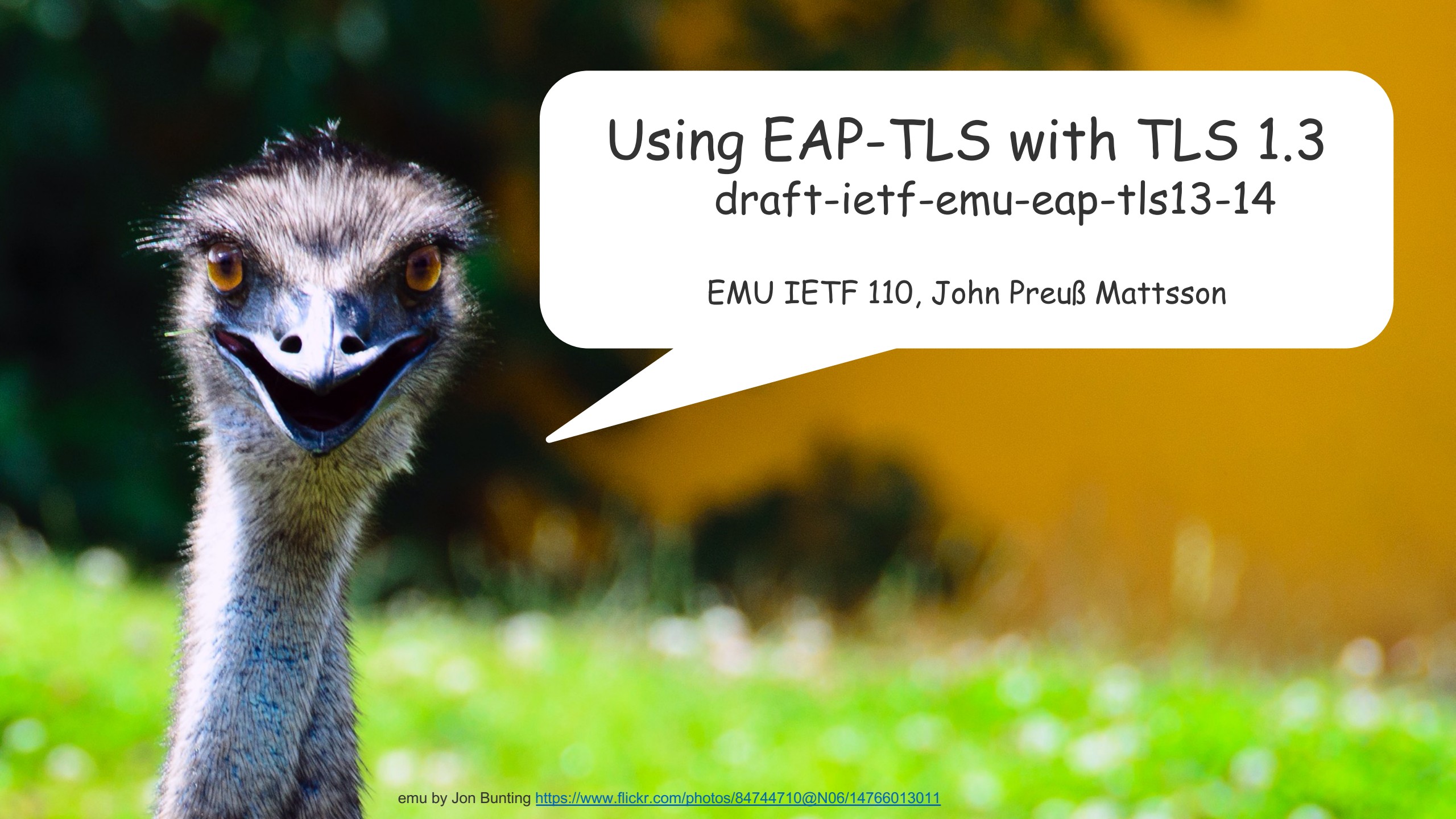

Using EAP-TLS with TLS 1.3

draft-ietf-emu-eap-tls13-14

EMU IETF 110, John Preuß Mattsson

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

Clarifications since draft-ietf-emu-eap-tls13-12

Clarifications between -12 and -14 was based on comments from IETF LC, directorates, IESG, TLS WG.

- Clarified that EAP-TLS 1.3 always provides forward secrecy.
- Clarified that all message flow are specific to TLS 1.3 and do not apply to TLS 1.2.
- Clarified what EAP-TLS means with “privacy” (peer username is not disclosed) and clarified that EAP-TLS 1.3 never disclose the peer identity.
- Clarified that in TLS 1.3, Error alerts are always fatal.
- Clarified that OCSP status cannot be sent for the trust anchor.
- Clarified maximum ticket lifetime and that servers MUST respect this maximum ticket lifetime.
- Clarified that client can use the ticket request extension to specify the desired number of tickets.
- Clarified what a TLS 1.3 ticket contains and that the PSK cannot be pre-computed when client auth is used.
- Clarified TLS 1.3 ticket issuing and use, and that ticket reuse enables an attacker to track the client.
- Clarified that server identity is only protected against passive attackers.
- Clarified that an implementation MAY enforce limited authorization before revocation checking has been done.
- Clarified that psk_dhe_ke is needed for forward secrecy.
- Clarified different ways to do revocation checking.
- Better examples of encrypted usernames.
- Corrected maximum length of TLS record.
- Corrected description of TLS 1.3 alerts (errors are always fatal and all other alerts are closure alerts)

Changes since draft-ietf-emu-eap-tls13-12

Changes between -12 and -14 was based on comments from IETF LC, directorates, IESG, TLS WG.

- Change from application data commitment message to close_notify alert
  - As close_notify cannot be together with Finished, some message flows have one more flight.
  - Note that in -14 the definition is still "commits to not send any more handshake messages"

- Modernized key derivation by using different labels for MSK and EMSK. No derivation of IVs. Removed specification of Enc-RECV-Key and Enc-SEND-Key which are defined elsewhere.

- Type-Code is not used as context but instead appended to the labels.

```
Type-Code  = 0x0D
MSK        = TLS-Exporter("EXPORTER_EAP_TLS_MSK_" + Type-Code,       "", 64)
EMSK       = TLS-Exporter("EXPORTER_EAP_TLS_EMSK_" + Type-Code,      "", 64)
Method-Id  = TLS-Exporter("EXPORTER_EAP_TLS_Method-Id_" + Type-Code, "", 64)
Session-Id = Type-Code || Method-Id
```

GitHub Issue #32

Exporter use of context

No Context:

```
Type-Code    = 0x0D
MSK        = TLS-Exporter("EXPORTER_EAP_TLS_MSK",  Type-Code, 64)
EMSK       = TLS-Exporter("EXPORTER_EAP_TLS_EMSK", Type-Code, 64)
Method-Id = TLS-Exporter("EXPORTER_EAP_TLS_Method-Id", Type-Code, 64)
Session-Id   = Type-Code || Method-Id
```

Context:

```
Type-Code  = 0x0D
MSK        = TLS-Exporter("EXPORTER_EAP_TLS_MSK_" + Type-Code,       "", 64)
EMSK       = TLS-Exporter("EXPORTER_EAP_TLS_EMSK_" + Type-Code,      "", 64)
Method-Id  = TLS-Exporter("EXPORTER_EAP_TLS_Method-Id_" + Type-Code, "", 64)
Session-Id = Type-Code || Method-Id
```

Review by Alan: "It defines key label prefixes, not key labels. Either the spec needs to be updated, or the IANA registry requirements needs to be updated. Preferably after consultation with the TLS WG."

Pull Request #42 and #44

- **Reference to ietf-emu-tls-eap-types**

  "This document specifies EAP-TLS 1.3 and does not specify how other TLS-based EAP methods use TLS 1.3. The specification for how other TLS-based EAP methods use TLS 1.3 is left to other documents such as [I-D.ietf-emu-tls-eap-types]."

  "Other TLS based EAP methods can use the TLS exporter in a similar fashion by replacing the EAP-TLS Type-Code with their own Type-Code (encoded as a hexadecimal string), see [I-D.ietf-emu-tls-eap-types]."

- **Message flows are example message flows**

  "All message flow are example message flows specific to TLS 1.3 and do not apply to TLS 1.2."

GitHub Pull Request #45

Some clarifications why resumption is needed

"NewSessionTicket messages (each associated ~~containing~~ a PSK, a PSK identity, a ticket lifetime, and other parameters)"

"The NewSessionTicket can be sent in the same flight as the TLS server Finished or later."

"EAP-TLS is typically used with client authentication and typically fragments the TLS flights into a large number of EAP requests and EAP responses. Resumption significantly reduces the number of round-trips and enables the EAP-TLS server to omit database lookups needed during a full handshake with client authentication."

GitHub Issue #48 and Pull Request #51

— **Guidance regarding NewSessionTicket**

"Servers should take into account that fewer newSessionTickets will likely be needed in EAP-TLS than in the usual HTTPS connection scenario. In most cases a single newSessionTicket will be sufficient. A mechanism by which clients can specify the desired number of tickets needed for future connections is defined in [I-D.ietf-tls-ticketrequests]."

Recent comment on the email list by Alan DeKok:
https://mailarchive.ietf.org/arch/msg/emu/_FAkvNlQN23bJ9i8WGulTTHl0Wk/

RFC 8446: *"Servers that issue tickets SHOULD offer at least as many tickets as the number of connections that a client might use; for example, a web browser using HTTP/1.1 [RFC7230] might open six connections to a server. Servers SHOULD issue new tickets with every connection. This ensures that clients are always able to use a new ticket when creating a new connection."*

"I suggest therefore that the draft be updated to reference RFC 8446 Section C.4 in respect to this topic. Also, that the draft require the EAP server sends only one session ticket. And that the EAP peer expect only one session ticket. And that if more than one session ticket is received by the EAP peer, that it chooses to save one and discards the rest."

GitHub Issue #36 and Pull Request #41

— **Packet Modification Attacks**

"As decscribed in [RFC3748] and Section 5.5 of [RFC5216], the only information that is integrity and replay protected in EAP-TLS are the parts of the TLS Data that TLS protects. All other information in the EAP-TLS message exchange including EAP-Request and EAP-Response headers, the identity in the identity response, EAP-TLS packet header fields, Type, and Flags, EAP-Success, and EAP-Failure can be modified, spoofed, or replayed. This section updates Section 5.5 of [RFC5216]."

"Protected TLS Error alerts are protected failure result indications and enables the EAP-TLS peer and EAP-TLS server to determine that the failure result was not spoofed by an attacker. Protected failure result indications provide integrity protection and replay but MAY be unauthenticated. Protected failure result does not significantly improve availability as TLS 1.3 treats most malformed data as a fatal error."

Text on protected success indications is missing. Based and Bernard/Mohit discussion, is seems a bit unclear what the text should say.

GitHub Issue #49 and Pull Request #50

Change terminology from Master to Main

- RFC8446bis updates the terminology for TLS 1.3 and TLS 1.2:
  - OLD "Master Secret"
  - NEW "Main Secret"
  - OLD "exporter_master_secret"
  - NEW "exporter_secret"

GitHub Issue #38 and Pull Requests #46

Must send Error alert

> "In TLS 1.3 [RFC8446], Error alerts are not mandatory to send after a fatal error condition. Failure to send TLS Error alerts means that the peer or server would have no way of e determining what went wrong. EAP-TLS 1.3 strengthen this requirement. Whenever an implementation encounters a fatal error condition, it MUST send an appropriate TLS Error alert."

The user_cancelled behavior needs to be clear. This is currently missing.

- user_cancelled behavior depends on protected indication choices (success and failure)
- user_cancelled behavior in TLS 1.3 is unclear. RFC 8446 is contradictory and and issue has been opened for RFC8446bis.

GitHub Issue #33 and PR #40 (EAP State Machine)

This is a new section when compared to [RFC5216] and only applies to TLS 1.3 (or higher). [RFC4137] offers a proposed state machine for EAP.

TLS 1.3 [RFC8446] introduces Post-Handshake messages. These Post-Handshake messages use the handshake content type and can be sent after the main handshake. Examples of Post-Handshake messages are NewSessionTicket, which is used for resumption and KeyUpdate, which is not useful and not expected in EAP-TLS. After sending TLS Finished, the EAP-TLS server may send any number of Post-Handshake messages in separate EAP-Requests.

To provide a protected success result indication and to decrease the uncertainty for the EAP-TLS peer, the following procedure MUST be followed:

When an EAP-TLS server has successfully processed the TLS client Finished and sent its last handshake message (Finished or a Post-Handshake), it commits to not sending any more handshake messages by sending a TLS close_notify alert. The TLS close_notify alert is a protected success result indication, as defined in [RFC3748]. Implementations following [RFC4137] sets the alternate indication of success variable altAccept after sending or receiving close_notify. After sending the TLS close_notify alert, the EAP-TLS server may only send an EAP-Success. The EAP-TLS server MUST NOT send an TLS close_notify alert before it has successfully processed the client finished and sent its last handshake message.

TLS Error alerts SHOULD be considered a failure result indication, as defined in [RFC3748]. Implementations following [RFC4137] sets the alternate indication of failure variable altReject after sending or receiving an Error alert. After sending or receiving a TLS Error alert, the EAP-TLS server may only send an EAP-Failure. Protected TLS Error alerts are protected failure result indications, unprotected TLS Error alerts are not.

The keying material can be derived after the TLS server Finished has been sent or received. Implementations following [RFC4137] can then set the eapKeyData and aaaEapKeyData variables.

The keying material can be made available to lower layers and the authenticator after the authenticated success result indication has been sent or received. Implementations following [RFC4137] can set the eapKeyAvailable and aaaEapKeyAvailable variables.

This section depends on protected indication choices (success and failure)

GitHub Issue #47

More information in key derivation and bidding down

Seems to be agreement that RFC 5216 requirements and guidance is lacking. Below is a recent suggestion from Joseph Salowey. No PR for this issue yet.

"The 5216 guidance here is lacking. I'm thinking we should add some things to the security considerations for this, something like

- *Peers should allow for configuring a unique trust root and name to match against SANs of type DNS name.*
- *List the problems if your trust roots are overly broad and you do not have host name matching*
- *Recommend that implementations provide ways to automate the provisioning of such names and trust roots.*
- *Perhaps say that implementation MAY implement a TOFU mechanism that allows a user to trust a particular certificate (I'm not sure how well this works in practice)"*

Alan DeKok Review of draft-ietf-emu-eap-tls13

Summary:

- The document says it defines EAP-TLS with TLS 1.3 (or higher). This is not true, and all references to "or higher" should be removed.
- It allows for multiple session tickets, going against the recommendations of RFC 8446, and against the consensus from implementors. It implicitly permits session tickets to be sent before the client certificate has been validated. These tickets are useless, and serve only to bloat the packet exchange.
- It adds "no peer authentication", but is silent on the security considerations of that change.
- It claimes to be "backwards compatible" with earlier versions of EAP-TLS, but does not explain how that is achieved.
- It discusses HelloRetryRequest, with no explanation as to how that affects EAP-TLS. From a reading of the document and RFC 8446, it appears that it doesn't.
- It should be updated to suggest that resumption uses the same EAP Identity as from the first full authentication. This suggestion solves a number of potential privacy and interoperability issues.
- It defines key exporters for other TLS-based EAP methods. The WG already has a document which covers this topic, accepted as a WG document prior to these changes being added to this document.
- It leaves significant portions of the protocol as implementation-defined.