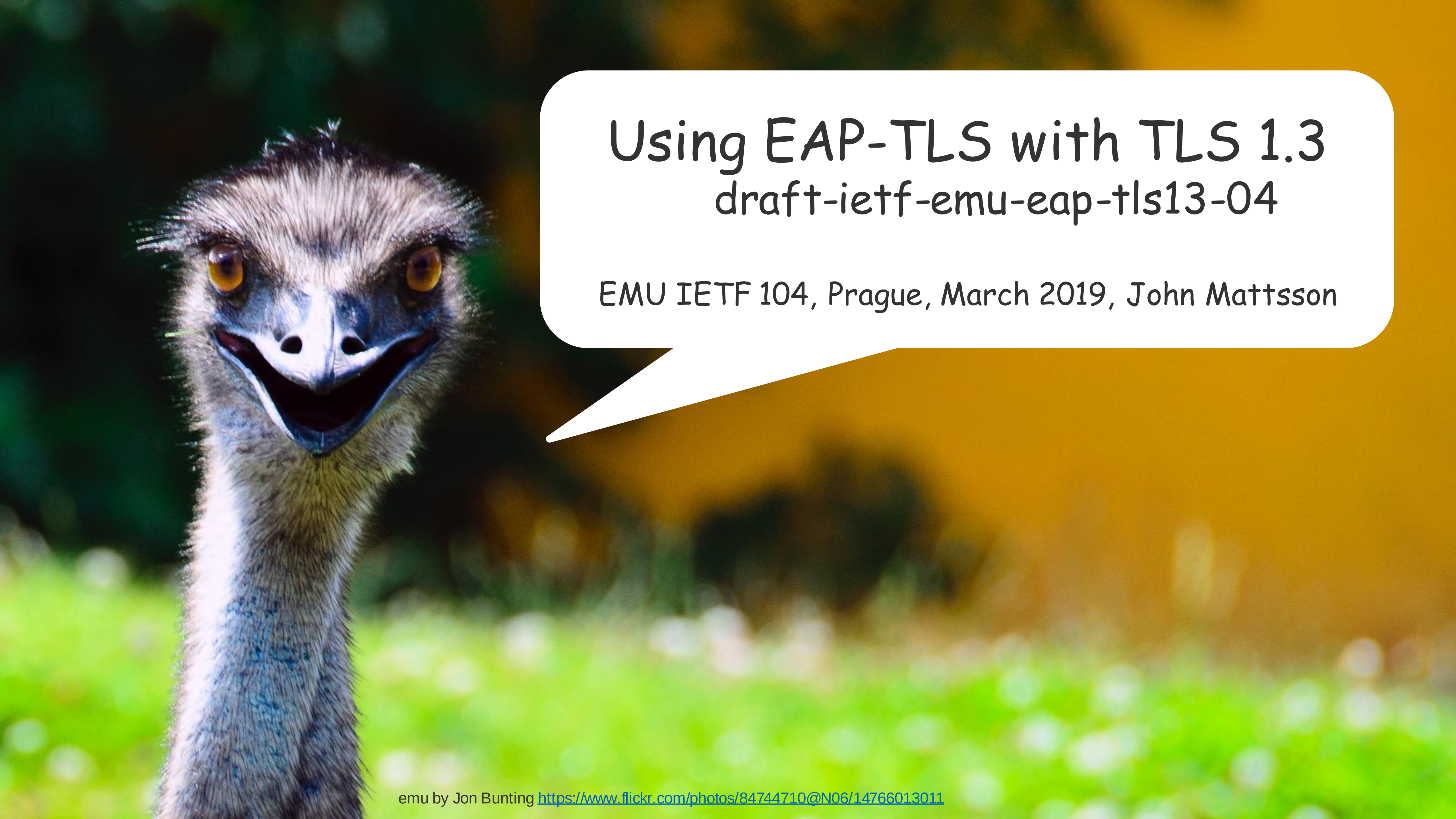
Using EAP-TLS with TLS 1.3

draft-ietf-emu-eap-tls13-04

EMU IETF 104, Prague, March 2019, John Mattsson



emu by Jon Bunting https://www.flickr.com/photos/84744710@N06/14766013011

DRAFT-IETF-EAP-TLS13-04

- **Changes between draft-ietf-emu-eap-tls13-02 and draft-ietf-emu-eap-tls13-03**
  - Pervasive monitoring: mandatory privacy protection of identities.
  - Differentiate between TLS fatal alerts and warning alerts.
  - Some reformulations and clarifications.

DRAFT-IETF-EAP-TLS13-04

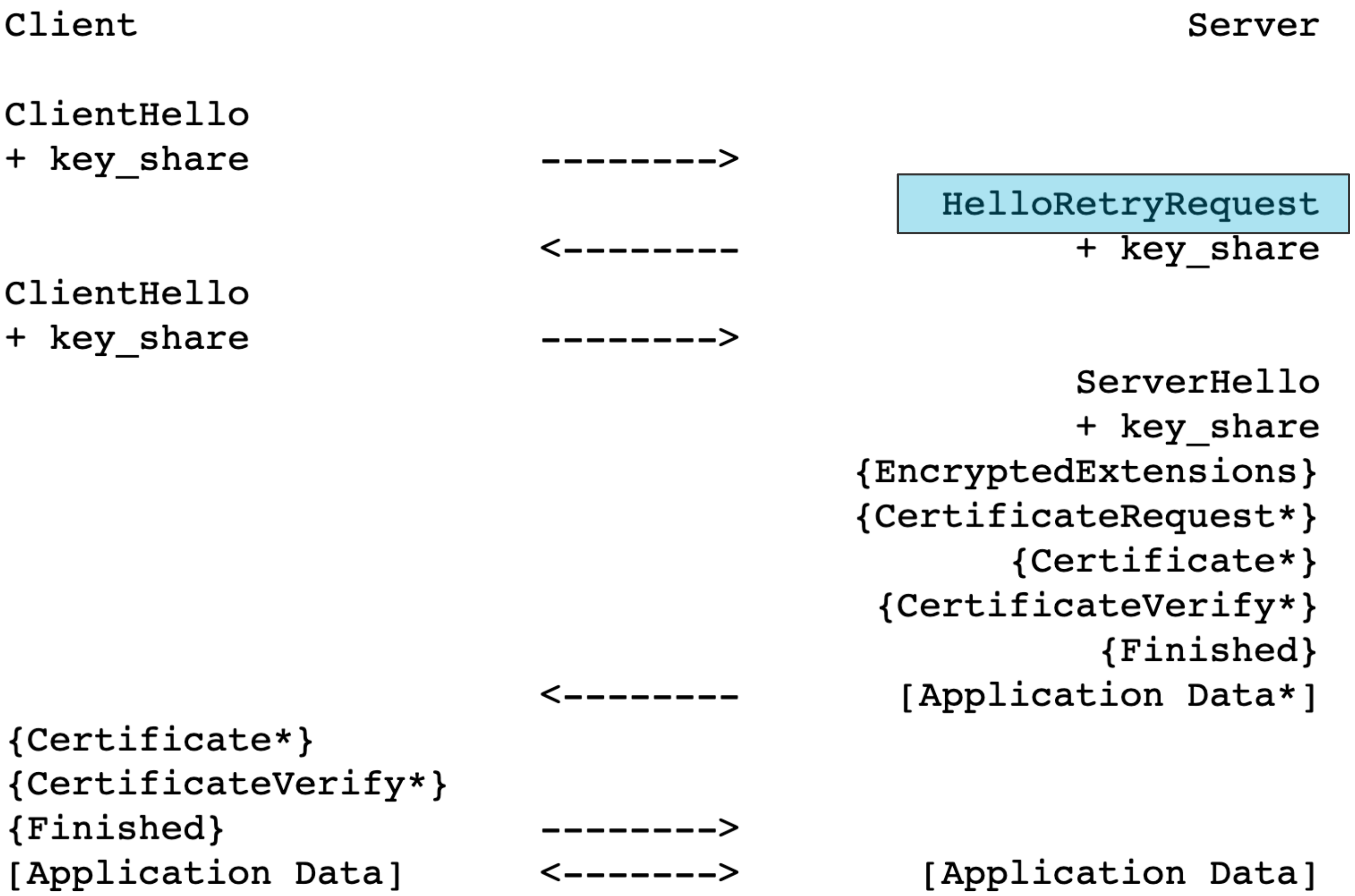
- **Changes between draft-ietf-emu-eap-tls13-03 and draft-ietf-emu-eap-tls13-04**
  - Borrowed the term privacy-friendly identities from RFC5448bis.
  - Figure describing EAP-TLS without peer authentication (e.g. emergency services)
  - Mandatory revocation checking and mandatory OSCP stapling.
  - Security considerations on discovered vulnerabilities, privacy, and revocation
  - **Clarifications on key derivation (as suggested by Alan DeKok).**
    - Type-Code, key length, other TLS based EAP methods, how to derive MSK, EMSK etc.
  - **Text on identity verification, authorization and resumption (draft text by Alan DeKok),**
  - **Clarification on unfragmented messages and the L bit (as suggested by Oleg Pekar).**
  - **Removed RFC 5216 requirement to not protect application data, early data still forbidden.**

UNFRAGMENTED MESSAGES AND THE L BIT

- Ambiguous in RFC 5216 (as pointed out by Oleg Pekar).
- EAP-TTLS (RFC) states that:
  - “Unfragmented messages MAY have the L bit set and include the length of the message (though this information is redundant).”
- No other RFCs or drafts discusses this.
- Some implementations reject unfragmented messages with the L bit set.
- https://github.com/emu-wg/draft-ietf-emu-eap-tls13
  - ”Implementations MUST NOT set the L bit in unfragmented messages, but MUST accept unfragmented messages with and without the L bit set.”

TLS HELLO RETRY REQUEST

- HelloRetryRequest is a new message type in TLS 1.3
- draft-ietf-emu-eap-tls13-04 does not say anything about HelloRetryRequest.
- Should probably mention it.
    - Early data?
    - Figure?



```
Client                                                  Server

ClientHello
+ key_share             -------->
                                           HelloRetryRequest
                        <--------                + key_share
ClientHello
+ key_share             -------->
                                                 ServerHello
                                                 + key_share
                                       {EncryptedExtensions}
                                       {CertificateRequest*}
                                              {Certificate*}
                                        {CertificateVerify*}
                                                  {Finished}
                        <--------        [Application Data*]
{Certificate*}
{CertificateVerify*}
{Finished}              -------->
[Application Data]      <------->         [Application Data]
```

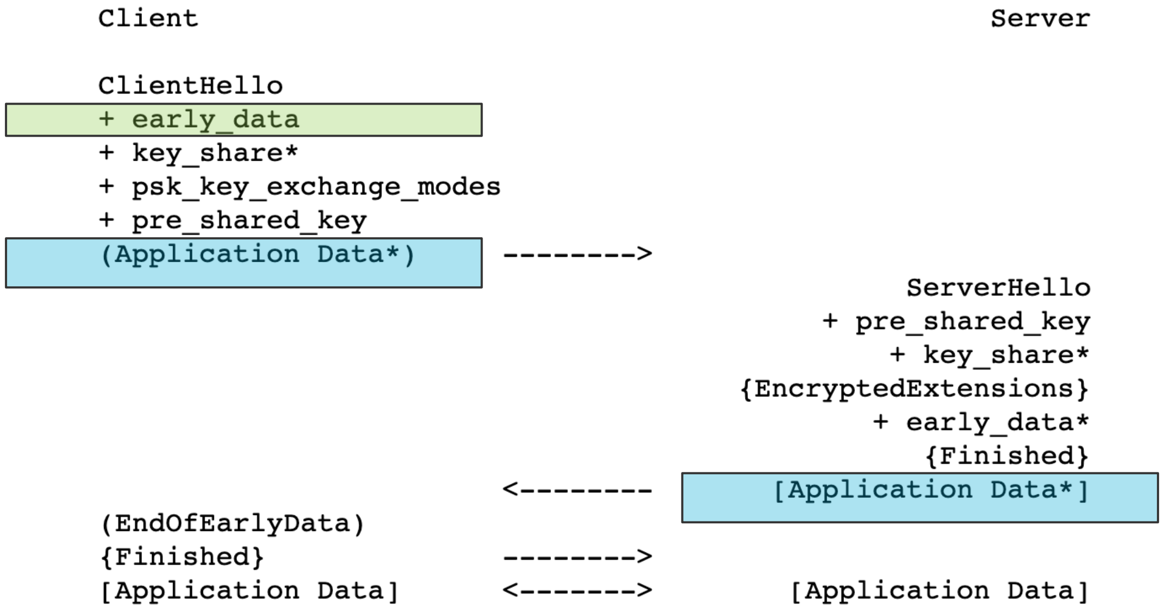
Figure 2: Message Flow for a Full Handshake with Mismatched Parameters

KEY DERIVATION

```
Type-Code    = 0x0D
Key_Material = TLS-Exporter("EXPORTER_EAP_TLS_Key_Material",
                            Type-Code, 128)
IV           = TLS-Exporter("EXPORTER_EAP_TLS_IV",
                            Type-Code, 64)
Method-Id    = TLS-Exporter("EXPORTER_EAP_TLS_Method-Id",
                            Type-Code, 64)
Session-Id   = Type-Code|| Method-Id
```

- "Other TLS-based EAP methods can perform similar key derivations by replacing the Type-Code with the value of their EAP type. The Type-Code is defined to be 1 octet for values smaller than 256, otherwise it is a 32-bit number (four octets), in network byte order. Additional discussion of other EAP methods is outside of the scope of this document."
- Extended types? `Session-Id = 0xFE || Vendor-Id || Vendor-Type || Method-Id`

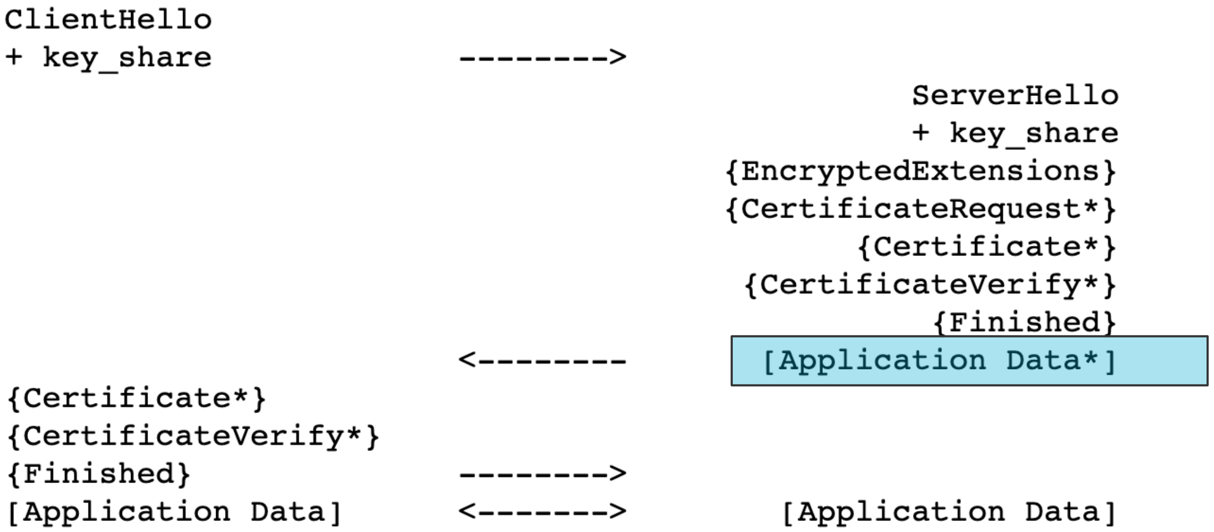
EARLY DATA



```
       Client                                               Server

       ClientHello
       + early_data
       + key_share*
       + psk_key_exchange_modes
       + pre_shared_key
       (Application Data*)     -------->
                                                       ServerHello
                                                  + pre_shared_key
                                                      + key_share*
                                             {EncryptedExtensions}
                                                     + early_data*
                                                        {Finished}
                               <--------       [Application Data*]
       (EndOfEarlyData)
       {Finished}              -------->
       [Application Data]      <------->        [Application Data]
```

0-RTT data resumption with early_data extension



```
ClientHello
+ key_share             -------->
                                                ServerHello
                                                + key_share
                                      {EncryptedExtensions}
                                      {CertificateRequest*}
                                             {Certificate*}
                                       {CertificateVerify*}
                                                 {Finished}
                        <--------       [Application Data*]
{Certificate*}
{CertificateVerify*}
{Finished}              -------->
[Application Data]      <------->        [Application Data]
```

Early data without extension

OTHER TLS-BASED EAP METHODS

- How should draft-ietf-emu-eap-tls13-04 deal with other TLS-based EAP methods? Division between draft-ietf-emu-eap-tls13-04 and draft-dekok-emu-tls-eap-types-00?
- **Key derivation**
  - “**Other TLS-based EAP methods can perform similar** key derivations by replacing the Type-Code with the value of their EAP type. The Type-Code is defined to be 1 octet for values smaller than 256, otherwise it is a 32-bit number (four octets), in network byte order. **Additional discussion of other EAP methods is outside of the scope of this document.**”
- **Protection of application data including early data**
  - “A server which receives an "early_data" extension **MUST ignore** the extension or respond with a HelloRetryRequest as described in Section 4.2.10 of RFC 8446.”
  - “**While EAP-TLS does not** protect any application data, the negotiated cipher suites and algorithms **MAY** be used to secure data as done in **other TLS-based EAP methods**.”
- **Cross method resumption?**

TEXT ON IDENTITY VERIFICATION, AUTHORIZATION AND RESUMPTION

- Involves several different sections:
  - `2.1.2 Resumption`
  - `2.2 Identity Verification`
  - `5.4 Revocation (Security Consideration)`
  - `5.6 Authorization (Security Consideration)`
  - `5.7 Resumption (Security Consideration)`
- Jim suggestion to move some text from security considerations to main body.
- “other authentication information” which information should be cached? And which information should draft-ietf-eap-tls13 discuss?
  - Alan suggested text discussing caching of information such as information about the authenticator, information about the EAP peer, or information about the protocol layers below EAP (MAC addresses, IP addresses, port numbers, WiFi SSID, etc.).
  - Jim suggest only caching information from the TLS handshake.

WANTED

REVIEWS

IMPLEMENTATIONS

INTEROP

WGLC…