

MashSSL - Extending SSL for securing mashups

Siddharth Bajaj
VeriSign Inc.

sbajaj@verisign.com

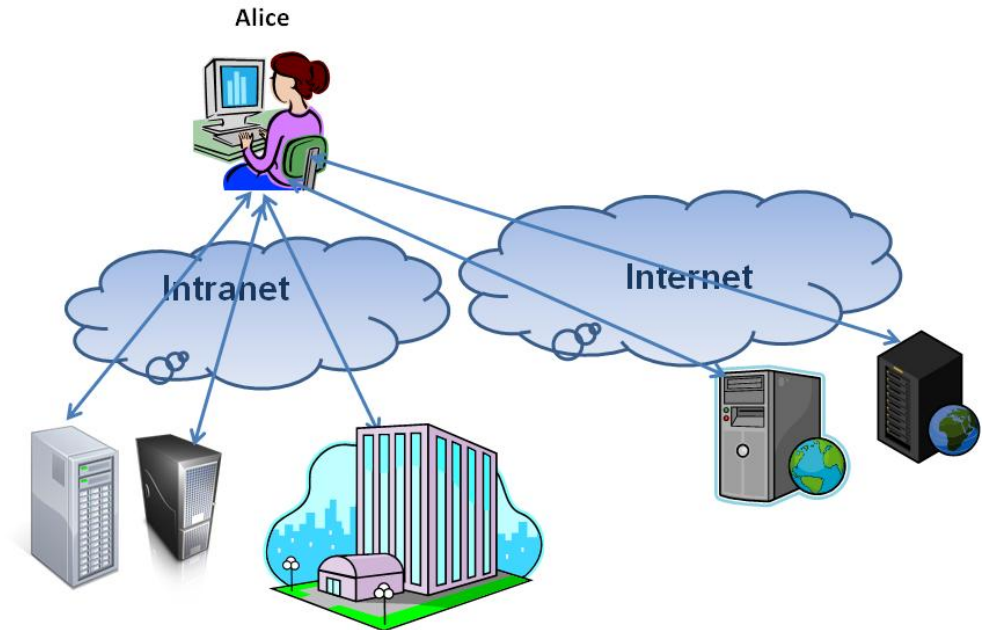
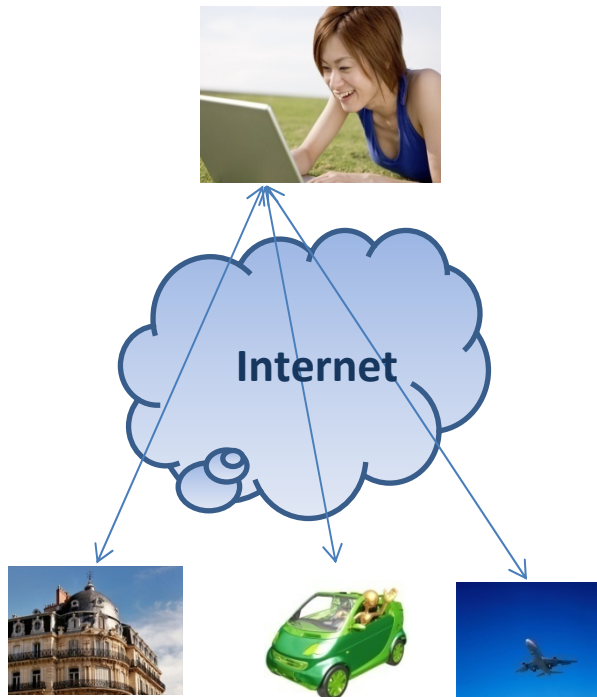
www.verisign.com

info@mashssl.org

www.mashssl.org

- Motivation for a new security protocol
- Why start with SSL/TLS?
- Introducing MashSSL
 - ▶ How MashSSL and TLS work together
- Bringing it all together – some examples
- Towards standardization...
- Q&A

What are mashups?



Mashups are composite applications that combine data and/or functionality from multiple sources to deliver rich functionality. Are gaining traction in both consumer and enterprise markets.

The buzz about mashups

Mashups are hot...

- Gartner Group: *"By 2010, Web mashups will be the dominant model (80 percent) for the creation of composite enterprise applications."*
- Forrester Research: *"Mashups will mature and eat into other major markets".*
- Tim Berners-Lee: *"It's about creating a seamless web of all the data in your life."*

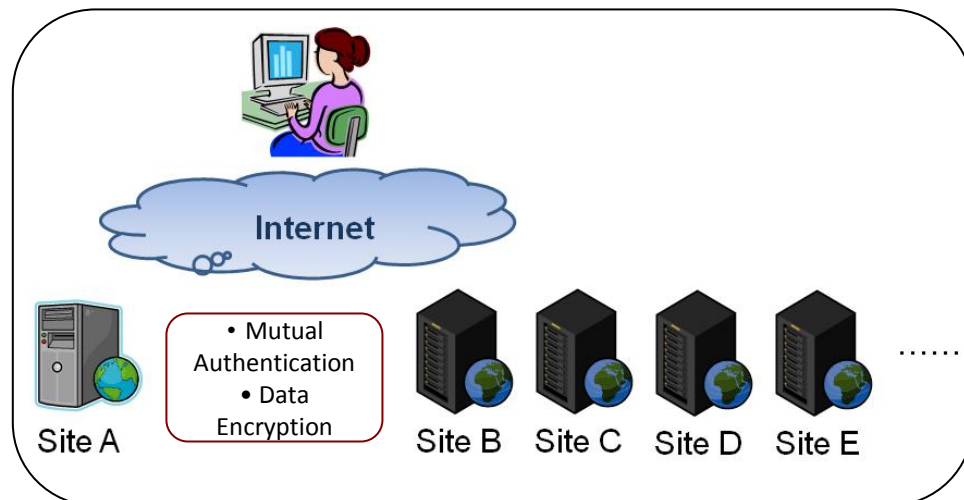
"Mashups are the most interesting development in the last 20 years", Douglas Crockford, Architect, Yahoo!

But...

- Gartner Group: *"The potential for security risks increases as more business users morph into application developers by building mashups."*
- KPMG: *"Survey of 472 executives found that half of them viewed security problems as a limiting factor in the uptake of Web 2.0-type tools in the enterprise."*

"Mashups are insecure. Mashups must not have access to any confidential information or any privileged connections to any servers. Until this is fixed, mashups are constrained to only trivial applications." Douglas Crockford, Architect, Yahoo!

Motivation for MashSSL

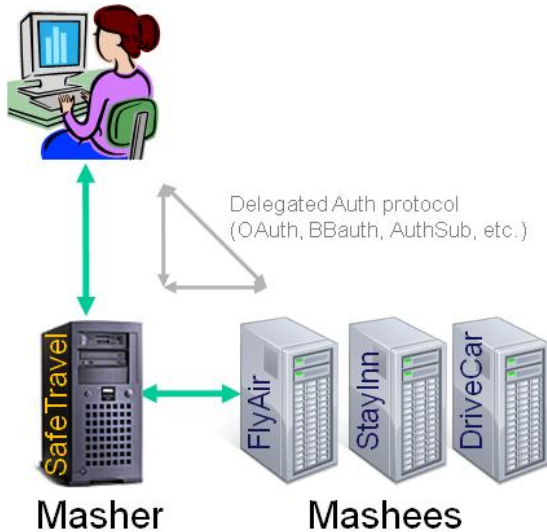


- Site A “talks” through Alice to various other sites...
 - Site B provides a payment button
 - Site C acts as an Identity Provider to Site A
 - Site D is an OAuth Service Provider to Site A
 - Site E wants to accept cross domain XHR from Site A
 - And so on...
- Web experiences are increasingly mashups!

- But Alice could be Evil Eve! How do sites authenticate each other via browser?
- Today for each problem different underlying crypto protocols are ginned up and credentials distributed. Site A has crypto s/w and credential management headache!

Would be nice if we could build standard secure pipe from site to browser to site.

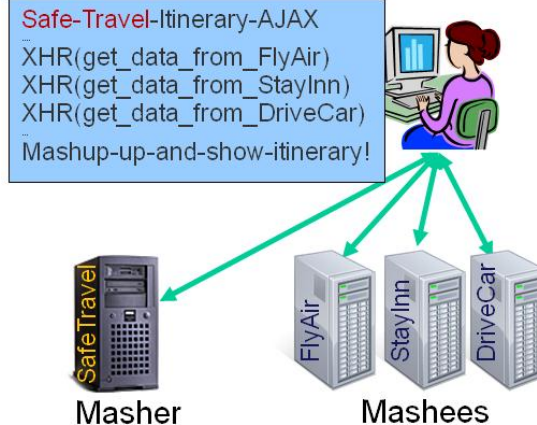
Deconstructing mashups



- In *server side mashups* the user authorizes the masher to get her data from multiple sources and serve it to her mashed.
- Industry moving towards standard called OAuth to avoid user having to give up her credentials.

All the parties in the mashup need to either provision or obtain OAuth credentials and manage them. And they have to repeat expensive cryptographic operations for each mashup.

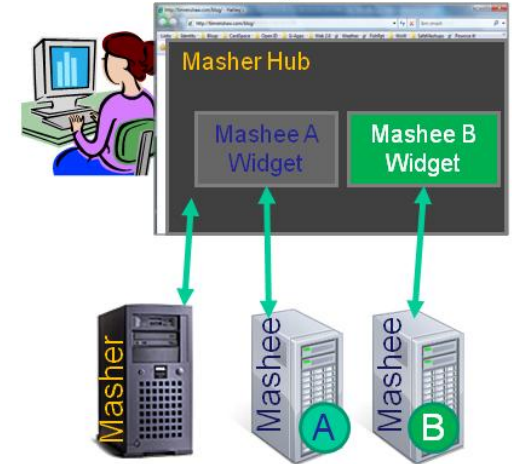
Using Cross Domain XHR



- For good security reasons XHR was restricted by the Same Origin Policy.
- But the new cross domain XHR specification allows for creation of client side mashups.
- The mashees examine ORIGIN header and their Access Control List (ACL) to allow or deny access.

Does not protect the business. ORIGIN header can easily be spoofed. Managing ACLs can be cumbersome. Additional new cryptography and new credentials are likely required to layer in real trust, as cross domain XHR can otherwise be very dangerous.

Hub & Widget Based



- This architecture is also gaining popularity, e.g. OpenAjax.
- Microsoft is working on approaches to “sandbox” widgets
- IBM has contributed ways to police inter-widget communication.
- Google’s CAJA provides ways of writing safer widgets.

All these efforts have merit and this line of work in increasing mashup security is important and must continue. But they do not address the issue of WHERE a widget came from. Joe the CSO would like to ensure his users are only downloading widgets from trusted partners.

Why start with TLS?

■ Pros:

- ▶ New protocols take years to mature.
- ▶ TLS handshake is probably the world's most carefully studied mutual authentication and key exchange protocol.
- ▶ Very efficient: Only initial handshake needs PKI processing.
- ▶ Trust infrastructure exists (get and manage SSL certificates)

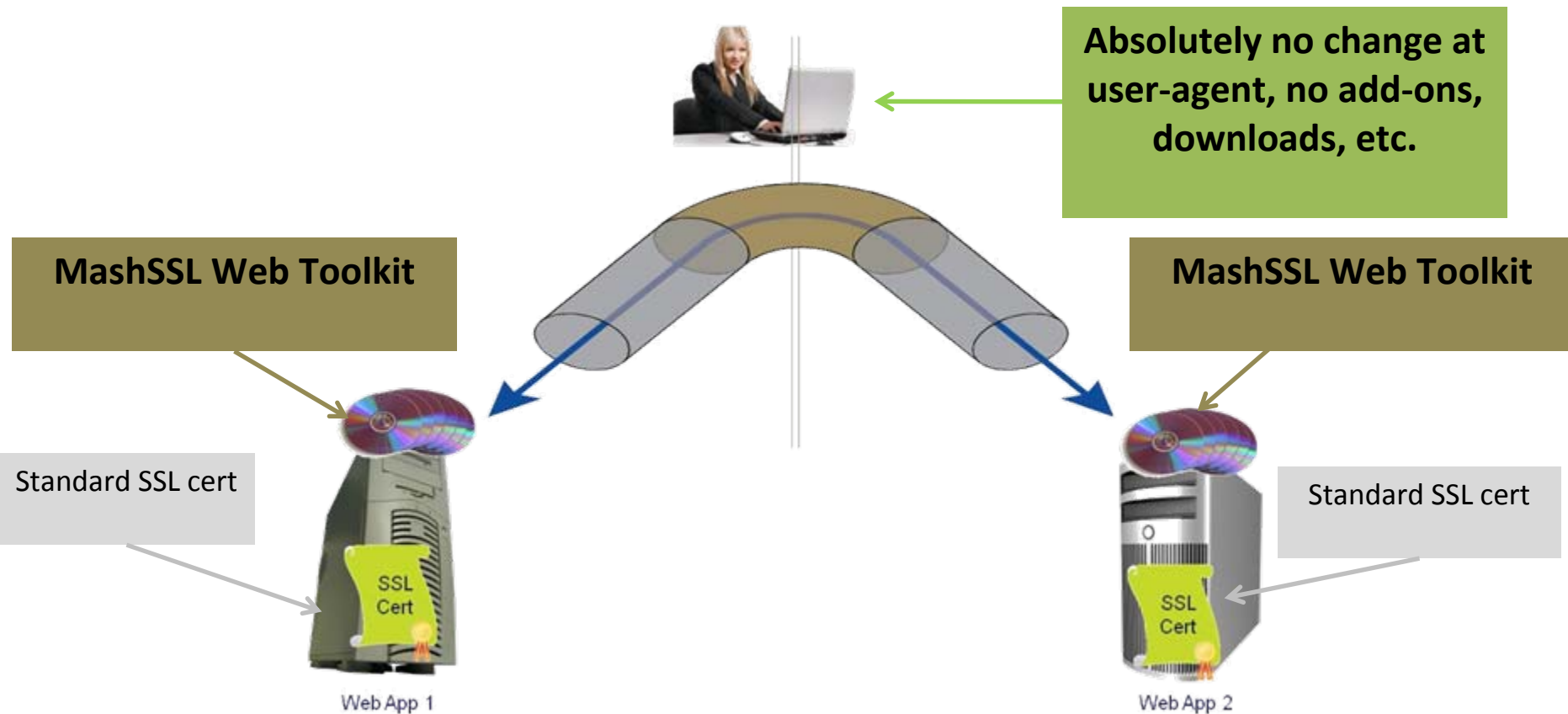
■ Cons:

- ▶ TLS does not allow MITMs. Our problem is multi-party.
- ▶ TLS usually runs over TCP. For us the "transport" is HTTP.
- ▶ TLS (for very good reason) is large and complex. World seems to want simple RESTful protocol...

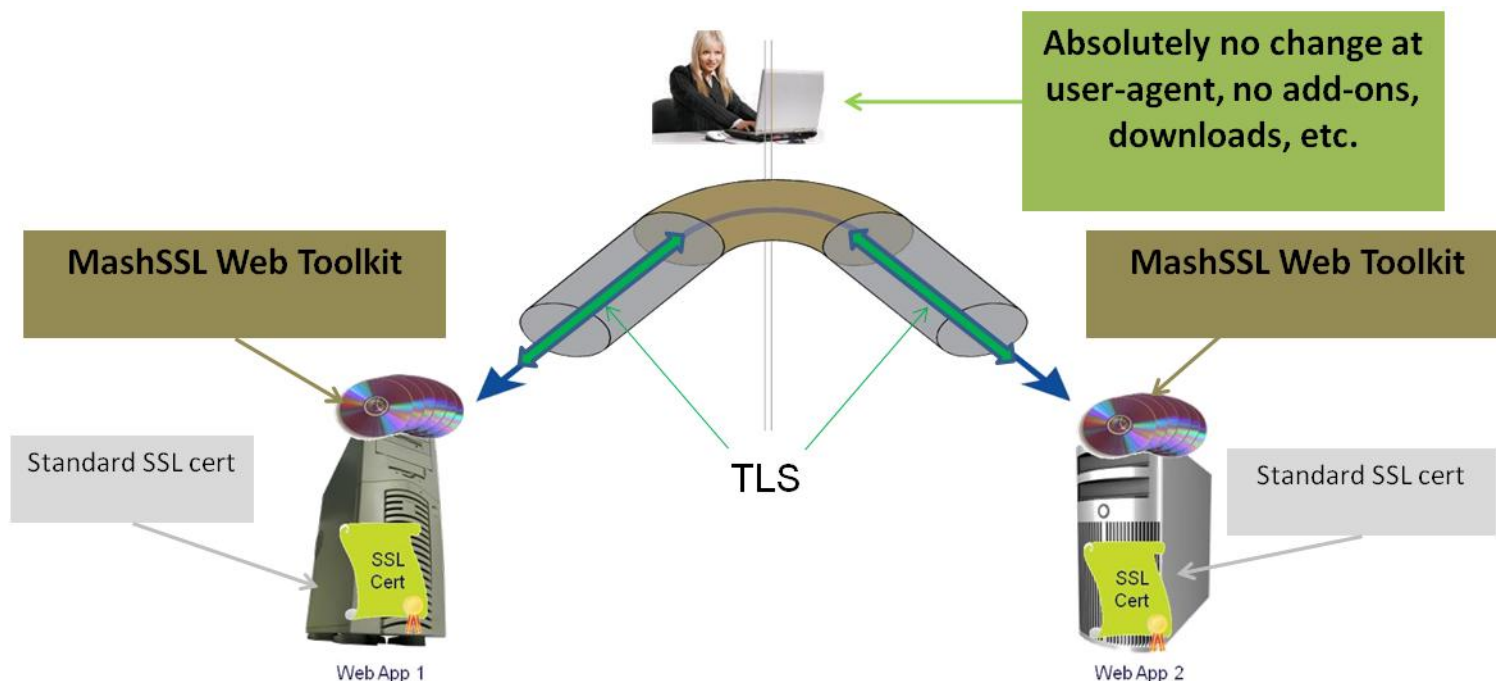
Very powerful "pros" suggest we start with TLS, and try and address the "cons".

- Introduces concept of “friend in the middle” to make TLS multi-party.
- Takes RFC 5246 handshake messages and exchanges them as name value pairs over HTTP via browser.
- Only implements core subset of TLS (e.g. no renegot!) to keep it simple.
- Adding optimization to make abbreviated handshake two step.
 - <http://www.ietf.org/mail-archive/web/tls/current/msg05728.html>
- In general never defines something already defined in RFC 5246 and simply points to it.

MashSSL in a picture



Result: Secure pipe between two web apps over which any “mashup protocol” can be run.







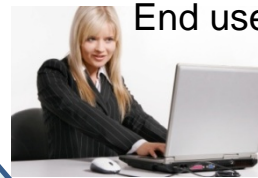
■ Likely scenario:

- ▶ Two independent TLS sessions over TCP pipes
- ▶ Over which are two independent HTTP sessions
- ▶ Over which is overlaid a MashSSL pipe
- ▶ On top of which one can run OAuth, OpenID, cdXHR, etc.

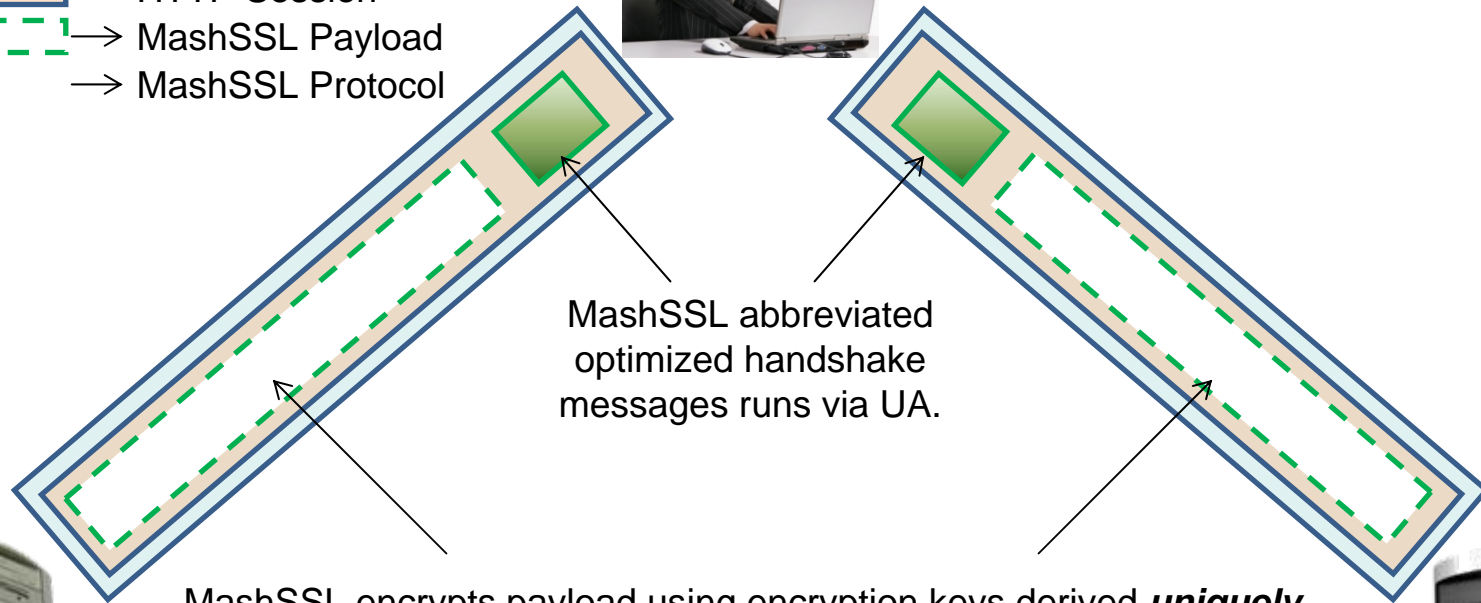
MashSSL: Actors and Layers

Legend

-  → (opt) SSL Pipe
-  → HTTP Session
-  → MashSSL Payload
-  → MashSSL Protocol



End user at User Agent (UA)



MashSSL encrypts payload using encryption keys derived **uniquely** for each session from master_secret and abbrev. handshake params.



Web App 1



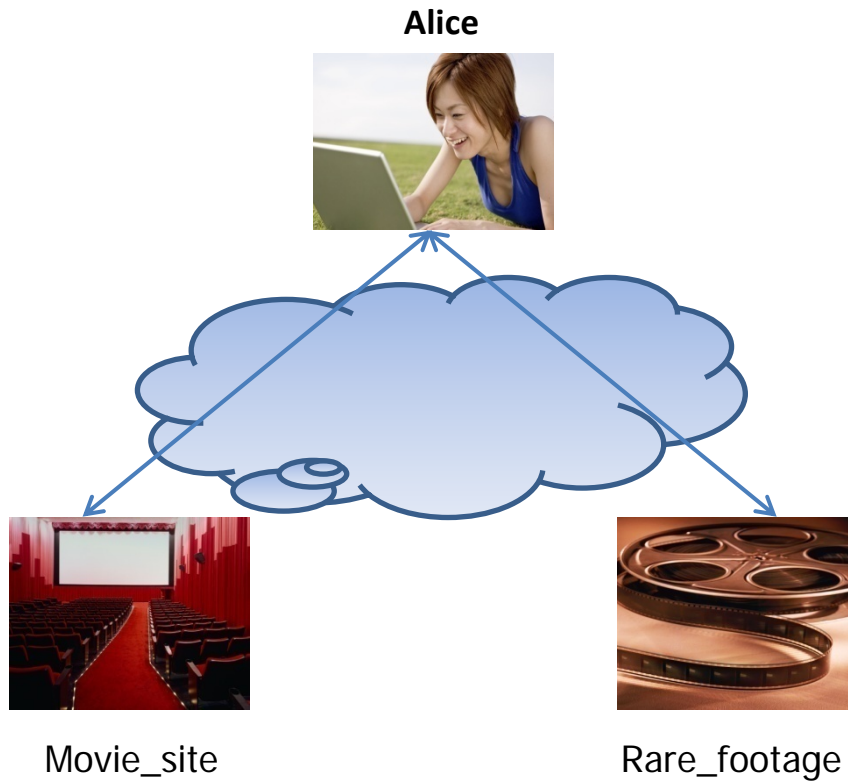
MashSSL full handshake runs directly between Web App 1 and Web App 2. Uses long term SSL certificate and private key to exchange short term "master_secret".



Web App 2



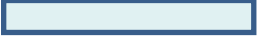




e.g. MashSSL for Cross Domain XHR

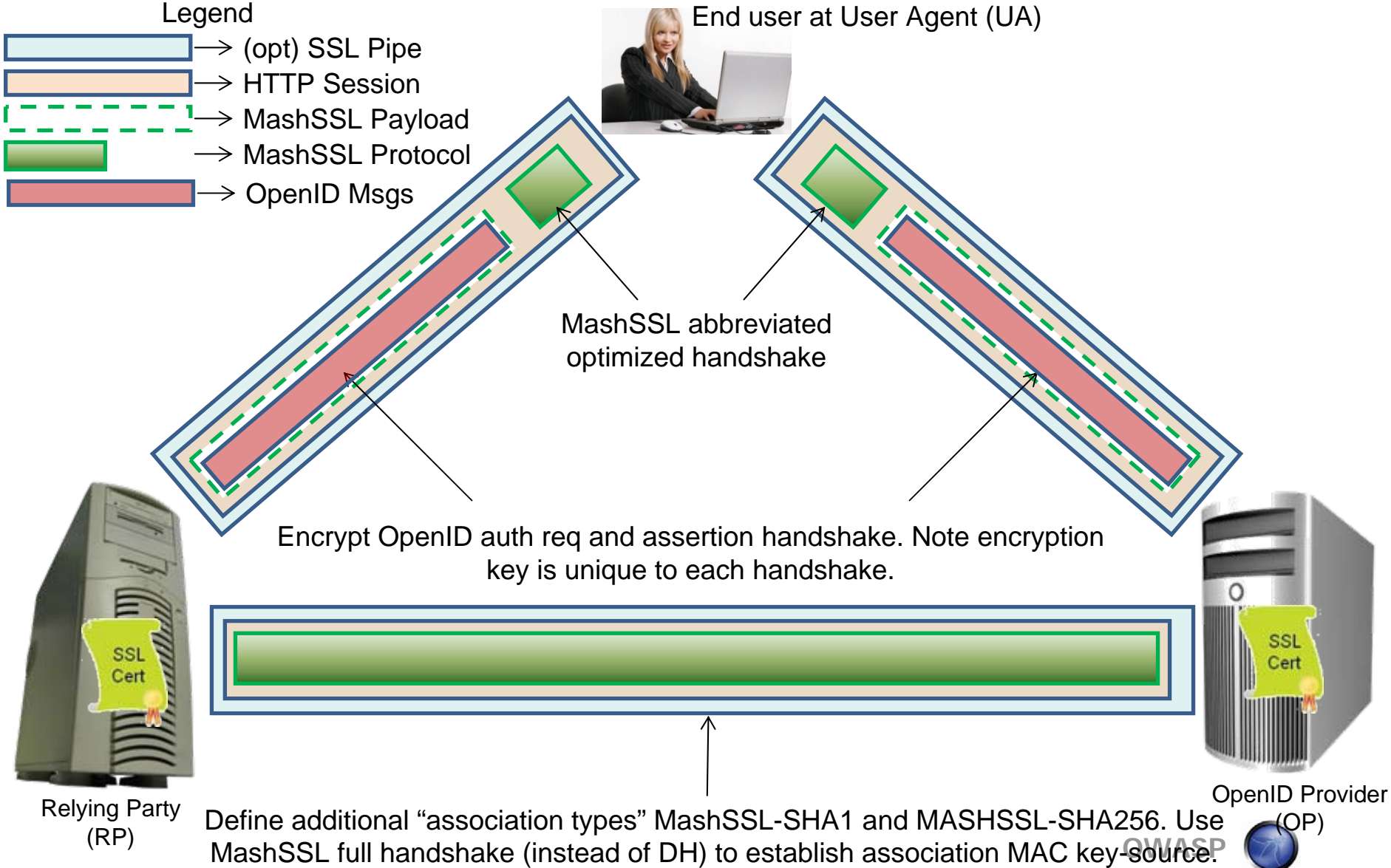


- Alice is on “Movie_site”
- Alice clicks on “Rare_footage” link. Cross domain XHR invoked.
- “Rare_footage” runs MashSSL with “Movie_site” via Alice’s browser.
- Once “Rare_footage” is convinced it is dealing with a valid business partner, it returns appropriate response in ACL and browser allows transaction.
- Note browser not trusted – “Rare_footage” is the policy enforcement point.

e.g. OpenID over MashSSL

Legend

-  → (opt) SSL Pipe
-  → HTTP Session
-  → MashSSL Payload
-  → MashSSL Protocol
-  → OpenID Msgs



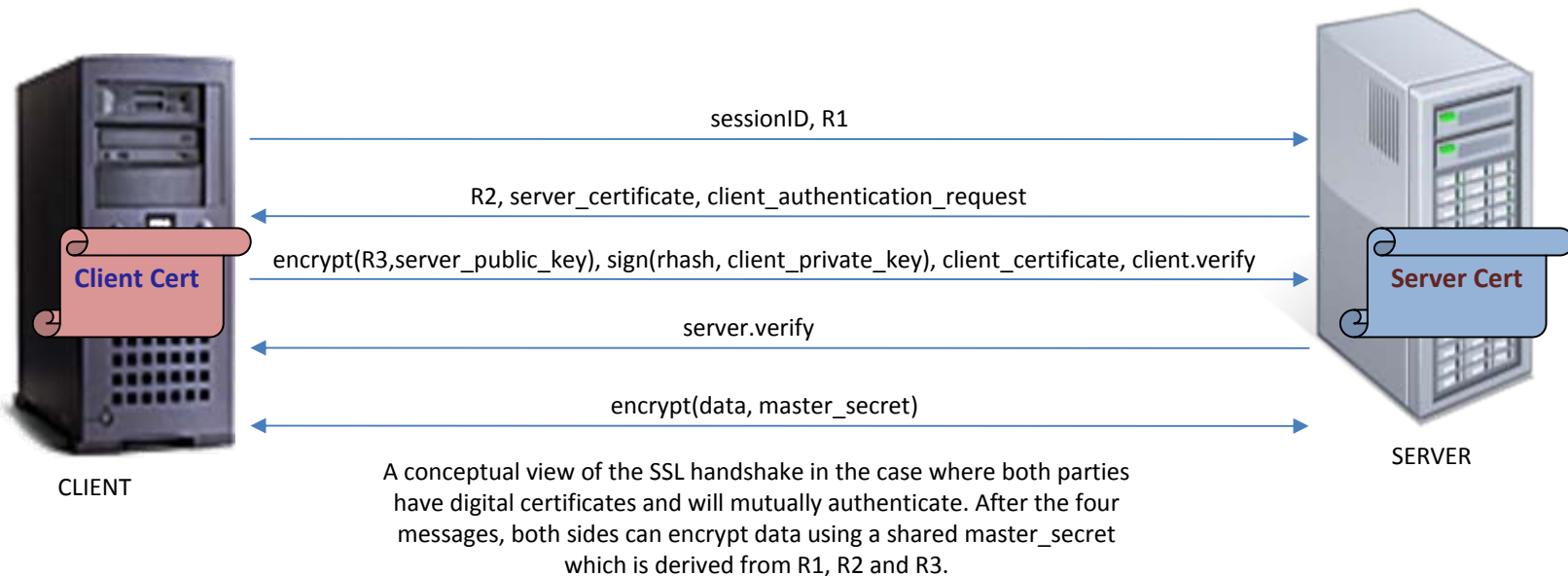
How does MashSSL stack up?

-  1. Single solution for all situations where problem manifests.
 - *MashSSL is a fundamental Internet building block that has countless uses.*
-  2. Lightweight RESTful application level protocol (HTTP).
 - *Standard defined in simple RESTful fashion.*
-  3. No new crypto protocol please. Takes up to a decade to build trust.
 - *Reuses SSL. Reuses whatever authentication is in place for scrambling.*
-  4. Trust browser as little as possible.
 - *Browser cannot spoof either web application!*
-  5. Don't ask us to get and manage new credentials from new authorities.
 - *Standard SSL certificates can be used.*
-  6. Don't use user authentication as proxy for B2B authentication.
 - *Web applications authenticating each other (through browser)*
-  7. Think scale. Do not repeat expensive PKI operations.
 - *Reuses SSL abbreviated handshake to avoid repeating PKI operations.*

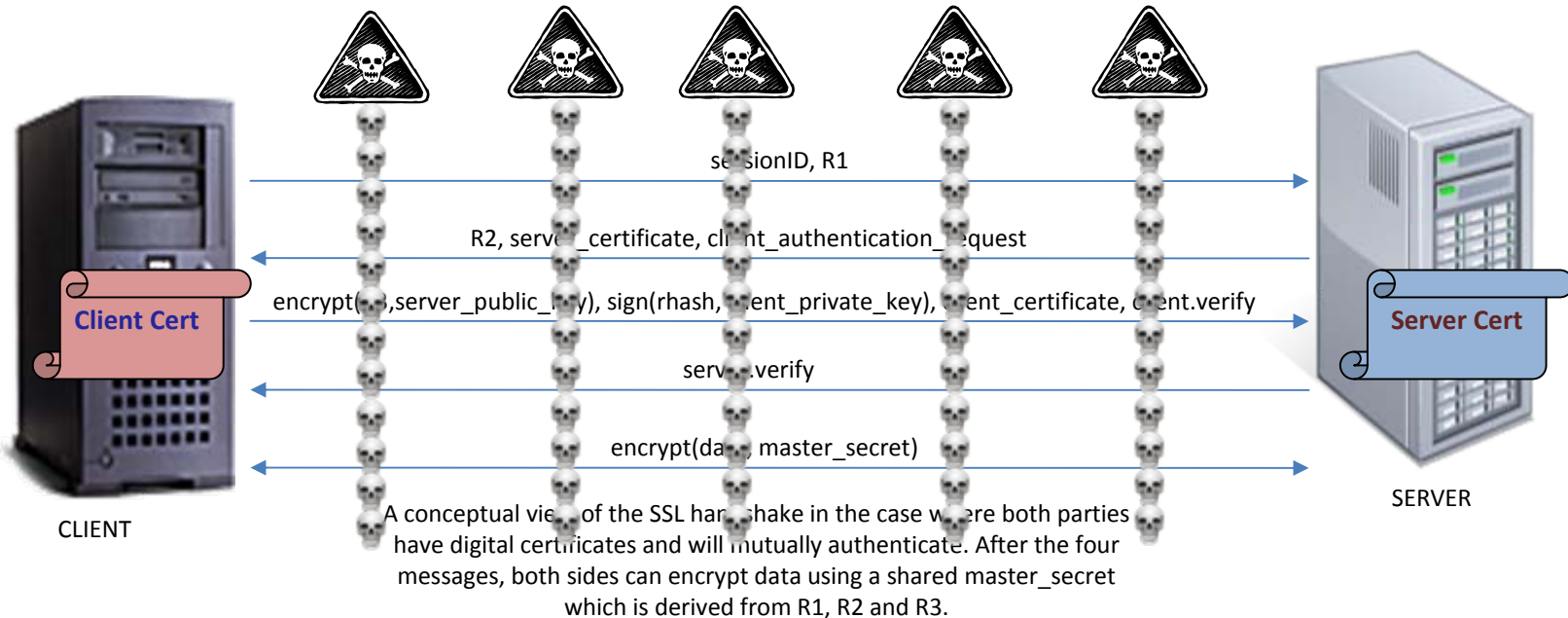
- For current specification and open source software please visit MashSSL Alliance site.
 - ▶ <http://www.mashssl.org>
- W3C Incubator:
 - ▶ <http://www.w3.org/2005/Incubator/MashSSL>
 - ▶ Final Report in a few weeks.
- Welcome participation from all!



Thank You!

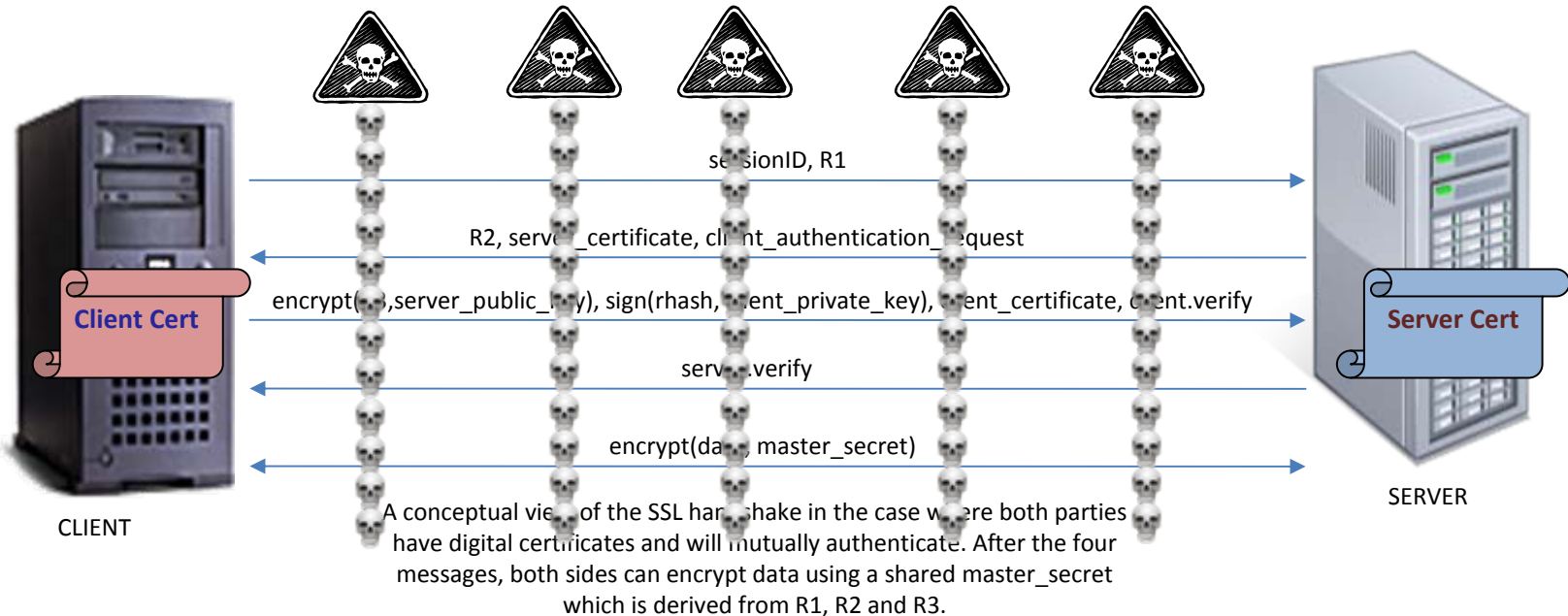


SSL protocol is widely used and trusted. Ability to reuse session without repeating PKI operations. Trust infrastructure CAs already exist. And is constantly being improved (TLS 1.2, EV certs). Many mashup apps already have SSL certs!



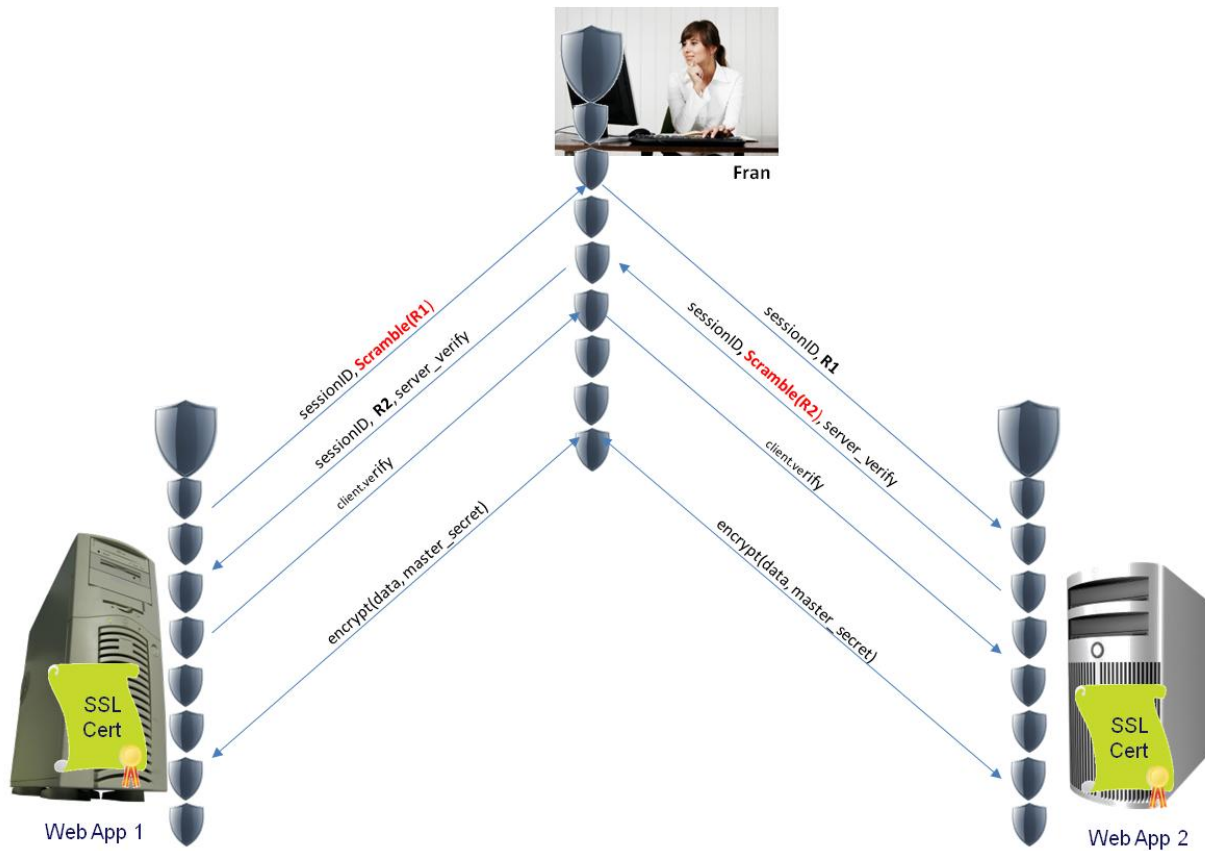
And, it is believed that the protocol is secure even in the presence of one or more MITM(s). **One of the strongest claims one can make of a crypto protocol!**

Lets ask for the impossible...



- Can we insert MITMs that manipulate protocol messages, but which
1. Allow successful execution of protocol
 2. Do not compromise underlying security of protocol

The very surprising non-intuitive answer is: YES WE CAN!



And, if they subsequently do mashup for another user Fran they simply use the SSL abbreviated handshake which avoids repeating the PKI operations. (In practice we will use optimized handshake and encrypted payload will accompany initial message.)