Financial Crypto and Data Security – Mar. 3, 2011

Mercury: Recovering Forgotten Passwords Using Personal Devices

M. Mannan

NSERC PDF, University of Toronto

m.mannan@utoronto.ca

Collaborators: D. Barrera, C.D. Brown, D. Lie, P.C. van Oorschot

Why do we need password recovery?

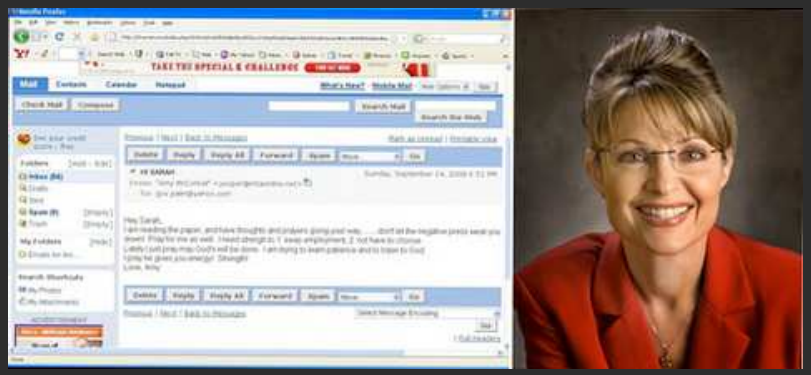
none is immune to forgetting

recall-based authentication needs reset/recovery

Why recovery must be secure?

"So in war, the way is to avoid what is strong and to strike at what is weak." (The art of war, 6:30)

Recovery/reset techniques are weaker than password?



Recovery vs. reset

In many cases users are forced to choose a new password

- lack of a secure transmission channel for passwords?
- cleartext passwords are not stored?

... but good passwords are not easy to generate

Our focus: recover the original password

‘I forgot my password’: now what?

1. Small, local env: ask the admin (secure, not scalable)
2. Large org, networked env: email, PVQ (scalable, insecure)
   - help desk calls are expensive

Our design goals: scalable, secure, deployable

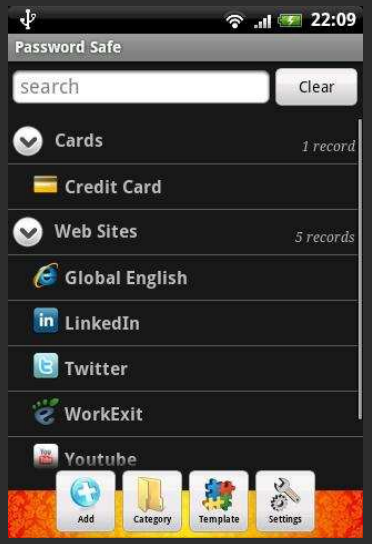
State of the art

1. Password managers: in all platforms
2. Email, SMS, phone: ownership, "secure" media
3. Personal verification questions (PVQs): more secrets!
   - related: Facebook social auth, Blue moon

No academic proposals for recovery?

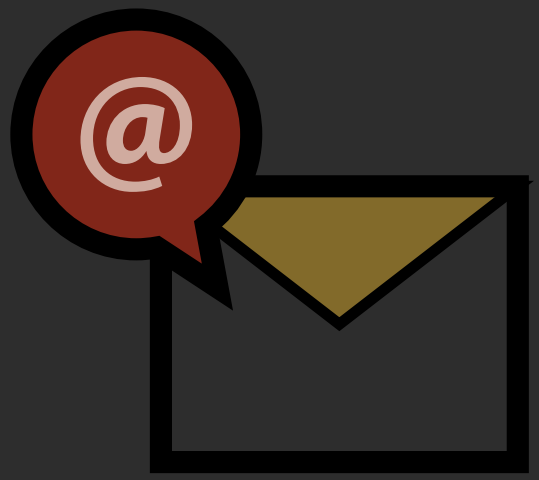
Password managers

1. Online encrypted storage (LastPass)
2. Offline encrypted storage (KeePass)
3. Issues:
   - trust: third parties?
   - password update: extra step?
   - master password: weak or none?



Email password reset/recovery

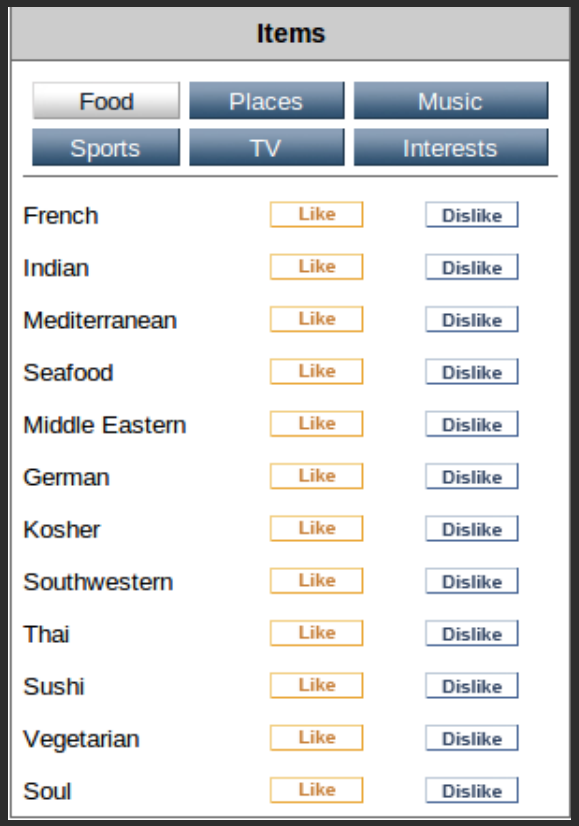
1. Widely used
2. Issues:
   - trust email providers
   - trust ISPs, wifi providers
   - check spam, keep waiting...
   - reset vs. recovery



Facebook social auth

1. Used for account verification
   - e.g., Captcha replacement
2. Issues:
   - abstract/pet images
   - barely known friends
   - privacy issues?

Blue moon: preference-based auth



Better than regular PVQs?

... but no password recovery

Our proposal: Mercury

1. Key idea
   - use end-to-end encryption for safe password retrieval
2. Mechanism
   - user generates a key pair for password recovery
   - shares the public key with a site during account setup
   - the site sends encrypted password during recovery

Mercury: components

1. User PC (i.e., the primary login machine)
2. Remote server
3. Personal mobile device (PMD) – for portability
4. Mercury software on: PC + PMD + Server
5. Local communication channel: PC $\leftrightarrow$ PMD

Mercury: design features and usage

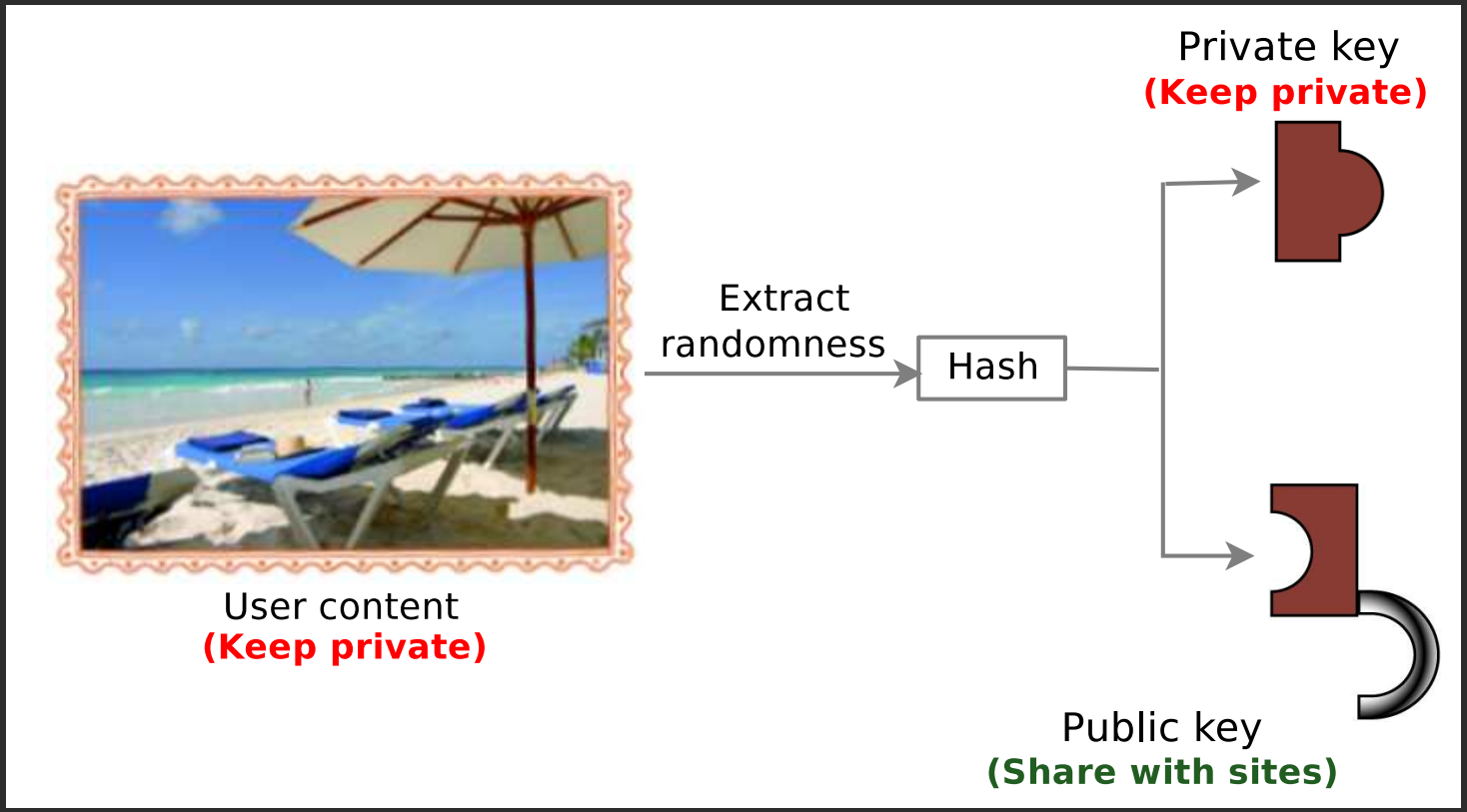
1. Key design features
   - use familiar technologies:
     smart-phones, QR codes
   - personal-level public keys, but no PKIs

2. Examples: online accounts, desktop password recovery

Mercury: steps

1. Key generation and backup
2. Key sharing
3. Password recovery

Key generation: personal objects



see also: Object-based password (HotSec'08)

Key generation: random seed

1. Same seed $\Rightarrow$ same keys
2. Save offline: print the QR coded seed

Key sharing with remote parties

1. Users can upload the public key from the primary PC
   - unique key per site, or
   - one key for all sites
2. Keys can be sent directly from the PMD

Password recovery steps

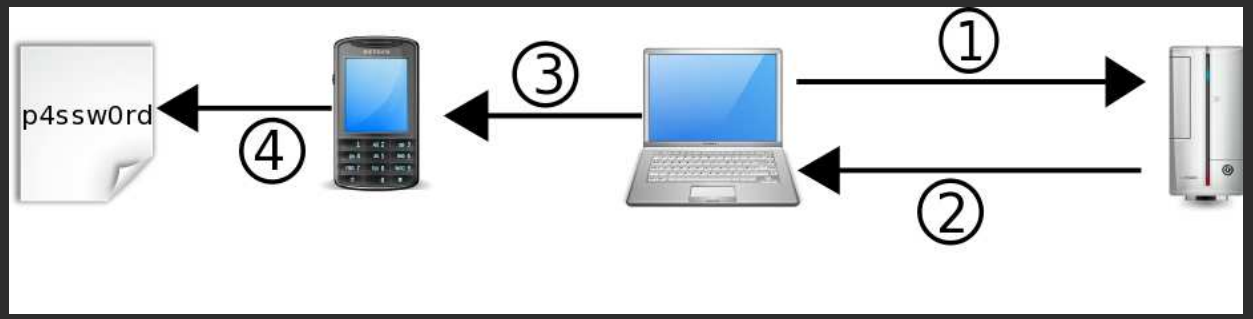
User (U) — Server (S)

ID_U, "forgot my password" →

Retrieves P, $pubU$

← $m = encode(\{P\}_{E_{pubU}})$

U transfers m to PMD,
retrieves $P = \{decode(m)\}_{D_{privU}}$



What if: the server does not store cleartext password?

Server-side password storage

1. Original password (plaintext or encrypted)
2. Hashed password
   - store public-key encrypted password
   - use reset password

PC-to-device channel examples

1. QR code: requires camera
2. Audio: universal availability
3. Direct email access from device

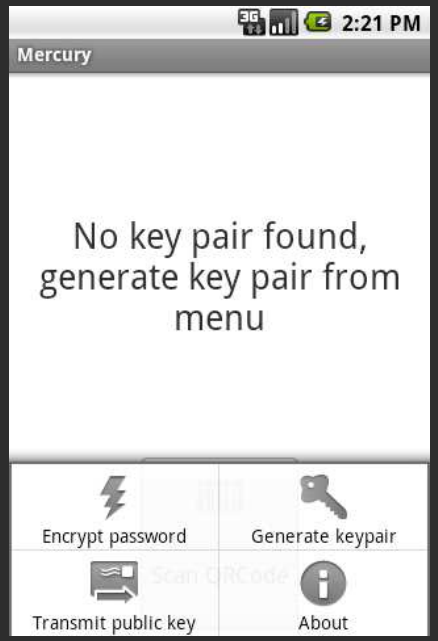
Features, advantages

1. Secure recovery: allows users to keep the same password
2. No third parties: user$\leftrightarrow$password$\leftrightarrow$server
3. Password update remains the same (for the primary mode of Mercury)
4. Key restoration after device update: usability?
5. Cheap two-factor auth (sort of)

Limitations

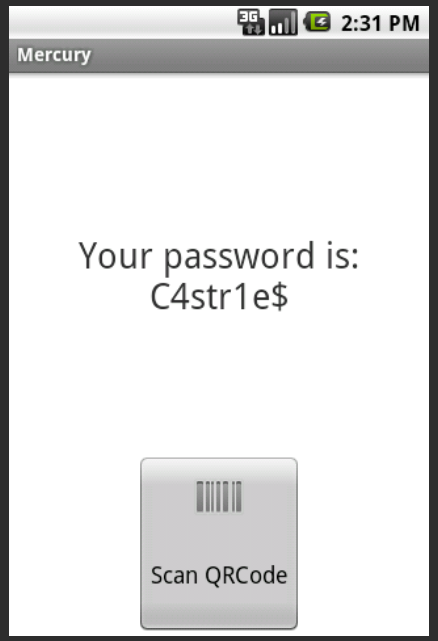
1. Require: server-side assistance + personal device (optional)
2. Device issues: compromised, lost, stolen
3. User level key management
   - leaked keys, key-gen objects

Mercury Android app and test website



(a) startup

(b) web-based recovery



(c) decrypted passwd

Open issues

1. How to bring service providers on-board?
   - trusted third parties – Google/Firefox Sync?
2. What more can we do with user-level public keys?
   - pk-based auth?

Android app and test site:

`http://www.ccsl.carleton.ca/software/mercury/`