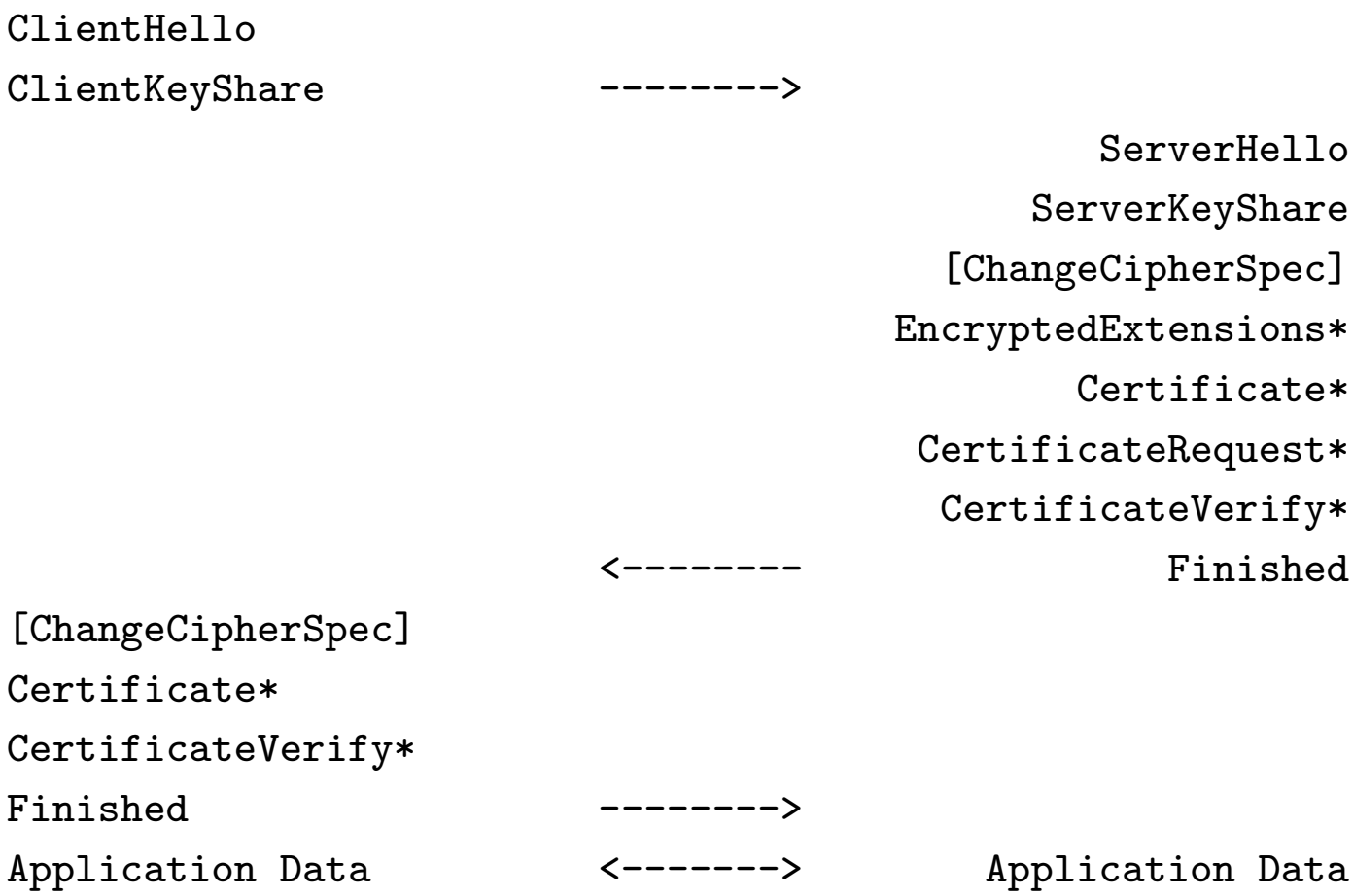
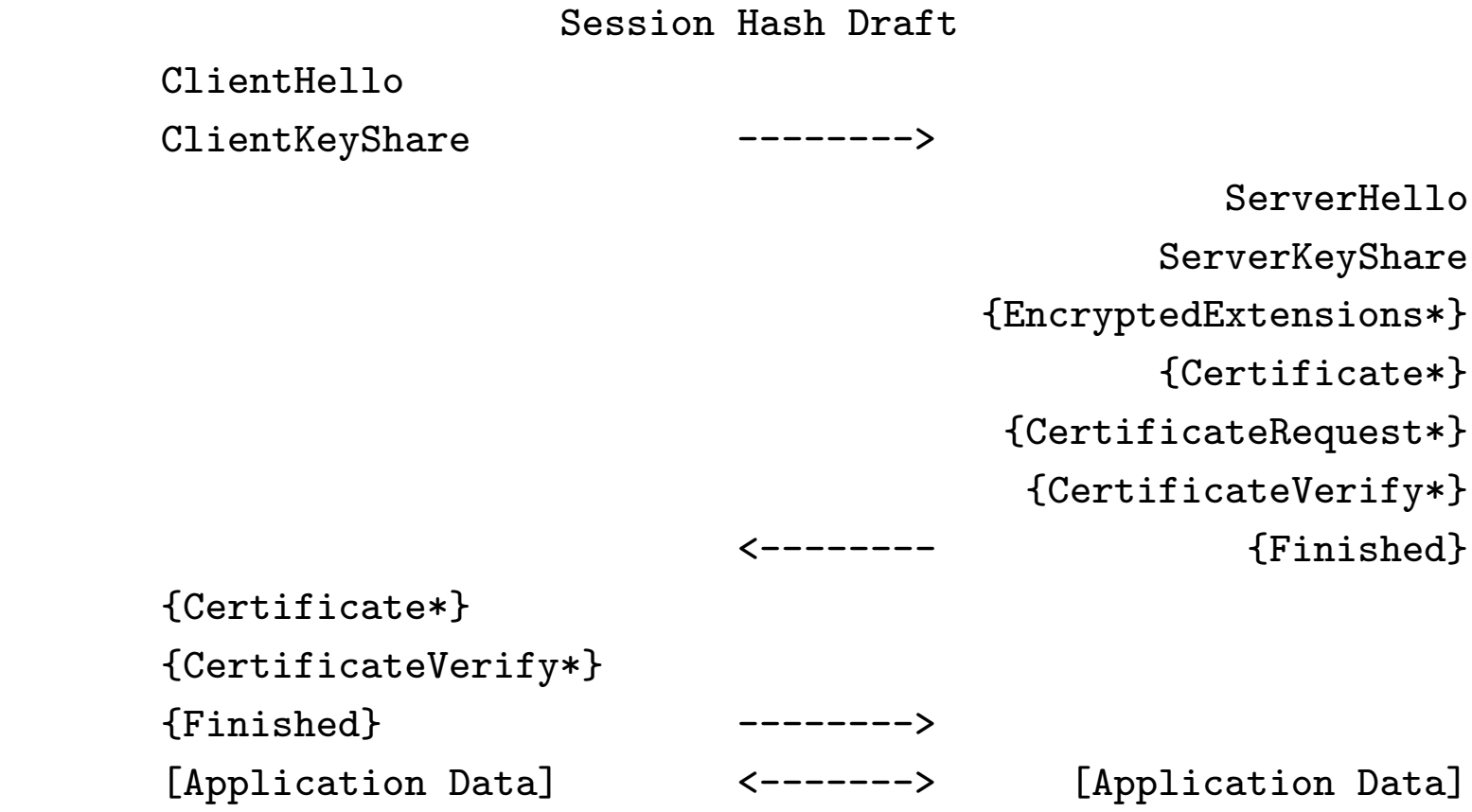
Existing Draft

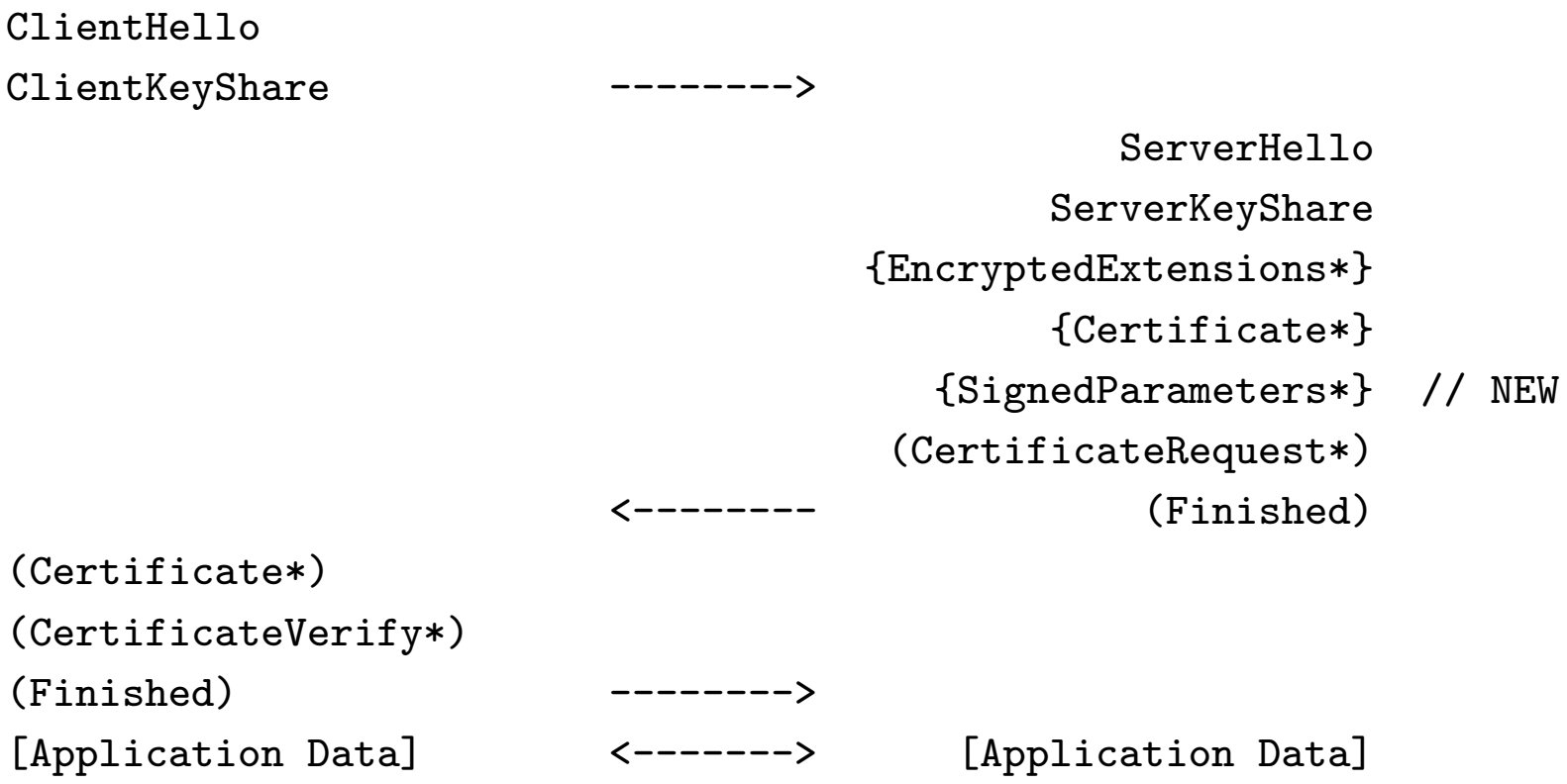




{} Indicates messages protected using keys derived from the handshake master secret.

[] Indicates messages protected using keys derived from the master secret.

Hugo's Proposal (slightly adapted, server-side only)



{} encrypted under g^xy

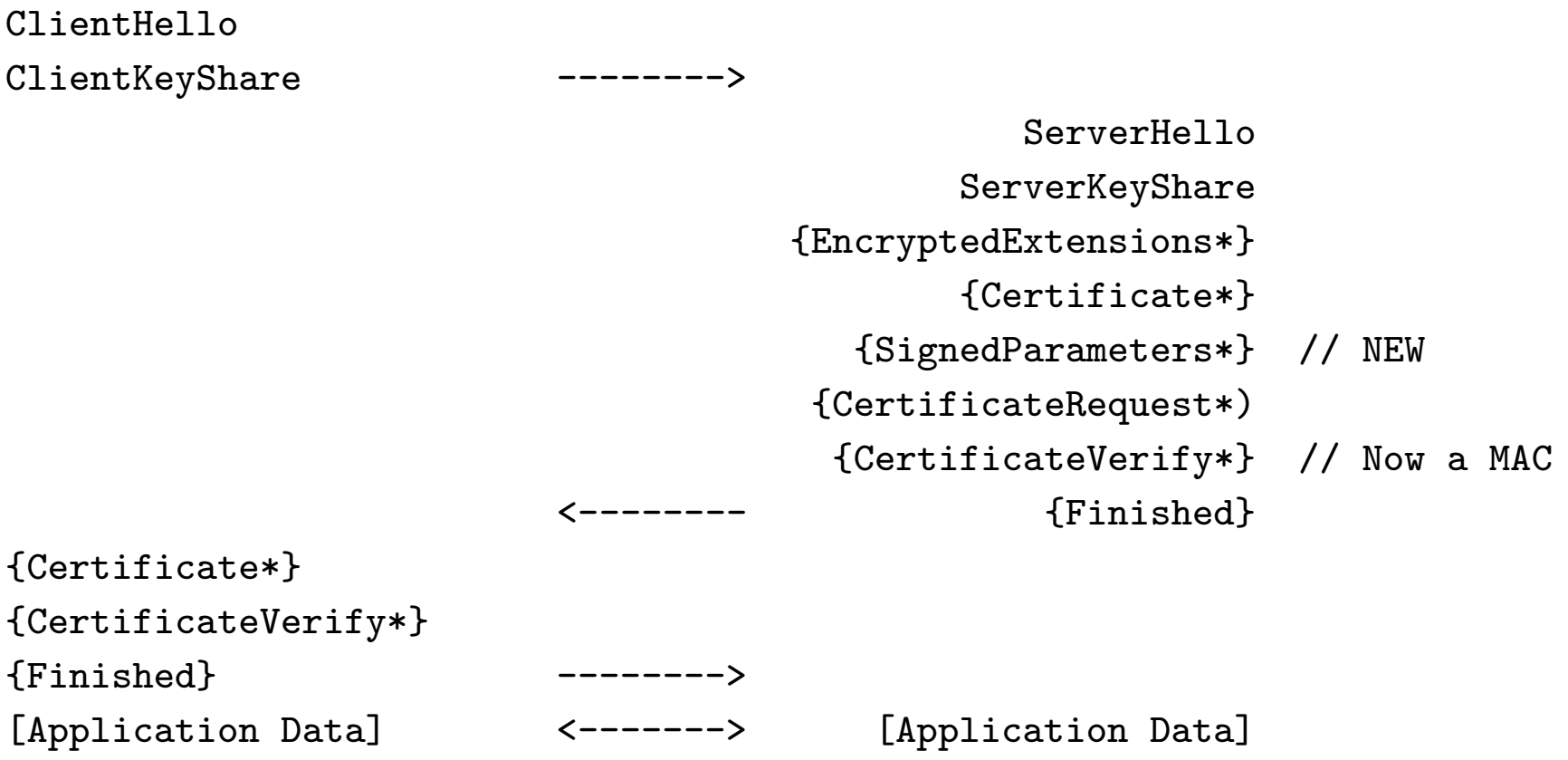
() encrypted under g^xy, g^xs (handshake key)

[] encrypted under g^xy, g^xs (application key)

```
struct {
  opaque identifier[16];
  uint64 not_before;
  uint64 not_after;
  NamedGroup group;
  opaque key_exchange;
} UnsignedParameters;

struct {
    UnsignedParameters parameters;
    digitally-signed struct {
      opaque zeros[64];
      UnsignedParameters parameters;
    };
 } SignedParameters;
```

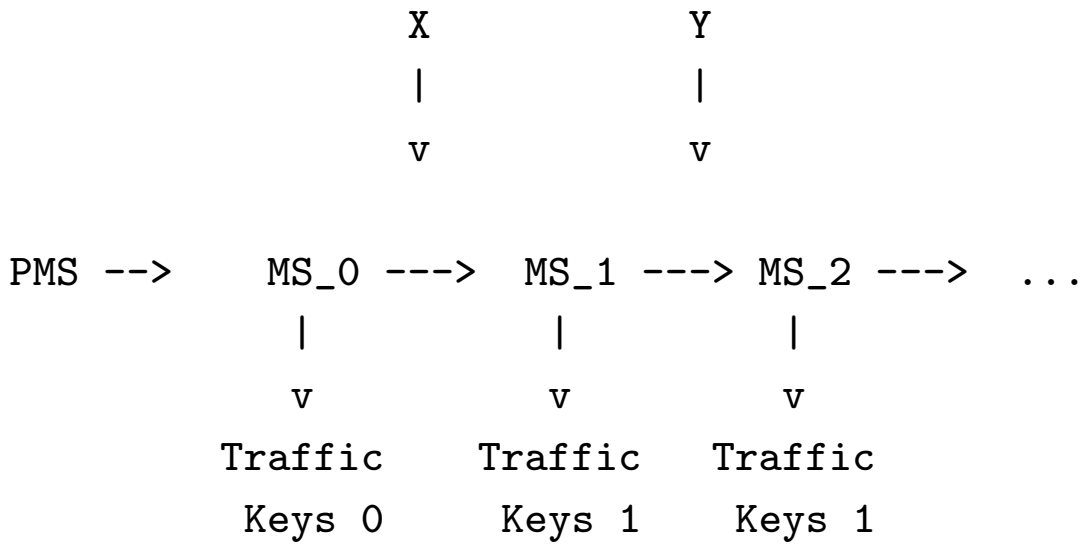
Variant of Hugo's proposal (server side only)

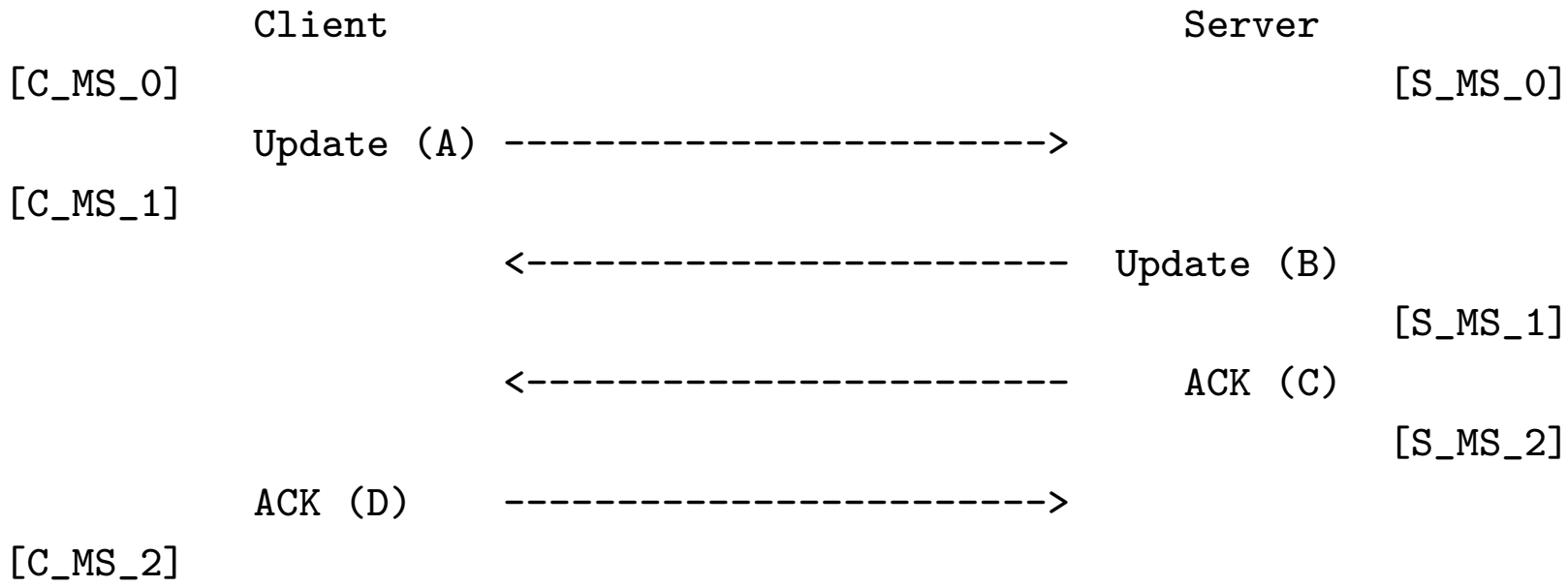


Server CertificateVerify is computed as HMAC(g^xs, transcript)

{} encrypted under the handshake master secret (g^xy)

[] encrypted under the master secret (g^xy + g^xs)

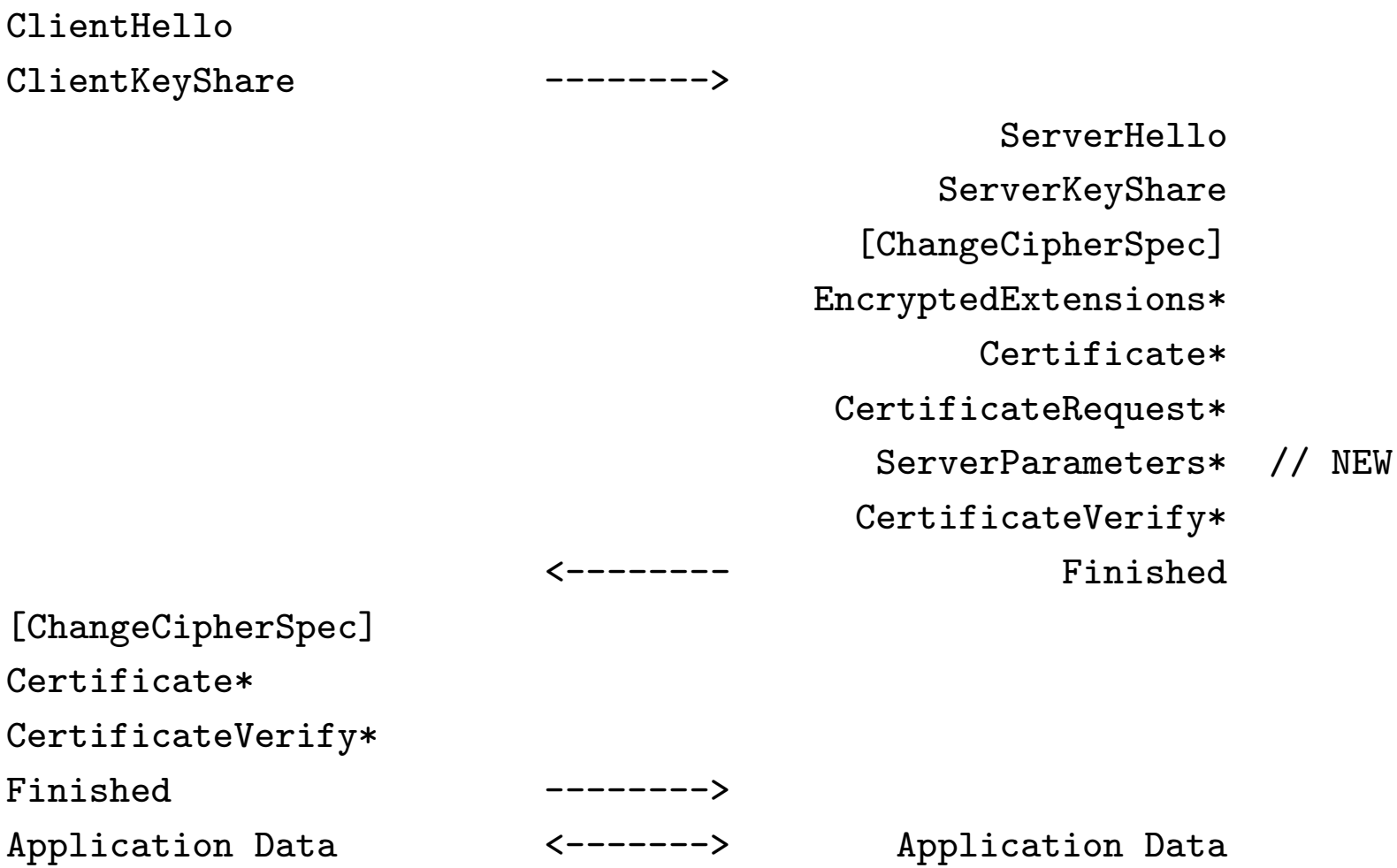

X
Y
PMS -->
MS_0 --->
MS_1 --->
MS_2 --->
...
Traffic
Keys 0
Traffic
Keys 1
Traffic
Keys 1



The key computations are as follows:

```
C_MS_1 = PRF(C_MS_0, A)
S_MS_1 = PRF(S_MS_0, B)
C_MS_2 = PRF(C_MS_1, D) [D depends only on B]
S_MS_2 = PRF(S_MS_1, C) [C depends only on A, but S_MS_2 depends
                         on S_MS_1 which depends on B].
```

First Handshake for 0-RTT



0-RTT Handshake (very Sketchy, no client auth)