Practical Anonymous Subscriptions

Alan Dunn, Jonathan Katz, Sangman Kim, Michael Lee, Lara Schmidt, Brent Waters, Emmett Witchel

University of Maryland

The University of Texas at Austin

Practical Anonymous Subscriptions

Alan Dunn, Jonathan Katz, Sangman Kim, Michael Lee, Lara Schmidt, Brent Waters, Emmett Witchel

UNIVERSITY OF MARYLAND

THE UNIVERSITY OF TEXAS AT AUSTIN

Anonymous subscriptions

Provide registered/paid u ability to *log in* and *acc*

Music/video streaming
reading news articles
transit pass

…while remaining **anonymous**…

…yet still allowing the server to enforce **admission control**

I.e., users cannot share their login with friends

System model

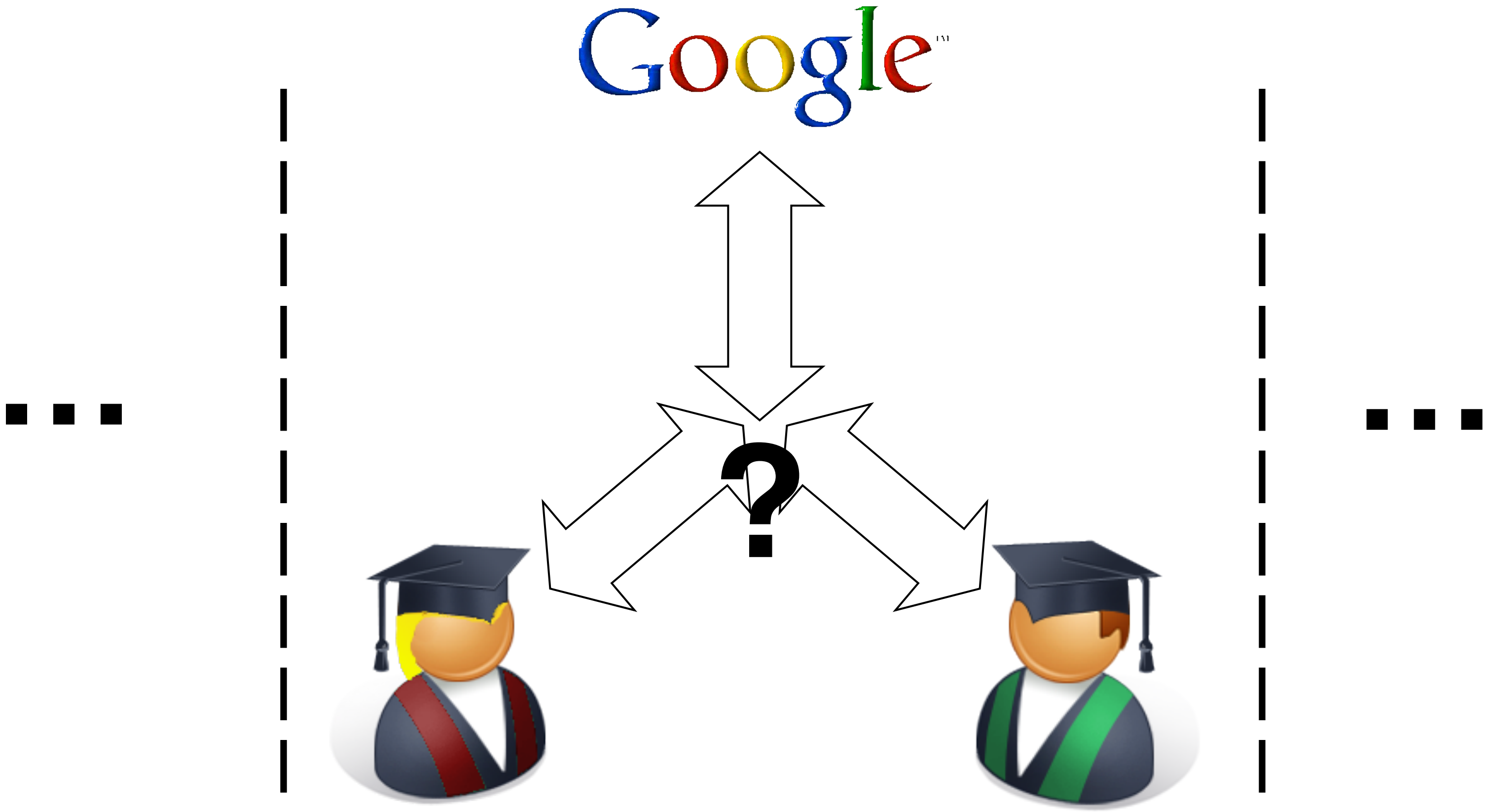
Time broken into a series of well-defined *epochs*

Anonymity/unlinkability

- Cannot link a **user login** to a **user registration**

- Cannot link **logins by the same user** (in different epochs) to each other

Anonymity/unlinkability



Admission ctr’l (“soundness”)

- Each registered user can only have **one** active login per epoch
  - I.e., a user cannot freely share their login information with their friends
  - (Formal definition later)

Soundness

sk_1 sk_1

How long is an epoch?

Shorter epochs ⇒ better anonymity

Longer epochs ⇒ less computation

How long is an epoch?

- Here: **conditional linkability**
  - Logged in user can choose to “re-up” his login for the next epoch
    - Re-up is cheaper than a login
  - Allows server to link user across epochs
    - User decides when this is acceptable
    - User can do a full login if unlinkability is desired

Related (but different)

- Anonymous credentials, DAA, group signatures
  - Anonymity, but no admisison control
- Anonymous blacklisting systems
  - Anonymity, revocation, but no notion of per-epoch admission control
- E-cash
  - Anonymity, double spending detected, but no notion of unlimited re-use

Related work

- Unclonable authentication [Damgård, Dupont, Østergaard]
- n-time anonymous authentication [Camenisch et al.]
  - Uses prior ideas from e-cash [Camenisch, Hohenberger, Lysyanskaya]
  - Different model – multiple verifiers, traceability after the fact

Our contributions

- **More efficient, simpler** construction
  - "Weaker" cryptographic assumptions
  - Cleaner definitions
- **Conditional linkability** for improved efficiency
- **Implementation** and system evaluation

What we do *not* prevent

- Users sharing login information to use at **different** times
- **Other ways** of breaking anonymity
  - Traffic analysis, IP addresses
  - User behavior
  - History of accessed content

  - Address using complementary techniques

Functional definition I

- **Setup** – server generates public/private keys; initializes state including **cur/next**
- **Registration** – user/server interact; user obtains secret key sk (or error)

Functional definition II

- **Login** – Using sk and the current epoch number, user logs in to server
    - Server increments **cur**

- **Link** (“re-up”) – User currently logged in during epoch *t* can log in for epoch *t +1*
    - Server increments **next**

Functional definition III

- **EndEpoch** – server refreshes state; updates **cur/next**
  - **cur** = **next**; **next** = 0

Security definitions

- (Honest) user is **logged in** at some pont in time if (1) that user previously ran Login in that epoch, or (2) at some point in previous epoch, user was logged in and ran Link

- (Honest) user *i* is **linked** at some point in time if at some previous point during that epoch, user was logged and ran Link

Soundness (informal)

- Attacker registers any number N of users; honest users also register
- Attacker interacts with server abritrarily
- Honest users login/link (so affect server state), but attacker cannot observe
- Attacker controls when epochs end

Attacker **succeeds** if, at any point in time, **cur** > N + #honest users logged in

Anonymity (informal)

- Phase 0
  - Attacker outputs arbitrary public key
  - Two honest users register (and get secret keys)
- Phase I
  - Attacker induces honest users to Login/Link
- Phase II – neither user logged in
  - Users either permuted or not
  - Attacker induces honest users to Login/Link

Attacker **succeeds** if it can guess whether users were permuted in Phase II (with significantly better than ½ probability)

Construction (intuition)

- **Registration**: user gets “anonymous credential” C (i.e., a re-randomizable blind signature) on PRF key k
- **Login** in epoch t: user sends C’ + $F_k(t)$ + ZK proof of correctness
    - Server verifies signature and proof; checks that $F_k(t)$ not in table; stores $F_k(t)$ in table
- **Link** in epoch t: user sends $F_k(t)$ + $F_k(t+1)$ + ZK proof of correctness
    - Look up $F_k(t)$ in table; verify proof; add $F_k(t+1)$

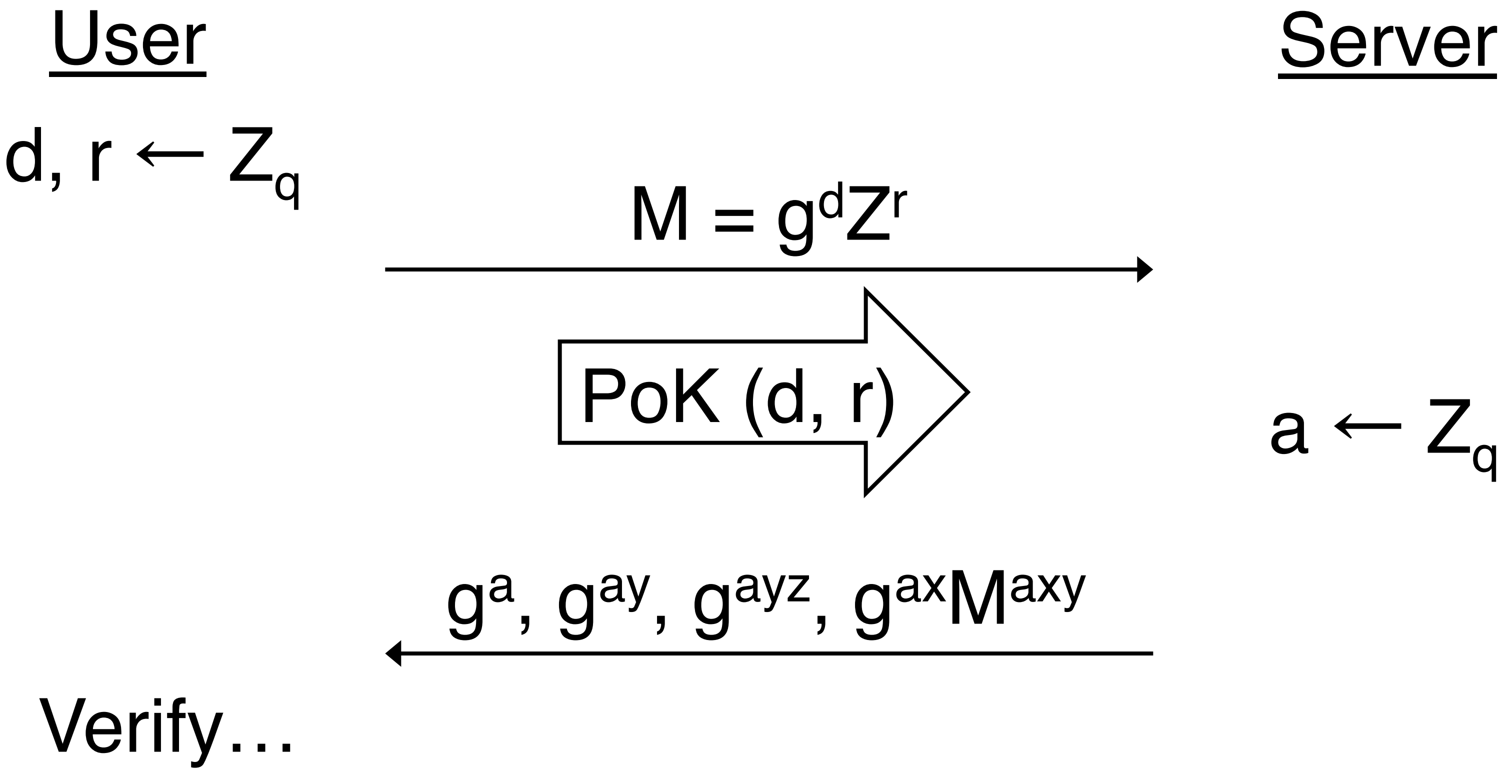
Construction (further detail)

- Anonymous credential is based on variant of **Camenisch-Lysyanskaya signatures**
  - Public key = (g^x, g^y, g^z)
  - Signature on (d, r) is $(g^a, g^{ay}, g^{ayz}, g^{ax}(g^d Z^r)^{axy})$
  - Re-randomizable, blindable, efficient ZK proofs

- **Dodis-Yampolskiy PRF**
  - $F_k(t) = g^{1/(k+t)}$
  - Compatible with various efficient ZK proofs

Construction (further detail)

- **Registration**



User

Server

$d, r \leftarrow Z_q$

$M = g^d Z^r$

PoK (d, r)

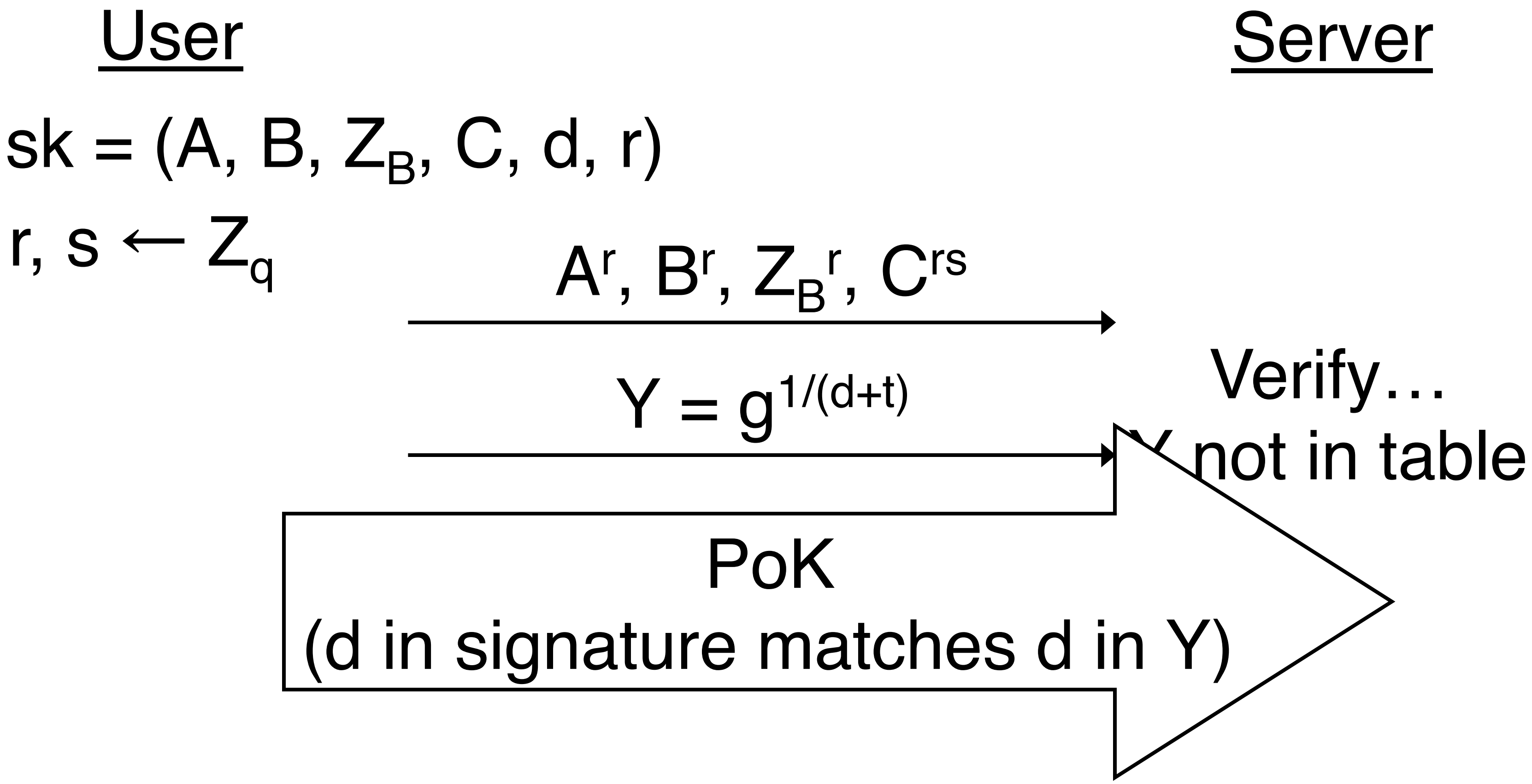
$a \leftarrow Z_q$

$g^a, g^{ay}, g^{ayz}, g^{ax}M^{axy}$

Verify…

Construction (further detail)

- **Login** (epoch t)



User

Server

sk = (A, B, Z_B, C, d, r)

r, s ← Z_q

$A^r, B^r, {Z_B}^r, C^{rs}$

$Y = g^{1/(d+t)}$

Verify…

Y not in table

PoK

(d in signature matches d in Y)

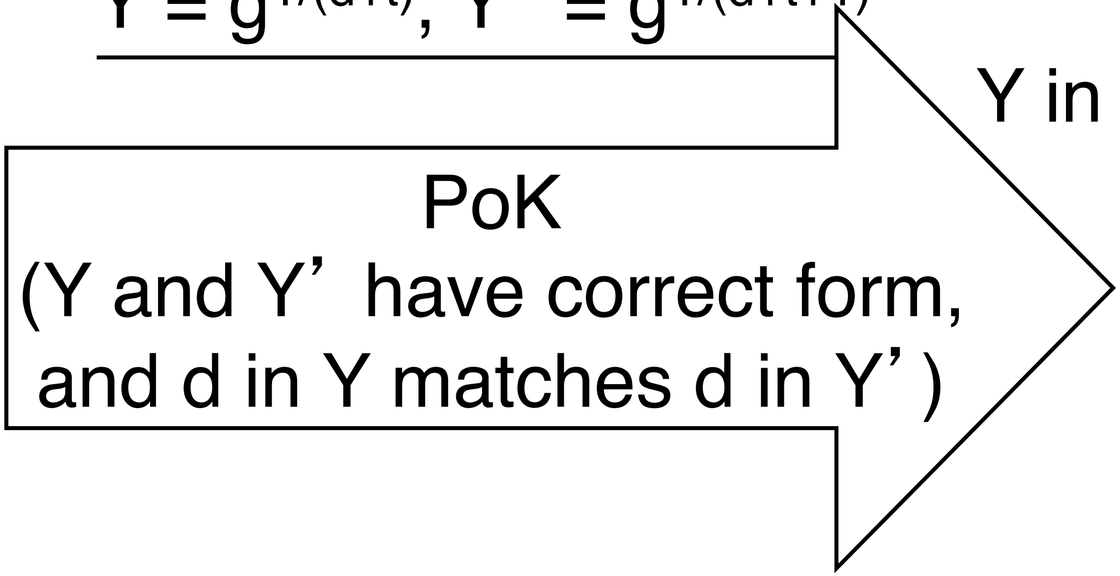
Construction (further detail)

- **Link** (epoch t)

User

Server

sk = (A, B, Z_B, C, d, r)



$Y = g^{1/(d+t)}$, $Y' = g^{1/(d+t+1)}$

Y in table?

PoK
(Y and Y' have correct form,
and d in Y matches d in Y')

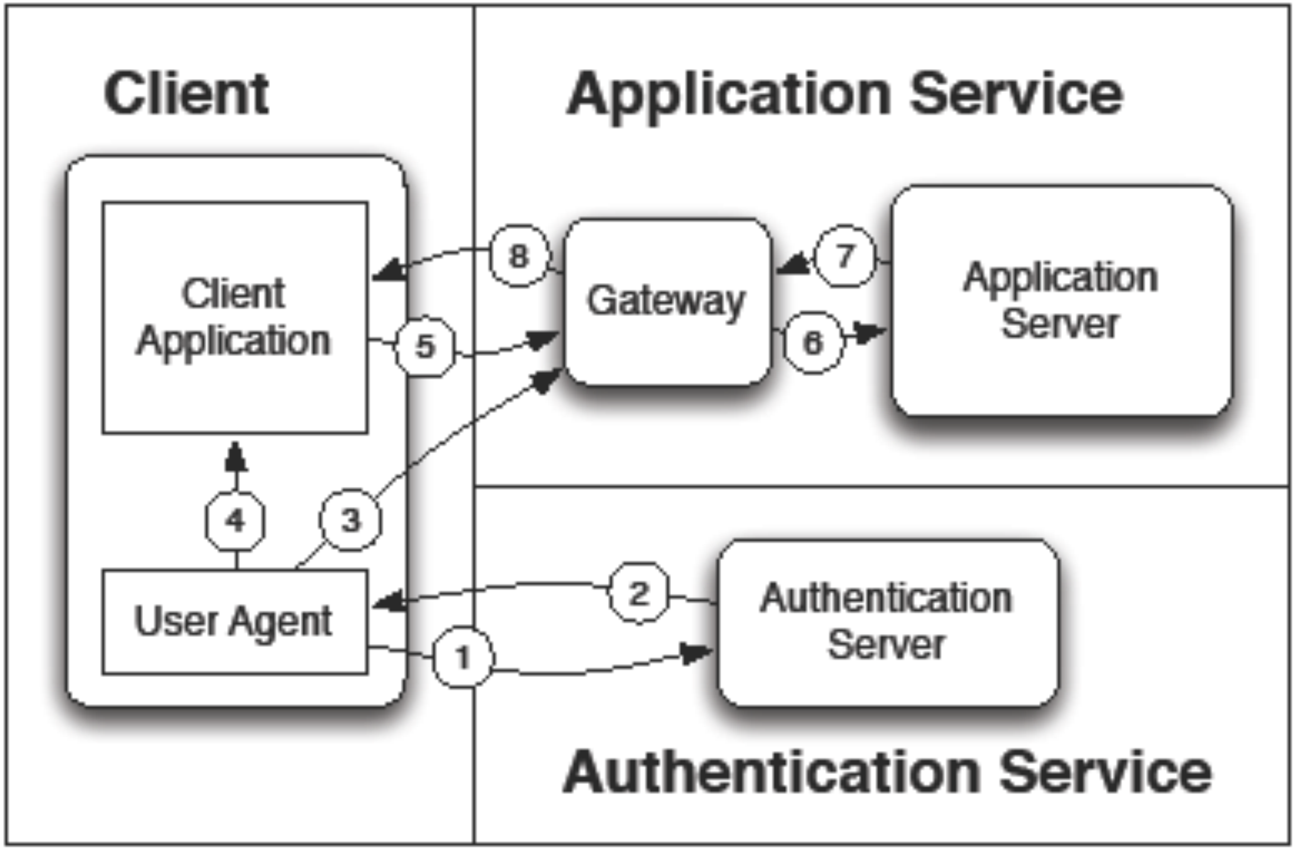
Construction (further detail)

- ZK proofs (of knowledge) fairly standard
  - Made non-interactive using Fiat-Shamir

Security guarantees

- Soundness holds under **LRSW assumption** (essentially, unforgeability of CL signatures)
- Anonymity holds under **DDHI assumption**
  - $g^{1/x}$ “looks random” even given g^x, …, g^{x^n}
- Note: in our security proofs, we assume extraction from all ZKPoKs is possible
  - Can be enforced if interactive proofs are used and sequentiality is enforced
  - Heuristic security if Fiat-Shamir proofs are used

System architecture



Notes

- Only loose synchronization needed
  - Server sends timestamp when connection is established
  - User caches previous timestamp to prevent rollback attacks on anonymity

- Login + (multiple) link(s) are done more efficiently than running Login, Link, …

Implementation

- Using PBC library [Lynn] and PolarSSL
  - Symmetric pairing; 160-bit elliptic-curve group over 512-bit field
- 1400 loc
- Pre-processing used when possible

Raw performance

	User	**Server**
Login	13.5 ms	7.9 ms
Link	1.3 ms	0.72 ms

(quad-core 2.66 GHz Intel Core 2 CPU, 8GB RAM)

Evaluation I

- Integrated our system into a streaming-music service
  - 7500 users
  - Epoch length = 15 seconds
  - Acceptable performance in terms of playback delay/latency; details in paper

Evaluation II

- Anonymous public-transit passes
  - Epoch length = 5 minutes
  - Estimate <10 servers could handle BART peak-traffic volumes

- Implemented user agent as Android app
  - Login message displayed as QR code for physical scanner to read
  - No network connectivity required
  - Login time: 220 ms (HTC Evo 3D)

Conclusions

- Design, implementation, and evaluation of a system providing **anonymous subscriptions**
- Formal definitions, cryptographic proofs
- Performance acceptable for practical applications