Enrollment over Secure Transport

Max Pritikin, Cisco

Peter Yee, AKAYLA

What is EST?

- Simple X.509 certificate enrollment protocol using existing tools:
  - PKI (CMC)
  - HTTP
  - TLS
- Specifies both client and server elements

EST Functions

- Limited functionality
  - Enroll
  - Re-enroll/renew
  - Get CA certificates
  - Server-side key generation
  - “Full” CMC

TLS Features

- Client-certificate authentication when possible
  - Client authorization can use TLS credentials
- Server authentication by cert or manually
  - Server authorization varies by authentication scheme and source of server URI
- Channel binding to assist proof-of-possession for client private key
- Naturally provides confidentiality and integrity

HTTP Features

- HTTP client authentication as a fallback
- Existing content types for conveying requests and responses
  - Derived from CMC and RFC 5280
- Individual URIs for EST function selection

Distribution of CA Certs

- Returns a bag of CA certs
  - Minimum of current root CA certificate
  - May include CMP-defined cert set for CA rollover
  - Must include any certs needed to form chain if using a subsidiary CA

Enroll/Re-Enroll Certificate

- Simple CMC messages (PKCS#10)
  - True of both Enroll and Re-Enroll/Renew
  - Re-enrollment affects TLS usage only
- Returns PKIX certificate

Full CMC Support

- Transports for CMC messages when simple CMC cert requests are insufficient
- No special services provided

Server-side Key Generation

- Latest addition, so not fully integrated in draft
- Allows server to supply private and public keys in response
- Client-supplied public key ignored
  - Only present owing to use of CMC/PKCS#10 request format

… and Running Code

- Three implementations in progress
- Reference implementation targeted for open source

… and Running Code

- Three implementations in progress
- A reference implementation targeted for open source:

  Disclaimers. Simplistic. Not for production.

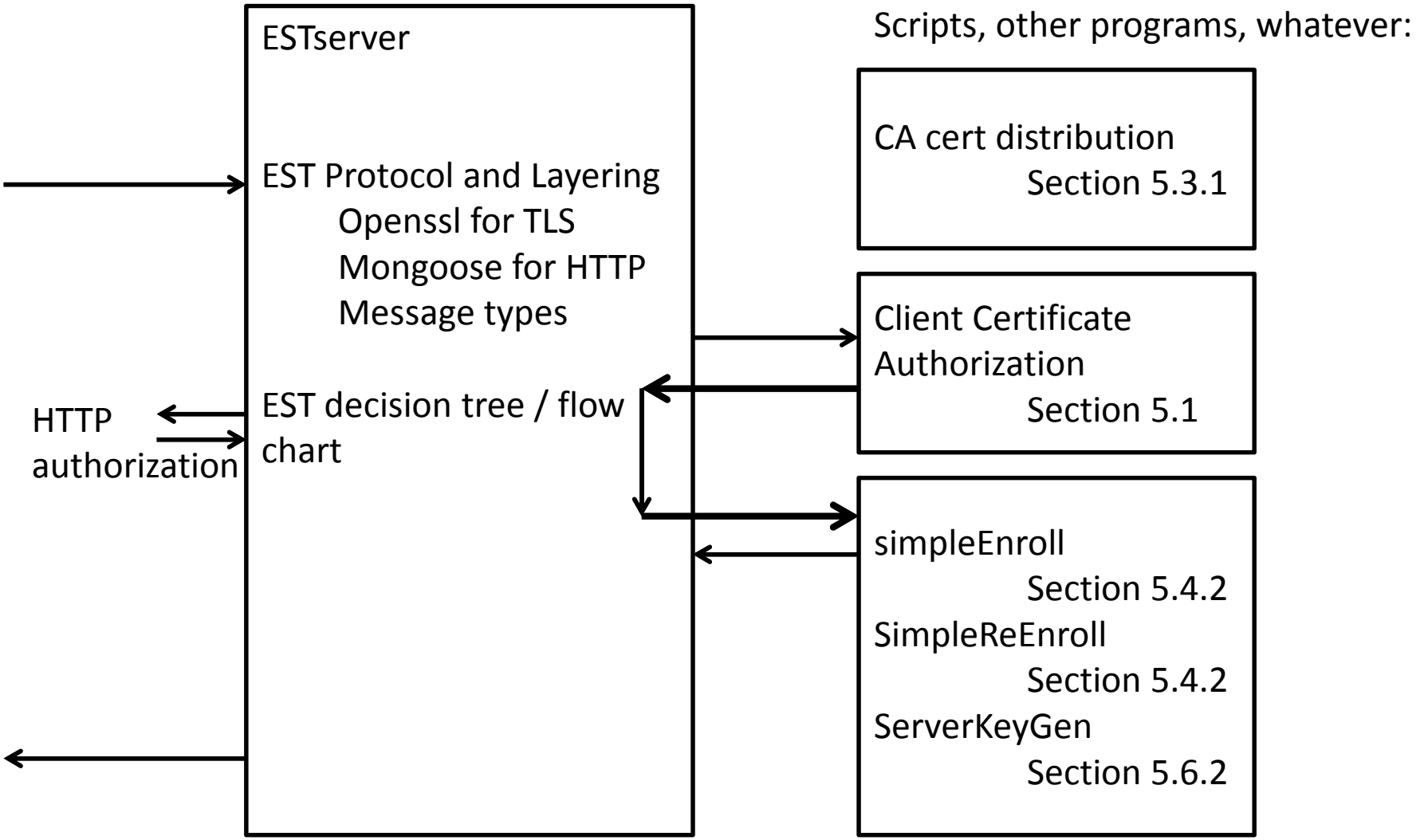
  "Sample minimal CA application" – openssl ca

  Server uses "mongoose" (MIT license)

  Client uses 'curl' and 'c-code' (libcurl)

  For Section 4.5 (Linking Identity and POP information)

Designed for experimentation



Server Scripts

- Replace these with whatever you want
- Reference implementation uses Bash
  - Which in turn uses openssl commands
- Environment variables passed in include
  - Client certificate
  - TLS_UNIQUE_SR value
  - Request
  - Username/password
- Anything sent to stdout goes to client
  - Think simple:“CGI”

Client

A really simple implementation:

$CURLCMD https://$ESTSRVR:$ESTPORT/$ESTURL
--anyauth –u usr:pwd
--cacert $EST_DEMOCLIENT_CACERT
--data-binary @$EST_DEMOCLIENT_REQ
-H "Content-Type: application/x-est-pkcs10”

And for Section 4.5:

./estclient enroll –u https://$ESTSRVR:$ESTPORT/$ESTURL
-p estuser:estpwd
-c $EST_DEMOCLIENT_CACERT
-R ESThandlerC_genp10.sh

Script is called to generate the actual request

Environment variable contains the TLS_UNIQUE_SR

Current Status Caveats

Expected to be made available ASAP

4.2.3 HTTP-based client authentication	Currently global (handled by server not callout script)
4.3.1.1 TLS-Based server authentication	Using a pinned server certificate
4.3.1.2 TLS-Based client authentication	Currently assumed
4.5 Linking Identity and POP information	One big switch to make this mandatory or optional (no per client switches)
5.1 Client authorization	Need to cleanup environment variables passed to authorization script
5.2 Server authorization	Client assumes authorization (w/o checking RA authorization values)
5.3.1 Distribution of CA certs	CA key rollover incomplete
5.6.1 Server-side Key Generation (experimental)	Works but the response format isn't compliant yet

Questions?