UNIVERSITY OF SURREY

Secure Modular Password Authentication for the Web using Channel Bindings

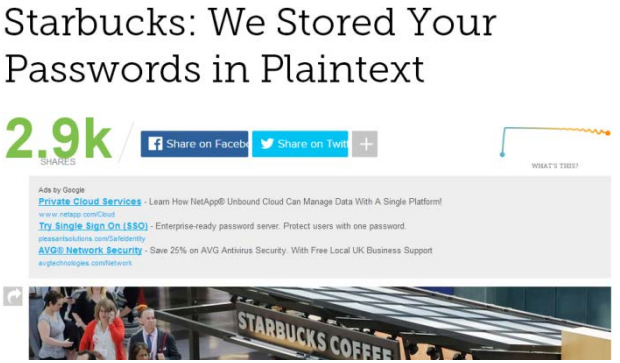
Mark Manulis, Douglas Stebila, Nick Denham

SSR 2014@RHUL December 17, 2014

General motivation: Passwords & Threats

Passwords

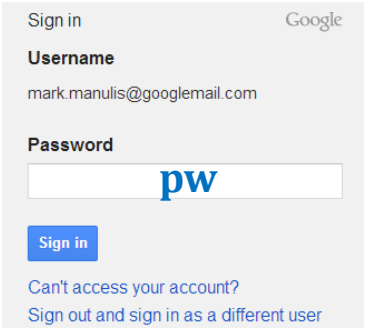
- “easy to remember” but “hard to guess”
- most widespread authentication method on the Web
- has become very popular with mobile apps

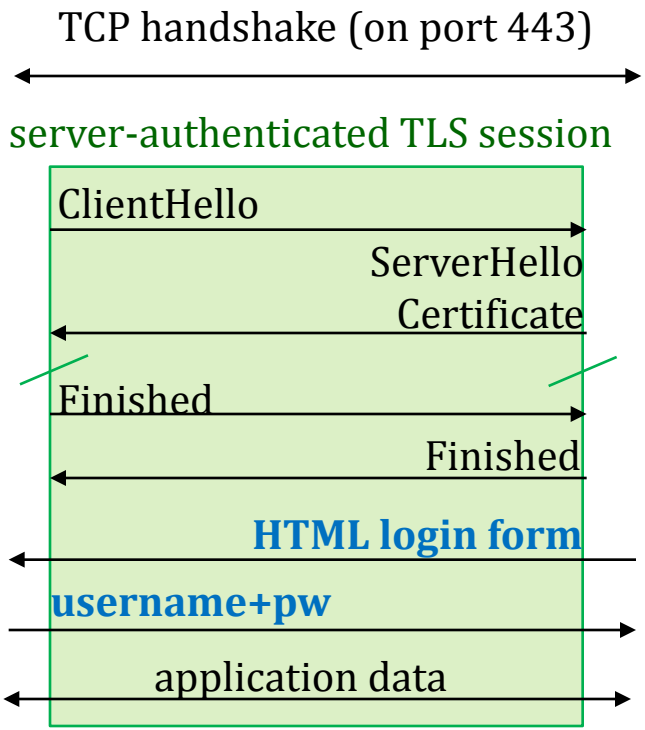


Threats

- online guessing attacks
- stolen password databases from web servers
- improperly stored passwords
  - without randomized hashing (PBKDF2, bcrypt)
- offline dictionary attacks on weak password
  - password crackers, brute-force, rainbow tables
- social engineering attacks
  - phishing and co

Current approach: HTML forms-over-TLS





X.509 certificate

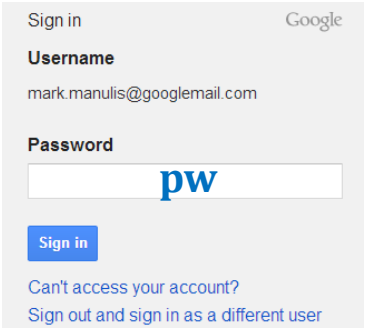
username	salt	pwhash
mark...	0x32bf..	0xa2fd..

check with database

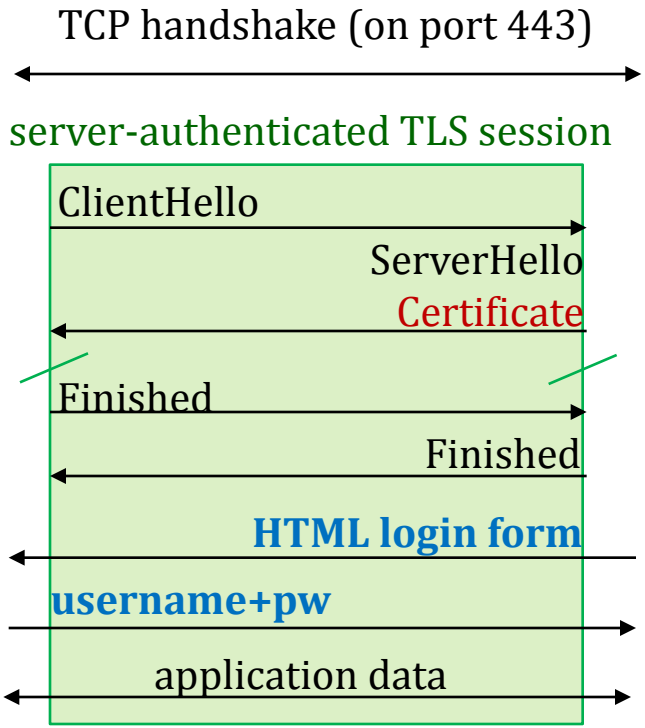
SSL/TLS

- Authenticated and Confidential Channel Establishment (ACCE) protocol [JKSS'12]
- for (server-)authentication fully relies on X.509 PKI

Phishing on HTML forms-over-TLS



if the user accepts the certificate...



rogue X.509 certificate

... then user's password is stolen

There exists many „solutions" helping users to reject rogue certificates:

- „usable" phishing indicators, browser personalization, black lists, etc...

Can we span a safety net to guarantee security irrespective of user's decision?

Cryptographic Password Authentication

UNIVERSITY OF SURREY

(Verifier-based) PAKE

- offers password authentication and key exchange
- does not depend on PKI

[BM92,BMP00,BPR00,AP05,GL06, G08,HR08,ACCP08,BBC+13]

protects against offline-dictionary attacks

[BM93,GMR06,BP13,KM14]

adds resilience to service compromise attacks

- many protocols and models, e.g. with random oracles, CRS, UC-secure
- design frameworks, e.g. based on smooth-projective hashing [GL06,G08]

PAKE protocols in Practice

UNIVERSITY OF SURREY

Stand-alone PAKE in practice

- at the moment there is no PAKE standard that is supported by browsers
- ISO-IEC 11770-4, ITU-T X.1035, IEEE P1363.2
- e.g. provably-secure SPAKE [AP05] is in ISO-IEC 11770-4, IEEE-P1363.2
- J-PAKE [HR08] is part of OpenSSL, OpenSSH, NSS, used in Firefox Sync
  - no security proof so far

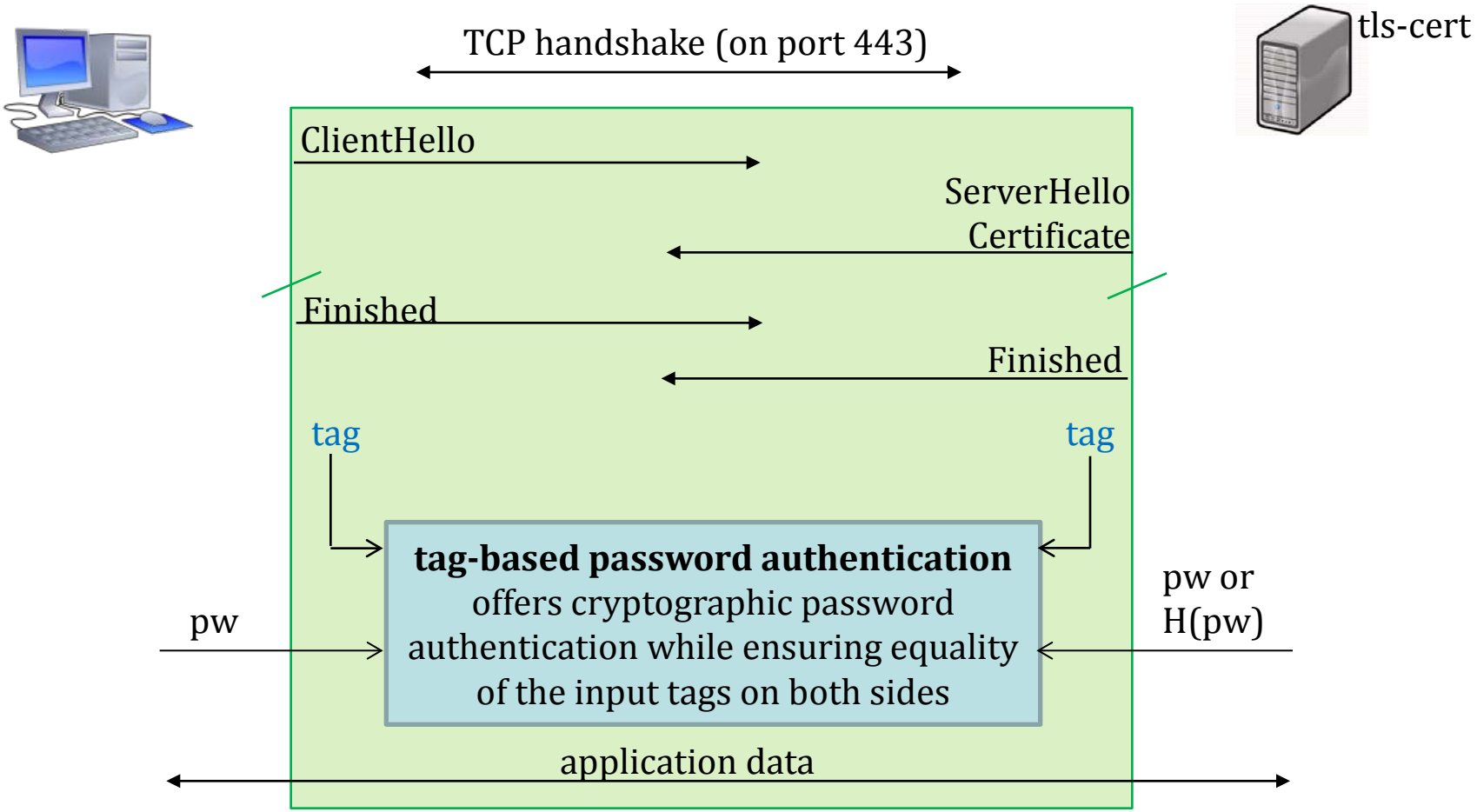
Invasive PAKE integration with TLS

- "invasive" integration implies changing TLS standard
  - RFC 6631, RFC 6628, RFC 5931, RFC 5054, challenging to adopt in real world
- SRP [Wu98] as a new TLS cipher suite in RFC 5054, has some patent issues
- SOKE [ABC+06] proposed a new TLS cipher suite

Our goal: PAKE-over-TLS

- "non-invasive" integration on top of TLS
- approaches from [OTW+09,DAT12] use PAKE in the application layer

A Modular Approach using Channel Bindings



- Rip authentication off the TLS channel and move it into the application layer
- Bind authentication in the application layer with the TLS session using tags
- Use channel binding mechanisms for TLS from RFC 5929 to realize tags

Tag-based Password Authentication

UNIVERSITY OF SURREY

Tag-based Authentication

- introduced for public key-based authentication by Jager et al (Asiacrypt'00)
- aims to authenticate not only parties but also their input tags
- tag is some public information, possibly controlled by the adversary

Tag-based Password Authentication (tPAuth)

- extends the concept to the password-based authentication setting
- accounts for online dictionary attacks
- many existing PAKE protocols can be modified to achieve tPAuth security
  - either by using H(pw, tag) instead of pw, may not work for verifier-based PAKE
  - or by using tag as an additional input to the key derivation function

Tag-based PAKE (an example protocol)

tSOKE: tagged Simple Open Key Exchange

- extends provably secure SOKE by Abdalla et al (ASIACCS'06) with tags
- used in our implementation, ECC nistp192, gen G, G' of order n

id, pw

registration over TLS channel

salt ← $\{0,1\}^{128}$, counter c ∈ ℕ
h ← PBKDF2(SHA256, pw, salt, c, 256)

id, salt, c, h →

id salt c h

tag	authentication over TLS channel	tag
x ← {2,..., n – 1}; X = xG	id, X →	y ← {2,..., n – 1}; Y = yG Y* ← Y + (h mod n)G'
Y ← Y* – (h mod n)G' Z ← xY pms ←SHA256(id, h, tag, X, Y*, Z)	← salt, c, Y*	Z ← yX pms ←SHA256(id, h, tag, X, Y*, Z)
	A_1←SHA256(pms, 'a1') →	
check A_2	← A_2←SHA256(pms, 'a2')	check A_1

Channel Bindings for TLS

UNIVERSITY OF SURREY

Channel Bindings for TLS by Altman et al. in RFC 5939, 2010

- standardizes three mechanisms to bind applications to TLS channels
- partially supported by Firefox extensions API

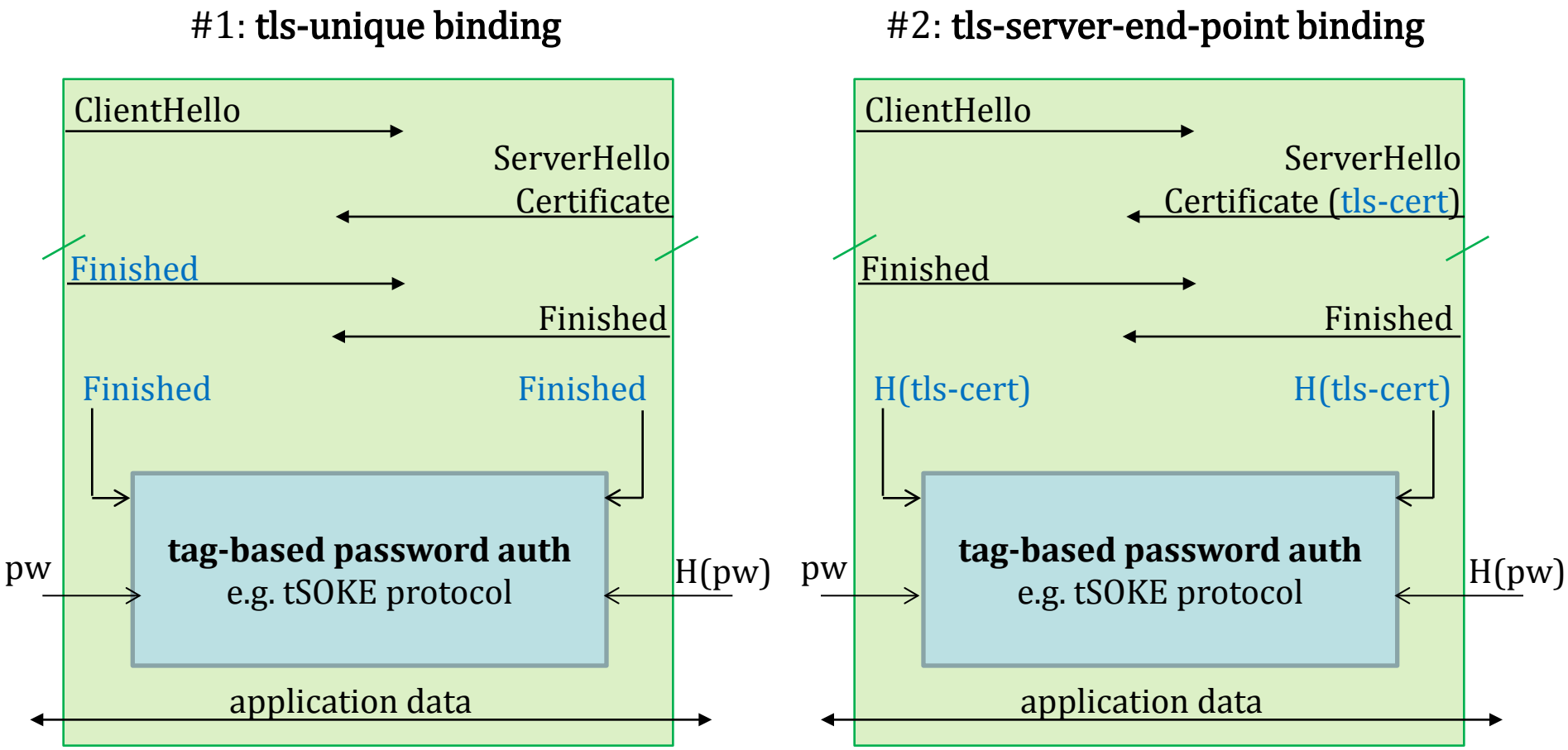
tls-unique channel binding

- uses client's (plaintext) *Finished* message
- computed as PRF(tls-key, "client finished", H(tls-transcript))
- tls-transcript includes all messages from ClientHello to ChangeCipherSpec
- this binding works for all TLS cipher suites

tls-server-end-point channel binding

- uses the hash of the server's TLS certificate
- computed as H(tls-cert)
- this binding works for TLS cipher suites with server-side certificates

Two Modular Constructions



How can we analyse security of these constructions?

Is there any difference between the two channel bindings?

How do we analyze security of this approach?

UNIVERSITY OF SURREY

Security of TLS channels

- TLS Handshake + Record Layer is a secure ACCE protocol
  - ACCE: Authenticated and Confidential Channel Establishment
  - proposed in [JKSS12] with the analysis for TLS-DHE channels
  - TLS-DH and TLS-RSA channels were analysed in [KPW13, KSS13]
- ACCE model assumes full trust into PKI for authentication purposes
- ACCE model does not capture password authentication

Use of TLS in our constructions

- no tls-cert on the client side
- no trust into PKI, i.e. server‘s tls-cert is not trusted (e.g. phishing)
  - we need to consider adversarial key registration to the ACCE model
- we do not rely on TLS to provide authentication
  - mutual authentication comes from tPAuth

Password-Authenticated Confidential Channels

UNIVERSITY OF SURREY

We modify the ACCE model to handle password-based authentication.

Mutual authentication between client and server

$\mathcal{A}$ breaks *mutual authentication* if there exists an instance π such that

- π accepted, its password $pw_{C,S}$ is not corrupted, and k not revealed
- there exists no partnered instance π‘
- with probability at least $\epsilon + O(n/|D|)$

Confidentiality of the application data (authenticated encryption)

$\mathcal{A}$ breaks *confidentiality* if it can output $b' = b$ such that

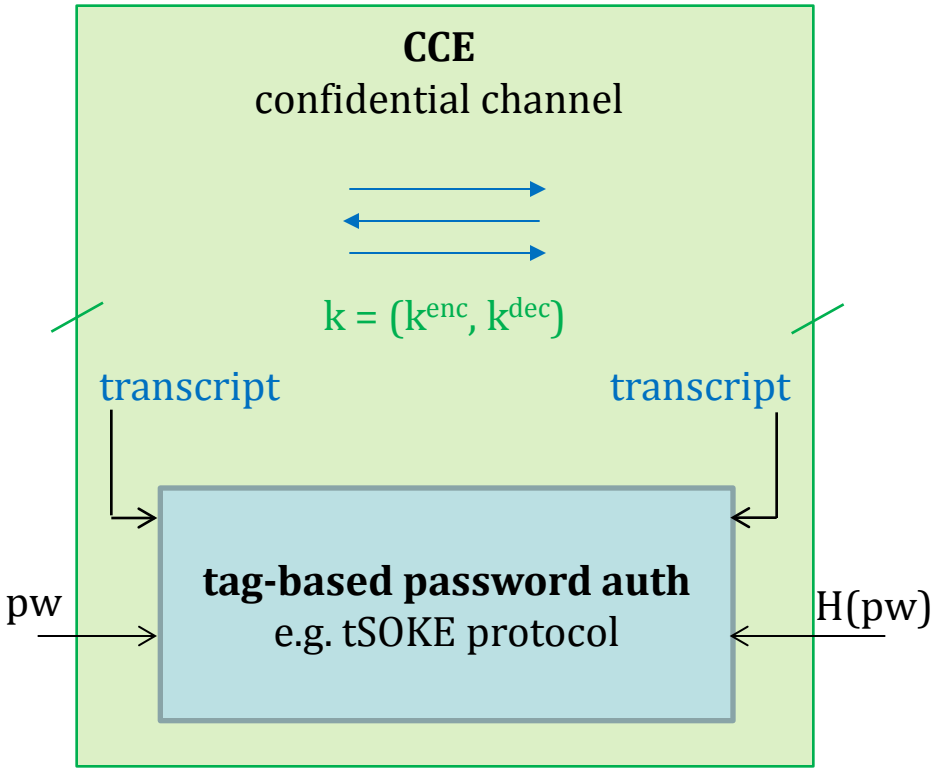
- π accepted, its password $pw_{C,S}$ is not corrupted, and k not revealed
- with probability at least $1/2 + O(n/|D|)$

Our goal is to show that both channel bindings achieve PACCE security.

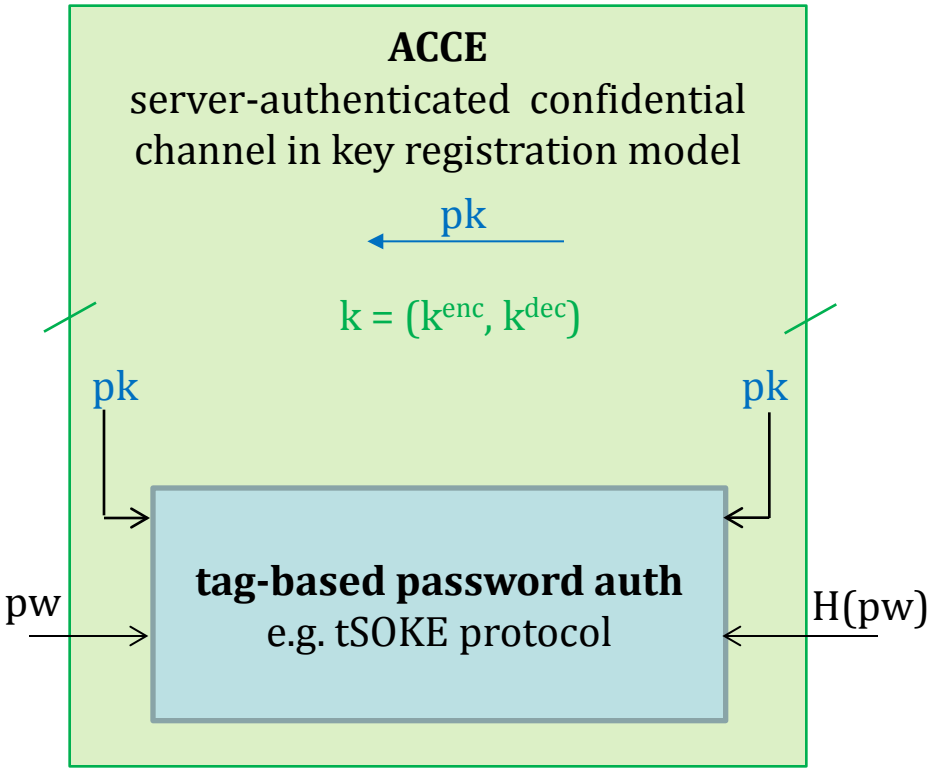
General results for TLS and beyond

We prove PACCE security of two general constructions.

CCE with transcript binding

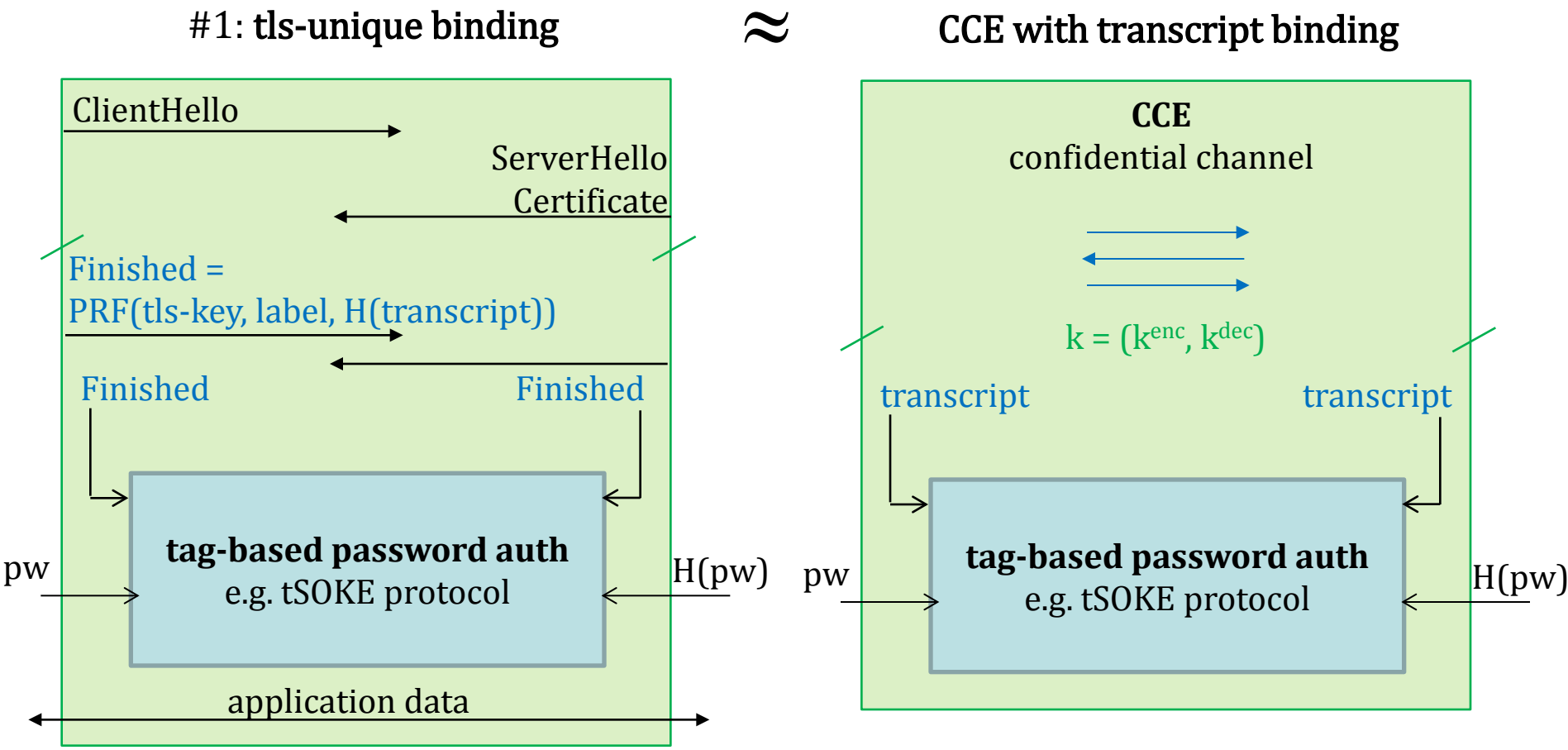


ACCE with public key binding



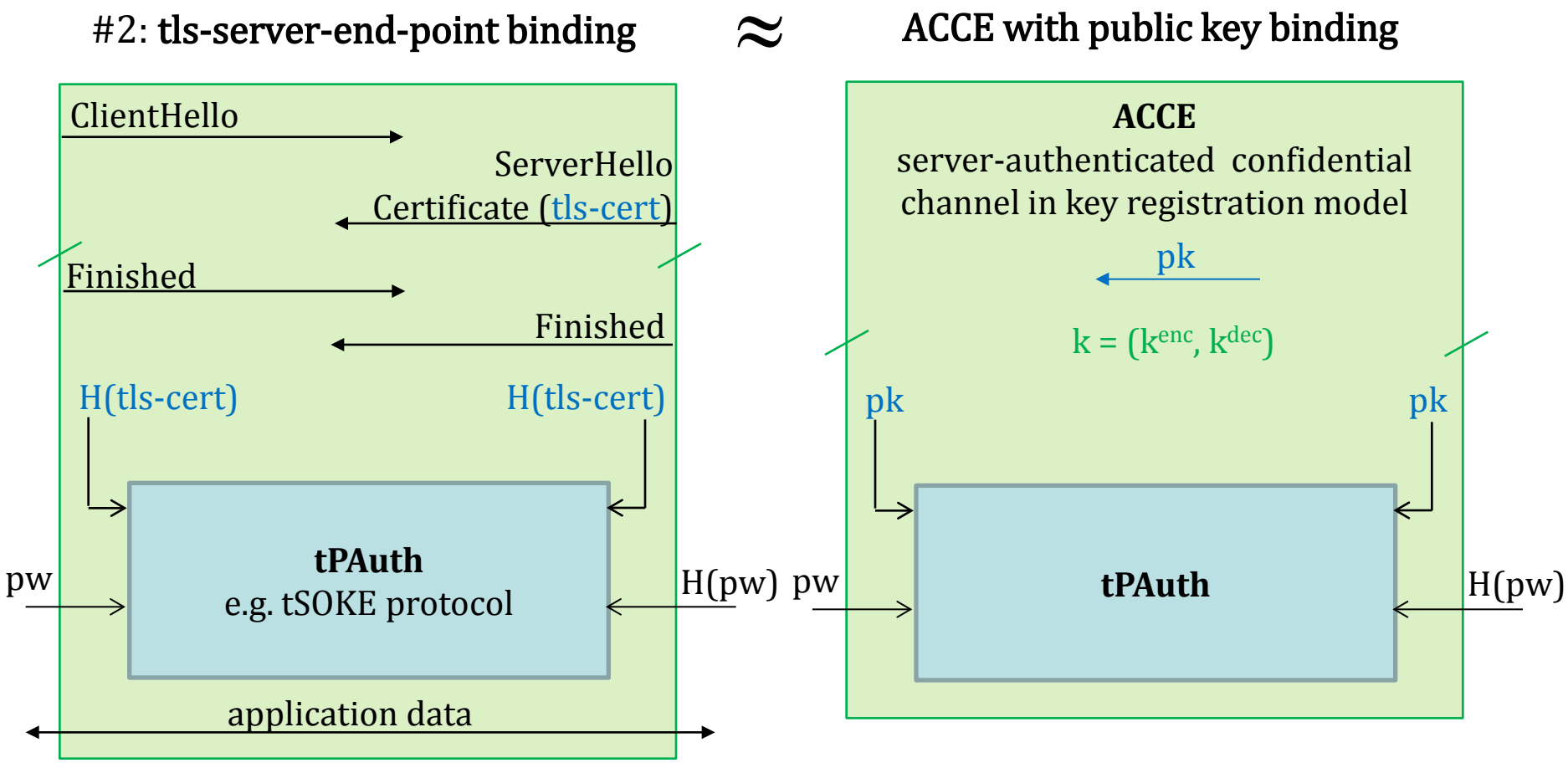
How do these proofs translate to TLS channel bindings?

How this translates to TLS channel bindings?



- TLS is ACCE and ACCE ⇒ CCE.
- We need further security for TLS PRF and collision-resistance for H.

How this translates to TLS channel bindings?



- TLS is ACCE and hence server-authenticated CCE.
- We need further second pre-image resistance for H.

tls-server-end-point binding: Implementation details and deployment

UNIVERSITY OF SURREY

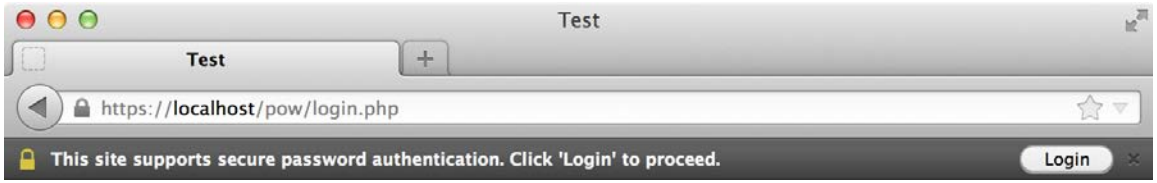
Client side

- currently only Mozilla Firefox has API that gives access to tls-cert
- our Firefox extension uses Javascript with libraries for ECC and PBKDF2
- automatic detection of PACCE support on the accessed website
  - by sending first tPAuth message as an HTML microobject and remaining via HTTP POST requests with responses formatted as JSON messages (compatible with cookie-based session management)
  - another approach is to define a new HTTP Authentication method [FHBH+99]
- trusted UI with customizable branding in the login dialog box

Server side

- PHP application using custom library for ECC
- uses Apache CGI variable $_SERVER['SSL_SERVER_CERT']
- support for PACCE can be added to the Apache webserver at run-time

PACCE with tls-server-end-point: Performance measurements



Operation	Pure Javascript (ms)	Native C (ms)	HTML forms-over-TLS (ms)
	our working demo	if supported in browsers	today's approach
Client cryptographic computations	354.06 ± 5.12	5.32 ± 0.23	N/A
Server cryptographic computations	36.27 ± 2.20	6.34 ± 0.48	N/A
Total runtime	487.72 ± 49.93	109.04 ± 47.96	66.16 ± 27.80

Software: Mozilla Firefox 21.0, Apache 2.2.22, PHP 5.4.14, GMP 5.1.1, MySQL 5.5.28, OpenSSL 1.0.1e, Mac OS X 10.8.3
Hardware: 2.6 GHz Intel Core i7, 16 GB of RAM
Network: Corporate network, ping time 48.55 ± 37.76 ms

Links to Demos and Papers

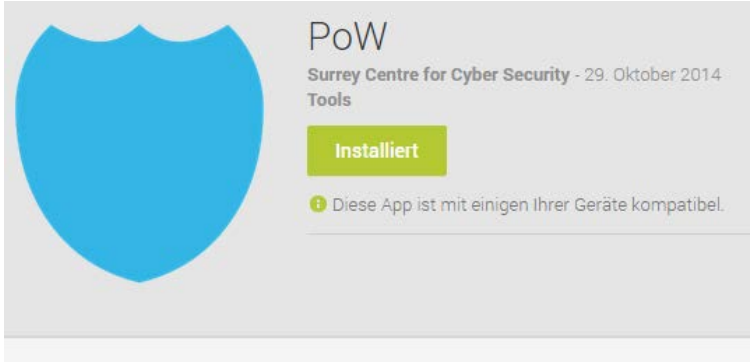
UNIVERSITY OF SURREY

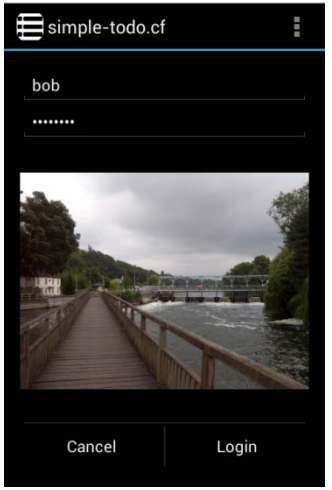
PACCE in desktop browsers

- M. Manulis, D. Stebila, N. Denham: Secure Modular Password Authentication for the Web using Channel Bindings. SSR 2014, https://eprint.iacr.org/2014/731
- Website http://www.douglas.stebila.ca/research/papers/MSD14/

PACCE in mobile browsers

- F. Kiefer, M. Manulis, D. Stebila: Password-based Authentication for Mobile Browsers. Available from https://www.franziskuskiefer.de/data/pow.pdf
- Test website https://www.simple-todo.cf
- search in Google Play for PoW and SCCS





Summary and Outlook

UNIVERSITY OF SURREY

Theoretical contributions

- 2 general PACCE constructions using channel bindings
  - CCE with transcript binding
  - ACCE with pk binding
- First security analysis of channel bindings for TLS (RFC 5929)

Practical contributions

- Two PACCE constructions with TLS channel bindings
  - tls-unique binding
  - tls-server-end-point binding
- Implementation of tSOKE with tls-server-end-point
- Integration into existing architectures for web and mobile browsers
  - Firefox browser extension for desktop browsers
  - An app-based solution for mobile browsers on Android

Thank you!