Surrey Centre for Cyber Security SCCS

UNIVERSITY OF SURREY

Password-Authenticated Searchable Encryption (over the Web)

Mark Manulis

Surrey Centre for Cyber Security

mark@manulis.eu

National Cyber Security Centre

Academic Centre of Excellence in Cyber Security Research

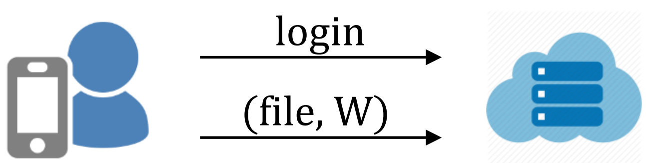
EPSRC Engineering and Physical Sciences Research Council

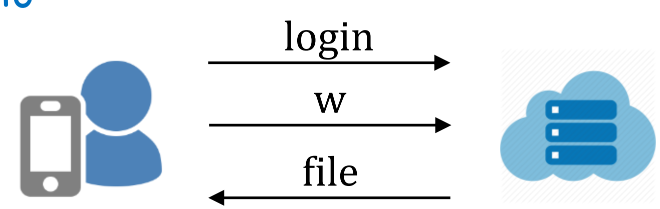
Outsourcing and Retrieval of Data

Cloud storage

- Widely used today for business and personal use
- Cloud storage = „infinite“ storage ressource
- Flexibility in upload/retrieval of data, multiple devices, synchronization, etc

basic scenario





(file, W) : $w \in W$

$W = w_1, \ldots, w_n$: keywords

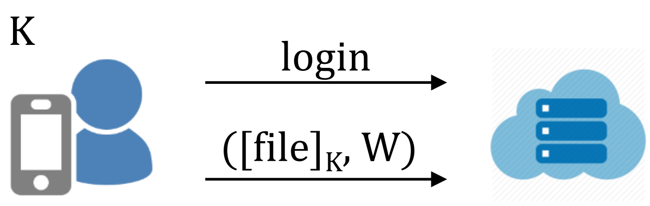
Trust issues

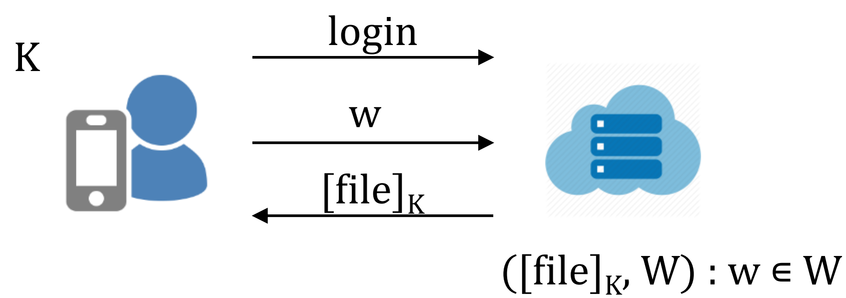
- Cloud is trusted with availability (replication would be needed to address this)
- Cloud is trusted to provide integrity of outsourced files
- Cloud has access to all user‘s files (confidentiality) and is trusted to enforce access control

Outsourcing and Retrieval of Encrypted Data

Outsourcing encrypted data

- Untrusted cloud necessitates the use of encryption on the client side
- Encryption can be based on public key crypto or symmetric crypto
- Client authentication/login phase is explicit, prior to outsourcing/retrieval





Encrypting keywords

- Keywords require encryption
  - otherwise, they may leak information about files
  - or limit the flexibility in defining the keywords

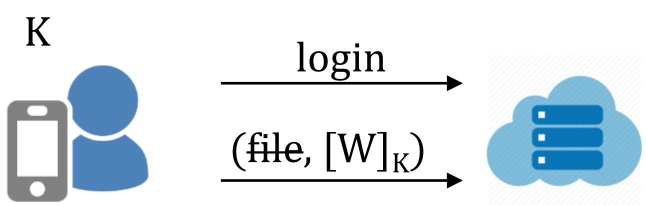
- But if keywords are encrypted – $[W]_K$ – then how can the client search?

Searchable Encryption (SE)

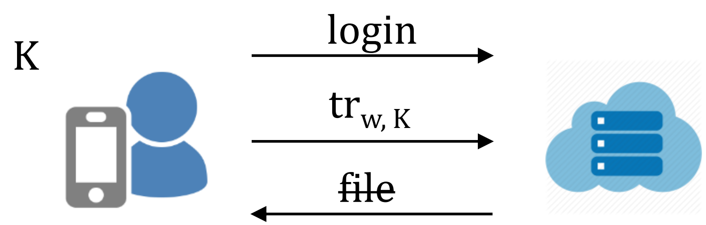
UNIVERSITY OF SURREY

SE functionality

- Enables authorised search over encrypted keywords (separate from file encryption)
- Hides keywords from the cloud throughout
  - Symmetric Searchable Encryption (SSE) [CGKO11]
  - Public Key Encryption with Keyword Search (PEKS) [BDOP04, HL07, ABC+08, KM15]



$[W]_K = [w_1]_K, \dots , [w_n]_K$: encrypted keywords



(~~file~~, $[W]_K$) : $w \in W$

$tr_{w,K}$: trapdoor / authorisation token for w

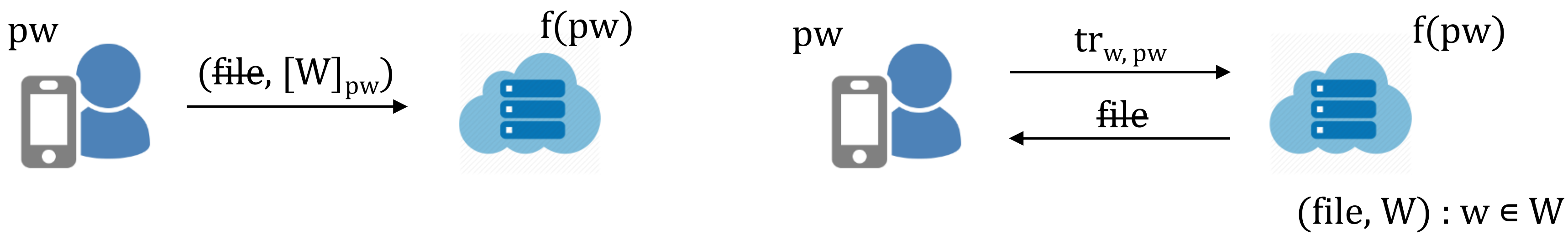
Current problems

- No client authentication
  - Existing SE schemes rely on external user authentication mechanisms
  - Without authentication encrypted keywords can possibly be injected
- SE schemes require users to manage their own high-entropy keys
  - Solutions are not device-agnostic
  - If user looses the key then all data is lost

Password-Only SE Solution

Benefits of password-only SE

- Memorable password would be sufficient for all operations on the client side
  - no key management leads to a device-agnostic solution
- Client authentication will become implicit (at no additional cost)
  - stronger protection for the outsourcing phase



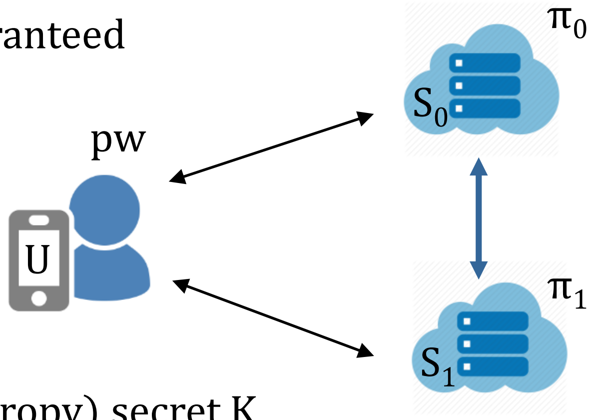
Challenge

- Increased threat from offline dictionary/guessing attacks on pw and W
  - due to their low entropy no single-server solution would offer protection
- Need for <u>distributed trust</u> (as with many other password-only cryptosystems)
  - 2*-server approach* (practical) or more general t-out-of-n server approach
  - Immediate gain for availability (by keeping files on more than one server)

Distributed Password Authentication

2-Server password authentication (and key exchange)

- After registration user U stores pw. Each server S_b stores pw-share π_b or encrypted pw.
  - additive/XOR ($pw = \pi_0 + \pi_1$) : [SK05, KMTG12, KM14]
  - multiplicative ($pw = \pi_0 \times \pi_1$): [BJKS03]
  - encryptions of pw : [JKK14]
- Unless both servers are corrupted secrecy of pw is guaranteed



Further extensions

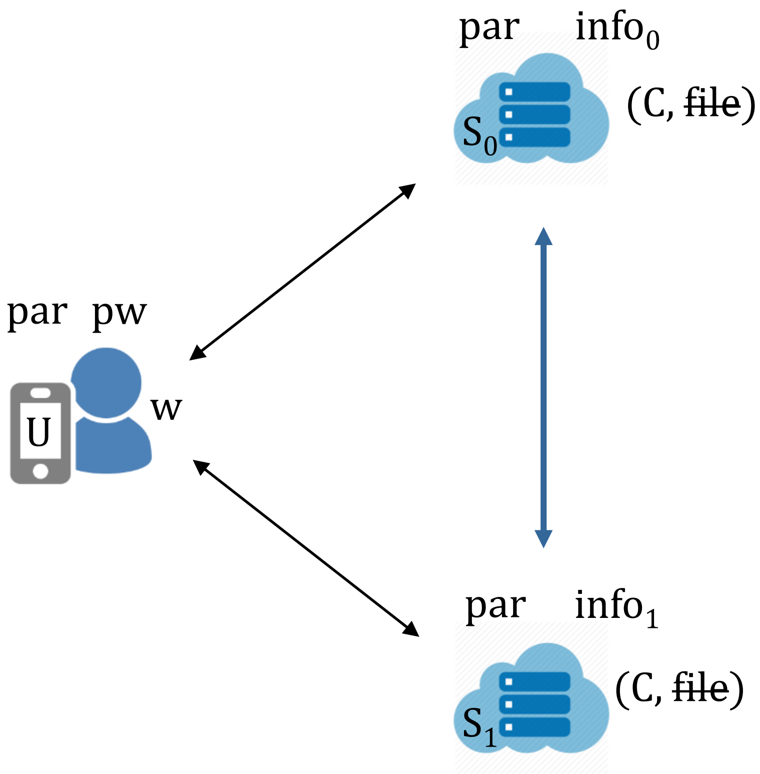
- Password-Authenticated Secret Sharing (PASS)
  - U can use pw to outsource/reconstruct a (high-entropy) secret K
  - password-only setting: [JKK14, YHCL14, JKKX16]

- Blind Password Registration (BPR) [KM16]
  - Let f be a server-defined policy for the lenth and composition of pw (e.g. ASCII chars)
  - U can prove that f(pw) = true without disclosing pw to the servers

Password-Authenticated Keyword Search

PAKS architecture

- **Setup**: initialise the system parameters par known to user U and servers S_0 and S_1



- **Register**: protocol to register a new user U with password pw, each server stores info$_d$
- **Outsource**: protocol to outsource (w, ~~file~~) to S_0 and S_1, each server stores (C, ~~file~~)
- **Retrieve**: protocol to retrieve ~~file~~ based on w

Main security properties

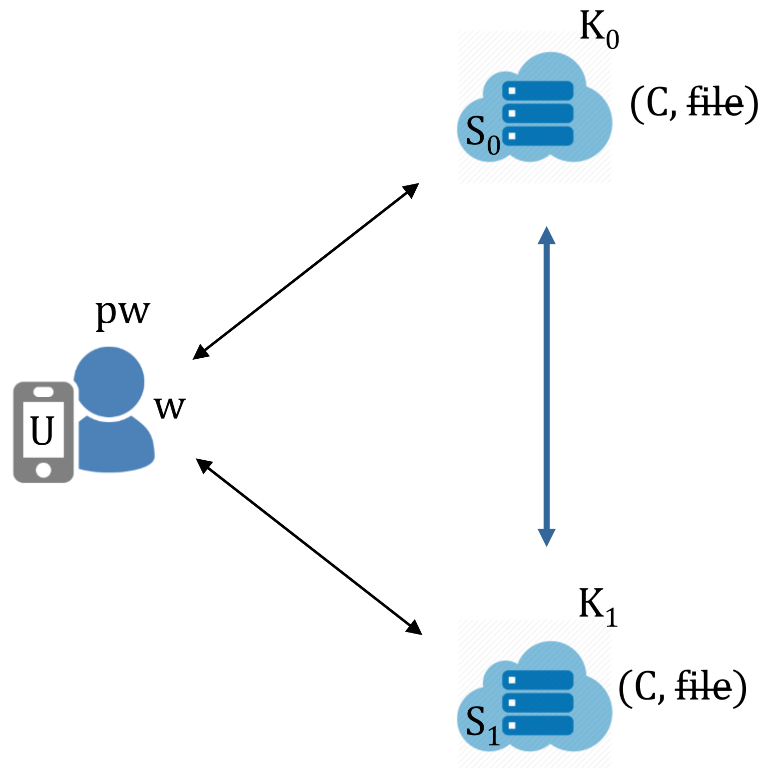
- **IND-CKA** to protect secrecy of the outsourced keywords
- **Authentication** to ensure that only legitimate users can outsource and retrieve

[HMC17] Password-Authenticated Keyword Search. IEEE Symposium on Privacy-Aware Computing, 2017

Intuition for building PAKS

Intuitive approach

- Black-box combination of PASS with SSE
- Registration: U picks pw and K and uses PASS to secret-share K across S_0 and S_1
- Outsource: U uses PASS to reconstruct K and uses K with SSE to outsource (w, ~~file~~)
- Retrieve: U uses PASS to reconstruct K and uses K with SSE to retrieve ~~file~~



Problem with black-box combination

- Current PASS and SSE schemes are not compatible
  - differences in syntax, functionality, security properties
  - for instance, PASS protocols do not have a registration step
- Black-box combinations are often less efficient than direct constructions

A direct PAKS scheme

Register:

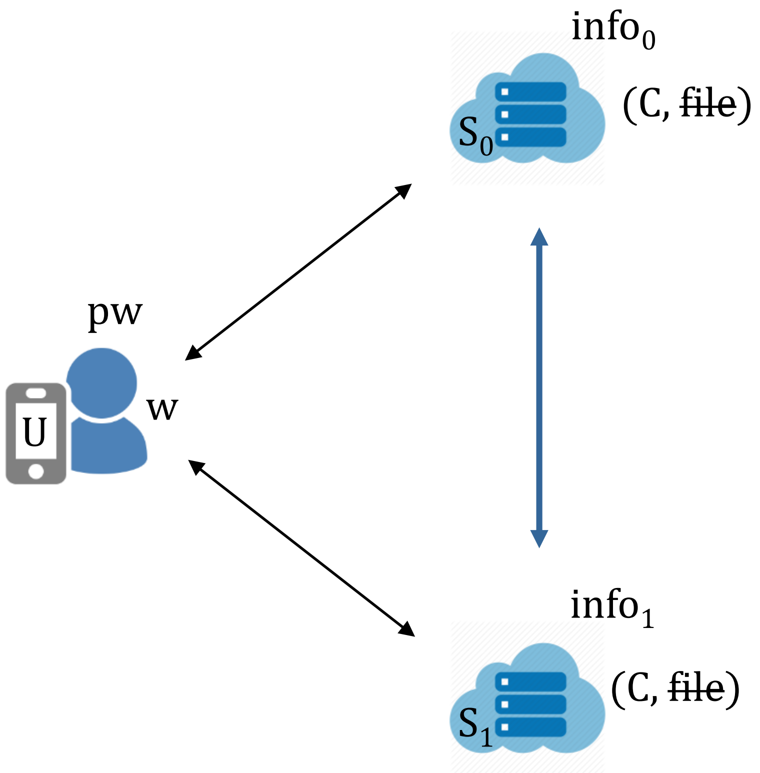
- $info_d = (x_d, g^{r_1}, g^{r_2}, C_{pw}, K_d, mk_d)$: sent by U to S_d
- $X = g^{x_0 + x_1}$: public key with distributed secret key
- $C_{pw} = X^{r_2} h^{pw}$: password ciphertext
- $K_0 K_1 = X^{r_1} K$: secret sharing of key K
- mk_d server's MAC key derived from K

Outsource:

- U reconstructs K and authenticates each S_d
- Outsources (C, ~~file~~) and authenticates using mk_d
  - $C = (e, v, \mu)$: e is random, $v = PRF(t, e)$, MAC μ
  - $t = KDF(K, w)$: keyword trapdoor

Retrieve:

- U reconstructs K and authenticates each S_d
- Sends trapdoor t to S_d and authenticates itself
- Obtains (C, ~~file~~) from S_d and checks integrity of both using μ



* we use encrypted pw approach

Changing / resetting passwords

Any password change must leave K unchanged; otherwise all outsourced keywords will no longer be decryptable.

PAKS offers two possibilities to replace $(g^{r2}, C_{pw} = X^{r2}h^{pw})$

Each server knows:

$info_d = (x_d, g^{r1}, g^{r2}, C_{pw}, K_d, mk_d)$

$X = g^{x0 + x1}$

$C_{pw} = X^{r2}h^{pw}$

$K_0K_1 = X^{r1}K$

mk_d server's MAC key

User still knows pw

- User can use pw to reconstruct K and mk_d
- Receives authenticated (g^{r2}, C_{pw}) and re-encrypts to $(g^{r2r^*}, (C_{pw}h^{-pw})^{r^*}h^{pw^*})$ using new pw*
- Authenticates changes to each server using mk_d

User has lost the original pw

- Requires secure out-of-band channel (or 2nd factor) from the servers to the user
- User receives g^{x_d} from each server to reconstruct X
- User picks new pw* and sends $(g^{r2^*}, C_{pw^*} = X^{r2^*}h^{pw^*})$ to both servers

Implementation and Demonstration

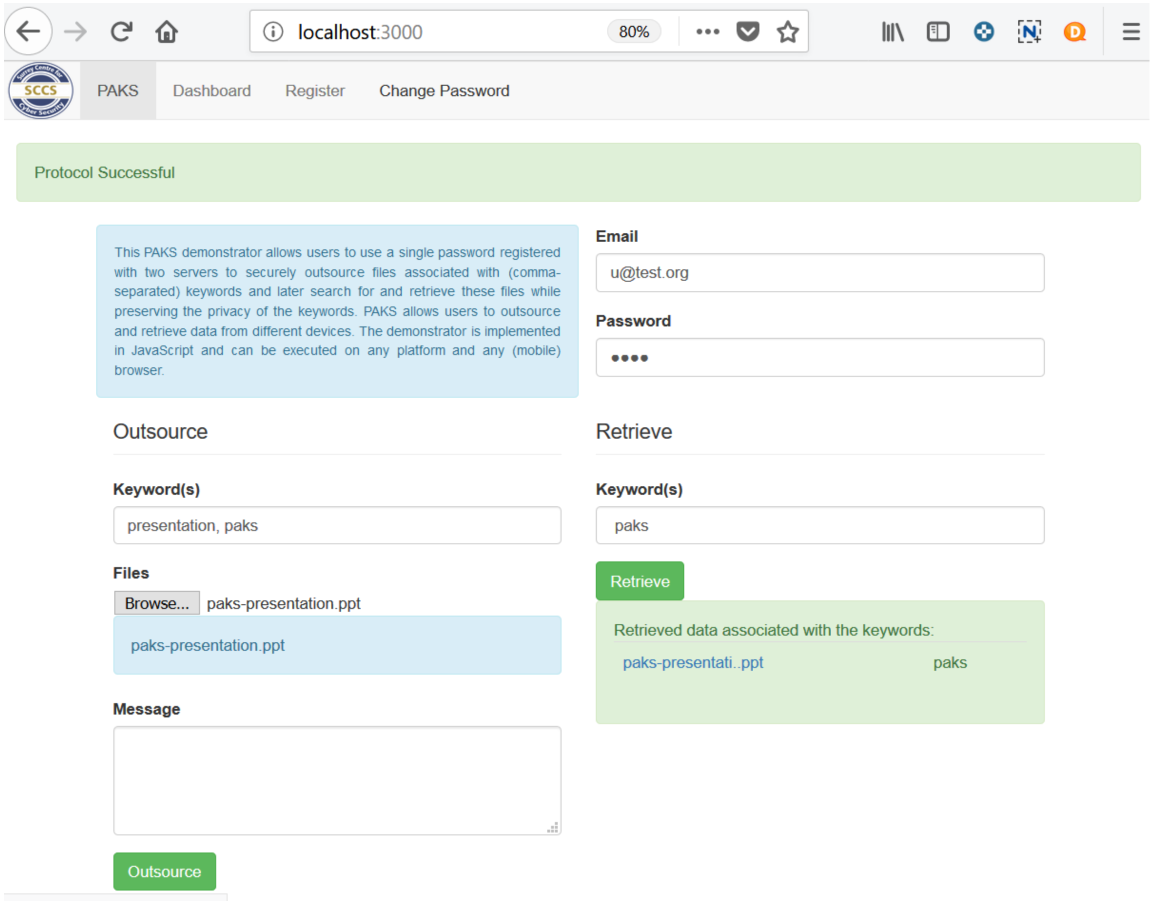
UNIVERSITY OF SURREY

Crypto choices

- based on Stanford JS Crypto Library (SJCL)
  - NIST P384 elliptic curve
  - SHA256 hash function
  - PBKDF2 (256 bits)
  - AES-GCM (256 bits)
  - HMAC

Web-based demonstrator

- JS implementation for any web browser (incl. mobile)
  - Node.JS (servers) + NodeCache + AJAX
  - MongoDB (good for files)
- Supports password change, multiple keywords
- Currently moving to WebCrypto API to support file encryption



Performance and Scalability

Performance evaluation

- (*) Average of 1000 executions with random keywords of 5 to 10 characters
- No network latency

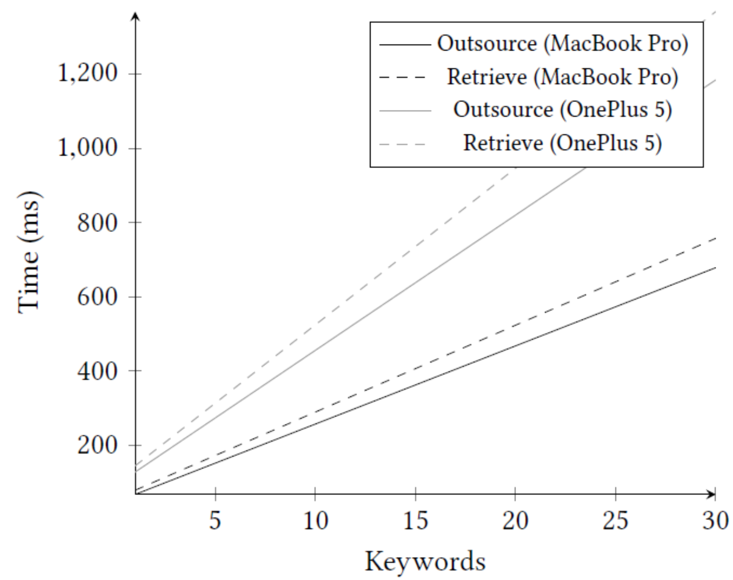
Scalability

- Appears scalable on commodity user devices
  - less than 400ms to process 10 keywords on a laptop
  - less than 1s to process 10 keywords on a smartphone

Optimisation potential

- Trials show possible speed-up of up to 15 times with WebCrypto API instead of SJCL

Procedure	Client		Each server
	MacBook Pro 2.2GHz Intel Core i7 16GB RAM	OnePlus 5 Snapdragon 835 2.45GHz 8GB RAM	MacBook Pro 2.2GHz Intel Core i7 16GB RAM
Registration	180 ms	360 ms	15 ms
Reconstruction of K	100 ms	195 ms	229 ms
Outsourcing (*)	23 ms	68 ms	22 ms
Retrieval (*)	26 ms	73 ms	28 ms



Thank you!

Mark Manulis
mark@manulis.eu
www.manulis.eu

Surrey Centre for Cyber Security
University of Surrey

Guildford, GU2 7XH

sccs@surrey.ac.uk
http://sccs.surrey.ac.uk
@SCCS_UniSurrey