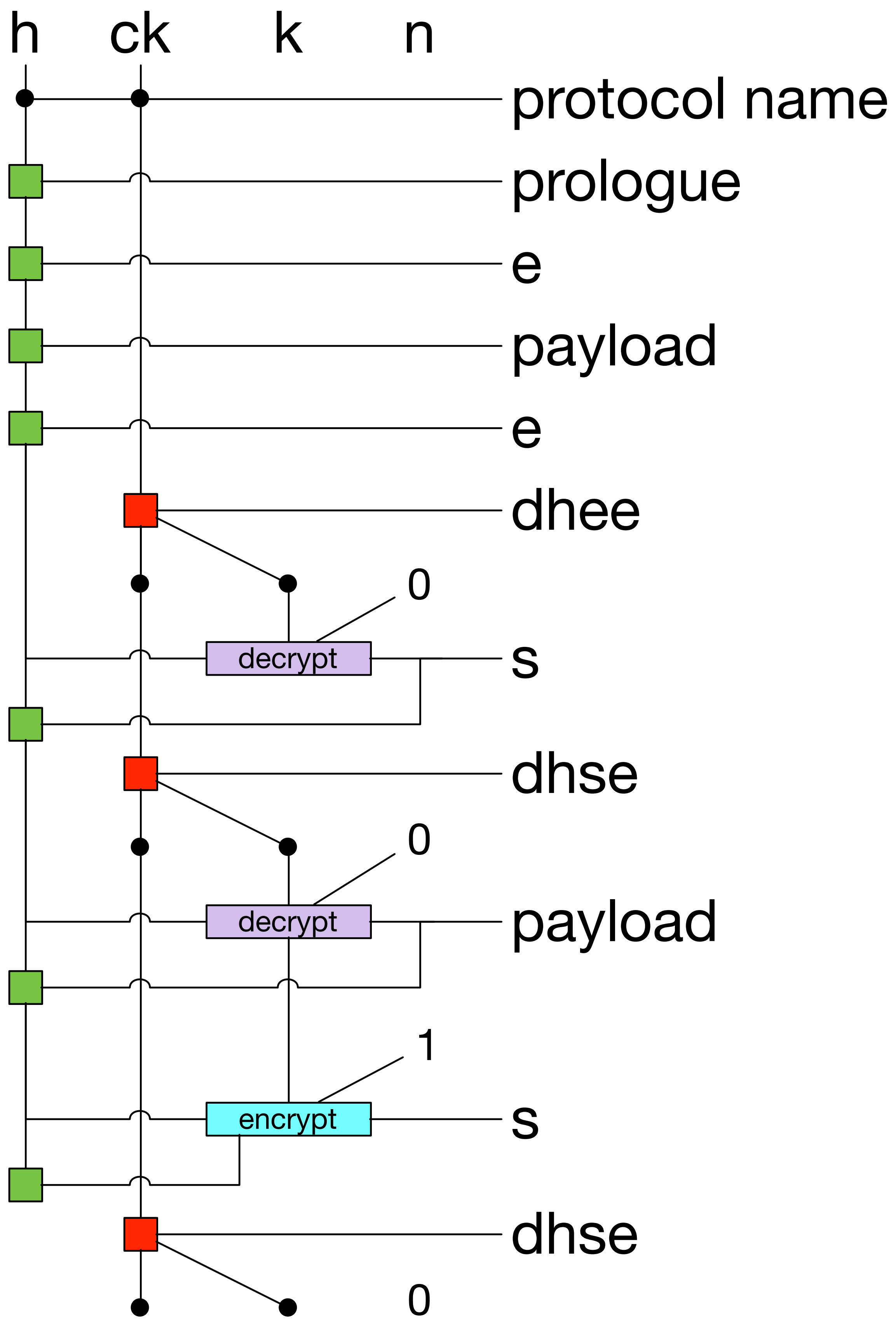
→ **s,** dhse



→ e

← e, dhee, s, dhse

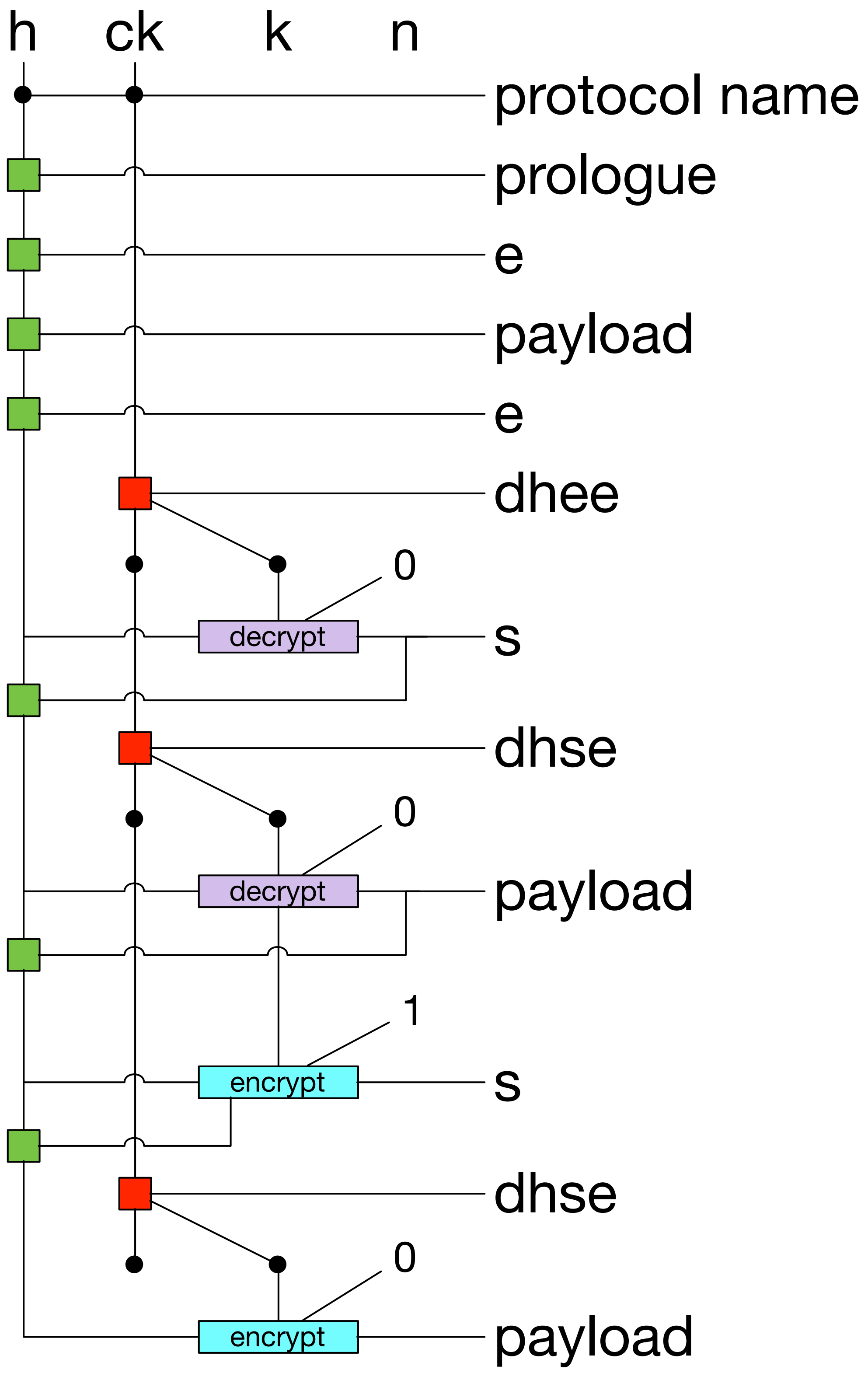
→ s, **dhse**



→ e

← e, dhee, s, dhse

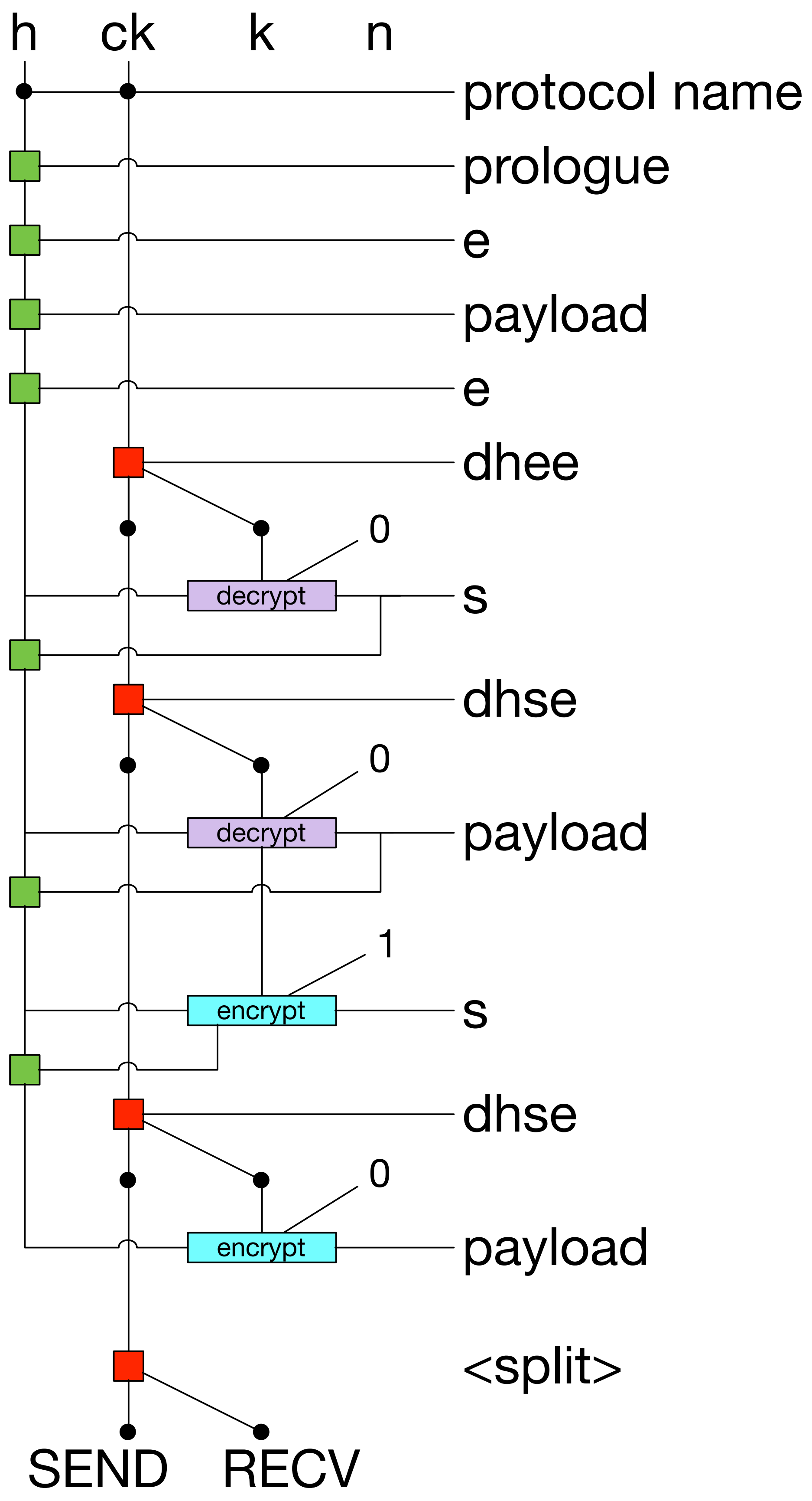
→ s, dhse



→ e

← e, dhee, s, dhse

→ s, dhse



The future?

- Find more users
- Get more analysis
- New patterns and crypto functions
- Extend the language, new symmetric crypto?
- https://noiseprotocol.org