

# A Proof of Concept Implementation and Evaluation of a Zero Configuration Authentication Option for EAP-TLS

**Carolin Latze, Ulrich Ultes-Nitsche**  
**University of Fribourg**

# Very Short Introduction Into EAP-TLS

- Authentication protocol for wired and wireless networks
- EAP = Extensible Authentication Protocol
- EAP-TLS = TLS in EAP
- X.509 TLS Certificates used for user authentication

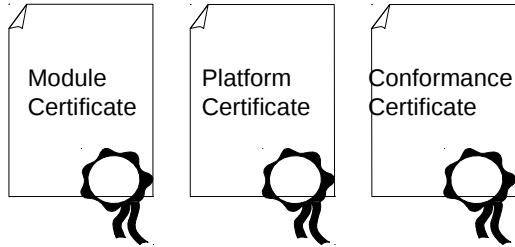
# TPM and AIKs

- TPM = trusted module in almost every new computer (hashing, signing, storage)
- AIK = Attestation Identity Key

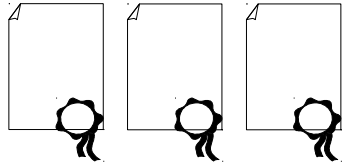
# AIK Retrieval in Detail



# AIK Retrieval in Detail



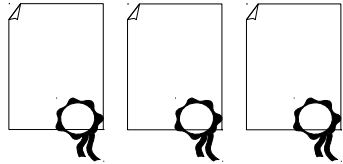
# AIK Retrieval in Detail



Generate new AIK



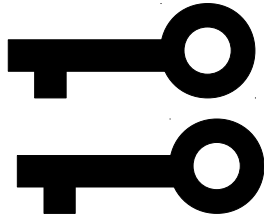
# AIK Retrieval in Detail



PCA

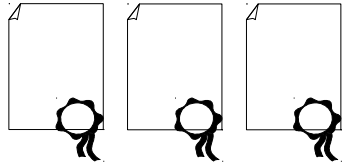


pubAIK/ label/ AIK(Hash(PCA,label),pubAIK)/  
Module, Platform, Conformance Certificate



$K(x)$  – x signed with key K  
 $K\{x\}$  – x encrypted with key K

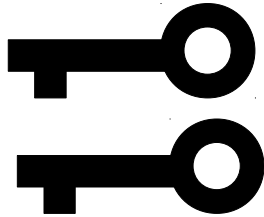
# AIK Retrieval in Detail



PCA



pubAIK/ label/ AIK(Hash(PCA,label),pubAIK)/  
Module, Platform, Conformance Certificate



symkey{AIK Cert}/pubek{symkey,Hash(pubAIK)}

$K(x)$  –  $x$  signed with key  $K$   
 $K\{x\}$  –  $x$  encrypted with key  $K$

# Protocol Model

*Client*  $= (X, \hat{Y})[$   
  *new*  $A_{\hat{X}};$   
  send  $\hat{X}, \hat{Y}, A_{\hat{X}}, \text{"label"},$   
   $SIG_{\bar{A}_{\hat{X}}}(HASH(\hat{Y}, \text{"label"}), A_{\hat{X}}), SIG_{TP\hat{M}E}(\hat{X}, E_{\hat{X}}),$   
   $SIG_{\hat{P}E}(\hat{X}, P_{\hat{X}}), SIG_{\hat{C}E}(\hat{X}, C_{\hat{X}});$   
  receive  $\hat{Y}, \hat{X}, \text{encaik}, \text{encsym};$   
  match  $\text{encsym} / ENC_{E_{\hat{X}}}(\text{symkey}, HASH(A_{\hat{X}}));$   
   $\text{symkey}, \text{hash} := \text{dec } ENC_{E_{\hat{X}}}(\text{symkey}, HASH(A_{\hat{X}})), \bar{E}_{\hat{X}};$   
  match  $\text{hash} / HASH(A_{\hat{X}});$   
  match  $\text{encaik} / ENC_{\text{symkey}}(SIG_{\hat{Y}}(\hat{X}, A_{\hat{X}}));$   
   $\text{cert} := \text{dec } ENC_{\text{symkey}}(SIG_{\hat{Y}}(\hat{X}, A_{\hat{X}})), \text{symkey}; ]_X$

*Server*  $= (Y)[$   
  receive  $\hat{X}, \hat{Y}, A_{\hat{X}}, \text{"label"}, \text{aiksig}, \text{certek}, \text{certp}, \text{certconf};$   
  match  $\text{aiksig} / SIG_{\bar{A}_{\hat{X}}}(HASH(K_{\hat{Y}}, \text{"label"}), A_{\hat{X}});$   
  match  $\text{certek} / SIG_{TP\hat{M}E}(\hat{X}, E_{\hat{X}});$   
  verify  $SIG_{TP\hat{M}E}(\hat{X}, E_{\hat{X}}), TP\hat{M}E;$   
  match  $\text{certp} / SIG_{\hat{P}E}(\hat{X}, P_{\hat{X}});$   
  verify  $SIG_{\hat{P}E}(\hat{X}, P_{\hat{X}}), \hat{P}E;$   
  match  $\text{certconf} / SIG_{\hat{C}E}(\hat{X}, C_{\hat{X}});$   
  verify  $SIG_{\hat{C}E}(\hat{X}, C_{\hat{X}}), \hat{C}E;$   
   $\text{aikc} := \text{sign}(\hat{X}, A_{\hat{X}}), \hat{Y};$   
  new  $\text{symkey};$   
  send  $\hat{Y}, \hat{X}, ENC_{\text{symkey}}(\text{aikc}), ENC_{E_{\hat{X}}}(\text{symkey}, HASH(A_{\hat{X}})); ]_Y$

# Correctness Proof

- Since  $X$  is honest, and since the predicates are ordered, the actions are ordered as well.

$$\begin{aligned} & \text{Send}(X, \hat{X}, \hat{Y}, A_{\hat{X}}, \text{"label"}, \text{SIG}_{\bar{A}_{\hat{X}}}(\text{HASH}(\hat{Y}, \text{"label"}), A_{\hat{X}}), \\ & \text{SIG}_{\text{TPME}}(\hat{X}, E_{\hat{X}}), \text{SIG}_{\text{PE}}(\hat{X}, P_{\hat{X}}), \text{SIG}_{\text{CE}}(\hat{X}, C_x) < \\ & \text{Receive}(X, \hat{Y}, \hat{X}, \text{ENC}_{\text{symkey}}(\text{SIG}_{\hat{Y}}(\hat{X}, A_{\hat{X}})), \text{ENC}_{E_{\hat{X}}}(\text{symkey}, \text{HASH}(A_{\hat{X}}))) \end{aligned}$$

# Correctness Proof

- Since X is honest, and since the predicates are ordered, the actions are ordered as well.
- Since sender and receiver are located on different machines and since X is honest, X knows – after receiving the AIK certificate – that there is some other entity who generated and send out the message.

$$\begin{aligned} & \theta_{zero}[\text{receive } \hat{Y}, \hat{X}, \text{encaik}, \text{encsym}; \\ & \text{match } \text{encsym} / \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})); \\ & \text{match } \text{encaik} / \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}})); ]_X \\ & \text{Receive}(X, \hat{Y}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})), \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) \supset \\ & \text{Send}(Z, \hat{Z}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})), \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) \wedge \\ & (\text{Send}(Z, \hat{Z}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})), \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) < \\ & \text{Receive}(X, \hat{Z}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})), \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}})))) \end{aligned}$$

# Correctness Proof

- Since X is honest, and since the predicates are ordered, the actions are ordered as well.
- Since sender and receiver are located on different machines and since X is honest, X knows – after receiving the AIK certificate – that there is some other entity who generated and send out the message.
- If an entity Z is able to encrypt data with a key K, it needs to have the data as well as the key K.

$$\begin{aligned} \text{Has}(Z, ENC_{E_{\hat{X}}}(symkey, HASH(A_{\hat{X}}))) &\equiv \text{Has}(\hat{Z}, symkey) \wedge \\ \text{Has}(\hat{Z}, HASH(A_{\hat{X}})) \wedge \text{Has}(\hat{Z}, E_{\hat{X}}); \\ \text{Has}(Z, HASH(A_{\hat{X}})) &\equiv \text{Has}(\hat{Z}, A_{\hat{X}}); \\ \text{Has}(Z, ENC_{symkey}(SIG_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) &\equiv \text{Has}(\hat{Z}, symkey) \wedge \text{Has}(\hat{Z}, A_{\hat{X}}); \end{aligned}$$

# Correctness Proof

- Since X is honest, and since the predicates are ordered, the actions are ordered as well.
- Since sender and receiver are located on different machines and since X is honest, X knows – after receiving the AIK certificate – that there is some other entity who generated and send out the message.
- If an entity Z is able to encrypt data with a key K, it needs to have the data as well as the key K.
- If an entity has symkey, it either created or decrypted it. The latter implies that another entity created and sent it.

$$\begin{aligned} Has(\hat{Z}, symkey) &\equiv New(Z, symkey) \vee \\ Decrypts(Z, ENC_{E_{\hat{X}}}(symkey, HASH(A_{\hat{X}}))) &\supset Z = X \vee \exists Z. New(Z, symkey) \end{aligned}$$

# Correctness Proof

- Since sender and receiver are located on different machines and since did not send those messages, the entity cannot be X.

$$\begin{aligned} & \theta_{zero}[\text{receive } \hat{Y}, \hat{X}, \text{encaik}, \text{encsym}; ]_X \\ & \text{Honest}(\hat{X}) \wedge \\ & \text{Receive}(X, \hat{Y}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(\text{symkey}, \text{HASH}(A_{\hat{X}})), \text{ENC}_{\text{symkey}}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) \supset \\ & Z \neq X \end{aligned}$$

# Correctness Proof

- Since sender and receiver are located on different machines and since did not send those messages, the entity cannot be X.
- Since Y generates symkey and sends it only encrypted (decryptable by X only), the entity must be Y.

$$\begin{aligned} & \text{Honest}(\hat{X}) \wedge \text{Honest}(\hat{Y}) \supset \\ & \exists Z. \text{Has}(Z, \text{ENC}_{E_{\hat{X}}}(\text{symkey}, \text{HASH}(A_{\hat{X}}))) \wedge \\ & \text{Has}(Z, \text{ENC}_{\text{symkey}}(\text{SIG}_{\hat{Y}}(\hat{X}, A_{\hat{X}}))) \wedge \\ & \text{Send}(Z, \hat{Z}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(\text{symkey}, \text{HASH}(A_{\hat{X}})), \text{ENC}_{\text{symkey}}(\text{SIG}_{\hat{Y}}(\hat{X}, A_{\hat{X}}))) \wedge \\ & Z = Y \end{aligned}$$

# Correctness Proof

- Since sender and receiver are located on different machines and since did not send those messages, the entity cannot be X.
- Since Y generates symkey and sends it only encrypted (decryptable by X only), the entity must be Y.
- Since X is honest, it received the message from Y only after Y encrypted and sent it.

$$\begin{aligned} & \text{Honest}(\hat{X}) \wedge \text{Honest}(\hat{Y}) \supset \\ & \text{Send}(Y, \hat{Y}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})), \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) < \\ & \text{Receive}(X, \hat{Y}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(symkey, \text{HASH}(A_{\hat{X}})), \text{ENC}_{symkey}(\text{SIG}_{\hat{Z}}(\hat{X}, A_{\hat{X}}))) \end{aligned}$$

# Correctness Proof

- Since sender and receiver are located on different machines and since did not send those messages, the entity cannot be X.
- Since Y generates symkey and sends it only encrypted (decryptable by X only), the entity must be Y.
- Since X is honest, it received the message from Y only after Y encrypted and sent it.
- Since the message sent by Y requires Y to process data sent by X, Y has a valid sequence too.

$$\begin{aligned} & \text{Honest}(\hat{X}) \wedge \text{Honest}(\hat{Y}) \supset \\ & \text{Receive}(Y, \hat{X}, \hat{Y}, A_{\hat{X}}, \text{SIG}_{TPME}(\hat{X}, E_{\hat{X}}), \text{SIG}_{PE}(\hat{X}, P_{\hat{X}}), \text{SIG}_{CE}(\hat{X}, C_{\hat{X}})) < \\ & \text{Send}(Y, \hat{Y}, \hat{X}, \text{ENC}_{E_{\hat{X}}}(\text{symkey}, \text{HASH}(A_{\hat{X}})), \text{ENC}_{\text{symkey}}(\text{SIG}_{\hat{Y}}(\hat{X}, A_{\hat{X}}))) \end{aligned}$$

# Correctness Proof

- Since sender and receiver are located on different machines and since did not send those messages, the entity cannot be X.
  - Since Y generates symkey and sends it only encrypted (decryptable by X only), the entity must be Y.
  - Since X is honest, it received the message from Y only after Y encrypted and sent it.
  - Since the message sent by Y requires Y to process data sent by X, Y has a valid sequence too.
- 
- The certificate request protocol is correct.
    - ^ The client is communicating with the PCA.
    - ^ The messages sent by the PCA are meant for the client.

# EAP-TLS with AIK Certificates

- Trusted certificate retrieval without the need for user interaction
- Allow to run the certificate requesting during the TLS handshake after the server announced his acceptable CAs

# Correctness Proof

## TLS Invariants

- No other protocol should create and send out the *finished* message.
- The TLS *secret* is not transmitted over a hidden channel.

# Correctness Proof TLS Invariants

- No other protocol should create and send out the *finished* message.
- The TLS *secret* is not transmitted over a hidden channel.
- Not affected by certificate request protocol.

# Implementation

- GnuTLS handshake callbacks
- Webserver as PCA with REST API

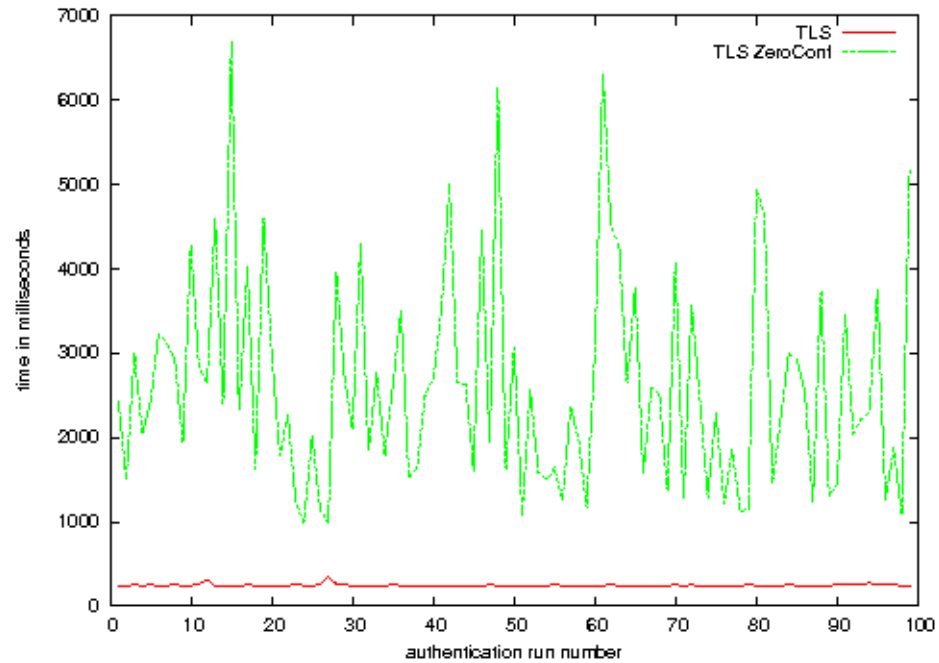
```
$CLIENTREQ="zero-request.pem";  
$CLIENTCERT="zero-cert.pem";  
$CACERT = "ca.pem";  
$CAKEY = "ca-key.pem";
```

```
$target_path="/";  
$target_path = $target_path . basename($_FILES['uploadedfile']['name']);
```

```
if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path)) {  
    $cmd="certtool --generate-certificate --load-request ".$CLIENTREQ." --outfile ".$CLIENTCERT."  
        --load-ca-certificate ".$CACERT." --load-ca-privkey ".$CAKEY." --template certtool.cfg";  
    system($cmd);  
    header('Content-type: application/octet-stream');  
    header('Content-Disposition: attachment; filename="zero-cert.pem"');  
    readfile('zero-cert.pem');
```

```
} else{  
    echo "There was an error uploading the file, please try again!";  
}
```

# Evaluation



# Questions?

**Thanks for your attention.**