Building Privacy-Preserving Cryptographic Credentials from Federated Online Identities

John Maheswaran
PhD Dissertation Defense, 6/24/2015
Department of Computer Science
Yale University

Committee:
Bryan Ford (adviser)
Joan Feigenbaum
Ramakrishna Gummadi
Anil Somayaji (Carleton University)
Roadmap

1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
Roadmap

1. **Background**

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
Background: Federated Authentication
Background: Federated Authentication
Background: Federated Authentication
Background: Federated Authentication
Background: Federated Authentication
Background: Federated Authentication

- Popular for managing online identities
- Examples: Facebook and PayPal
- Authentication protocols such as OpenID/OAuth
- Privacy cost: ID provider and applications can track users across all sites
Federated Authentication \textbf{Privacy Concerns}

- ID providers learns every application user logs into
- ID providers learns login time to every application for a user
- ID provider can impersonate user on applications
- Applications learn the user’s true identity
- Applications learn user profile details e.g. friends lists, location
Federated Authentication **Privacy Concerns**

- Applications can edit user profile on ID provider e.g. post to timeline, edit personal info
- Applications can link user behavior across sites
- User data can be tracked and sold to advertisers
- Compromised federated ID account can log in as that user to all applications
Motivating Use Case: Wikipedia Anonymous Editing

- Privacy preserving login to Wikipedia
- In favor of anonymous editing
- Anonymous editing often abused - vandalism/spam
- Anonymous yet abuse resistant editing
- Allow users to edit pages without revealing their identities
- Allow admins to sanction site abusers
Motivating Use Case: Group Authenticated SecureDrop

• Verifiable whistleblowing without compromising privacy

• Allow a journalist to authenticate leaked documents without compromising source anonymity

• A whistleblower authenticates as a member of a group and signs document

• Journalist knows that the document came from a director at Evil Corp. Inc. but does not know which one
Related Work

• **PseudolD** Dey and Weis. [HotPets ’10]
  - privacy protected federated login
  - does not handle key assignment or Sybil resistance

• **Location privacy via private proximity testing** Narayanan et al. [NDSS ’11]
  - Proposed using social network as a PKI

• **Opaak** Maganis et al. [MobiSys ’12]
  - provides Sybil resistance by relying on a cellphone as scare resource.

• **SudoWeb** Kontaxis et al. [Information Security 2011]
  - looked at limiting the amount of Facebook information disclosed to third party sites
  - did not consider anonymous online IDs
Roadmap

1. Background

2. **Work Overview**

3. System Architecture

4. Credential Producers and Consumers
   - At-Large Credentials
   - Group Credentials

5. Evaluation

6. Conclusions
Work Overview

• [Poster] Crypto-Book: Privacy Preserving Online Identities; John Maheswaran, David Isaac Wolinsky, Bryan Ford; SOSP ’13 Poster Session (Symposium on Operating Systems Principles); and Diversity ’13 Poster Session (Workshop on Diversity in Systems Research)

• [Extended abstract/WIP] Crypto-Book: Privacy Preserving Online Identities; John Maheswaran, David Isaac Wolinsky, Bryan Ford; SOSP ’13 Works In Progress (WIP) Session (Symposium on Operating Systems Principles)

• [Paper] Crypto-Book: An Architecture for Privacy Preserving Online Identities; John Maheswaran, David Isaac Wolinsky, Bryan Ford; HotNets ’13 (Hot Topics in Networks ’13)
Work Overview


- [Paper (under submission)] **Building Privacy-Preserving Cryptographic Credentials from Federated Online Identities**; John Maheswaran, Daniel Jackowitz, Ennan Zhai, David Isaac Wolinsky, Bryan Ford; CoNEXT ’15 (ACM Conference on emerging Networking Experiments and Technologies)
Press coverage

• The workshop on diversity in systems research 2013; Christopher Stewart and Vishakha Gupta; *ACM SIGOPS Operating Systems Review* 48.1 (2014): 103-106.

• The federation of our digital identities; *Is Nerd Science blog*; http://isnerd.co/2014/07/05/federated-identity-privacy-namecoin/

Online resources

• Open source code is available on GitHub:
  • github.com/jyale/cobra

• Project websites:
  • www.crypto-book.com
  • www.cryptobook.ninja
Roadmap

1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
System Components

- Client
- Federated ID Provider
- Applications
- Credential Producers
- Credential Consumers
System Components

- **Client**
- **Federated ID Provider**
- **Credential Producers**
- **Applications**
- **Credential Consumers**

Verify a client’s ID with federated ID provider, then issue client with privacy preserving credentials.

Verify a client’s privacy preserving credentials and authenticate client to applications.
Security Properties

• **Anonymity** No single party can unmask a pseudonym to a federated ID

• **Unlinkability** It is not possible to tell if two pseudonyms are controlled by the same person

• **Accountability** (abuse resistance) A user can be punished if they misbehave (e.g. spam/troll)

• **Unforgeability** (no impersonation) No one can act as the user and authenticate as them
Threat Model: **Threats**

- **Clients** post low quality content/spam

- **Federated ID providers and applications**
  - de-anonymize client
  - learn what applications client accesses

- **Multiple applications** link client’s identity across sites
Threat Model: Assumptions

- At most \((t-1)\) of \(n\) credential producers are dishonest. Others are honest-but-curious.

- Do not consider network level attacks. Clients can connect to system components via anonymous networks (e.g. Tor).

- Anonymous network communication/cryptographic primitive compromise are outside of scope.
Client

• Person browsing the web
• Interacts with other system components via browser
• Interacts with all other components in system
• Goal is to login to and use a web application
Application

• A web site that someone wants to use

• Client authenticates to log in to their account on that website

• Many applications now support federated authentication (e.g. Log in with Facebook/Log in with LinkedIn etc)

• Examples:

  ![Quora](https://via.placeholder.com/150)

  ![Pinterest](https://via.placeholder.com/150)

  ![Stack Overflow](https://via.placeholder.com/150)

  ![Slideshare](https://via.placeholder.com/150)

  ![Venmo](https://via.placeholder.com/150)
Non-federated client-application interaction
Non-federated Client-Application interaction
Non-federated Client-Application interaction

1. User navigates to website
Non-federated Client-Application interaction

1. User navigates to website

2. Site prompts user for username and password

Client

Application
Non-federated Client-Application interaction

1. User navigates to website
2. Site prompts user for username and password
3. Username and password

Application

Client
Non-federated Client-Application interaction

1. User navigates to website
2. Site prompts user for username and password
3. Username and password
4. Application hashes password and checks it against the password hash stored in database for that username
Non-federated Client-Application interaction

1. User navigates to website
2. Site prompts user for username and password
3. Username and password
4. Application hashes password and checks it against the password hash stored in database for that username
5.(a). If password hash matches saved hash, authenticate the client as “username”
5.(b). If password hash does not match saved hash, do not authenticate the user, display an error message and ask user to retype their username and password
Federated Identity Provider

- Authenticates users for applications
- Often a social network or other identity provider
- Financial ID providers (e.g. PayPal) require real world verification - Higher barrier to entry
- Authorize access/modification of profile data
- Examples:
Federated Authentication Interaction

High level

Client

Federated ID Provider

Application
Federated Authentication Interaction (high level)
Federated Authentication Interaction (high level)

1. Prove this is your account

Client

Federated ID Provider

Application
Federated Authentication Interaction *(high level)*

1. Prove this is your account
2. Request OAuth token for that account

---

Client

Federated ID Provider

Application
Federated Authentication Interaction (high level)

1. Prove this is your account
2. Request OAuth token for that account
3. OAuth token

Client

Federated ID Provider

Application
Federated Authentication Interaction *(high level)*

1. Prove this is your account
2. Request OAuth token for that account
3. OAuth token
4. OAuth token
Federated Authentication Interaction (high level)

1. Prove this is your account
2. Request OAuth token for that account
3. OAuth token
4. OAuth token
5. Verify OAuth token and access user data
System Architecture

1. Verify identity

2. OAuth API

3. Obtain credentials

4. Authenticate using credentials

5. OAuth API

6. Use applications

Client

Federated ID Provider(s)

Credential Producers

Credential Consumers

Applications
Federated ID Authentication

Detailed view
Federated Authentication Interaction

Client

Federated ID Provider

Application
Federated Authentication Interaction

1. User navigates to website

Client

Federated ID Provider

Application
Federated Authentication Interaction

1. User navigates to website.
2. Login page.

Client

Federated ID Provider

Application
Federated Authentication Interaction

1. User navigates to website
2. Login page
3. User clicks to “Log in with X”
Federated Authentication Interaction

1. User navigates to website
2. Login page
3. User clicks to "Log in with X"
4. Redirect client to federated ID login page
Federated Authentication Interaction

4. Redirect client to federated ID login page
Federated Authentication Interaction

4. Redirect client to federated ID login page

5. Client is redirected and requests federated ID login page
Federated Authentication Interaction

3. Client is redirected and requests federated ID login page
Federated Authentication Interaction

5. Client is redirected and requests federated ID login page
6. Login page

Federated ID Provider

Client

Application
Federated Authentication Interaction

5. Client is redirected and requests federated ID login page
6. Login page
7. Fed ID username and password
Federated Authentication Interaction

5. Client is redirected and requests federated ID login page
6. Login page
7. Fed ID username and password
8. Verify username and password. **Prompt user to authorize app**
Federated Authentication Interaction

5. Client is redirected and requests federated ID login page
6. Login page
7. Fed ID username and password
8. Verify username and password. Prompt user to authorize app
9.(a). Successfully verified, issue OAuth token
9.(b). Authentication error, display "login failed" error message
Federated Authentication Interaction

9.(a). Successfully verified, issue OAuth token.
9.(b). Authentication error, display "login failed" error message.

10. OAuth token via redirect as URL parameter: example.com/page.php?access_token=AFB34
Federated Authentication Interaction

10. OAuth token via redirect as URL parameter:
example.com/page.php&access_token=AFB34
Federated Authentication Interaction

Client

Federated ID Provider

Application

10. OAuth token
Federated Authentication Interaction

Client

10. OAuth token

Federated ID Provider

11. OAuth token

Application
Federated Authentication Interaction

11. OAuth token
Federated Authentication Interaction

11. OAuth token

12. Verify OAuth token
Federated Authentication Interaction

Client

11. OAuth token

Federated ID Provider

12. Verify OAuth token

Application

13. Verification result
Federated Authentication Interaction

11. OAuth token

14. Request user data, e.g. user ID

Client

Federated ID Provider

Application

12. Verify OAuth token

13. Verification result
Federated Authentication Interaction

11. OAuth token

12. Verify OAuth token

13. Verification result

14. Request user data, e.g. user ID

15. User ID
Federated Authentication Interaction

Federated ID Provider -> Application

15. User ID
Federated Authentication Interaction

15. User ID

16. Look up user ID in database, retrieve user data
Federated Authentication Interaction

15. User ID

16. Look up user ID in database, retrieve user data

17. Welcome page for user logged in with that user ID
System Architecture

1. Verify identity

Client

3. Obtain credentials

Federated ID Provider(s)

2. OAuth API

Credential Producers

4. Authenticate using credentials

Credential Consumers

5. OAuth API

Applications

6. Use applications
1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
Definition: Privacy Preserving Credential

- A client uses a privacy preserving credential to prove they own a pseudonym, without revealing their true identity
- Using privacy preserving cryptographic techniques
Credential Producers

• Several credential producer servers collectively act to assign credentials to clients

• (t,n) threshold model - t of n servers can collectively assign a credential to a client

• Acts as an “application” in OAuth protocol to authenticate client with federated ID provider
System Architecture

1. Verify identity
2. OAuth API
3. Obtain credentials
4. Authenticate using credentials
5. OAuth API
6. Use applications
System Architecture

1. Verify identity

2. OAuth API

3. Obtain credentials

4. Authenticate using credentials

5. OAuth API

6. Use applications
Credential Assignment Mechanism

Obtaining OAuth tokens
Credential Assignment Mechanism

1. Username and password

Client

facebook

App1 App2 App3
Credential Assignment Mechanism

1. Username and password

2. Login result

Client

App1

App2

App3

facebook
Credential Assignment Mechanism

1. Username and password
2. Login result
3. Request OAuth token (one per app)

Requests performed in parallel. Automated by a Chrome extension so user does not have to manually repeat the same task.
Credential Assignment Mechanism

4. Do you want to authorize this app?
Credential Assignment Mechanism

4. Do you want to authorize this app?
Credential Assignment Mechanism

1. Authorize apps

5. Authorize apps

Client

facebook

App1  App2  App3
Credential Assignment Mechanism

6. OAuth token for App1
6. OAuth token for App2
6. OAuth token for App3

Client

facebook

App1

App2

App3
Credential Assignment Mechanism

Client now has one OAuth token per app. Each app corresponds to one credential producer server.

OAuth token for App1
OAuth token for App2
OAuth token for App3
Multiple ID provider use case: This process is performed for each federated ID provider. The user only has to enter their username and password once per federated ID provider. The other steps are automated by a Chrome extension.
Credential Assignment Mechanism

Client

facebook

App1  App2  App3

PayPal™

App1  App2  App3
System Architecture

1. Verify identity

2. OAuth API

3. Obtain credentials

4. Authenticate using credentials

5. OAuth API

6. Use applications
System Architecture

1. Verify identity

2. OAuth API

3. Obtain credentials

4. Authenticate using credentials

5. OAuth API

6. Use applications
Credential Assignment Mechanism

Obtaining credentials
Credential Assignment Mechanism
Credential Assignment Mechanism

Credential Producers
1. App1 OAuth token
1. App2 OAuth token
1. App3 OAuth token

Client

1. App3 OAuth token
1. App2 OAuth token
1. App1 OAuth token
Credential Assignment Mechanism

Client

2. App1 OAuth token

2. App2 OAuth token

2. App3 OAuth token

Credential Producers
Credential Assignment Mechanism

3. Each app verifies corresponding token

2. App1 OAuth token
2. App2 OAuth token
2. App3 OAuth token

Client

Credential Producers
Credential Assignment Mechanism

3. Each app verifies corresponding token

4. Verification result

Credential Producers

Client
5. If OAuth token verified successfully, each credential producer returns its share of the credential to the client.
Credential Assignment Mechanism

6. Credential shares
Credential Assignment Mechanism

7. Client combines credential shares to obtain overall credential.
System Architecture

1. Verify identity
2. OAuth API
3. Obtain credentials
4. Authenticate using credentials
5. OAuth API
6. Use applications
System Architecture

1. Verify identity

2. OAuth API

3. Obtain credentials

4. Authenticate using credentials

5. OAuth API

6. Use applications
Credential Consumers

Authenticating with and using privacy preserving credentials
Credential Consumers

- Map credentials to pseudonyms
- Pseudonyms produced are not linkable back to federated IDs

**OAuth provider consumers:** Expose pseudonym IDs to applications via OAuth. Easily integrate with applications already using federated authentication

- Application-embedded consumer directly in application
System Architecture

Credential Producers

Credential Consumer

Client
System Architecture

Client

Credential Producers

Credential Consumer

0. (a). Challenge (in web page)
System Architecture

0.(b). Client signs challenge using credentials (signing performed by browser extension)

Client

Credential Producers

Credential Consumer

0.(a). Challenge (in web page)
System Architecture

1. Browser extension fills in hidden form with signature

Client

Credential Producers

Credential Consumer
System Architecture

1. Browser extension fills in hidden form with signature

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4317913668590612421815701</td>
<td>1257858443935756067593364</td>
</tr>
<tr>
<td>8352791428378484108076735</td>
<td>1430165293341</td>
</tr>
<tr>
<td>1430165293560</td>
<td>groups/221288</td>
</tr>
</tbody>
</table>
2. Form containing signature is submitted by clicking “login” button.
System Architecture

1. Consumer requests public key(s)

2. Credential Producers

3. Credential Consumer
System Architecture

Credential Producers

4. Public keys

Credential Consumer

Client
System Architecture

Credential Producers

Credential Consumer

5. Consumer verifies client credentials
6.(a). If credential verifies successfully, issue OAuth token.
6.(b). Otherwise issue login error message
System Architecture

Client

Credential Producers

Credential Consumer

Application

7. OAuth token
System Architecture

Client

Credential Producers

Application

Credential Consumer

8. OAuth token
System Architecture

8. OAuth token

9. Consumer verifies token

Credential Producers

Client

Credential Consumer

Application
System Architecture

Credential Producers

Client

Credential Consumer

Application

8. OAuth token

9. Consumer verifies token

10. Verification result, pseudonym
System Architecture

Credential Producers

Client has now successfully authenticated to the application

Client

Credential Consumer

1. Logged in web page for user as pseudonym

9. Consumer verifies token

10. Verification result, pseudonym

Application
System Architecture

1. Verify identity

2. OAuth API

3. Obtain credentials

4. Authenticate using credentials

5. OAuth API

6. Use applications

Client

Federated ID Provider(s)

Credential Producers

Credential Consumers

Applications
Roadmap

1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
At-Large Credential Scheme

Can use for privacy preserving Wikipedia login
At-Large Credential Scheme

• Represents that the user has been verified as the owner of some federated identity.

• Anonymity set is implicitly the users who have collected a credential

• Accountability through rate limiting: producers restrict number of credentials a federated ID gets within a period of time

• Can include credential attributes, such as “age over 18” or “identity active for at least one year”
Technical Building Block: Blind Signatures

1. Request a signature on a blinded message

2. Signer cannot learn message content

3. Third party can verify unblinded signature

\[ m \rightarrow m' \rightarrow m', s' \rightarrow m, s \]
Technical Building Block: Blind Signatures

- Client is the requester
- Each credential producer is a signer
- Credential consumers are verifiers
At-Large Credential Scheme

Credential Producers

Client

Credential Consumers
At-Large Credential Scheme

1. Producers publish initialization info

Credential Producers

Credential Consumers
At-Large Credential Scheme

1. Producers publish initialization info

2. Client blinds message using published info

Credential Consumers

Credential Producers
At-Large Credential Scheme

1. Producers publish initialization info

2. Client blinds message using published info

3. Blinded message $m'$

Credential Producers

Credential Consumers
At-Large Credential Scheme

1. Producers publish initialization info
2. Client blinds message using published info
3. Blinded message $m'$
4. Signs blinded message $(m',s')$

Credential Producers

Credential Consumers
At-Large Credential Scheme

1. Producers publish initialization info
2. Client blinds message using published info
3. Blinded message \( m' \)
4. Signs blinded message \((m', s')\)
5. \((m', s')\)
At-Large Credential Scheme

1. Producers publish initialization info

2. Client blinds message using published info

3. Blinded message \( m' \)

4. Signs blinded message \((m',s')\)

5. \((m',s')\)

6. Unblinds message using \((m',s') \rightarrow (m,s)\)

Credential Producers

Credential Consumers
At-Large Credential Scheme

1. Producers publish initialization info

2. Client blinds message using published info

3. Blinded message \( m' \)

4. Signs blinded message \( (m', s') \)

5. \( (m', s') \)

6. Unblinds message using \( (m', s') \rightarrow (m, s) \)

7. \( (m, s) \)

Credential Producers

Credential Consumers
At-Large Credential Scheme

1. Producers publish initialization info
2. Client blinds message using published info
3. Blinded message \( m' \)
4. Signs blinded message \((m',s')\)
5. \((m',s')\)
6. Unblinds signature \((m',s') \rightarrow (m,s)\)
7. \((m,s)\)
8. Verifies \((m,s)\) against producer’s public key.
At-Large Credential Scheme

1. Producers publish initialization info
2. Client blinds message using published info
3. Blinded message $m'$
4. Signs blinded message $(m',s')$
5. $(m',s')$
6. Unblinds message using $(m',s') \rightarrow (m,s)$
7. $(m,s)$
8. Verifies $(m,s)$ against producer’s public key.
9. If $(t,n)$ threshold is reached client is authenticated to application.
Roadmap

1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
Group Credential Scheme

Provides k-anonymous authentication

Verifiable whistleblowing/private chat room use cases
Group Credential Scheme

• Allows a client to authenticate explicitly as some member of a larger, well defined set of users (e.g. a Facebook group)

• The group credential scheme provides $k$-anonymity, the client is anonymous among a set of $k$ people

• Based on linkable ring signatures
Technical Building Block: Linkable Ring Signatures

• Created by member of a group of users

• Third party can verify:
  – Some member of the group created signature
  – Whether two signatures were created by same signer

• Third party cannot discover
  – Which specific user created the signature
Technical Building Block: Linkable Ring Signatures

• LRS has *linkage tag*
  – If a client generates two LRSs, will have the same linkage tag
  – Means LRSs can be linked across time

• Linkage tag provides *accountability*
  – privacy preserving mapping between fed IDs and pseudonyms
Group Setup
Group Credential Scheme

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

1. List of group members

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

1. List of group members

Chat Room Application
Group Credential Scheme

1. List of group members

Client

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

1. List of group members

2. Creates a chat room with those members
Group Credential Scheme

- The client collects their private key shares from at least $t$ of $n$ credential producers.

- Client combines shares to give private key, saved in browser extension.

- Client collects public keys from credential producers (no authentication).

- Credential consumers issue challenge to client, which client signs with LRS and is the authenticated to application.
Group Credential Scheme

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

1. Client requests public keys

Client

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

1. Client requests public keys
2. Public keys

Client

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

1. Client requests public keys
2. Public keys
3. Client requests to log in to a chat room
Group Credential Scheme

1. Client requests public keys
2. Public keys
3. Client requests to log in to a chat room
4. Challenge $m$

Client → Credential Producers → Client → Credential Consumer → Chat Room Application
Group Credential Scheme

1. Client requests public keys
2. Public keys
3. Client requests to log in to a chat room
4. Challenge $m$
5. Client signs challenge using private key and public key list to give a linkable ring signature (LRS)
Group Credential Scheme

5. Client signs challenge using private key and public key list to give a linkable ring signature (LRS)
Group Credential Scheme

5. Client signs challenge using private key and public key list to give a linkable ring signature (LRS)

6. LRS

Credential Producers

Credential Consumer

Chat Room Application
Group Credential Scheme

Client

Credential Producers

Credential Consumer

Chat Room Application

6. LRS

7. Verify LRS against public keys
Group Credential Scheme

1. **Credential Producers**
2. **Credential Consumer**
3. **Client**
4. **Chat Room Application**

**Steps:**

8. OAuth token giving access to chat room
7. Verify LRS against public keys
Group Credential Scheme

1. Credential Producers
2. Credential Consumer
3. Chat Room Application
4. OAuth token giving access to chat room
5. OAuth token to access chat room
6. Verify LRS against public keys
7. Client
8. OAuth token giving access to chat room
9. OAuth token to access chat room
Group Credential Scheme

Client

Credential Producers

Chat Room Application

8. OAuth token giving access to chat room

9. OAuth token to access chat room

10. OAuth token

11. Verify OAuth token

12. Verification result
Group Credential Scheme: Chat Room

DeDiS Group anonymous comment board

Write your comment below to post to the anonymous comment board below.

You are logged in as: Anonymous radiator

Post:

All comments

Colored word is anonymous username, text next to it is the message

Anonymous radiator fgghghgh
Anonymous radiator good stuff
Anonymous radiator good stuff
Anonymous radiator lol
Anonymous knee hghhkhh
Anonymous radiator Hello
Anonymous radiator BEAST!
Anonymous radiator this is awesome
Anonymous radiator hahaha
Anonymous word good stuff
Anonymous word nice!!!!

Pseudonym is based on linkage tag of LRS. User will always be given same pseudonym for a given chat group.

Users post comments here to the chat group.

Comments are displayed live to other group members of the group.

Other group members have different pseudonyms.

Shareable Link

Use this link to share this chat group with other members of the group.

http://cryptobook.ninja/cobra2:
Roadmap

1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   - At-Large Credentials
   - Group Credentials

5. Evaluation

6. Conclusions
Evaluation
Evaluation: Experimental Setup

- **Clients:** consumer laptops
  - 2.4GHz Intel Core i5 processors
  - 8GB of RAM.

- **Credential producers:** PlanetLab nodes
  - 2.4GHz Intel Xeon processor
  - 4GB of RAM

- **Credential consumers:** commercial shared hosting
  - 2.4GHz Intel Xeon processors
  - 16GB of RAM

- Client performs this setup step only once, the first time they use the system.
Evaluation: Producing At-large Credentials

Blind Signature Size (bandwidth)

<table>
<thead>
<tr>
<th>Key Parameters</th>
<th>Signature Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1024,160)</td>
<td>210</td>
</tr>
<tr>
<td>(2048,224)</td>
<td>287</td>
</tr>
<tr>
<td>(2048,256)</td>
<td>325</td>
</tr>
<tr>
<td>(3072,256)</td>
<td>326</td>
</tr>
</tbody>
</table>

• Network overhead between client and producer depends on the size (and hence strength) of the signature.
Evaluation: Producing/Consuming At-large Credentials

- For a 2048-bit signing key, credential production takes approximately 50ms of computation time, verification takes less than 20ms,
Evaluation: Producing Group Credentials

- **Key pair generation:** The first time a key pair is requested it is collectively generated by the producers
Evaluation: Producing Group Credentials

- **Key retrieval**: requests to all producers are performed in parallel. Private keys include Facebook authentication.
Evaluation: Consuming Credentials

End-to-end group credentials evaluation

<table>
<thead>
<tr>
<th>Entity</th>
<th>Operation</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Produce LRS</td>
<td>0.257</td>
</tr>
<tr>
<td>Credential Consumer</td>
<td>Fetch Public Keys</td>
<td>1.011</td>
</tr>
<tr>
<td></td>
<td>Verify LRS</td>
<td>0.035</td>
</tr>
<tr>
<td>Client-Consumer Network Latencies</td>
<td></td>
<td>0.304</td>
</tr>
<tr>
<td><strong>Total User-Observable</strong></td>
<td></td>
<td><strong>1.607</strong></td>
</tr>
</tbody>
</table>

- **Group credential**: ten Facebook identities for DeDiS group
- 1.2s overhead vs non-anonymous federated authentication
Evaluation: Consuming Credentials

- For ring size \(\sim 100\) (2048-bit keys), operations <1s
Evaluation: Consuming Credentials

- For ring sizes ~100 (2048-bit keys), signatures <10KB.
Roadmap

1. Background

2. Work Overview

3. System Architecture

4. Credential Producers and Consumers
   • At-Large Credentials
   • Group Credentials

5. Evaluation

6. Conclusions
Conclusions and Future Directions

• Crypto-Book is a pluggable architecture for providing privacy preserving credentials based on federated identity providers.

• Experimental evaluations show acceptable overheads

• Privacy conscious applications can be developed on top of this platform

• Pluggable nature means other privacy preserving technologies can be integrated in future
Acknowledgements

• Thanks to my adviser, Bryan Ford and committee members, Joan Feigenbaum, Ramki Gummadi, Anil Somayaji

• Collaborators: Danny Jackowitz, Ennan Zhai, David Isaac Wolinsky and DeDiS research group members Ewa Syta, Weiyi Wu and Jose Faleiro

• Undergraduate adviser: The late Robin Milner (University of Cambridge, UK)

• PhD funding sources: Yale University, NSF grant CCF-0916389, DARPA SAFER grant N66001-11-C-4018

• Thanks to the everyone in the Yale Computer Science department and everyone else for attending
In conclusion,

AAAAAAAAAAAAAA!!!

The best thesis defense is a good thesis offense.
Thanks!
[Subsequent slides are were removed from presentation and may be incomplete]
“Two Principles of Deadlines:
1. All deadlines converge on the same day—Deadline Day.
2. Every day is Deadline Day.”

—Bryan Ford
Federated Authentication Interaction
Credential Assignment Mechanism
Credential Assignment Mechanism

Client

Federated ID Provider(s)

App1
App2
App3

Credential Producers
System Architecture

Client

Credential Producers

Credential Consumer

Application
At-Large Credential Scheme

Client

Credential Producers

Credential Consumers
Federated Authentication Interaction

1. User navigates to website with that user ID

2. Login page

3. User clicks to "Log in with X"

4. Redirect client to federated ID login page

5. Client is redirected and requests federated ID login page

6. Login page

7. Fed ID username and password

8. Federated ID provider

9. (a). Successfully verified, issue OAuth token

9. (b). Authentication error, display "login failed" error message

10. OAuth token via redirect as URL parameter: example.com/page.php?access_token=AFB34

11. OAuth token

12. Verify OAuth token

13. Verification result

14. Request user data, e.g. user ID

15. User ID

16. Look up user ID in database, retrieve user data

17. Welcome page for user logged in with that user ID
Chat Room Application

1. Client requests public keys
2. Public keys