**Abstract**

# Building Privacy-Preserving Cryptographic Credentials from Federated Online Identities

John Maheswaran

2015

Third-party applications such as Quora or StackOverflow allow users to log in through a federated identity provider such as Facebook (Log in with Facebook), Google+ or Twitter. This process is called federated authentication. Examples of federated identity providers include social networks as well as other non-social network identity providers such as PayPal.

Federated identity providers have gained widespread popularity among users as a way to manage their online identity across the web. While protocols like OAuth and OpenID allow users to maintain a single set of credentials for federated authentication, such federated login can leak privacy-sensitive profile information, making the user's online activity more easily tracked.

To protect themselves, users could forego using such identities altogether, or limit the content of their profiles. Ideally, users could leverage their federated identities but in a way as to prevent third party applications from accessing sensitive information. While anonymous authentication techniques have been proposed, their practicality depend on such technologies as PGP or complex encryption algorithms which most users lack the knowledge or motivation to use effectively.

While federated identity providers offer a convenient and increasingly popular mechanism for federated authentication, unfortunately, they also exacerbate many privacy and tracking risks. We present Crypto-Book, a privacy preserving layer enabling federated authentication while reducing these risks.

Crypto-Book relies on a set of independently managed servers that collectively assign each federated identity credentials (either a public/private keypair or blinded signed messages). We propose two components, "credential producers" that create and issue clients with privacy preserving credentials, and "credendial consumers" that verify these privacy preserving credentials for authentication of clients to third party applications.

The credential producer servers have split trust and use a (t,n)-threshold cryptosystem to collaboratively generate client credentials. Using their credentials, clients can then leverage anonymous authentication techniques such as linkable ring signatures or blind signatures to log into third party applications via credential consumers, while preserving privacy.

We have implemented our system and demonstrate its use with four distinct applications: a Wiki system, an anonymous group communication system, a whistle blower submission system based on SecureDrop, and a privacy preserving chat room system. Our results show that for anonymity sets of size 100 and 2048-bit DSA keys, Crypto-Book ring signature authentication takes 1.641s for signature generation by the client, 1.632s for signature verification on the server, and requires 8.761KB of communication bandwidth. Similarly for partially blind signature authentication, each phase takes under 0.05s and requires 0.325KB of bandwidth.

Crypto-Book is practical and has low overhead: We deployed a privacy preserving chat room system built on top of the Crypto-Book architecture. Within the deployment within our research group, Crypto-Book group authentication took 1.607s end-to-end, an overhead of 1.2s compared to traditional non privacy preserving federated authentication.

# Building Privacy-Preserving Cryptographic Credentials from Federated Online Identities

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
John Maheswaran

Dissertation Director: Bryan A. Ford

December 2015

# Contents

# List of Figures

# List of Tables

# Acknowledgments

First and foremost, I would like to thank my advisor, Bryan Ford, for all his guidance, help and advice in the process of preparing this dissertation and throughout the course of my PhD at Yale. I would also like to thank all my dissertation committe members, Joan Feigenbaum, Ramki Gummadi and Anil Somayaji. I would also like to thank the members of our research group, Danny Jackowitz, Ennan Zhai, Weiyi Wu, Ewa Syta and David Isaac Wolinsky for all their help, support and encouragement and for all the useful feedback received at All Hands meetings. I also thank Jose Faleiro.

I also thank my undergraduate final year project advisor and Turing Award laureate, the late Robin Milner, for sparking my interest in computer science research.

Additionally, I would like to acknowledge Google and the Security team especially Andrew Sacamano, Cory Hardman and Daryl Seah for facilitating work towards this disseration, providing me with industry experience and education of security and cryptography in a software engineering setting. I thank the uProxy team at Google Ideas and University of Washington, especially Lucas Dixon.

I want to thank the entire computer science department at Yale including all the faculty. I also want to thank Yale University as a whole. I am extremely grateful to have had the opportunity to complete my doctoral education here.

I also want thank all my friends for making my time at Yale enjoyable. I also

want to thank my family.

Finally I want to acknowledge funding sources. I am thankful to be a recipient of a Yale University Fellowship which supported me during my first year.

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR Systems Center Pacific, SAFER Contract No. N66001-11-C-4018. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA) or SPAWAR Systems Center Pacific.

# Chapter 1

# Introduction

*Third-party applications* (such as Quora [84] or StackOverflow [99]) allow users to log in through a *federated identity provider* such as Facebook (*Log in with Facebook*) or Google+. This process is called *federated authentication*. Examples of federated identity providers include social networks as well as other non-social network identity providers such as PayPal.

Federated identity providers have gained widespread popularity among users as a way to manage their online identity across the web. While protocols like OAuth [49, 50] and OpenID [86] allow users to maintain a single set of credentials for federated authentication, such federated login can leak privacy-sensitive profile information [46], making the user's online activity more easily tracked.

To protect themselves, users could forego using such identities altogether, or limit the content of their profiles. Ideally, users could leverage their federated identities but in a way as to prevent third party applications from accessing sensitive information. While anonymous authentication techniques have been proposed [103, 104, 5, 4, 3, 6, 10, 13, 14, 24, 51, 55, 58, 61, 62, 79, 105], their practicality depend on such technologies as PGP or complex encryption algorithms which most users lack the

knowledge or motivation to use effectively [113].

Crypto-Book interposes an anonymity layer between existing federated identity providers and the third party applications users may wish to log into. Crypto-Book prevents the federated identity provider from learning which applications a user accesses, and prevents the application from learning which user accessed their application or which *other* application the same user has accessed.

## Roadmap

Chapter 2 motivates Crypto-Book and discusses OAuth and federated login background as well as related work. Chapter 3 defines technical terms used in this dissertation. Chapter 4 offers an overview of Crypto-Book's design and outlines its threat model. Chapter 5 introduces credential producers. Chapter 6 details Crypto-Book's mechanism to assign credentials to federated identities. Chapter 7 introduces credential consumers. Chapter 8 discusses the idea of "at large" credentials and Chapter 9 discusses group credentials. Chapter 10 details privacy preservation within Crypto-Book. Chapter 11 specifies the system's security properties, outlines potential attackers and analyses how these attackers may attempt to violate system security properties and how well the system defends against these attacks. Chapter 12 discusses the current implementation and Chapter 13 lays out three application case-studies, which Chapter 14 experimentally validates. Chapter 15 discusses limitations and future work, and Chapter 16 concludes.

# Chapter 2

# Background and Related Work

**Background and Motivating Use Cases** It is often difficult to strike an appropriate balance between the following objectives:

- Supporting free speech and free association, and fighting censorship and oppression.
- Improving the quality of public discourse. Hidden behind an anonymous veil, people often say or do things they might otherwise not.

These two objectives are often at odds with each other. While Wikipedia would like to allow anonymous editing, such privileges are often abused for vandalism or sock-puppetry. We would like a system that allows users to edit pages without revealing their identity but at the same time allows the Wikipedia site administrators to sanction site abusers.

Another example placing free speech and quality of discourse are at odds is the realm of anonymous chat. A covert organization, for example, may wish to discuss sensitive issues without revealing their individual identities while at the same time limiting access to their discusssion to people within their organization, in an effort to prevent repressive authorities or other undesirable outsiders from viewing their

communications.

Whistleblowing provides another such scenario, as a journalist taking possession of sensitive documents may wish to authenticate the documents without compromising the anonymity of the source.

We used the Crypto-Book framework to implement applications motivated by the above three scenarios - CB-Wiki, CB-Dissent and CB-Drop, respectively, which we describe in Chapter 12.

## 2.1 Federated Identity Authentication

We now give a brief overview of federated identity, focusing on OAuth [49, 50]. A federated identity protocol allows a user to present credentials from an identity provider with which they have an account and authenticate themselves to a third party application without revealing their actual login information (password). Many protocols, such as OAuth, also enable the third party to gain limited access to the user's resources stored by the identity provider. OpenID [86, 87] is another widely used federated identity protocol. A typical OAuth login proceeds as follows, supposing a Facebook user wishes to log into StackOverflow.

1. The user clicks *Log in with Facebook* on StackOverflow
2. StackOverflow redirects the user to Facebook where they login using their Facebook credentials
3. The user gives permission for StackOverflow to access and/or modify their data (read contacts, post status updates and so on)
4. Facebook generates a temporary OAuth access token that corresponds to the granted permissions
5. Facebook redirects the user back to StackOverflow, passing along the access token

6. StackOverflow can now use the token to access the user's Facebook resources in line with the permissions granted by the user

### 2.1.1 Privacy Concerns with Federated Identity

Using federated authentication, a user can log in to third party applications without having to maintain separate accounts for each application. This convenience brings privacy risks, however, of which Crypto-Book focuses on the following:

- The ID provider learns every application or site the user logs into using the identity, and every *time* they use it.
- Third party sites and applications learn the user's true identity and often many profile details such as friends lists.
- Companies can link and profile a user across many sites and applications, sharing or selling profiles to advertisers.
- A compromised ID provider account gives an attacker access to the user's accounts on many third-party sites.

Additionally, whenever you visit any page containing ID provider's "Like" or "Share" button, the ID provider learns that you visited that page which enables even more detailed tracking and sale of personal information. Web applications often demand access to profile information, contacts lists, and even write access (permission to post on user's behalf) and it is sometimes unclear to users what these permissions will be used for, and not obvious after-the-fact how they were actually used.

### 2.1.2 Motivating Use-Cases for Crypto-Book

In our evaluation we use three different motivating applications which we outline here.

**Privacy-preserving "Log In with Crypto-Book"** Crypto-Book may be used to provide privacy preserving log in functionality to third party websites. A website may choose to include a "Log in with Crypto-Book" button which allows users to be authenticated via Crypto-Book. The difference between Crypto-Book privacy preserving login and existing federated authentication, such as "Log in with Twitter" is that Crypto-Book login preserves the user's privacy, while also protecting the third party websites from abuse. The website learns that the user has been authenticated by Crypto-Book, and learns a pseudonym for the user, but does not know the user's identity, nor can the website map back from the pseudonym to the user's identity.

For example, while Wikipedia would like to allow anonymous editing, such privileges are often abused for vandalism or sock-puppetry. We would like a system that allows users to edit pages without revealing their identity but at the same time allows the Wikipedia site administrators to sanction site abusers. CB-Wiki leverages Crypto-Book to provide anonymous, yet linkable editing in a collaborative environment wiki system. We built CB-Wiki using MediaWiki [73], the software behind Wikipedia.

**Abuse-resistant anonymous communication** Another example placing free speech and quality of discourse are at odds is the realm of anonymous chat. A covert organization, for example, may wish to discuss sensitive issues without revealing their individual identities while at the same time limiting access to their discusssion to people within their organization, in an effort to prevent repressive

authorities or other undesirable outsiders from viewing their communications.

The Crypto-Book architecture could be used to give web sites and services a reason not block Tor, by authenticating users anonymously in an abuse resistant manner. This would allow websites to counter anonymous abuse without compromising anonymity. The Tor Project issued a "call to arms" seeking solutions to this issue [1].

CB-Dissent shows how the integration of anonymous authentication with anonymous communication systems can better protect the identities of the users of those systems. We built CB-Dissent using Dissent [28, 114] an anonymous group communication tool.

**Ring-authenticated whistleblowing via SecureDrop** Whistleblowing provides another such scenario, as a journalist taking possession of sensitive documents may wish to authenticate the documents without compromising the anonymity of the source. CB-Drop provides anonymous document signing using Crypto-Book identities, allowing for verifiable leaks without compromising privacy. A whistleblower authenticates as a member of a "ring" so a journalist can verify that the leak came from one of the members of the ring, yet does not know which specific member leaked the document. CB-Drop extends the SecureDrop [93] open-source whistleblower platform.

Crypto-Book's focus is providing convenient, abuse resistant, anonymous authentication, but it does not by itself address the general problem of network level anonymous communication, especially under traffic analysis, as systems like Tor [38], Dissent [28, 115], and Aqua [60] do. Crypto-Book is synergistic with such systems but also usable independently, when a casual level of anonymity is desired but the user

[1] https://blog.torproject.org/blog/call-arms-helping-internet-services-accept-anonymous-users

does not wish to incur the performance costs of full anonymous forwarding.

This dissertation makes the following contributions:

- Anonymous and abuse resistant authentication using federated identities.
- A pluggable credential design, demonstrated with two different schemes (public/private key pairs with LRS, and blind signatures).
- A multiple identity provider credential assignment protocol preventing any federated identity provider from impersonating a user.
- Security properties and attack analysis of the system.
- Experiments demonstrating the practicality of Crypto-Book for authentication.

## 2.2 Related Work

Existing work on anonymous credential systems include BLAC [103, 104, 5] which supports blacklisting of anonymous credentials in certain situations. There have been a wide variety of other approaches to anonymous credential systems including those based on group signatures [58, 24, 10, 6, 3, 4], dynamic accumulators [79, 61, 14] and Nymble systems [105, 62, 55, 51].

Felt and Evans [44] examine privacy protection in social networking APIs. The deployment of public key cryptography over social networks was considered by Narayanan *et al.* [76] where they considered key exchange over social networks. They considered using social networks as a public key infrastructure (PKI), they did not implement any applications that use the public keys.

Various schemes have been proposed to protect user data *within* an online social network [69, 68, 48, 30, 54], by encrypting the content stored within the social network. However these schemes did not consider the privacy risks involved when a user uses their online social networking identity to identify themselves with third

parties such as logging into other websites using their Facebook credentials. Dey and Weis [36] proposed PseudoID, a similar system based on blind signatures [16] for privacy protected federated login, however their scheme does not handle key assignment or Sybil resistance as our work does. A similar blind signature based system was proposed by Khattak *et al.* [57]. Watanabe and Miyake [112] made initial efforts towards account checking however still did not consider key assignment. Opaak [70] is a system that attempts to provide some Sybil resistance through relying on a cellphone as a scare resource. SudoWeb [59] looked at limiting the amount of Facebook information disclosed to third party sites but did not consider fully anonymous online IDs.

Identity based encryption (IBE) refers to an encryption system where a public key can be an arbitrary string, for example a user's email address or social security number. The idea was first proposed by Shamir [95] and since then several IBE systems have been proposed [34, 53, 72, 107, 101].

# Chapter 3

# Definitions

We now introduce the technical terms that we use in this dissertation. We present the relevant terms along with the definitions as we are using them in this dissertation to clarify our intended meaning of them.

- **Federated identity provider** - An identity provider such as Facebook that allows a user to log into other websites using their username and password for the identity provider (for example Log in with Facebook, Log in with LinkedIn). Examples include social networks such as Facebook, Google+, Twitter, LinkedIn as well as other systems that support identity management such as PayPal. Also referred to as an *identity provider, ID provider* and *federated ID provider*.
- **Third party application** - A website or application that allows the user to log in through a federated identity provider. Examples include Quora [84] and StackOverflow [99]. Also referred to as an *app, third party app, application, web app* or *third party website*.
- **Federated authentication** - The process whereby a user logs into a third party application through a federated identity provider.
- **Client** - A user of a system. in this dissertation we often refer to the client as

Alice.

- **Anonymity** - The state of being not identifiable within a set of subjects.
- **Anonymity set** - To enable anonymity of a subject, there always has to be an appropriate set of subjects with potentially the same attributes. The anonymity set is the set of all possible subjects who might have caused an action. The number of subjects in the anonymity set is the *size* of the anonymity set.
- **$k$-anonymity** - A subject has $k$-anonymity [100] if they are a member of an anonymity set of size $k$.
- **Authentication** - A process that establishes the source of information, provides assurance of an entitys identity or provides assurance of the integrity of communications sessions, messages, documents or stored data [80].
- **Anonymous authentication** - The process of authenticating yourself without revealing your identity [63].
- **Pseudonym** - An identifier of a subject (a client). The subject that may be identified by the pseudonym is the holder of the pseudonym. Pseudonymity allows the holder of a pseudonym to establish a reputation while using the same pseudonym. When a user anonymously authenticates to a system they may be provided with a pseudonym to use within that system.
- **Pseudonymity** - The use of pseudonyms as identities.
- **Ring signature** - A ring signature [92] makes it possible to specify a set of possible signers without revealing which member actually produced the signature. RSA-based ring signatures [92] have the property of *deniability* [88].
- **Deniability** - The property where if you reveal your private key your previous ring signatures cannot be deanonymized [88].
- **1-out-of-$n$ group signature scheme** - A 1-out-of-$n$ group signature scheme [64] allows any member of a group of $n$ signers to generate a signature such that any

public verifier can determine if the signature is generated by a group member. A 1-out-of-$n$ group signature scheme provides $n$-anonymity where the anonymity set is the set of group members.

- **Linkable ring signature (LRS)** - A 1-out-of-$n$ group signature scheme [64] which satisfies three properties:
  1. Anonymity, or signer-indistinguishability.
  2. Linkability: That two signatures by the same signer can be linked.
  3. Spontaneity: No group secret, and thus no group manager or secret-sharing setup stage.
- **Blind signature** - A blind signature [96, 18, 15, 12, 19, 22, 20, 21, 23, 45, 82] is a form of digital signature in which the content of a message is disguised (blinded) before it is signed. The resulting blind signature can be publicly verified against the original, unblinded message in the manner of a regular digital signature.
- **Partially blind signature** - A partially blind signature [1] is a blind signature in which part of the message is unblinded and part is blinded before it is signed.
- **Trust level** - The degree of trust that someone has in a particular identity. A Facebook account that has existed for three years and has 300 friends will have a higher trust level than one that has existed for one day and has zero friends. Similarly an identity that has been verified against a bank account (such as PayPal) and has an associated three year old, 300 friend Facebook account with the same name and date of birth will have a higher trust level.
- **Identity based encryption (IBE)** - A form of encryption based on elliptic curve cryptography (ECC) proposed by Boneh-Franklin [9] where a user's public key is a public string such as their email address.
- **Threshold cryptosystem** - A $(t, n)$-threshold cryptographic primitive [11, 29, 33, 35] where $(t \leq n)$ is a function, $y = f(x_1, x_2, \ldots, x_n)$, where $n$ parties want

to collaboratively compute the result $y$ and each of them holds an individual secret share $x_i$. We say the function $f$ is $(t, n)$-threshold if any $t$ of $n$ parties can cooperate to construct $y$ and no group of $t-1$ or fewer parties can compute $y$.

- **Key share/private key share/public key share** - Several private key share can be combined to form a private key. Similarly for public keys. For example for DSA keys, the private key exponents of two private keys can be added to give a new private key, where the first two private keys are private key shares of the second. Similarly DSA public keys can be added to make a new DSA public key.
- **Verfiable secret sharing (VSS)** - Secret sharing [94] is a method for distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. Verfiable refers to the fact that participants can check their shares are consistent.
- **Distributed private key generator (PKG)** - A distributed PKG [83] is a system that uses VSS for multiple parties to collaboratively compute private keys for clients.
- **OAuth** - OAuth [49, 50] is an open protocol to allow secure authorization from web and mobile applications. The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service (for example a federated identity provider).
- **OAuth provider** An identity service that provides an API for clients to connect to and use [49, 50]. Most federated identity providers are OAuth providers.
- **OAuth client** A third party application that connects to an OAuth provider (such as a federated identity provider) in order to use authenticate users [49, 50].
- **Credential** Something that can be used to prove someone is who they say they are (or prove that they are a member of a group). This is used to authenticate with a third party. For example a public/private key pair and a set of public

keys can be used as a credential to generate an LRS to authenticate oneself with a third party. A signed blinded message is another example of a credential. The message can be unblinded and the signature used to authenticate with a third party.

- **Credential producer** An entity or collection of entities that creates credentials for clients.
- **Key server** A credential producer that serves private and public key shares to clients. A group of key servers collectively act as a distributed PKG to distribute private and public keys to clients.
- **Signing server** A credential producer that signs blinded messages for a client. A client blinds a message, sends it to a signing server which signs the message and returns it to the client. The client can then unblind the message and signature and use the signature to authenticate to a third party application. Several signing servers act collectively to sign several messages for a client.
- **Credential consumer** An entity that supports authentication of a client using credentials (for example an LRS made with a set of public keys and a private key, or a blind signature that has been unblinded). There are three different types of credential consumer:
    1. A party distinct from the application, credential producers and identity providers may act as an OAuth provider that verifies credentials of clients and authenticates them on behalf of the third party. In this case the credential verifier is itself acting as a federated identity provider.
    2. A third party application may run its own OAuth provider internally so they do not need to trust someone else to verify credentials.
    3. A third party application may directly verify credentials themselves, in the form of a module integrated into their application.

- **Credential assignment** The process of a credential producer giveing a client a credential (or set of credentials).
- **Dissent** An anonymous group messaging system [28, 115].
- **MediaWiki** Open source software to support Wikis (websites with collaborate editing) [73]. The software used by Wikipedia.
- **SecureDrop** Open source software for an anonymous drop box where whistleblowers can submit leaked documents to journalists [93].
- **Sock puppetry** A sockpuppet is an online identity used for purposes of deception. The term refers to a false identity assumed by a someone while pretending to be another person.
- **Abuse resistance** A system is abuse resistant if it has some way to sanction, caution or ban users who violate the system's terms and rules.
- **Trolling** Offensive or provocative online posting intended to upser or anger other people.
- **Spamming** Sending or posting or large amounts of unsolicited information, often advertising related.
- **Sybil attack** An attack where a single entity can gain control of multiple identities and use these to abuse a distributed system and defeat system redundancies [40].
- **Collusion attack** An attack on a system where two or more different parties collude together in order to attack the system or a third party in the system.
- **Timing analysis attack** An attack where the time of different events occurring are used to subvert system properties or deanonymize parties.
- **Traffic analysis attack** An attack where network traffic is examined by an adversary in order to subvert a system or deanoymize parties.
- **Correlation attack** A form of traffic analysis attack where traffic entering a

network from a client is correlated by an adversary with traffic leaving the network in order to deanonymize clients.

- **Intersection attack** An attack where anonymity sets for different actions overlap in such a way that the true actor is known to be a member of the intersection of the intersection of the sets and hence has reduced anonymity [85, 56, 31].

# Chapter 4

# Architecture Overview

Figure 4.3 shows a high level overview of the architecture. Figure 4.1 shows the overall Crypto-Book architecture. Traditionally a federated identity provider provides an API to an application via OAuth or a similar protocol. This, however, exposes the user's identity to the application. We insert the credential producer and credential consumer layers between the non-anonymous identity API from the federated identity provider and a privacy preserving API for applications. The lower parts of the system stack (federated identity provider and credential producers) do not communicate with the upper parts of the system stack (credential consumer and applications).

Crypto-Book addresses the user's identity concern by interposing two additional, disjoint layers between the federated identity APIs and the third-party applications that consume them. *Credential producers* interact with the federated identity provider to collectively map federated identities to privacy-preserving cryptographic credentials. Clients then supply these credentials to *credential consumers*, which enable the user to create accounts on and authenticate with cooperating third-party applications using these credentials in place of their original federated identities. Since Crypto-Book credential producers derive credentials from OAuth-based iden-

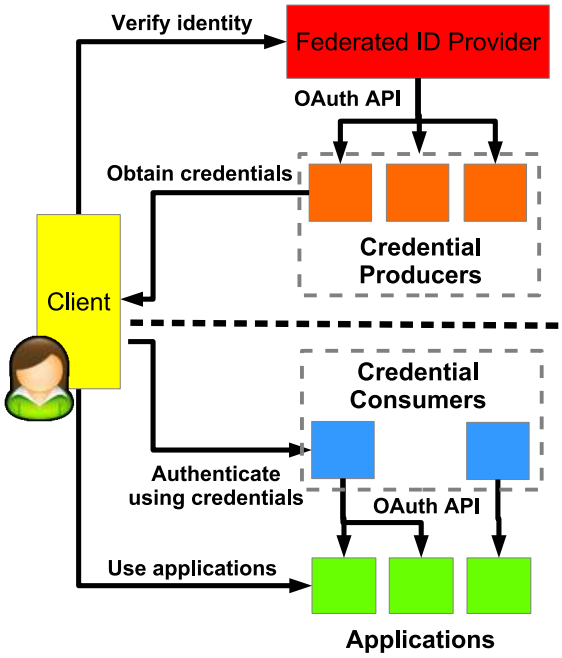


Figure 4.1: The Crypto-Book architecture

tities, and Crypto-Book credential consumers can offer OAuth-based identities to cooperating applications, Crypto-Book in effect acts as a fully backward compatible privacy-protection layer for OAuth identities.

The Crypto-Book architecture ensures that credentials are assigned in such a way that, subject to our threat model, throughout the entire authentication process only the client ever learns enough to complete the mapping, in either direction, between an underlying federated identity and a Crypto-Book identity on a given third-party site. This threat model is defined more fully in Section 4.4, and the process by which Crypto-Book produces credentials is discussed further in Section 5. Finally, Section 7 describes the various ways in which cryptographic credentials can be consumed to produce anonymous identities.

## 4.1 Credential Producers

Crypto-Book credential producers are a set of independently managed servers responsible for producing cryptographic credentials from verified federated identities. We assume each server is run by a respected, technically competent, and administratively independent anonymity service provider. We envision several commercial or non-profit organizations each deploying a cluster of the credential producer servers, as either a for-profit or donation-funded community service.[1]

To obtain cryptographic credentials, Crypto-Book client contacts a threshold $t$ of the $n$ credential producers, each of which independently authenticates the user with respect to one or more federated identity providers. This process is outlined in Figure 6.1, and proceeds as the following three steps.

Step 1. Each credential producer prompts the user to perform a non-anonymous OAuth federated authentication with each of the federated identity providers (e.g. Facebook and PayPal). Each credential producer, then, redirects the client to the federated identity provider's login page for authentication. For each of such authentications, if the client can successfully log into, the client would receive a unique OAuth token corresponding to the specific federated identity provider.

Step 2. The client sends these tokens to the producer who initiated that authentication. With these tokens, the corresponding credential producers perform limited access to the user's profile information. Credential producers request only the minimum access necessary to verify that the identity is valid. Each credential producer verifies via federated identity provider's profile-access API (e.g., the Facebook API) that the federated identity for which the OAuth token was obtained corresponds to the federated identity (e.g. Facebook ID) that the user claimed to have. For multiple

[1]The assumption on servers has been demonstrated to be reasonable in practice [38, 114]

federated identity providers (e.g. Facebook and PayPal), each credential producer also verifies that the user attributes (i.e., date of birth and email address) are the same for both the Facebook and PayPal accounts. In order to obtain a verified PayPal account, a user needs to connect her real world bank account or credit card, which requires showing her real-world identity (e.g. driver license) in person at a bank. This provides a higher barrier to entry and makes it much more difficult for someone to assume a fake identity.

Step 3. After each producer verifies all identities with their respective providers and, if all verify successfully, returns a share of the credential to the client. The client then combines the shares and stores the resulting cryptographic credential for use in future privacy-protected logins.

It is crucial that each producer performs its own OAuth authentication and receives its own OAuth token. A strawman design uses only one OAuth workflow with a single OAuth token and forwards this token to each of the credential producers. The problem with this is that each credential producer can forward the token to other producers to impersonate the user. Having separate OAuth workflows for each producer protects against this.

While security requires each of multiple credential providers to verify one or more of the user's federated identities, we do not wish to subject the *human* user to a tedious process of typing passwords into many federated identity provider login dialogs in succession. The Crypto-Book client hides the multiple-independent-authentications from the human user, on the client side using a Chrome plugin.

Crypto-Book credential producers support multiple cryptographic credentials, requiring only that the credential be adaptable to a $(t, n)$-threshold cryptosystem; any set at least $t$ honest producers must be able to produce a valid credential which will be accepted by any honest credential consumer, while any set of fewer than $t$

producers must not be able to produce such a credential.

We later introduce two such credentials, an *"at-large"* credential using partially blind signatures and a *group credential* based on linkable ring signatures. We discuss the details of credential production for each scheme in §8 and §9, respectively.

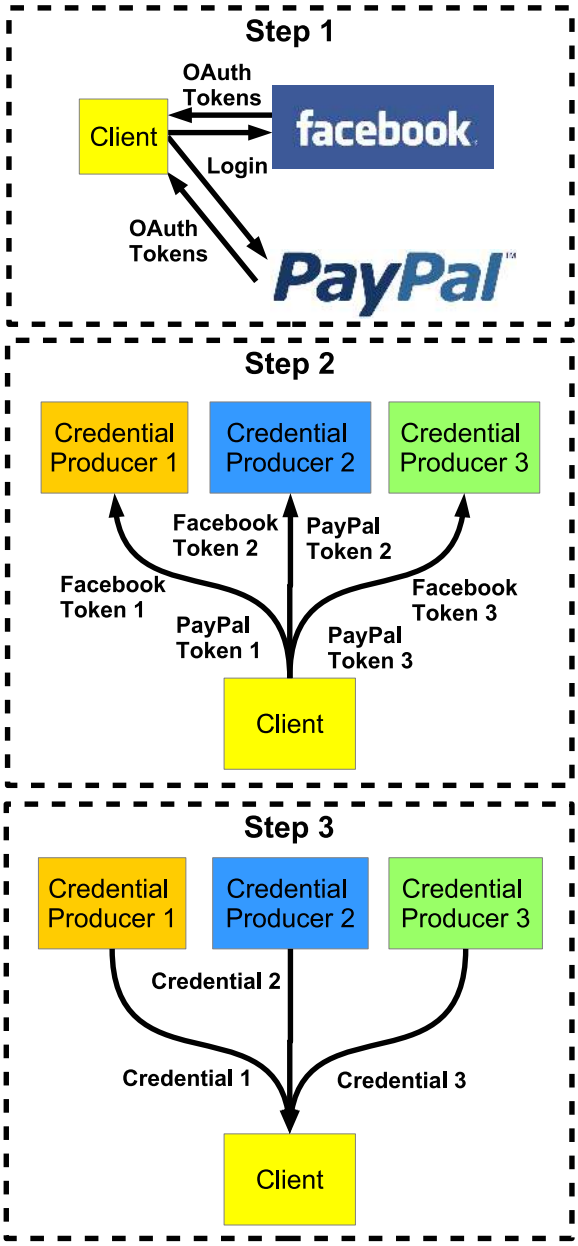


Figure 4.2: Client collects credentials from multiple credential producers.

## 4.2 Credential Consumers

Credential consumers map cryptographic credentials to cryptographic pseudonyms that can then be used to authenticate with third-party applications. The cryptographic pseudonyms produced by credential consumers are unlinkable to the actual federated identities from which they are derived. Credential consumers typically take one of two forms: OAuth provider or application-embedded consumer.

OAuth provider consumer. OAuth provider consumers operate externally to third-party applications. They map cryptographic credentials to anonymous identities and then expose those identities to third-party applications via the OAuth protocol. Applications interact with OAuth provider consumers just as they would directly with conventional federated identity providers. Third-party applications already using federated authentication require little modification to support OAuth provider credential consumers; such integration, however, requires that the application trust the OAuth provider.

Application-embedded consumer. An application-embedded credential consumer exists directly within a third-party application, either via an imported library or a custom implementation of the consumer. Using this approach the application need not trust an external provider, but at the cost of ease of integration with existing authentication mechanisms.

## 4.3 Workflow

The client initially authenticates with a federated identity provider. The client then collects their credentials from the credential producer servers.

The *credential producer* maps credentials to each of the identities from the identity provider. The credentials are then used by the client to anonymously authenticate with *credential consumers*. Credential consumers generate privacy protected identities (pseudonyms) for the user and provide them to the application layer through an OAuth API. We propose two pluggable credential designs:

1. Public/private key pairs
2. Blind signatures [18].

The client generates an anonymous signature using these credentials (for example using the key pair, or by unblinding the blind signature) and sends that to a credential consumer that verifies their credentials and allows them to anonymously log in to third party applications.

The credential producer servers have an $(t, n)$-*threshold cryptosystem* [11, 29, 33, 35] split trust design such that $t$ of $n$ servers can collectively supply a user's credentials and no group of $t-1$ or fewer servers can do so. Hence $t$ servers must be compromised to acquire a user's credentials. The credential producer servers make use of existing technologies, such as OAuth, to verify the user's federated identity before issuing them with their credentials.

**Key pair design** In the public/private key pair design, users obtain a component of their public/private key pair from each key server and use a client-side module to combine these parts and produce their composite key pair. The key servers additionally supply a user with the public keys corresponding to any other federated identities.

A *ring signature* [92, 90] makes it possible to specify a set of possible signers without revealing which member actually produced the signature. RSA-based ring signatures [92] have the property of *deniability* [88] where if you reveal your private key your previous ring signatures cannot be deanonymized. DSA [43] based linkable ring signatures (LRS) [64] provide linkability, that two signatures by the same signer can be linked.

The user anonymously authenticates [63] with the acquired private and public keys using linkable ring signatures, which prove that the signer owns the private key corresponding to *one* of a list of public keys (the *anonymity set*), without revealing *which* key. This provides $k$-anonymity [100] for an anonymity set of size $k$. This property is useful in scenarios where trust is associated with a group rather than an individual, such as a journalist verifying that the source of a leaked document is a member of a particular organization.

We use the linkability property of linkable ring signatures [64] to maintain a 1-to-1 correspondence between pseudonyms and the federated identities, which both ensures credibility of pseudonyms and allows applications to block abusive users.

**Blind signature design** A *blind signature* [96, 18, 15, 12, 19, 22, 20, 21, 23, 45, 82] is a form of digital signature in which themessage is disguised by the requester before it is signed. The resulting blind signature can be unblinded and verified against the original unblinded message. In the blind signature design each credential producer server signs a message blinded by the user. The user must obtain at least $t$ signed messages (from $n$ servers) as we have a $(t, n)$-threshold cryptosystem design. The user unblinds these signed messages and sends them to a credential consumer which verifies the signatures and allows the user to anonymously authenticate to third party applications.

In the blind signature design blind signatures are application specific; the signed

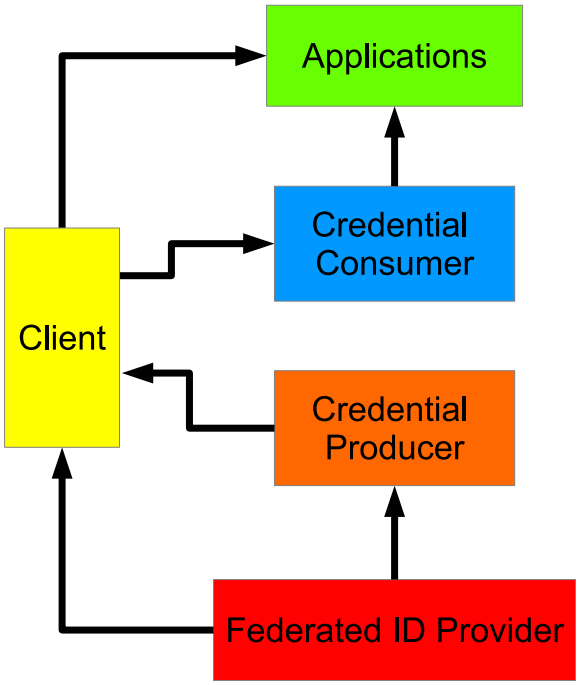


Figure 4.3: High level system diagram

messages contain the name of the application they are valid for. Users are also limited in collecting signed messages, so they can only obtain a certain number of signed messages for an application within a given timeframe (for example per month). This allows applications to block pseudonyms used for abusive purposes for the remainder of that timeframe. This provides abuse resistance while maintaining user anonymity.

Crypto-Book addresses some common concerns with online anonymity. A Sybil attack [40] is where a single user can gain control of multiple identities and use these to abuse a distributed system. Crypto-Book preserves whatever Sybil attack resistance the underlying federated identity provider offers and limits the number of pseudonyms a user can generate from their federated identity. For LRS authentication the linkage tag maintains a 1-to-1 correspondence between pseudonyms and federated identities. For partially blind signature based authentication, pseudonyms are time limited so for a given application a user can only obtain one pseudonym per

time period (such as one per month).

Due to the split trust credential producer server design, a user need only trust that no more than $t - 1$ of the credential producer servers have been compromised.

We present a prototype implementation and evaluation of the Crypto-Book framework using three applications: CB-Wiki, CB-Dissent, and CB-Drop. CB-Wiki leverages Crypto-Book to provide anonymous, yet linkable editing in a collaborative environment. CB-Dissent shows how the integration of anonymous authentication with anonymous communication systems can better protect the identities of the users of those systems. CB-Drop provides anonymous document signing using Crypto-Book identities, allowing for verifiable leaks without compromising privacy. We built CB-Wiki using MediaWiki [73], the software behind Wikipedia, and CB-Dissent using Dissent [28, 114], an anonymous group communication tool. CB-Drop extends the SecureDrop [93] open-source whistleblower platform. Our experimental results show that for anonymity sets of size 100 and 2048-bit DSA keys, Crypto-Book ring signature authentication takes 1.641s for signature generation by the client, 1.632s for signature verification on the server, and requires 8.761KB of communication bandwidth. Similarly for partially blind signature authentication, each phase takes under 0.05s and requires 0.325KB of bandwidth.

## 4.4 Threat Model

We make the following assumptions about a potential adversary in the context of our system. We provide security properties and attack analysis under these assumptions as well as in the case where we relax these assumptions in Chapter 11.3.

- We assume that the client has the ability to connect to the credential producer servers through an anonymity network such as Tor [38].

- Fewer than $t$ of the $n$ credential producer servers are compromised. The remaining *honest* servers do not share their master key or private keys with anyone else. We use a $(t, n)$-threshold scheme for the credential producer servers.
- A key server provides consistent public and private keys; given two requests for the same key, it does not return two different results (we consider the case where we relax this assumption in Chapter 11.3.1).
- Dishonest servers may collude with each other to share master secrets or private keys.
- Credential producer servers can see the IP addresses of clients that connect to them.
- If an adversary compromises a server they have access to the master key and all private keys from that server from that epoch, but not from previous epochs (epoch details are described in Chapter 6.5).
- For the *single identity provider* credential assignment scheme we assume that the resource server provided by the federated identity provider is honest. In this case we rely on the federated identity provider to provide identities and assume that the federated identity provider does not impersonate its users.
- For *multiple identity providers* credential assignment, we assume that different identity providers do not collude with each other to obtain a user's private key. In this case a single federated identity provider cannot compromise any user's private key.
- Different parties (user, ID provider, applications, credential producer) may collude to try to compromise system security properties (see Chapter 11.3).

Crypto-Book's focus is on authentication and does not directly address network-based surveillance or active attacks [77]. For users desiring protection from behavior tracking, linking, or de-anonymization via network-based attackers, Crypto-Book is

synergistic with and designed to be easy to use with anonymous communication systems such as Tor [38], Dissent [28, 115], and Aqua [60].

# Chapter 5

# Credential Producers

In this section we explain how credential producers assign credentials to users of federated identities. We outline at the threshold server model for credential producers and how credential producers generate credentials for users. Credential producer servers use a $(t, n)$-threshold design to split trust among the servers. An adversary would need to compromise at least $t$ of $n$ servers to compromise user credentials.

Crypto-Book credential producers are a set of independently managed servers responsible for producing cryptographic credentials from verified federated identities. To obtain cryptographic credentials, the user's Crypto-Book client contacts a threshold $t$ of the $n$ credential producers, each of which independently authenticates the user with respect to one or more federated identity providers. This process is outlined in Figure 6.1, and proceeds as follows. Each credentials producer first prompts the user to perform a non-anonymous OAuth federated authentication with each of the identity providers (for example Facebook and PayPal). Each credentials producer redirects the client to the ID provider's login page to give the credential producer limited access to the user's profile. After the client logs into each ID provider, for each authentication the client receives a unique OAuth token which it passes along to

the producer which initiated that authentication. With these tokens the producers receive limited access to the user's profile information through the provider's identity API. Credential producers request only the minimum access necessary to verify that the identity is valid.

Each credential producer verifies via ID provider's profile-access API (e.g. the Facebook graph API) that the federated ID account for which the OAuth token was obtained corresponds to the federated ID (e.g Facebook ID) that the user claimed to have.

For multiple ID providers (e.g. Facebook and PayPal), each credential producer also verifies that the user attributes (name, date of birth and email address) are the same for both the Facebook and PayPal accounts. In order to obtain a verified PayPal account, a user needs to connect their real world bank account or credit card, which requires showing their ID (e.g. driver's license) in person at a bank. This provides a higher barrier to entry and makes it much more difficult for someone to assume a fake identity.

After each producer verifies all identities with their respective providers and, if all verify successfully, returns a share of the credential to the client. The client then combines the shares and stores the resulting cryptographic credential for use in future privacy-protected logins.

It is crucial to have each producer do its own OAuth authentication and receive its own OAuth token. A strawman design uses only one OAuth workflow with only one OAuth token and forwards this to each of the credential producers. The problem with this is that each credential producer can forward the token to other producers to impersonate the user. Having one OAuth workflow per producer protects against this.

While security requires each multiple credential providers to verify one or more

of the user's federated identities, we do not wish to subject the *human* user to a tedious process of typing passwords into many federated ID provider login dialogs in succession. The Crypto-Book client hides the multiple-independent-authentications from the human user, on the client side using a Chrome plugin.

Crypto-Book credential producers support multiple cryptographic credentials, requiring only that the credential be adaptable to a $(t, n)$-threshold cryptosystem; any set at least $t$ honest producers must be able to produce a valid credential which will be accepted by any honest credential consumer, while any set of fewer than $t$ producers must not be able to produce such a credential. We later introduce two such credentials, an *"at-large"* credential using partially blind signatures and a *group credential* based on linkable ring signatures.

## 5.1 Threshold Server Model

We use a $(t, n)$-threshold server model in our architecture. We have $n$ credential producer servers and credentials from $t$ of them are required to construct a user's overall credential.

We leverage a threshold cryptosystem in our system architecture design. Under our threat model (Chapter 4.4), we assume that at most $t - 1$ of the credential producer servers are compromised and all other servers may be colluding to try to compromise a client's credentials.

Leveraging a threshold server model means that we do not have to rely on a single trusted server that would be an obvious point of attack. In a single server model an adversary would have to get access to only that server's private key to compromise a client, however in the threshold model an adversary would have to get access to the private keys on at least $t$ of $n$ servers in order to compromise the client's credentials.

## 5.2 Credential Generation

We developed two different pluggable forms of credentials. These are **public/private key pairs** and **blind signatures**. Key pairs can be used to anonymously authenticate by using them to create a ring signature that proves a user is a member of a group. Blind signatures can be used to authenticate the user unblinding them and using the signatures to anonymously authenticate to an applications. Our pluggable credential assignment design means that alternative forms of credentials could also be used instead of key pairs or blind signatures.

### 5.2.1 Key Generation

For the case of keys as credentials, we use a $(t, n)$-threshold cryptosystem to generate key pairs in a distributed manner. For DSA keys Pedersen's private key generator (PKG) algorithm can be used [83]. Boneh and Franklin proposed a similar threshold algorithm for generating RSA keys [7, 8].

In the PKG, for each client key $k$ key server $i$ holds a private key share $k_i$. The client collects key shares from at least $t$ of $n$ key servers then uses some key combining function $f$ that takes each of the private key parts $k_1, k_2, \ldots, k_t$ and combines them into their private key $k = f(k_1, k_2, \ldots, k_t)$ according to the appropriate PKG algorithm (Pedersen or Boneh-Franklin).

### 5.2.2 Blind Signature Generation

For the case of blind signatures as credentials we again use a $(t, n)$-threshold model. The client first generates $n$ random strings to use as the messages. Each message $m$ is then blinded by the client using a blinding factor to give a blinded message, $m'$. The blinded message $m'$ is sent to a credential producer server (signing server)

which signs the blinded message and sends the blinded signature $s'$ back to the client. The client then unblinds this to reveal the signature $s$. The client must obtain at least $t$ signatures $s_1, s_2, \ldots, s_t$ from $n$ servers which the client can use to authenticate anonymously to applications.

We use *partially blind* signatures [1] to provide two additional features, *abuse resistance* and *attribute verification*. Partially blind signatures include an *info* field as part of the message that is not blinded.

**Abuse resistance** To provide abuse resistance the client must put in the *info* fields the names of the applications they wish to access (one application per message). They must then send a set of $k$ messages to each signing server to be signed. The signing server rate limits these requests (for example users might only be able to get one message signed for the StackOverflow application per week or per month). The user then uses these application specific signed messages to authenticate to specific applications. If a user abuses that application, that pseudonym may be blocked and they will not be able to obtain another signed message for that application until the time period (for example one week or one month) has expired. This provides abuse resistance. We will also have generic messages that do not have application specific information in them. Applications will be able to decide whether or not they choose to allow authentication using generic messages (which do not provide abuse resistance) or if they require application specific messages that provide abuse resistance.

**Attribute verification** To provide attribute verification clients add additional attributes in the unblinded part of the message (for example, their age or location). When they submit the message to be signed, the signing server verifies these attributes against their federated identity and only signs the message if the attributes can be successfully verified. Applications learn these attributes and can in turn use

them to provide appropriate content without learning the user's identity.

# Chapter 6

# Credential Assignment Mechanism

We now explain how a client collects their credentials. The primary steps for the client are as follows:

- Anonymously request a link to collect their private key (only for *keys* as credentials design).
- Authenticate with one or more federated identity providers.
- Collect their credentials from the credential producer servers.
- Anonymously authenticate using credentials through credential consumers to applications.

*Key* credential assignment has an initialization phase (not required for *blind signature* credentials). Key distribution begins with a client request for their private key. If the client simply connected to the key servers and requested their private key, this could compromise their anonymity when authenticating with third party sites as a dishonest key server could collude with a third party site to perform a timing analysis attack correlating private key pickups with subsequent authentications to the third party site. To mitigate this, the client anonymously requests a list of Crypto-Book invitations to be sent out.

Suppose Alice wants to collect her private key. She anonymously connects to a key request server (through an anonymity network) and supplies a list of federated identities (say Bob's and Charles'), to form an anonymity set. Each key server then sends out an invitation link to each of the federated identity accounts (for example via social networking message in the case of a social network federated identity provider) to sign up and use the Crypto-Book service.

Alice then receives an invitation to sign up to Crypto-Book. When she clicks the link she is redirected and has to authenticate herself with a federated identity provider.

## 6.1 Alternative Identity Providers

Crypto-Book supports alternative identity providers for example different social networks or other online identity providers such as financial institutions. Crypto-Book integrates with any federated identity provider that is OAuth compliant. Users may want to use different identity providers. For example Alice may not have a Twitter account, but has a Facebook account whereas Bob may be the other way around. Support for multiple identity providers means that more people will have access to the service. Additionally, these different identity providers may provide different *trust levels*. For example bank-confirmed PayPal accounts have a higher barrier to entry than Facebook accounts. Using a PayPal accounts may lead to a higher degree of trust in the identities they represent.

## 6.2 Trust levels

Our design incorporates the idea of identities having various *trust levels*. Some pseudonyms are likely to be more trusted than others. For example a Facebook account that has been active for three years and has 1000 friends is more trusted that one that was created one day ago and has zero friends. If a pseudonym was generated by the user authenticating with multiple federated identity accounts, this will be more highly trusted than if they authenticated with only a single federated identity account. If one of these accounts has had they employer or school email verified (for example LinkedIn or Facebook networks) the identity will be more trusted. If one of the accounts has been bank account or SSN verified (for example PayPal or online banking/financial services provider) the identity will be yet more trusted.

## 6.3 Combined Identities

In addition to supporting alternative identity providers, Crypto-Book also allows users to authenticate with *multiple* identity providers and hence obtain credentials that are tied to all of these accounts. This attests to the fact that a user has, for example, *both* a Facebook and a PayPal account. Credential producer servers may also verify that these accounts are in the same name and/or have other consistent identifying information associated with them such as date of birth, address or phone number. In the combined identities case, it is not possible for a single federated identity provider to impersonate the user and obtain their credentials, as they would also need access to the user's accounts on the other identity provider sites.

## 6.4 Credential Collection

Returning to our example, Alice must now authenticate with one or more federated identity providers to collect her credentials. Upon authenticating, Alice receives one OAuth token per credential producer server from each federated identity provider. Note that each token is only compatible with a single credential producer server. If only one token were provided for all servers, a malicious server could forward the token onto other servers to impersonate Alice. Step 1 of Figure 6.1 shows the client authenticating with Facebook and PayPal.

Since there is one token per server, if there are $n$ servers, there will be $n$ corresponding federated identity provider apps (for example Facebook apps), each requiring user authorization. To simplify this process, we provide a Chrome extension that the user can download to automatically authorize all server apps with a single sign in (our Chrome extension is designed to authorize Facebook apps).

Alice then forwards the appropriate OAuth tokens to the servers (each server receiving a token from each identity provider) as shown in step 2 of Figure 6.1. In the case of *blind signature* credentials, Alice also forwards a blinded message to each server along with the OAuth token. The servers contact the identity providers, for example Facebook and PayPal, to verify the tokens (and any attributes present in the case of *partially blind signature* credentials). If verification succeeds, the key server distributes its share of the client's private key to the client as shown in step 3 of Figure 6.1.

In the case of *key* credentials, once the client has received all private key shares, they combine them to get their private key.

An alternative approach to having the client locally run software to collect the credentials would be to have an intermediary server acting as a trusted web proxy

whose job is is to authenticate the user, then collect their credentials on their behalf. The advantage of this is that the user does not have to run any program locally themself however they have to trust the web proxy with their credentials.

## 6.5 Compromised Credential Producer Servers

One threat that we need to consider is what happens if a credential producer server's storage is compromised. For example if the master key is leaked or if a thief breaks into the data center and illicitly obtains the key server's physical hardware. The adversary would obtain access to the master private key. While this does not in itself compromise the user's credentials it is still undesirable. We propose an epoch based scheme to mitigate this vulnerability.

Under this scheme we have the server work in epochs, where the server's master private key is valid only during a given epoch and gets randomly reinitialized in each successive epoch. If we want previously generated ring signatures/blind signatures to still be verifiable after the epoch, the server must maintain a list of public keys containing all the public keys generated in that epoch. Then in subsequent epochs the server will be able to serve requests for older public keys to allow for verification of old ring signatures/blind signatures.

Since the master private key gets randomly reinitialized in each successive epoch, each user can thus get new credentials in each server epoch. The server is only able to serve a client's credentials for the current epoch so that in case of compromise, only the user credentials in the latest epoch are compromised, not those of all past epochs. We envisage epochs being realtively long, of the order of months, or new epochs triggered in the event of a key server compromise.

The epoch based server scheme used in conjunction with the threshold cryp-

tosystem credential generation/assignment scheme significantly reduces the risk of a client's private key being compromised by an adversary.

We include a more detailed analysis of attacks and defenses in Chapter 11.3.

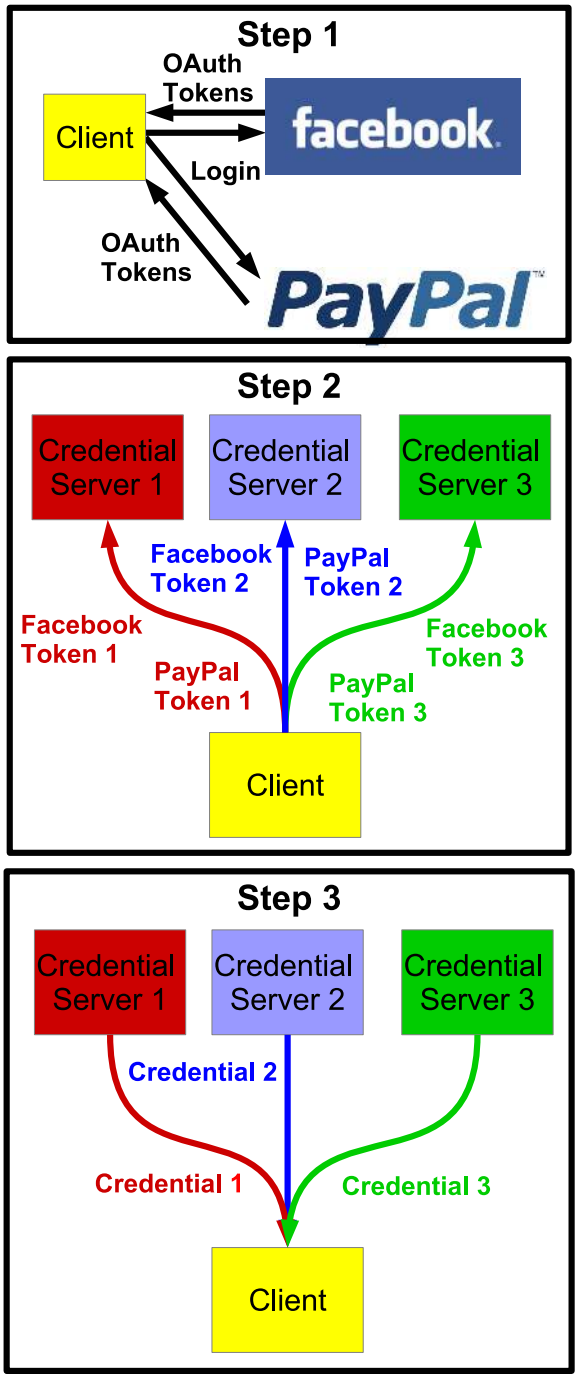


Figure 6.1: Client collects credentials from servers

# Chapter 7

# Credential Consumers

A credential consumer is an entity that supports authentication of a client using credentials (for example a LRS made with a set of public keys and a private key, or a blind signature that has been unblinded). In this section we explain how credential consumers fit into our design.

There are three different types of credential consumer:

1. A party distinct from the application, credential producers, and identity providers may act as an OAuth provider that verifies credentials of clients and authenticates them on behalf of the third party. In this case the credential consumer is itself acting as a federated identity provider.
2. A third party application may run its own OAuth provider internally so they do not need to trust someone else to verify credentials.
3. A third party application may directly verify credentials themselves, in the form of a module integrated into their application.

These three options have different tradeoffs in terms of usability and trust. Using an external OAuth provider to verify credentials makes it easy to integrate Crypto-Book authentication with existing applications providing the applications are willing

to trust the provider with verification. We implemented an OAuth compliant credential consumer. A third party application may choose to run their own OAuth provider in house to verify credentials. This requires the additional overhead of the application of setting up an maintaining the OAuth provider server, however they do not have to trust someone else to perform credential verification. Finally, a third party may choose to forego OAuth providers altogether instead opting to use a custom credential verification module integrated into their application to verify user credentials. The application does not have to trust anyone externally with credential verification but has to bear the overhead of integrating the module.

## 7.1 OAuth Provider Credential Consumer

We now consider the case of a credential consumer as an OAuth provider in more detail.

The Crypto-Book architecture serves as an OAuth compliant identitity provider allowing websites that include a *Log in with Facebook* button to authenticate users to similarly include a *Log in with Crypto-Book* button to do so anonymously and accountably. Crypto-Book as an OAuth identity provider works as follows, starting from when the third-party redirects the user to Crypto-Book to authenticate:

In the LRS authentication design, Crypto-Book presents a challenge to a the user in the form of a random string that the user is requested to sign using a linkable ring signature. In the blind signature case the user obtains signed blinded messages from credential producers and then unblinds them. The user uploads their signatures (LRS or unblinded blind signature) to Crypto-Book which then verifies the signature. If the signature is successfully verified, Crypto-Book obtains the pseudonym corresponding to that user by hashing the *linkage tag* (in the case of LRS) or the $m$ message value

(in the case of blind signatures). Crypto-Book then generates an OAuth token and associates it with that pseudonym, storing the pair in a database, and redirects the user back to the third party site passing the OAuth token as a parameter.

The third party site now knows that the user successfully authenticated with Crypto-Book. The site can then include the OAuth token in future requests to query Crypto-Book for the user's pseudonym and the anonymity set they authenticated with (for LRS). The fact that the pseudonym is derived from the linkage tag means that if the user tries to authenticate multiple times they will always be given the same pseudonym and thus can be prevented from repeatedly conducing abuse on the third party application in the same way as a non-anonymous user. Hence Crypto-Book allows any third-party site or app that is an OAuth-compatible client to provide anonymous, abuse resistant login.

## 7.2 Application-Embedded Consumer

An application-embedded credential consumer exists directly within a third-party application, either via an imported library or a custom implementation of the consumer. Using this approach the application need not trust an external provider, but at the cost of ease of integration with existing authentication mechanisms.

## 7.3 LRS Anonymous Authentication

This section discusses how anonymous authentication is performed in the case of LRS as unlike blind signatures, LRS have the additional feature of an anonymity set. Once a client has obtained their own private key and a list of public keys corresponding to other users' federated identities, the client constructs a ring signature [92, 90] with

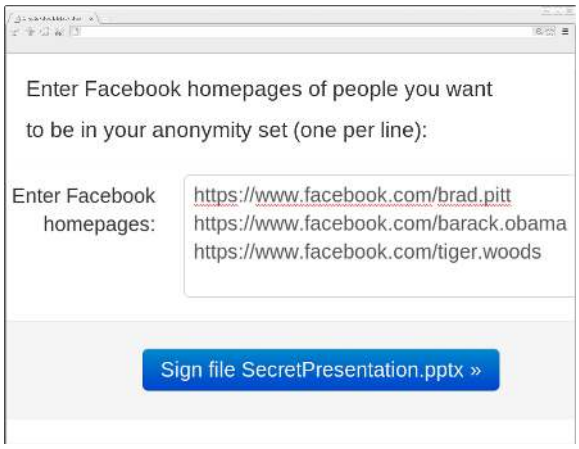


Figure 7.1: Crypto-Book anonymity set selection

all the identities as the anonymity set. Figure 7.1 shows our system where a user is choosing to have Brad Pitt, Barack Obama and Tiger Woods in their anonymity set by entering their Facebook profiles. A ring signature has the property that a third party can verify using only the public keys that the signature was created by one of the members of the anonymity set. However they cannot determine which person in the anonymity set specifically created the signature. Hence they can be used to protect the anonymity of the user.

The ring signature is now used by the user as a form of anonymous online identity and could be used in a multitude of different scenarios. For example the user could anonymously sign a document to give a credible leak, join an anonymous chat group open only to a specific set of users, or anonymously comment on blog posts.

A credential consumer verifies an LRS against the list of public keys of those in the anonymity set (the LRS ring) in order to authenticate a user to an application.

# Chapter 8

# At-Large Credentials

We first discuss *at-large credentials*, which do not explicitly constrain the anonymity set of a user but instead represent that the user has been verified as the owner of *some* federated identity. The anonymity set for each at-large credential is then implicitly all users who have ever collected a credential in the time period before the credential was used.

We then extend at-large credentials to optionally include *credential attributes*, such as "age over 18" or "identity active for at least one year", which are also verified with the identity provider at the time of credential production. Section 8.5 discusses credential attributes in detail.

The following sections introduce an at-large credential construction which uses only existing, standard cryptography in both the production and consumption of credentials; Section 8.8 then discusses ways in which more specialized cryptography could lead to improved constructions.

## 8.1 Background: Blind Signatures

A *blind signature* [96, 18, 15, 12, 19, 22, 20, 21, 23, 45, 82] is a form of digital signature in which the signer never learns the contents of the message they are signing. To obtain a signature on message $m$, the requester first obscures their message with a secret blinding factor to produce blinded message $m'$ and sends $m'$ to the signer. The signer then produces a signature on this blinded message and returns the blinded signature $s'$ to the requester. The requester, with knowledge of the secret used to blind the message, removes the blinding factor to reveal the unblinded signature $s$. The orginal message $m$ and the unblinded signature $s$ can then be verified by a third-party using only the signer's public key, just as for a traditional digital signature.

## 8.2 Building Block: Blind Signatures

The blind signature [2, 17] is a cryptographic primitive, where a *requester* can request a *signer* to sign one or more messages, while the signer cannot learn the signed message's content. Given the message-signature pair, a public *verifier* is able to verify the legitimacy of the signature. In our architecture, the client is the requester, each credential producer is a signer and the credential consumer is a verifier (of multiple signatures).

## 8.3 Producing At-Large Credentials

To obtain a threshold $t$ at-large credential for use with consumer with identity $id_c$, a client first generates a random value $r$ which identifies the credential. The client hashes this value $r$ with the identity of the consumer to produce message $m = H(r, id_c)$. The client then contacts at least $t$ of the $n$ credential producers with

signature requests, uniquely blinding the message $m$ to produce $m'_i$ for each request.

Before signing the message, each credential producer verifies the client's federated identity and, if successful, returns blinded signature $s'_i$ to the client. The client unblinds the signatures from each of the credential producers to obtain a vector of unblinded signatures $s_1, s_2, \ldots, s_t$ which serves at the at-large credential for anonymous identity $r$ with credential consumer $c$.

## 8.4 Consuming At-Large Credentials

To authenticate with a credential consumer requiring a threshold $t$ at-large credential, a client must provide the credential consumer with the value $r$ defining their anonymous identity along with a vector $s_1, s_2, \ldots, s_t$ of signatures from at least $t$ *unique* credential producers. The consumer first hashes this value with it's own identity to produce message $m = H(r, id_c)$. The consumer, using the public keys of the credential producers, then verifies that each signature is, in fact, valid for message $m$ and, if successful, authenticates the client as anonymous identity $r$.

## 8.5 Credential Attributes

Credential attributes allow credential consumers to enforce general restrictions on the at-large credentials they accept. For example, some credential consumers may require that all users be at least 18 years of age. Credential attributes can also be used to provide a higher barrier to entry by requiring, for example, that a Facebook identity has been active for at least one year.

The at-large credential scheme supports credential attributes by using partially blind signatures. *Partially blind signatures* [1] are a modification to blind signatures

in which part of the message, the *info* tag, remains visible to the signer. This allows the signer and verifiers to share additional information about the context of the blind signature.

Clients bind attributes to at-large credentials by including each desired attribute in the info tag of their signing requests. Each credential producer then additionally verifies all of the attributes with the federated identity provider and produces a signature only if all check out. Credential consumers enforce credential attributes by ensuring that each signature presented by the client contains all required attributes in the info tag. An inherent restriction on credential attributes is that they must be verifiable with the federated identity provider.

## 8.6 Rate-Limits via Attributes

Credential attributes additionally allow for finer control over the rate-limits imposed on at-large credential production. Rather than relying on producers to choose a proper default rate for all applications (*e.g.* $x$ credentials per identity per week), clients can instead specify a time interval (*e.g.* 3 days) with each credential request; only if the producer has not already issued the federated identity an at-large credential during the preceding interval does production succeed. As credential attributes are accessible by credential consumers, a consumer can inspect the interval associated with each credential and in turn elect to accept only those credentials satisfying criteria the consumer itself defines. Some consumers may accept 1-hour credentials while other, more abuse-conscious consumers may require 1-month credentials, allowing consumers themselves to determine the degree of accountability they wish to enforce.

## 8.7 Security/Privacy Properties

The at-large credential scheme is designed to provide the following properties:

- **Anonymity** During credential production, the credential producer learns the user's federated identity, but not the anonymous identity, $m$. The credential consumer learns $m$, but not the user's federated identity. Only the user himself knows both pieces needed to complete the mapping between identities.
- **Unlinkability** For any two at-large credentials, neither credential producers nor consumers can determine whether they correspond to the same federated identity.
- **Abuse Resistance** Each at-large credential is bound to anonymous identity $m$. If $m$ misbehaves, he can be banned by the consumer just as any non-anonymous identity. Producers provide abuse resistance by rate-limiting the credentials assigned to each federated identity.
- **Unforgability** Each at-large credential requires a signature from $t$ of the $n$ credential producers; no colluding group including fewer than $t$ dishonest producers can produce a forged credential that will be accepted by an honest consumer.

We provide a security analysis of how Crypto-Book provides these properties and which attacks it does and does not protect against in Appendix 11.

## 8.8 Discussions

Crypto-Book's current at-large credential scheme requires the client to obtain a separate blind signature for each third-party application or site the user wishes to visit for the first time, to protect the user from being linked across applications. This limitation may be an inconvenience, especially if Crypto-Book's rate-limits interfere with a user's legitimate attempts to explore several third-party sites in a short time period. Adopting a more sophisticated cryptographic credential scheme such as

BLAC [103, 104, 5] might allow the client to pick up a single credential and then "re-blind" it for use across multiple sites while maintaining cryptographic unlinkability and abuse-resistance. Since Crypto-Book's immediate goal is not to find the "ultimate" cryptographic credential scheme but to fit cryptographic credentials (of any kind) into a usable OAuth-compatible architecture, we leave more advanced at-large credential schemes to future work.

# Chapter 9

# Group Credentials

## 9.1 Background: Ring Signatures

Proposed by Rivest *et al.* [92, 90], ring signatures build on group signatures [25] and allow third-parties to verify that a message was signed by one of a well-defined set of private keys, but not which specific key. Ring signatures are particularly useful for associating properties of a group as a whole, such as credibility in our CB-Drop example, with the signed message. We use ring signatures to provide credentials that provide anonymity within a specific group.

Liu *et al.*'s [64] proposed *linkable ring signatures* (LRS) extend ring signatures with the additional property of *linkability* - for any two linkable ring signatures, a third party can determine whether or not the two signatures were produced using the same private key by comparing the *linkage tag* properties of the signatures. We use the linkability property to add accountability to anonymous credentials. Creating a linkage ring signature requires the public DSA keys of all members of the group plus *one* corresponding private key. Crypto-Book solves the problem of the creation and retrieval of these keys.

Liu *et al.*'s [64] proposed *linkable ring signatures* (LRS) are based on discrete logarithm DSA [43] keys. Linkable ring signatures are similar to traditional RSA ring signatures with the additional property of *linkability*. Linkability refers to the fact that given any two signatures, a third party can determine whether or not they were produced by the same person. While LRSs provide linkability, they do not provide *deniability* in the way that RSA ring signatures do. If a private key is compromised by an attacker, then the attacker may use that to unmask previously generated LRSs to tell whether or not they were indeed produced by that private key.

## 9.2 Building Block: Ring Signatures

Ring signatures [92] rely on group signatures [26] and allow third-parties to verify that a message was signed by one of a well-defined set of private keys, but do not reveal which specific key. Ring signatures are particularly useful for associating properties of a group as a whole, such as credibility in our CB-Drop example, with the signed message. Liu *et al.* propose *linkable ring signatures* (LRS) [65]. LRS is an extended version of ring signatures with the additional property of *linkability* – for any two linkable ring signatures, a third party can determine whether or not the two signatures were produced using the same private key by comparing the *linkage tag* properties of the signatures. We use the linkability property to add accountability to anonymous credentials. Creating a linkage ring signature requires the public DSA keys of all members of the group plus *one* corresponding private key. Crypto-Book solves the problem of the creation and retrieval of these keys.

## 9.3 Producing Group Credentials

A ring credential consists of two components - the user's individual private key and the set of public keys of all other members of the desired anonymity set.

Credential production begins with an initialization phase. Suppose a user Alice wants to collect her private key. She first connects to a credential producer through an anonymity network (such as Tor) and supplies a list of federated identities, including her own, as her desired anonymity set. The credential producers then work together to create public/private key pairs for each federated identity using Pederson's PKG. Under this scheme, each of the $n$ credential producers holds a single share of each key, $t$ of which are necessary to reconstruct the key.

After all keys have been generated, each credential producer sends an invitation to each of the federated identities (via Facebook message, for example) inviting then to join the Crypto-Book service and collect their private key. This indirection is necessary as if a user directly requested their private key a credential producer colluding with a credential consumer could potentially deanonymize a user via a timing analysis attack by correlating the private key request with subsequent authentications.

Alice (and, independently, the other identities who have received invitations) then follows the invitations to collect a share of her private key from each of the credential producers. Before releasing the share, each producer first requires Alice to authenticate with her federated identity provider via OAuth, proving that she, in fact, owns the identity corresponding to the private key. After collecting shares from at least $t$ of the $n$ credential producers, Alice combines the shares to recover her private key.

Obtaining the set of public keys is much simpler as Alice need not prove her identity to the credential producers. This time, Alice directly contacts each credential

producer and requests a share of the public key for each identity in her anonymity set. After receiving at least $t$ shares of each key, Alice recovers the set of public keys. In practice, clients bundle all key requests for a given anonymity set and producers return all shares in a single response, to minimize transmission overhead and latency.

## 9.4 Consuming Group Credentials

This section discusses how anonymous authentication is performed in the case of LRS as unlike blind signatures, LRS have the additional feature of an anonymity set.

Credential consumers authenticate group credentials by requiring users to supply a valid linkable ring signature over a message of the consumer's choosing; typically, a fresh, random message. Such a challenge prevents replay attacks and ensures that the user knows a valid private key for the ring.

The user first contacts the credential consumer with an authentication request. The consumer then replies with a challenge, which the user signs using their group credential and returns to the consumer. In some instances, the user also supplies the anonymity set to which the group credential corresponds; in others, this set is implicitly determined by the consumer itself. In either case, the consumer first asserts that the anonymity set contains valid Crypto-Book identities for the given consumer.

The consumer then collects public key shares for all members of the anonymity set from at least $t$ credential producers and verifies the ring signature against the resulting public keys. If the signature verifies, authentication succeeds and the consumer maps the user to an anonymous account based on the linkage tag of the signature, creating a new account if this was the first authentication for the given tag.

## 9.5 Security/Privacy Properties

The group credential scheme is designed to provide the following properties:

- **Anonymity** During credential production, each credential producer learns only a single share of a user's private key. No colluding group of fewer than $t$ dishonest produces can recover the private key needed to de-anonymize the user.
- **Abuse Resistance** Linkability of the signatures produced by group credentials ensures that a given credential always maps to the same anonymous identity. Thus group credential identities can be banned just as any non-anonymous identity.
- **Unforgability** Any valid group credential must include a private key for a valid Crypto-Book identity. Shares for such a key must be obtained from at least $t$ credential producers. Thus no colluding group including fewer than $t$ dishonest producers can produce a forged credential that will be accepted by an honest consumer.

We provide a security analysis of how Crypto-Book provides these properties and which attacks it does and does not protect against in Chapter 11.

## 9.6 Discussions

Crypto-Book's current group credential scheme uses linkable ring signatures whose size is linear in the anonymity set size; their efficiency on large anonymity sets could be improved using accumulator-based schemes [14, 61, 62] at the cost of more complex computations.

Another disadvantage is that using any form of signatures for authentication leaves a non-repudiable trail, which might expose a user whose private signing key is later compromised. This limitation might be addressed by adopting techniques from

deniable authentication protocols [37, 75].

Finally, in practice it may be hard for users to pick "good" anonymity sets. If all the other users a whistleblower conscripts into his "anonymity set" turn out to be implausible for some reason to an investigating adversary, e.g., because none of the other members could have had access to the leaked document, then the chosen anonymity set may prove ineffective. We make no suggestion that group credentials are straightforward to use safely: only that, if the user's only other alternative is to disclose his identity completely (e.g., to persuade the journalist of his credibility), then group anonymity may be better (and perhaps at least more "plausibly deniable") than no anonymity.

A ring signature has the property that a third party can verify using only the public keys that the signature was created by one of the members of the anonymity set. However they cannot determine which person in the anonymity set specifically created the signature. Hence they can be used to protect the anonymity of the user.

## 9.7 Neff-Based Group Credential Scheme

One consideration for group credetials is the case of large sized groups. We would like to have a scheme that allows clients to authenticate to forums with large numbers of users for example. In these cases, we would rather not impose on the clients the computation overhead of linkable ring signature generation as this scales linearly with group size. We propose a group credential scheme based on the Neff key shuffle [78]. In the description below we use the use case of authenticating to an anonymous group chatroom.

## Initialization

The setup step is for creating the group. A chatroom configuration file is created stating the Facebook group which the chatroom will be created for. The configuration file also specifies a refresh period. The refresh period is how often to check whether the Facebook group has new members. This will be short, for example once per day.

## Group refresh

When the group is initially created and every refresh period afterwards, each credential producer server downloads the list of Facebook members in the Facebook group, obtained from Facebook. Each server publishes and commits it's list to all other servers. Servers then pick the group members that appear in each the majority of the lists as the chatroom members. This ensures that a malicious server cannot drop members or Sybil-attack the membership by adding many fake members.

After agreeing on the group membership list, each server looks up, or generates as necessary, their share of each Facebook ID's corresponding public/private keypair. The usual process is followed the same as if the server were handing public key requests for each group member.

Each server now has a list of Crypto-Book public keys, one for each member of the group, $G$. The servers now collectively perform the simple Neff verifiable key shuffle [78]. This Neff shuffle takes a list of public keys $Y_1 = g_1^x, \ldots, Y_n = g_n^x$, rebases the keys to use a new $h$ as the generator: $Y'_1 = h_1^x, \ldots, Y'_n = h_n^x$ and returns a permutation (shuffle) of these rebased public keys. The output of this process is a list of permuted public keys for all members of $G$, rebased to a generator $h$. The corresponding private keys are unmodified however no entity will know which rebased public key corresponds to which original public key.

## Client Authentication

When the client wishes to authenticate to the chatroom, the chatroom application queries the servers for the list of rebased public keys for group $G$. Additionally, the chatroom application retrieves the common generator $h$ for the rebased keys and sends it to the client as an authentication challenge. Note that only $h$ needs to be sent to the client, not the entire list of public keys.

The client verifies each server's signature on $h$ and computes their corresponding public key, $Y' = h^x$ where $x$ is the client's private key. The client sends $Y'$ to the chatroom application along with a plain digital signature generated using $x$. The chatroom verifies that $Y'$ is on the rebased membership list for group $G$ and verifies the client's digital signature. The chatroom server can now use $Y'$ as a unique pseudonym for the client in the chatroom in place of a linkage tag.

The client authenticates using a standard public-key digital signing operation, the cost to of client authentication is constant, not dependent on group size. Group refreshes scale linearly with group size. Although the clients pseudonymous public key $Y$ is not a secret, the Neff shuffle ensures that no entity can tell which original public key $Y$ corresponds to a client's rebased public key $Y'$, hence protecting the client's anonymity.

## Membership changes

On each refresh, the servers shuffle and rebase the public keys for all members of the group. If the keys were previously rebased using generator $h$, a new generator $h'$ must be used next time to avoid disclosing which new members of the group have been added. Whenever the group changes, each client gets a fresh rebased public key $Y$. The client uses their most recent rebased public key to authenticate to the

chatroom. Once authenticated the client may remain connected to the chatroom potentially across rebase events. Clients must reauthenticate if they disconnect from the chatroom. The chatroom may optionally require clients to periodically reauthenticate to prevent clients who have been removed from the group from remaining in the chatroom.

### Human-readable pseudonyms

Human readable pseudonyms (for example "anonymous fox") are assigned using a simple pseudonym allocation mechanism. When a client connects and authenticates to a chatroom, the client also requests a pseudonym for the duration of the connection. This will be the client's screen name. The screen name must be unique so if a client chooses a pseudonym that is already in use, the chatroom will require them to pick another one instead that is unique. Once logged in, the cleint will be able to keep that pseudonym for the duration of the connection. IRC systems use a similar mechanism to this.

### Multi-connection sock-puppetry

One potential hazard is that a client may open a new connection each refresh period and keep the old connections open in order to obtain multiple pseudonyms in the chatroom for a sock-puppetry attack.

To prevent this the chatroom can require each client to reauthenticate and reveal their new public key each refresh period. Say a client has authenticated using rebased public key $Y_1$ based on generator $g_1$. After a rebasing occurs the client now has a new rebased public key $Y_2$ based on generator $g_2$. The chatroom now informs the client that the group has been rebased from $g_1$ to $g_2$ and requires the client to produce and

sign $Y_2$ and prove its correspondence to $Y_1$. This now means that the chatroom knows the latest rebased public key for each connected client and so can reject attempts of an existing client making additional connections under different pseudonyms. This protects against sock-puppetry attacks.

# Chapter 10

# Privacy Preservation

In this section we consider how user privacy is preserved in our architecture. We first look at how anonymity set choice affects user privacy, then discuss how ring signatures are used to provide $k$-anonymous authentication, as well as blind signatures.

## 10.1 Anonymous Key Distribution

One possible threat comes from *intersection attacks* [85, 56, 31] where a compromised key server allows an adversary to see who is requesting their private key. When subsequent ring signatures are created and used, the adversary may be able to deduce who created a ring signature, for example if only a small subset of the anonymity set had actually collected their private keys. We propose an *anonymous key distribution* protocol that helps to protect client anonymity in the face of these intersection attacks.

Say that Alice wants to collect her private key part from a key server, but we do not want the key server to know that Alice has picked up her private key. Alice connects through an anonymity network to the key server using a secure SSL connection but this time does not *log in with Facebook* or otherwise identify herself to

the key server. Alice then requests that the key server deliver her private key. She does this by supplying a list of email addresses that includes her own email address along with other email addresses that form the anonymity set as shown in Figure 10.1. The server then generates a private key for each email address and encrypts each private key using the same single symmetric key. The server sends over the secure connection to Alice the symmetric encryption key along with instructions on how to decrypt her private key.

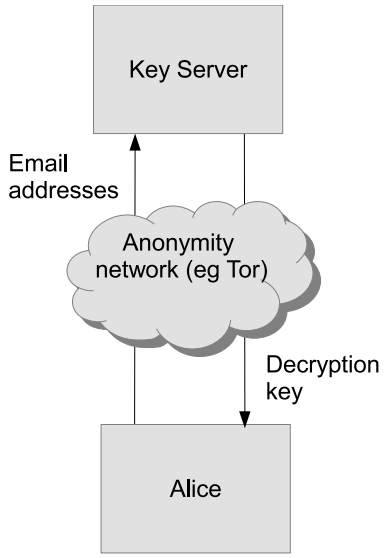


Figure 10.1: Alice anonymously requests her private key

Each encrypted private key is then attached to an email inviting that user to sign up to the service as shown in Figure 10.2. The emails are then sent out such that each email address only receives their own encrypted private key. Alice can now check her email, find the attachment to the invitation email and decrypt it to obtain her private key part. In order to prevent the system from sending out too many messages rate limiting could be employed.

Since her private key was encrypted, her email provider or anyone who compromises her email or intercepts is does not have access to it. Additionally since the server received an anonymous request and sent out multiple private keys to multiple

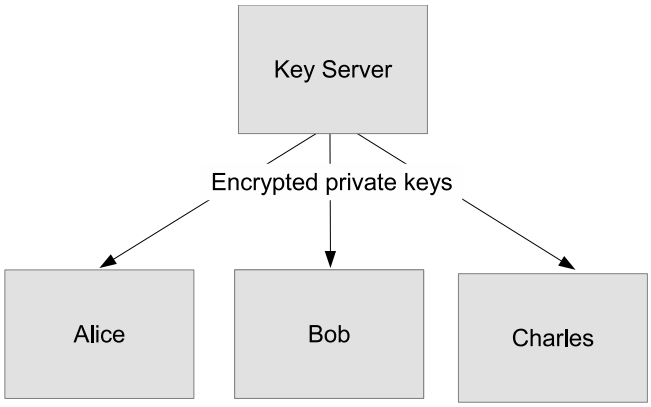


Figure 10.2: Invitation emails sent with encrypted private key attached. Alice can decrypt her private key.

emails, the server does not know who in the end was able to decrypt and use their private key, hence protecting Alice against intersection attacks from a compromised server.

Although this scheme may seem somewhat complicated, Alice only needs to carry it out once per key server (for example if there are three key servers, Alice need only do it three times). Alice can then construct her private key and keep it saved for future use so as to avoid having to participate in anonymous key distribution again.

In addition to being deployed over email, this protocol could also be deployed over other communication channels such as social network messaging or SMS text messaging.

## 10.2 Anonymity Set

One important consideration from the standpoint of maintaining user anonymity is the selection of the anonymity set used when generating the ring signature. One option is to leave it up to the user to construct their anonymity set. The benefit of this is that the user may want the set to correspond to some group of people in the real world, such as a group of employees all working in the same corporate

division or government department. One a hazard of allowing users to choose their own anonymity set, however, is that they may inadvertently choose a set that compromises their anonymity. An alternative approach is for the system itself to batch users together into groups.

In some cases users may benefit from having the option of defining explicit anonymity sets. For example, a company employee may want to leak documents of public interest to the press but at the same time show that they come from a credible source. In this case the employee would like to have the option of choosing their own anonymity set where all members of the set are employees at the same company. This way, when a journalist verifies who signed the document, they know for sure that it came from an employee of the company, they just do not know which specific employee leaked the document. We implement an application that supports this functionality called CB-Drop that we describe in Chapter 13.3.

The way user groups are chosen will vary with application and may have implications for the degree of privacy protection afforded to users. The extent of these implications is an interesting area for future work which we discuss in Chapter 15.

A possible threat to user privacy comes from the fact that third party sites may collude to attempt to deanonymize and uniquely identify users. If a user authenticates themself as a member of a group across many third party sites, this vector of group membership may threaten the user's anonymity. The extent of this risk depends on how the anonymity set is chosen. If different sites use different groups, for example a user is in group A on site 1 but group B on site 2, there may be some risks to user privacy the extent of which would depend on the way groups are chosen. We consider such *intersection attacks* and defenses in Chapter 11.3.

However if groups are defined by Crypto-Book, users do not face this risk as they would always be in the same group regardless of which third party site they logged

into. We propose using this scheme where Crypto-Book puts users into groups in order to protect user privacy.

### Protecting users from intersection attacks when choosing anonymity sets

One possible attack we consider in Chapter 11.3 is the case where few people in an anonymity set collect their private keys. If the credential producer or identity provider colludes with applications, they may be able to eliminate certain people from the anonymity set and threaten user anonymity. The blind signature scheme is not succeptible to this attack. We propose the following scheme for picking an anonymity set in the case where a user wants to authenticate using LRS but is worried about other people not having collected their private key:

Previous work [74, 108, 66] has found that there are many public participants active in social networks who choose to join various *social actions*. We envisage groups being created on social networks which users who are interested in anonymous authentication join. All members of the group will have collected their private key (they may have to sign something to join the group using their private key). People will join the group to provide a large group of people that can be used in anonymity sets without worrying that they have not collected their private key. People will join for similar reasons that people volunteer to run Tor exit nodes.

Our solution is a dynamic anonymity set generation technique. First, clients construct their anonymity sets using real world communities such as a group of employees all working in the same corporate division or government department. The intuition behind this design is clients can hide themselves in a group of people with similar interests. Then, the clients randomly include some *active federated*

*identity users* with the similar interests in their anonymity sets.

The client then randomly chooses some member from the social network groups on anonymous authentication to include in their anonymity set. Combining the two sets of social network identities, (the members of the client's real world community and active members of an anonymous authentication social networking group) would form a sophisticated anonymity set for the client where the client knows at least some of the other anonymity set members have collected their private keys. Hence protecting the client from intersection attacks even if the identity provider or credential producer colludes with applications.

## 10.3 Ring signatures

Ring signatures are a cryptographic scheme proposed by Rivest *et al.* [92, 90]. Ring signatures build on group signatures [25] in that a message signed with a ring signature is endorsed by someone in a particular group of people however it is difficult to determine which of the group members' keys were used to produce the signature. Ring signatures differ from group signatures in that ring signatures do not require any initial setup and can be created on an ad hoc basis.

Rivest *et al.*'s ring signatures are based on each participant having a personal RSA public private keypair. The signer must obtain the public keys of all other parties in their anonymity set, in addition to their own RSA private key in order to generate a ring signature. Rivest *et al.*'s ring signatures provide the property of *deniability*: Even with access to the private keys, given an RSA ring signature it is not possible to unmask the original signer. Even if a private key is leaked it does not compromise the anonymity of existing ring signatures due to the deniability property.

### Linkable ring signatures

Liu *et al.*'s [64] proposed *linkable ring signatures* (LRS) are based on discrete logarithm DSA [43] keys. Linkable ring signatures are similar to traditional RSA ring signatures with the additional property of *linkability*. Linkability refers to the fact that given any two signatures, a third party can determine whether or not they were produced by the same person. While LRSs provide linkability, they do not provide *deniability* in the way that RSA ring signatures do. If a private key is compromised by an attacker, then the attacker may use that to unmask previously generated LRSs to tell whether or not they were indeed produced by that private key.

We cannot trust a single server to maintain the private keys. To counteract this, a distributed PKG such as Pedersen's scheme for DSA keys [83] or Boneh and Frankin's scheme for RSA keys [7] should be used to serve keys.

Linkable ring signatures are used in our architecture to provide an anonymity preserving identity to a user. The ring signature may be used to authenticate with a third party website or service so the third party knows that the user is a member of a group of users, but the specific user's identity is not revealed to the third party, protecting the user's privacy and anonymity.

## 10.4 Blind signatures

Blind signatures are a cryptographic scheme originally proposed by Chaum [18] and since then many other schemes have been proposed [96, 15, 12, 19, 22, 20, 21, 23, 45, 82]. Blind signatures provide a way for a signer to sign a message without being able to view the message content (a blinded message). Message $m$ is blinded by the requester to give a blinded message $m'$ which the requester sends to the signer. The signers signs the blinded message returning a blinded signature $s'$ to the requester.

The requester can then unblind the signed message to reveal a signature $s$ and send this along with the message $m$ to a third party verifier who can check the signature against the message.

## Partially blind signatures

Partially blind signatures were subsequently proposed [1] and are a modification of blind signatures where part of the message, called the *info* tag is not blinded. This allows some transparent information that can be read by the requester, signer and verifier to be included in the signed message.

In our architecture we use partially blind signatures for anonymous authentication including attributes in the *info* tag to provide user attributes to the application. We also include the application name in the *info* tag and limit the number of signed messages a user can obtain for any given application within a specific time period. This allows the user to be blocked by the application if they abuse the application and prevents the user from obtaining more signed messages *for that application* within that time period (they can still obtain signed messages for other applications). This provides abuse resistance in our system.

We have multiple signing servers (credential producers) and users must obtain $t$ signed messages from $n$ servers in order to authenticate through a credential consumer to an application. This means that an adversary must compromise at least $t$ of $n$ servers in order to compromise the system. Specific attack analysis details are given in Chapter 11.3.

# Chapter 11

# Security Analysis

This section presents a security analysis of the Crypto-Book architecture and its at-large and group credential schemes. We first outline the types of attackers primarily relevant to Crypto-Book, then cover a variety of threats these attackers can introduce and how Crypto-Book does (or does not) address them.

## 11.1 Security Properties

We now outline the different security properties that our system has. We analyse these against different potential attacks on the system in Chapter 11.3.

Our system design provides the following security properties.

- **Anonymity** Given a pseudonym on a third party application, it is not possible to determine from which underlying federated identity a pseudonym was generated, and vice versa. No single party other than the user can map forward from federated identity to pseudonym, or backward from pseudonym to federated identity.
- **Unlinkability** Given two pseudonyms, one on each of two different third party applications, it is not possible to determine whether they are controlled by the

same person (underlying federated identity).

- **Abuse resistance** A user can be punished if they engage in abuse such as spamming on a third party application. The user cannot get another pseudonym and continue abusing the application.
- **No impersonation** It should not be possible for any party to act on behalf of the user and authenticate as that user to an application.
- **Valid authentication** It should only be possible for a user to authenticate to applications using *their own* credentials. It should not be possible for any party to authenticate to applications without the use of such valid credentials. Users with valid credentials should be able to authenticate to applications and should not be denied service (except in the case of *abuse resistance* where the application decides to block that user).

## 11.2 Attackers

We now outline the different potential attackers that may act as adversaries in our design along with the different behaviors they may engage in. Parties in our design may themselves misbehave, or be compromised by an external adversary and be used for nefarious purposes. We analyse how our well our design withstands these different attackers and collusions between attackers in Chapter 11.3.

### 11.2.1 Application attacker

A third party appication may act as an attacker in the system. The application may try to compromise the user's privacy. The application can track when user's authenticate to them through a credential consumer and can track across time when different pseudonyms accessed the application. The application knows all activities

each user performs on that site and may keep track of and analyse this.

### 11.2.2 Identity provider attacker

A federated identity provider may act as an attacker. The identity provider may act as a *passive attacker* where it saves which users have authenticated with them, as well as when each user has authenticated with them to a credential producer (and hence knows with high probability when a user has collected their credentials from a credential producer). The identity provider may also act as an *active attacker* where the identity provider itself attempts to impersonate the user for nefarious purposes, for example to the credential producer.

### 11.2.3 Credential producer attacker

A credential producer attacker knows when a user has collected their credentials. It knows when a user has authenticated with a federated identity provider and which identity provider they authenticated with. In the case of the credential producer being a set of key servers, the key servers know when private keys are collected but do not know who has collected public keys as no authentication is required for this. In the key server case, each server knows one of the private key shares for each user. In the signing server case, each server knows their own private key that they use to sign blinded messages.

### 11.2.4 Credential consumer attacker

The credential consumer attacker knows which anonymous credentials have been used to authenticate to a third party application. They know which LRSs have been used to authenticate and who the members of the ring in the LRS are. If two users

authenticate at two different times using the same anonymity set, the attacker can determine whether or not they are the same user (using the linkability property of LRSs). In the blind signature case the attacker knows the messages that were signed and can keep track of these.

### 11.2.5 User attacker

The user may misbehave in the system in order to attack other parties. The user may attempt to troll or span an application. The user may create multiple accounts on a federated identity provider to perform Sybil attacks or sock puppetry attacks on an application. A user may also decide to publish their private key.

## 11.3 Attacks and Defenses

We now present a variety of different attacts on our system and discuss how well our design withstands them.

### 11.3.1 Single party attacks

Each attacker may individually try to compromise the overall system by themself. This section details such attacks.

#### Application attacks

**Logging pseudonym use** An application that keeps track of when pseudonyms are used cannot directly compromise anonymity by itself. It knows when pseudonyms are used but cannot determine which federated identities they correspond to (due to anonymity property of LRSs and blind signatures).

**Hosting malicious code** An application may host malicious code (for example drive-by-downloads) to try to attack the user by infecting their system with a virus or keylogger to deanonymize them. This is outside the scope of our attack model and our system does not protect against this. We propose using our system with existing works [41, 42, 89, 67, 97, 52] that protect against drive-by-downloads.

**Phishing/social engineering** An applcation may try to trick a user into providing their federated identity username and password or otherwise trick the user into deanonymizing themself. This is outside the scope of our attack model and our system does not protect against this. We propose using our system with existing works [116, 119] that protect agains phishing attacks.

### Identity provider attacks

**Passive attacks** An identity provider can save user logins and when credentials are collected. This information does not by itself compromise the anonymity of users.

**Active attacks** An identity provider may *impersonate* a user to a credential producer. In the case where only a single identity provider is used, the identity provider can collect the user's credentials and impersonate the user. The multiple identity provider design protects against this. Credential producers require users to authenticate with multiple identity providers (for example Facebook *and* LinkedIn *and* PayPal with matching name and date of birth) in order to collect their credentials. In this case a single identity provider cannot obtain a user's credentials.

### Credential producer attacks

**Key server publishes private key shares** A key server could publish its master private key or private key shares that it has generated for users. Our system is based on a $(t, n)$-cryptosystem so as long as fewer than $t$ servers publish their private key

shares, it is not possible to obtain a user's private key and so is not possible to violate their *anonymity*, *link* their different pseudonyms, or to *impersonate* that user.

**Signing server publishes private key** A signing server could publish its private key that it uses to sign blinded messages. Our system is based on a $(t, n)$-cryptosystem so it would require at least $t$ servers to be compromised in order to be able to sign the $t$ messages required to authenticate with an application. As long as fewer than $t$ servers are compromised it is not possible to sign the number of messages required to *impersonate* a user to an application.

**Credential server denial of service** A key server or signing server could be offline, not responding to requests or otherwise unavailable. So long as $t$ out of $n$ servers are still accessible, it will still be possible to collect private and public keys since our design uses a $(t, n)$ cryptosystem.

**Key server public keys inconsistent with private keys** Key servers sign every public or private key that they return using their own private key. A client can request their own public key at a different time or though a different network route. If the private key share they receive is inconsistent with their public key share they receive, they can publish both to show the server is misbehaving (as they are signed by the server). Publishing one private key share does not compromise user *anonymity* as it requires at least $t$ private key shares to construct the user's private key.

**Signing servers signs messages without verifying client** A signing server may sign messages without verifying the client with a federated identity provider, or without verifying the hash of the client's identity in the message is correct. $t$ servers would need to be compromised in order to have sufficient signed messages to *impersonate* the user to applications since we use a $(t, n)$ cryptosystem.

### Credential consumer attacks

**Invalid credential authentication** An OAuth provider credential consumer could authenticate users to applications without checking their credentials, or when their credentials are invalid. An credential consumer could also fail to authenticate users who have valid credentials to applications (a denial of service attack). These behaviors would violate the *valid authentication* security property. An application that does not want to trust an external OAuth provider to validate credentials could instead opt for one of the other credential consumer designs: either maintaining their own OAuth provider credential consumer interally, or directly verifying user credentials within the application itself.

### User attacks

**Trolling/spamming** A user may abuse their pseudonymity on an application to troll or spam that application, attempting to violate the *abuse resistance* system security property.

- The **partially blind signature** use case means that the application can block that pseudonym. The partially blind signature contains the application name in the unblinded part of the signature (the *info* field) and these are rate limited by credential producers so only a fixed number can be issued to a user per time period. If a user conducts abuse on an application, the application will block their pseudonym. The user will not be able to obtain another pseudonym for that application until the time period expires. Hence they will be blocked from conducting abuse for the remainder of that time period. This provides the *abuse resistance* security property on a per time period basis.
- The **LRS** use case provides *abuse resistance* as an LRS contains a linkage tag where if a user authenticates at two different times using the same anonymity set,

the LRSs will have the same linkage tag. The application can blacklist a linkage tag of an abusive user to prevent them from authenticating in the future. The linkage tag cannot be reversed to the user's federated identity without knowing the user's private key hench providing *abuse resistance* without violating user *anonymity*.

**Multiple social network accounts (Sybil attack)** A user may create many social network or other federated identity accounts in an attempt to perform Sybil or sock puppetry attacks on an application, violating *abuse resistance*. The multiple identity providers use case partially protects agains this as if a user has an account on a more trusted, verified identity provider, such as PayPal where identities are verified against bank accounts, then it will be difficult for a user to have several accounts without having multiple real life bank accounts in different names. However the general case of detecting Sybil accounts in social networks remains outside the scope of our attack model and is not protected against in our system. We suggest using existing system such as SybilGuard [118] or other Sybil defenses [117, 102, 110, 109, 111] along with Crypto-Book to detect and defend against Sybil accounts in social networks.

**User publishes private key** A user could opt to share their private key with other people or otherwise publish it. This would allow other people to impersonate the user, violating the *no impersonation* security property. However if the user themselves decides to publish their private key we know that they no longer desire this security property.

### 11.3.2 Collusion attacks

Attackers may collude with each other in order to have more collective power in order to compromise the system security properties. This section details such attacks.

**Key server collusion** Key servers may collude with each other to share private key shares of users. We use a $(t, n)$ cryptosystem so as long as fewer than $t$ key servers collude, it is not possible for them to obtain a user's overall private key and hence cannot *impersonate* a user or compromise their *anonymity*.

**Credential producer or identity provider colludes with application** A credential producer server (and the identity provider) knows which users collected their credentials and an application knows when a user authenticates to them but they do not know which user authenticated. A credential producer server (or identity provider) may collude with an application to perform a timing analysis attack to try to link credential pickups to authentications violating user *anonymity*. A user may mitigate such attacks by separating in time their credential pickup from their credential use. Another attack is that they might launch is on the LRS system. An application may collude with a credential server or identity provider to know which members of the anonymity set have collected their private key and which have not. This could threaten the *anonymity* of the users if only one or two of them have collected their private keys. The previously proposed *anonymous key distribution protocol* [71] could be used to anonymously distribute private keys. Alternatively if a user is worried that few or no other members of their anonymity set are likely to have collected their private key and they do not want to have the overhead of sending additional messages associated with the anonymous key distribution protocol, they could instead use the blind signature based scheme to anonymously authenticate to the application.

**User colludes with identity provider** A user attacker could collude with an identity provider to launch a Sybil attack on an application, violating the *abuse resistance* security property. The multiple identity provider use case prevents this if only one identity provider is colluding with the user. However if multiple identity

providers collude with a user they may attempt to launch a Sybil attack. Sybil resistance remains outsite the scope of our attack model and we suggest using existing systems such as SybilGuard [118] or other Sybil defenses [117, 102, 110, 109, 111] along with Crypto-Book to protect against such Sybil attacks.

**Credential producer colludes with user** If a user colludes with a credential producer, the credential producer could keep issuing the user fresh credentials to allow them to have an unlimited number of pseudonyms and hence conduct abuse on applications without being able to be blocked. This violates *abuse resistance*. Our design incorporates a $(t, n)$-threshold cryptosystem as the solution which means that at least $t$ of $n$ credential producer servers have to be compromised or agree to collude with the user to be able to launch this attack.

**Multiple identity providers collude to impersonate users** Multiple identity providers such as PayPal, Facebook and Twitter could collude in the multple identity provider use case to impersonate users. Colluding would enable them to violate the *no impersonation* security property as well as compromise user *anonymity*. Using a wide selection of diverse identity providers who mutually distrust each other (such as competing companies, those in different jurisdictions) would mitigate the risk of such an attack and likely make such an attack infeasible in practise.

**Apps collude to share anonymity sets (intersection attack)** Applications could collude to share the anonymity sets used to authenticate to them. They could attempt so see common members of the anonymity sets in order to perform an intersection attack [85, 56, 31] and threaten user *unlinkability*. To guard against this a user could ensure they always have some common members in their anonymity set when authenticating to multiple applications, especially if they are engaging in similar activities across multiple applications. Alternatively users could use the blind signature approach with distinct signed messages, one per application to protect

against this attack.

**Apps collude to share signed messages used to authenticate** If users reuse signed messages used to authenticate to one app, to authenticate to a second app, then apps could collude to share these messages an threaten user *unlinkability*. To protect against this users should not reuse signed messages across different applications.

### 11.3.3 Miscellaneous attacks

**ISP level adversary traffic analysis** An adversary who can observe large parts of the network may be able to correlate traffic between users and credential producers or identity providers, with subsequent traffic between users and applications. They may be able to use traffic analysis to perform a correlation attack to threaten user *anonymity*. Such a powerful adversary is outside the scope of our attack model and we suggest using anonymous communication systems such as Dissent [28, 60] along with Crypto-Book to guard against such attacks.

**LRS replay attack** An adversary may manage to illicitly obtain an LRS generated by another user (for example by hosting a malicious app) and then try to reuse the LRS to authenticate as that user to a different application. This would violate the *no impersonation* security property. To prevent against this credential consumers present a challenge with significant entropy to users which they must use to generate a fresh LRS on each time they authenticate.

**Credential producer hardware compromised** An adversary may physically break into a data center and gaining access to credential producer server hardware. They may take the server offline (denial of service) or copy the master private key. Since we use a $(t, n)$-threshold scheme, compromising fewer than $t$ servers does not compromise user private keys. Additionally so long as $t$ of $n$ servers are still online,

user private keys could still be served so the service would not be denied. An epoch scheme could be employed by servers whereby master private keys are rotated each epoch (for example every few months) or in the case of a server compromise. If a server is compromised, the adversary will get access to user private key shares for that epoch, but not previous epochs.

# Chapter 12

# Implementation

To demonstrate the feasibility of our architecture, we implemented a prototype of our system. The system allows a user to log in using Facebook, Paypal, or both, and to connect to the key servers and collect their private key. We also implemented three applications built on the Crypto-Book framework, CB-Wiki, CB-Dissent and CB-Drop. CB-Wiki is a Wikipedia style site where users can log in using Crypto-Book to edit the Wiki anonymously and accountably. CB-Dissent combines the Crypto-Book anonymous authentication architecture with the Dissent [28] chat system. CB-Drop is a verifiable whistleblowing application that allows users to anonymously yet credibly submit documents to journalists. We implemented OAuth provider functionality on top of Crypto-Book to allow other applications to be more easily built on top of Crypto-Book.

We used both Facebook and PayPal federated identity providers. We implemented DSA [43] and Boneh-Franklin IBE [9] key systems, RSA [91] based ring signatures [92] and DSA based LRS [64]. We implemented a $(t, n)$-threshold cryptosystem using Pedersen's distributed PKG [83] for credential producer servers to assign credentials to clients. We also implemented blind signature [96] and partially

blind signature schemes [1]. We implemented an OAuth compliant credential consumer and built three applications on top of it (CB-Wiki, CB-Dissent and CB-Drop). We deployed our credential producer servers across 10 servers globally using PlanetLab [27]. We also implemented a Google Chrome extension to automate the process of collecting a private key from the distributed servers. Since each key server is tied to a different Facebook app, the user must in turn authorize 10 Facebook apps - to automate this process we implemented a Chrome extension which requires only that the user log in to Facebook a single time in order to collect their private key.

## 12.1 Credential Assignment Mechanism

We implemented a credential assignment mechanism using keys as credentials. We implemented *two alternative ways* of collecting and assembling the key parts into a composite key:

- A downloadable application allowing the user to pickup and assemble the key parts on his own machine
- A trusted web proxy

Key distribution works as follows: the user logs into Facebook and PayPal and collects an OAuth token from each provider. The user then sends these tokens to each of the key servers to request their private key. Once a key server receives a private key request and corresponding OAuth token, it makes a request to the Facebook and PayPal APIs to verify that the credentials on the two accounts match and to obtain the user's corresponding Facebook username. If the authentication succeeds and a valid username is returned then the key server will lookup the corresponding private key in its database and return it to the requester (the proxy or the desktop app). For public key requests, the requester sends to the key server the Facebook

username that they want to obtain the public key for, and the key server looks up the key and returns it to the requester. If for any request the server does not already have a keypair saved for that Facebook username, the server will generate a keypair and store it in its database, returning the appropriate key to the requester.

Once the requester receives responses from all of the servers it will compute the composite private and public keys. The requester, now in possession of all necessary keys, generates the linkable ring signature for the specified file.

## 12.2 Credential Schemes Implemented

We implemented three different key credential schemes – one based on DSA [64], one basd on RSA [92] and one based on elliptic curve cryptography (Boneh-Franklin identity based encryption [9]). We implemented two different blind signature credential schemes - one based on Shen *et al.*'s blind signatures [96] and one based on partially blind signatures [1].

### 12.2.1 DSA-Based Scheme

We implemented an LRS scheme [64] based on DSA keys. DSA keys operate in a group $G$ of order $p$ and are of the form $Y = g^x \bmod p$ where $Y$ is the public key and $x$ is the private key. A composite key can be formed from a set of keys by adding the private keys and multiplying the public keys.

Our distributed key distribution relies on the fact that we can generate a composite private key $x_c$ from a list of private keys $x_0, x_1, \ldots, x_n$ by summing them such that $x_c = x_0 + x_1 + \cdots + x_n \bmod q$. The corresponding composite public key $Y_c = g^{x_0+x_1+\cdots+x_n \bmod q} \bmod p$. This is equivalent to multiplying the corresponding public keys $Y_c = y_0 * y_1 * \cdots * y_n \bmod p$ so we can calculate the composite public key

without knowledge of the private keys.

### 12.2.2 RSA-Based Scheme

We implemented ring signatures [92] using RSA keys. Unlike DSA based LRS, RSA based ring signatures are not linkable. However they provide the additional property of *deniability* so even if a user's private key is compromised, their previously generated ring signatures cannot be deanonymized. Hence RSA based ring signatures provide stronger anonymity guarantees than DSA based linkable ring signatures, although RSA based ring signatures do not provide the abuse resistance that DSA based LRS provides.

### 12.2.3 Boneh-Franklin Identity-Based Encryption Scheme

In addition to the DSA-based scheme, we also implemented a scheme using Boneh-Franklin [9] identity-based encryption (IBE) keys. Boneh-Franklin is a scheme based on elliptic curves where a string such as a user's Facebook ID *is* their public key. Their private key is generated for them by a private key generator (PKG). We implemented a distributed PKG where a user's private key is split among $n$ key servers using Shamir secret sharing [94].

A Boneh-Franklin PKG generates a user's private keys using a master private key, $s$. The PKG multiplies this by $Q_{ID}$ which is derived from the user's public key, to compute the user's private key, $Q_{priv} = sQ_{ID}$. The master private key $s$ can be split among $n$ key servers by giving each server a Shamir secret share $s_i$ of the master private key. Each key server now returns to the client a private key part $Q_{priv}^{(i)} = s_i Q_{ID}$. Using the appropriate Lagrange coeffiecients $\lambda_i$ the user can then construct their overall composite private key $Q_{priv} = \sum \lambda_i Q_{priv}^{(i)}$ from the private key

parts $Q^i_{priv}$.

We demonstrated the use of Boneh-Franklin keys by implementing an encrypted Facebook messaging app that allows a user to send an encrypted Facebook message to any other Facebook user using Boneh-Franklin identity based encryption. The recipient, even if they have not previously registered with the service, can then collect their private key from the key servers and decrypt and view the message.

### 12.2.4 Blind and Partially Blind Signature Schemes

We implemented two credential schemes using blind signatures as credentials, one based on blind signatures [96] and one based on partially blind signatures [1]. The partially blind signatures are more complex to implement but can be used to provide attributes and abuse resistant credentials. Our results in Chapter 14 show that they also give better performance than the traditional blind signatures.

# Chapter 13

# Applications

We built on our implementation from Chapter 12 to develop three realistic applications using Crypto-Book. These are CB-Wiki, CB-Dissent and CB-Drop described below in Chapter 13.1, Chapter 13.2 and Chapter 13.3 respectively.

## 13.1 CB-Wiki

CB-Wiki is a system based on the software behind Wikipedia that allows users to log in as an *accountable, anonymous* user instead of as a *personally identifiable* user. This allows users to edit Wiki pages without disclosing their identity but at the same time provides resistence to abuse.

CB-Wiki provides privacy preserving accounts instead of traditional accounts. Instead of having to create a Wikipedia account in order to be able to edit pages, users instead are able to *Log in with Crypto-Book* in a similar way that many sites allow users to *Log in with Facebook*. When a user chooses to log in with Crypto-Book, they are redirected to the Crypto-Book website. They are then required to submit a linkable ring signature (via file upload) to the Crypto-Book servers for verification. If it is valid, the servers redirect the user back to CB-Wiki where they will log-in

with an anonymized account.

The anonymous ID is derived from the linkable ring signature that the user provides. The signatures proves that they are a member of some group and the *linkage tag* is used to tell if two people signing in are the same user. If a user signs in with Crypto-Book and then abuses the Wiki site, the user's linkage tag can be blocked so they will no longer be able to access the site, ensuring accountability.

CB-Wiki site administrators are able to interact with that anonymized user in exactly the same way as with regular, non-anonymous users (which are also supported by CB-Wiki). For example administrators can message that user, interact with them on their talk page, view their edits to the Wiki and block or ban them if necessary. This allows for user accountability while simultaneously protecting user privacy.

## 13.2 CB-Dissent

CB-Dissent is a system that combines the Crypto-Book anonymous authentication architecture with the Dissent[28, 115] chat system. Dissent is a practical group anonymity system that offers provable anonymity with accountability. CB-Dissent is an anonymous authentication platform for Dissent [28, 115], consisting of two major components:

- Anonymous Dissent chat group creation
- Anonymous authentication to Dissent

**Dissent group requests** Using CB-Dissent, users request Dissent chat groups as follows: The client uses their web browser to connect to the the Crypto-Book servers. The client then generates a linkable ring signature using keys collected from the CB-Dissent key servers corresponding to the Facebook users they want in their anonymous Dissent chat group. The client sends the ring signature to the CB-Dissent

servers to be verified. The Crypto-Book servers then verify the ring signature. If the signature is successfully verified by CB-Dissent, then CB-Dissent checks its database to determine whether there is already a Dissent session running for that group. If there is no active Dissent session for that group then CB-Dissent starts three Dissent servers, computes a group authentication code and inserts a corresponding record into the database along with the group members.

When a client is successfully authenticated by CB-Dissent as being a member of the group, the servers send the group authentication code to the client. The client uses this authentication code to connect to the Dissent servers: They send the authentication code to another CB-Dissent service which checks the authentication code, looks up the Dissent servers for that group and connecccts the user's Dissent client to the Dissent anonymous chat group.

**Dissent anonymous authentication** We also implemented anonymous authentication directly within Dissent using linkable ring signatures. The Dissent servers maintain a ring of public keys for that group. When a client attempts to connect to a Dissent server they are challenged to provide a valid linkable ring signature corresponding to the ring of keys for that group. The client generates this using their private key (which corresponds to one of the public keys in the ring) and sends the signature to the server. The server verifies the signature against the ring of public keys and, if successful, allows the client to connect.

## 13.3 CB-Drop

CB-Drop builds on SecureDrop [93], an open-source whistleblower submission system managed by Freedom of the Press Foundation. SecureDrop allows journalists to accept sensitive documents from anonymous sources via a web interface. CB-Drop


Figure 13.1: The CB-Drop document source interface

adds credibilty to leaks by allowing a source to anonymously sign a document using a relevant anonymity set and the DSA based scheme described in Chapter 12.2.1 before submitting it via SecureDrop as shown in Figure 13.1. Upon retrieving the document, a journalist can then verify the signature, increasing confidence in the authenticity of the leak without compromising the source's anonymity.

To retrieve a verifiably leaked document, a journalist connects to the SecureDrop journalist web interface using Tor. They then download and decrypt any waiting documents using their private key. At this point, CB-Drop provides the additional option to verify the document by processing the signature and retrieving the signing anonymity set. This information, combined with past submissions associated with the codename, allows the journalist to make a more informed decision regarding the authenticity of the leak.

# Chapter 14

# Evaluation

To evaluate the practicality of Crypto-Book we consider both end-to-end measurements in expected deployment scenarios and microbenchmarks focusing on scalability of specific components. We first describe the experimental setup and then evaluate credential production and consumption in both proposed credential schemes. We conclude by evaluating Crypto-Book in the context of our example applications.

## 14.1 Experimental Setup

The experimental setup for the following evaluations include three classes of machines, based on role in the system:

- **Clients** are consumer laptops with 2.4GHz Intel Core i5 processors and 8GB of RAM.
- **Credential producers** are nodes on the geographically distributed PlanetLab [27] network. A typical PlanetLab node has a 2.4GHz Intel Xeon processor and 4GB of RAM.

- **Credential consumers** are commercial shared hosting providers also with 2.4GHz Intel Xeon processors and 16GB of RAM.

In selecting experimental key pairs we followed NIST recommendations [81] for DSA keys where the tuple $(L, N)$ specifies the bit-length of the $p$ and $q$ parameters, respectively. If a parameter tuple is not specified, it is assumed to be $(1024, 160)$.

## 14.2 Producing Credentials

In this section we evaluate the production of credentials in both the at-large and group credential schemes.

### 14.2.1 Facebook Application Authorization

In both the at-large and group credential schemes a first-time user must authorize a credential producer's application with a supported federated identity provider before retrieving any credentials from that producer. We evaluated the time taken to complete this step for a varying number of credential producers, with Facebook as the federated indentity provider. We used our Chrome extension to automate the authorization process and performed all authorizations in parallel. Results are presented in Figure 14.1, drawing distinction between time spent authenticating with Facebook and time spent authorizing the Crypto-Book application. The times shown are those of the last credential producer to return. **A client only ever needs to perform this setup step once, the first time they ever use Crypto-Book.**

### 14.2.2 At-Large Credentials

An at-large credential consists of partially blind signatures from $t$ credential producers. As each signature is obtained independently from all others, we focus on time

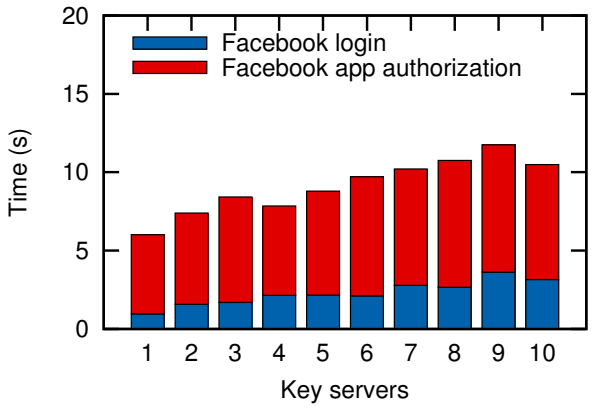


Figure 14.1: Facebook application authorization

| Key Parameters | Signature Size (Bytes) |
|---|---|
| (1024,160) | 210 |
| (2048,224) | 287 |
| (2048,256) | 325 |
| (3072,256) | 326 |

Table 14.1: Partially blind signature size (communication costs)

taken to obtain a single signature. As discussed in Section 8, a signature is generated collaborately by the client and a producer. Network overhead consists of two round trips between producer and client where the second trip is dependent on the size (and hence strength) of the signature. Signature sizes for varying signing key sizes are shown in Table 14.1.

The computation costs for partially blind signature operations are shown in Figure 14.2. Credential production consists only of the signing operation, which, for a 2048-bit signing key, takes approximately 50ms of computation time, with effort divided fairly evenly between client and producer.

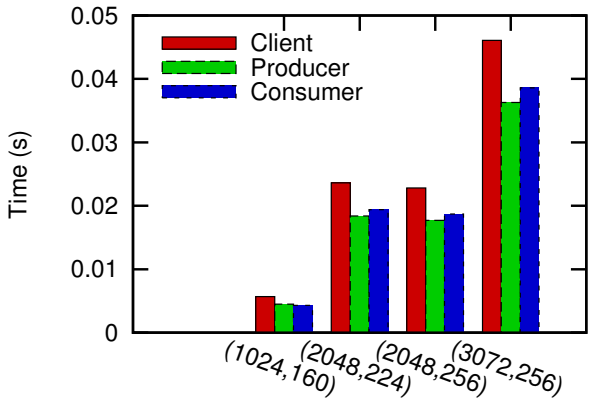


Figure 14.2: Partially blind signature operations (CPU costs)

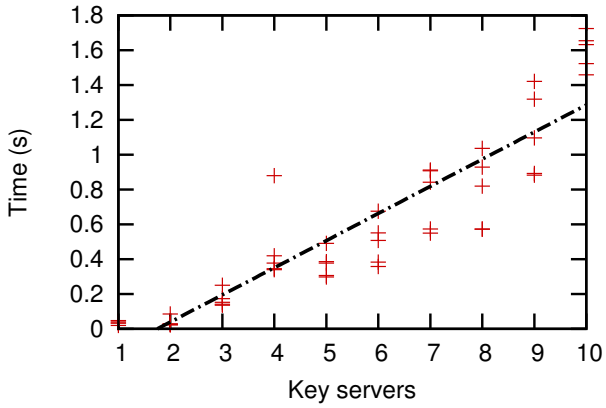


Figure 14.3: Distributed keypair generation

### 14.2.3 Group Credentials

**Keypair Generation** In our group credential scheme, the first time a key pair is requested it must be collectively generated by the credential producers. We evaluated the time required to generate a single key pair for a varying number of credential producers and present the results in Figure 14.3. Times shown include only operations involving producers; returning the keys to the client is evaluated separately. For a reasonable number of producers, we find that the results scale approximately linearly.

**Key Retrieval** We next evaluate the time taken for a client to retrieve a single

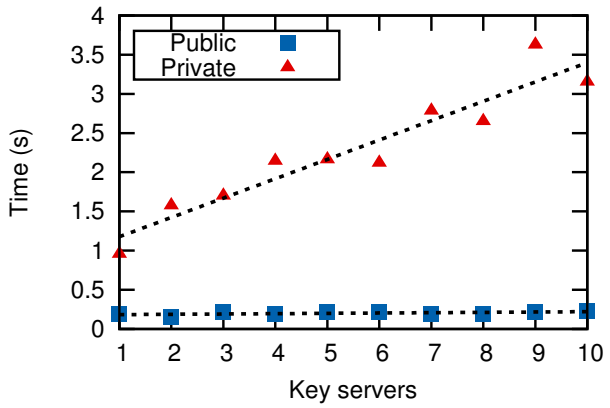


Figure 14.4: Retrieval of previously generated keys (communication costs)

key, either private or public, from the set of credential producers. Requests to all producers are performed in parallel. We present the results in Figure 14.4. For public keys, we find near constant response. Times shown for private keys include Facebook authentication, which accounts for the difference when compared to public key requests. **Private key requests will be rare, as after retrieval users retain their private keys locally, stored in the Chrome extension.**

### 14.2.4 Neff-Based Group Credentials

The primary cost involved with the Neff-based group credential scheme is the initial setup cost, that is the Neff key shuffle and rebasing. Once keys have been shuffled and rebased by credential producers, clients can simply authenticate to credential consumers using simple digital signature authentication (constant time). Figure 14.5 shows the setup cost of for Neff-based group credentials for 3, 5 and 7 servers (credential producers). Setup costs for Neff-based group credentials are significant in comparison to other schemes we propose, however may be acceptable for certain applications since the setup need only be carried out once. After that clients can authenticate in constant time. This is the tradeoff between setup costs (credential

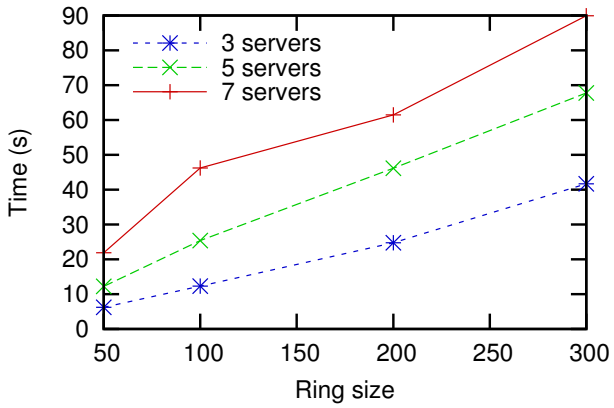


Figure 14.5: Neff-based group credential setup costs

production) and authentication costs (credential consumption).

## 14.3 Consuming Credentials

In this section we evaluate the consumption of credentials in both the at-large and group credential schemes.

### 14.3.1 At-Large Credentials

In evaluating the consumption of at-large credentials we assume that all credential producers' signing keys are well-known to all credential consumers. As a result, authentication via at-large credentials consists only of the transmission and verification of partially blind signatures. As shown previously in Figure 14.1, a threshold $t$ at-large credential is approximately $300t$ bytes in size; at these sizes credential upload times are heavily dependant on network properties.

To evaluate the costs of verification we considered $t = 1$ at-large credentials for varying key size. Results are shown in Figure 14.2; for a 2048-bit key verification takes less than 20ms. As each signature verification is completely independent of all others, for expected values of $t$ ($\leq 10$), signature verification can be performed

| Entity | Operation | Type of Cost | Time (s) |
|---|---|---|---|
| Client | Produce LRS | CPU | 0.257 |
| Credential Consumer | Fetch Public Keys | Communication | 1.011 |
| | Verify LRS | CPU | 0.035 |
| Client-Consumer Network Latencies | | Communication | 0.304 |
| | | **Total User-Observable** | **1.607** |

Table 14.2: Real World End-to-end Group Authentication for Group Size of 10

largely in parallel.

### 14.3.2 Group Credentials

In order to perform an **end-to-end evaluation** of authentication using group credentials, we first created a group credential including ten Facebook identities belonging to members of the author's research group. Users then each collected their group credential and used our Chrome extension to authenticate to an internal website with their Crypto-Book identity. We used an application-embedded consumer and three credential producers on networks different from the consumer's. We recorded timings for each phase of 117 authentications and present the averages in Table 14.2. On average, the total user-observable authentication time was 1.6 seconds; this is approximately a 1.2 second overhead compared to non-anonymous federated authentication with Facebook.

In addition to the real world deployment within our research group, we also simulated the experiment for group sizes of 100 and 250. The results are shown in Table 14.3 and 14.4 respectively. For group sizes of less that 100 which we would consider for practical usage, costs are primarily communication bound. For larger group sizes, costs become CPU bound. This scheme is reasonable for practical purposes. If faster authentications are desired, the Neff-based group credential scheme could be used. The tradeoff here is that there is a bigger initial setup cost when

| Entity | Operation | Type of Cost | Time (s) |
|---|---|---|---|
| Client | Produce LRS | CPU | 1.641 |
| Credential Consumer | Fetch Public Keys | Communication | 0.885 |
| | Verify LRS | CPU | 1.632 |
| Client-Consumer Network Latencies | | Communication | 0.453 |
| | | **Total User-Observable** | **4.612** |

Table 14.3: End-to-end Group Authentication for Simulated Group Size of 100

| Entity | Operation | Type of Cost | Time (s) |
|---|---|---|---|
| Client | Produce LRS | CPU | 4.263 |
| Credential Consumer | Fetch Public Keys | Communication | 0.799 |
| | Verify LRS | CPU | 4.120 |
| Client-Consumer Network Latencies | | Communication | 0.484 |
| | | **Total User-Observable** | **9.666** |

Table 14.4: End-to-end Group Authentication for Simulated Group Size of 250

initializing the group, however after that login times are constant as they consist of a simple digital signature signing and verification as opposed to a linkable ring signature. The scalability of setup times for Neff-based group credentials are shown in Figure 14.5.

To investigate how group credential authentication scales with group size, we considered each operation from Table 14.2 separately. We found that by bundling public key requests we were able to fetch up to 1000 public keys in near-constant time, identifying the ring signature operations as the limiting factor for larger groups. We then measured the computation time for each ring signature operation, varying the ring size between 1 and 1000. Results for signing and verification are shown in Figures 14.6 and 14.7, respectively. Both operations scale linearly with ring size and, for ring sizes near 100 and 2048-bit keys, both operations complete in one second.

We additionally measured the size of the linkable ring signature produced for varying ring sizes. This signature is what the client sends to the credential consumer with each authentication, thus overall client-consumer network latency depends on

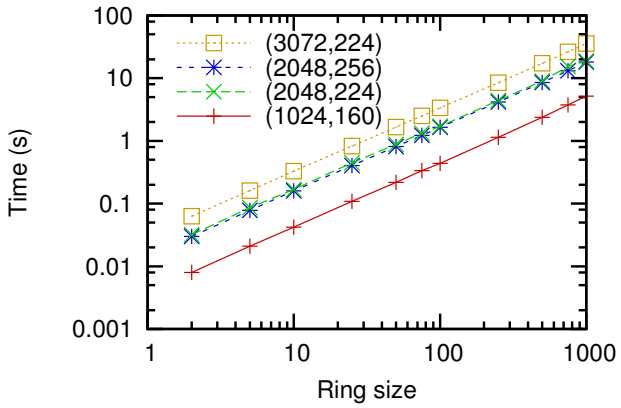


Figure 14.6: Linkable ring signature generation (CPU costs)

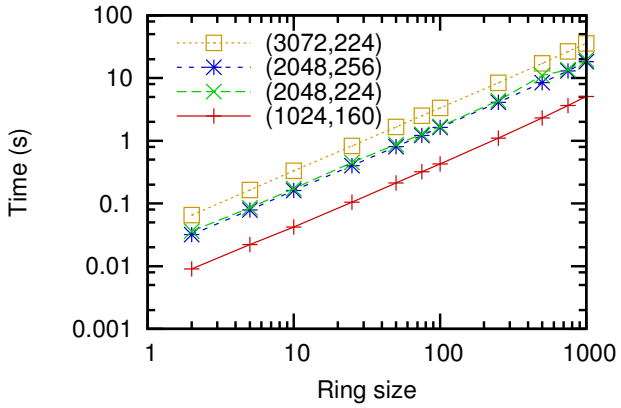


Figure 14.7: Linkable ring signature verification (CPU costs)

signature size. Results are shown in Figure 14.8. We found that signature size scales linearly with ring size and that, for ring sizes near 100 and 2048-bit keys, signatures are less than 10kB.

## 14.4 CB-Dissent Authentication

Finally, we evaluated group authentication in our CB-Dissent implementation. This experiment measures the time required for a client to authenticate with a single Dissent server acting as an application-embedded consumer. We again varied the group size between 1 and 1000, presenting the results in Figure 14.9. For a group

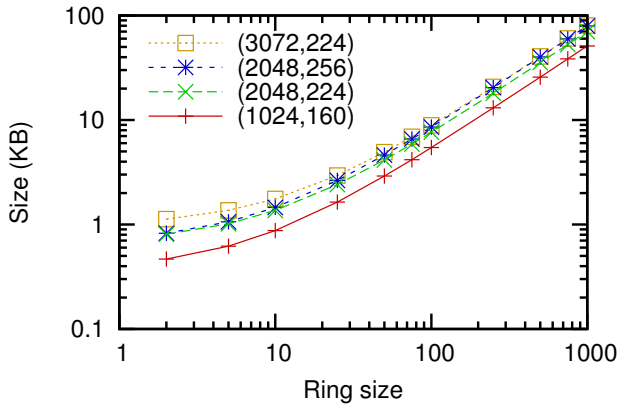


Figure 14.8: Linkable ring signature size (communication costs)

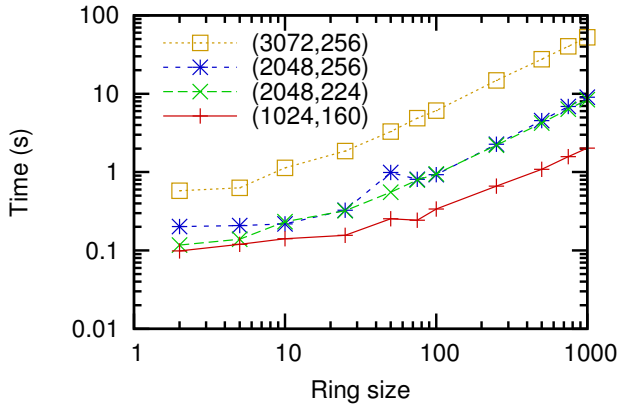


Figure 14.9: CB-Dissent authentication time

size of 100 and 2048-bit keys, authentication again takes under 1 second. While this is a significant overhead compared to traditional pre-exchanged key authentication, it remains well within practical limits.

## 14.5 Code modification

We used CLOC [32] to count the lines of code that we had to add to modify Media Wiki, Dissent, and SecureDrop to integrate with Crypto-Book. Only small modifications were required in all cases:

- **M**edia Wiki required only 96 additional lines of code to integrate it with our Crypto-Book login system (deployed externally of Media Wiki).
- **D**issent group request was implemented entirely outside of the Dissent codebase, making only command line level calls to Dissent services, so required no modification of Dissent code.
- **D**issent anonymous authentication was implemented as an additional module to Dissent requiring 574 additional lines of code. This is in comparison to Dissent's pre-existing authentication system that consisted of 838 lines of code.
- **S**ecureDrop integration with our Crypto-Book signing system required only 35 additional lines of code.

# Chapter 15

# Future Directions

An area for future work may be in investigating the impact different anonymity set choices have on user privacy protection. In this dissertation we have used a scheme where users are batched into groups by Crypto-Book and where these groups are shared across all third party sites and services. Future work is required to investigate how custom, per third party service group definition can be applied without threatening user privacy.

A limitation of the current system is that linkable ring signature size scales linearly with ring size. Dodis *et al.* [39] proposed a scheme for constant space ring signatures. These signatures also have the property that both the signer and the verifier can perform a one-time computation proportional to the size of the ring which then allows them to produce and verify many subsequent signatures in constant time. Future work could incorporate such a scheme to reduce signing and verification time. In subsequent work, Tsang *et al.* [106] proposed a scheme for constant space accumulator based linkable ring signatures. Future work could incorporate such schemes to reduce signature size.

Another interesting line of inquiry may be investigating how our privacy pro-

tecting identities can be tied back into anonymous posting within Facebook as is proposed in the Faceless framework [98]. This would allow for anonymous discussion within existing social networking sites. Other anonymous credential schemes such as BLAC [103, 104, 5] could be integrated with Crypto-Book.Finally, it may be worthwhile to look at what other applications could be developed on top of Crypto-Book using our privacy protecting identities. For example an anonymous group Twitter application similar to GroupTweet [47] with improved end user privacy guarantees that do not require the end user to trust the service provider with their identity.

# Chapter 16

# Conclusions

We have demonstrated Crypto-Book, a novel architecture for providing privacy preserving online identities based on federated identity providers. We have implemented three major applications, CB-Wiki, CB-Dissent, and CB-Drop, on top of Crypto-Book and shown them to have good scalability properties. We believe that Crypto-Book is a practical way to provide federated identity users with abuse resistant pseudonyms. There remain a large number of areas for future research based on our architecture as well as many applications that could be developed on top of our framework leaving open a wide range of areas for investigation building on our results in future work.

# Bibliography


[1] Abe, M., and Okamoto, T. Provably secure partially blind signatures. In *Advances in CryptologyCRYPTO 2000* (2000), Springer, pp. 271–286.

[2] Abe, M., and Okamoto, T. Provably secure partially blind signatures. In *20th CRYPTO* (2000).

[3] Ateniese, G., Camenisch, J., Joye, M., and Tsudik, G. A practical and provably secure coalition-resistant group signature scheme. In *Advances in CryptologyCRYPTO 2000* (2000), Springer, pp. 255–270.

[4] Ateniese, G., Song, D., and Tsudik, G. Quasi-efficient revocation of group signatures. In *Financial Cryptography* (2003), Springer, pp. 183–197.

[5] Au, M. H., Kapadia, A., and Susilo, W. Blacr: Ttp-free blacklistable anonymous credentials with reputation.

[6] Boneh, D., Boyen, X., and Shacham, H. Short group signatures. In *Advances in Cryptology–CRYPTO 2004* (2004), Springer, pp. 41–55.

[7] Boneh, D., and Franklin, M. Efficient generation of shared rsa keys. In *Advances in CryptologyCRYPTO'97*. Springer, 1997, pp. 425–439.

[8] Boneh, D., and Franklin, M. Efficient generation of shared rsa keys. *Journal of the ACM (JACM) 48*, 4 (2001), 702–722.

[9] Boneh, D., and Franklin, M. Identity-based encryption from the Weil pairing. In *Advances in Cryptology - CRYPTO 2001* (2001), Springer, pp. 213–229.

[10] Boneh, D., and Shacham, H. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM conference on Computer and communications security* (2004), ACM, pp. 168–177.

[11] Boyd, C. Digital multisignatures. *Cryptography and coding* (1986).

[12] Brands, S. Electronic cash systems based on the representation problem in groups of prime order. *Preproceedings of Advances in CryptologyCRYPTO93* (1993), 26–1.

[13] Brickell, E., and Li, J. Enhanced privacy id: A direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society* (2007), ACM, pp. 21–30.

[14] Camenisch, J., and Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in CryptologyCRYPTO 2002*. Springer, 2002, pp. 61–76.

[15] Camenisch, J. L., Piveteau, J.-M., and Stadler, M. A. Blind signatures based on the discrete logarithm problem. In *Advances in CryptologyEUROCRYPT'94* (1995), Springer, pp. 428–432.

[16] Chaum, D. Blind signatures for untraceable payments. In *Crypto* (1982), vol. 82, pp. 199–203.

[17] Chaum, D. Blind signatures for untraceable payments. In *CRYPTO* (1982).

[18] Chaum, D. Blind signatures for untraceable payments. In *Advances in cryptology* (1983), Springer, pp. 199–203.

[19] Chaum, D. Blind signature system. In *Advances in cryptology* (1984), Springer, pp. 153–153.

[20] Chaum, D. Privacy protected payment, smart card 2000, 1989.

[21] Chaum, D., den Boer, B., van Heyst, E., Mjølsnes, S., and Steenbeek, A. Efficient offline electronic checks. In *Advances in CryptologyEUROCRYPT89* (1990), Springer, pp. 294–301.

[22] Chaum, D., Fiat, A., and Naor, M. Untraceable electronic cash. In *Proceedings on Advances in cryptology* (1990), Springer-Verlag New York, Inc., pp. 319–327.

[23] Chaum, D., and Pedersen, T. P. Wallet databases with observers. In *Advances in CryptologyCRYPTO92* (1993), Springer, pp. 89–105.

[24] Chaum, D., and Van Heyst, E. Group signatures. In *Advances in CryptologyEUROCRYPT91* (1991), Springer, pp. 257–265.

[25] Chaum, D., and Van Heyst, E. Group signatures. In *Advances in Cryptology EUROCRYPT'91* (1991), Springer, pp. 257–265.

[26] Chaum, D., and Van Heyst, E. Group signatures. In *EUROCRYPT* (Apr. 1991).

[27] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review 33*, 3 (2003), 3–12.

[28] Corrigan-Gibbs, H., and Ford, B. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM conference on Computer and communications security* (2010), ACM, pp. 340–350.

[29] Croft, R., and Harris, S. Public-key cryptography and re-usable shared secrets. *Cryptography and Coding, Institute of Mathematics & Its Applications (IMA)* (1989), 189–201.

[30] Cutillo, L. A., Molva, R., and Strufe, T. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE 47*, 12 (2009), 94–101.

[31] Danezis, G., and Serjantov, A. Statistical disclosure or intersection attacks on anonymity systems. In *Information Hiding* (2005), Springer, pp. 293–308.

[32] Danial, A. Counting Lines of Code. http://cloc.sourceforge.net/.

[33] Desmedt, Y. Society and group oriented cryptography: A new concept. In *Advances in CryptologyCrypto87* (1988), Springer, pp. 120–127.

[34] Desmedt, Y., and Quisquater, J.-J. Public-key systems based on the difficulty of tampering (is there a difference between DES and RSA?). In *Advances in Cryptology - CRYPTO86* (1987), Springer, pp. 111–117.

[35] Desmedt, Y. G. Threshold cryptography. *European Transactions on Telecommunications 5*, 4 (1994), 449–458.

[36] Dey, A., and Weis, S. Pseudoid: Enhancing privacy in federated login. *Hot topics in privacy enhancing technologies* (2010), 95–107.

[37] Di Raimondo, M., Gennaro, R., and Krawczyk, H. Deniable authentication and key exchange. In *CCS* (2006).

[38] Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. Tech. rep., DTIC Document, 2004.

[39] Dodis, Y., Kiayias, A., Nicolosi, A., and Shoup, V. Anonymous identification in ad hoc groups. In *Advances in Cryptology - EUROCRYPT 2004* (2004), Springer, pp. 609–626.

[40] Douceur, J. R. The sybil attack. In *Peer-to-peer Systems*. Springer, 2002, pp. 251–260.

[41] Egele, M., Kirda, E., and Kruegel, C. Mitigating drive-by download attacks: Challenges and open problems. In *iNetSec 2009–Open Research Problems in Network Security*. Springer, 2009, pp. 52–62.

[42] Egele, M., Wurzinger, P., Kruegel, C., and Kirda, E. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2009, pp. 88–106.

[43] Federal Information Processing Standards Publication. Digital signature standard (DSS), July 2013. FIPS 186-4.

[44] Felt, A., and Evans, D. Privacy protection for social networking apis. *2008 Web 2.0 Security and Privacy (W2SP08)* (2008).

[45] Ferguson, N. Single term off-line coins. In *Advances in CryptologyEUROCRYPT93* (1994), Springer, pp. 318–328.

[46] Golijan, R. Consumer Reports: Facebook privacy problems are on the rise.

[47] GroupTweet. http://www.grouptweet.com/.

[48] Guha, S., Tang, K., and Francis, P. NOYB: Privacy in online social networks. In *Proceedings of the first workshop on Online social networks* (2008), vol. 1, ACM, pp. 49–54.

[49] Hammer-Lahav, E. RFC 5849: The OAuth 1.0 Protocol (2010).

[50] Hardt, D. The OAuth 2.0 Authorization Framework.

[51] Henry, R., Henry, K., and Goldberg, I. Making a nymbler nymble using verbs. In *Privacy Enhancing Technologies* (2010), Springer, pp. 111–129.

[52] Hsu, F.-H., Tso, C.-K., Yeh, Y.-C., Wang, W.-J., and Chen, L.-H. Browserguard: A behavior-based solution to drive-by-download attacks. *Selected Areas in Communications, IEEE Journal on 29*, 7 (2011), 1461–1468.

[53] Hühnlein, D., Jacobson, M., and Weber, D. Towards practical non-interactive public key cryptosystems using non-maximal imaginary quadratic orders. In *Selected Areas in Cryptography* (2001), Springer, pp. 275–287.

[54] Jahid, S., Mittal, P., and Borisov, N. EASiER: Encryption-based access control in social networks with efficient revocation. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (2011), ACM, pp. 411–415.

[55] Johnson, P. C., Kapadia, A., Tsang, P. P., and Smith, S. W. Nymble: Anonymous ip-address blocking. In *Privacy Enhancing Technologies* (2007), Springer, pp. 113–133.

[56] Kedogan, D., Agrawal, D., and Penz, S. Limits of anonymity in open environments. In *Information Hiding* (2003), Springer, pp. 53–69.

[57] Khattak, Z. A., Manan, J.-l. A., Sulaiman, S., et al. Analysis of open environment sign-in schemes-privacy enhanced & trustworthy approach. *Journal of Advances in Information Technology 2*, 2 (2011), 109–121.

[58] Kiayias, A., Tsiounis, Y., and Yung, M. Traceable signatures. In *Advances in Cryptology-EUROCRYPT 2004* (2004), Springer, pp. 571–589.

[59] Kontaxis, G., Polychronakis, M., and Markatos, E. P. SudoWeb: Minimizing information disclosure to third parties in single sign-on platforms. In *Information Security*. Springer, 2011, pp. 197–212.

[60] Le Blond, S., Choffnes, D., Zhou, W., Druschel, P., Ballani, H., and Francis, P. Towards efficient traffic-analysis resistant anonymity networks. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM* (2013), ACM, pp. 303–314.

[61] Li, J., Li, N., and Xue, R. Universal accumulators with efficient nonmembership proofs. In *Applied Cryptography and Network Security* (2007), Springer, pp. 253–269.

[62] Lin, Z., and Hopper, N. Jack: Scalable accumulator-based nymble system. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society* (2010), ACM, pp. 53–62.

[63] Lindell, Y. Anonymous authentication. *Journal of Privacy and Confidentiality 2*, 2 (2007), 4.

[64] Liu, J. K., Wei, V. K., and Wong, D. S. Linkable spontaneous anonymous group signature for ad hoc groups (extended abstract). In *ACISP* (2004), pp. 325–335.

[65] Liu, J. K., and Wong, D. S. Linkable ring signatures: Security models and new schemes. In *ICCSA* (May 2005).

[66] Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. Analyzing facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 61–70.

[67] Lu, L., Yegneswaran, V., Porras, P., and Lee, W. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM conference on Computer and communications security* (2010), ACM, pp. 440–450.

[68] Lucas, M. M., and Borisov, N. Flybynight: mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society* (2008), ACM, pp. 1–8.

[69] Luo, W., Xie, Q., and Hengartner, U. Facecloak: An architecture for user privacy on social networking sites. In *Computational Science and Engineering, 2009. CSE'09. International Conference on* (2009), vol. 3, IEEE, pp. 26–33.

[70] Maganis, G., Shi, E., Chen, H., and Song, D. Opaak: using mobile phones to limit anonymous identities online. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, pp. 295–308.

[71] Maheswaran, J., Wolinsky, D. I., and Ford, B. Crypto-Book: An architecture for privacy preserving online identities. In *HotNets-XII: Proceedings of the 12th ACM Workshop on Hot Topics in Networks* (2013), ACM.

[72] Maurer, U., and Yacobi, Y. Non-interactive public-key cryptography. In *Advances in Cryptology-EUROCRYPT'91* (1991), Springer, pp. 498–507.

[73] MediaWiki. http://www.mediawiki.org.

[74] Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P., and Bhattacharjee, B. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement* (2007), ACM, pp. 29–42.

[75] Naor, M. Deniable ring authentication. In *'02 CRYPTO* (2002).

[76] Narayanan, A., Thiagarajan, N., Lakhani, M., Hamburg, M., and Boneh, D. Location privacy via private proximity testing. In *Proc. of NDSS* (2011), vol. 2011.

[77] Neal, R. W. Edward snowden reveals quantum insert: Nsa and gchq used fake linkedin and slashdot pages to install spyware.[online] international business times, november 11, 2013.

[78] Neff, C. A. A verifiable secret shuffle and its application to e-voting. In *8th CCS* (Nov. 2001).

[79] Nguyen, L. Accumulators from bilinear pairings and applications. In *Topics in Cryptology–CT-RSA 2005*. Springer, 2005, pp. 275–292.

[80] NIST Special Publication 800: Recommendation for Key Management. http://web.archive.org/web/20140703231420/http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[81] The FIPS 186-4 Digital Signature Algorithm Validation System. http://csrc.nist.gov/groups/STM/cavp/documents/dss2/dsa2vs.pdf.

[82] Okamoto, T., and Ohta, K. Universal electronic cash. In *Advances in CryptologyCRYPTO91* (1992), Springer, pp. 324–337.

[83] Pedersen, T. P. A threshold cryptosystem without a trusted party. In *Advances in CryptologyEUROCRYPT91* (1991), Springer, pp. 522–526.

[84] Quora. http://www.quora.com.

[85] Raymond, J.-F. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies* (2001), Springer, pp. 10–29.

[86] Recordon, D., and Fitzpatrick, B. OpenID Authentication 1.1. *Finalized OpenID Specification, May* (2006).

[87] Recordon, D., and Reed, D. OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management* (2006), ACM, pp. 11–16.

[88] Reich, E. *Deniable Ring Signatures*. PhD thesis, Massachusetts Institute of Technology, 2007.

[89] Rieck, K., Krueger, T., and Dewald, A. Cujo: efficient detection and prevention of drive-by-download attacks. In *Proceedings of the 26th Annual Computer Security Applications Conference* (2010), ACM, pp. 31–39.

[90] Rivest, R., Shamir, A., and Tauman, Y. How to leak a secret: Theory and applications of ring signatures. In *Essays in Memory of Shimon Even* (2006), pp. 164–186.

[91] Rivest, R. L., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[92] Rivest, R. L., Shamir, A., and Tauman, Y. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001*. Springer, 2001, pp. 552–565.

[93] SecureDrop. https://pressfreedomfoundation.org/securedrop/.

[94] Shamir, A. How to share a secret. *Communications of the ACM 22*, 11 (1979), 612–613.

[95] Shamir, A. Identity-based cryptosystems and signature schemes. In *Advances in cryptology* (1985), Springer, pp. 47–53.

[96] Shen, V. R., Chung, Y. F., Chen, T. S., Lin, Y. A., et al. A blind signature based on discrete logarithm problem. *INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING INFORMATION AND CONTROL 7*, 9 (2011), 5403–5416.

[97] Song, C., Zhuge, J., Han, X., and Ye, Z. Preventing drive-by download via inter-module communication monitoring. In *Proceedings of the 5th ACM symposium on information, computer and communications security* (2010), ACM, pp. 124–134.

[98] Song, X., Wolinsky, D. I., and Ford, B. Faceless: decentralized anonymous group messaging for online social networks. In *SNS* (2012), E. Yoneki, D. Frey, and I. Brown, Eds., ACM, p. 13.

[99] StackOverflow. http://www.stackoverflow.com.

[100] Sweeney, L. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 05 (2002), 557–570.

[101] Tanaka, H. A realization scheme for the identity-based cryptosystem. In *CRYPTO* (1987), pp. 340–349.

[102] Tran, D. N., Min, B., Li, J., and Subramanian, L. Sybil-resilient online content voting. In *NSDI* (2009), vol. 9, pp. 15–28.

[103] Tsang, P. P., Au, M. H., Kapadia, A., and Smith, S. W. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *Proceedings of the 14th ACM conference on Computer and communications security* (2007), ACM, pp. 72–81.

[104] Tsang, P. P., Au, M. H., Kapadia, A., and Smith, S. W. Blac: Revoking repeatedly misbehaving anonymous users without relying on ttps. *ACM Transactions on Information and System Security (TISSEC) 13*, 4 (2010), 39.

[105] Tsang, P. P., Kapadia, A., Cornelius, C., and Smith, S. W. Nymble: Blocking misbehaving users in anonymizing networks. *Dependable and Secure Computing, IEEE Transactions on 8*, 2 (2011), 256–269.

[106] Tsang, P. P., and Wei, V. K. Short linkable ring signatures for e-voting, e-cash and attestation. In *Information Security Practice and Experience*. Springer, 2005, pp. 48–60.

[107] Tsujii, S., and Itoh, T. An ID-based cryptosystem based on the discrete logarithm problem. *Selected Areas in Communications, IEEE Journal on 7*, 4 (1989), 467–473.

[108] Viswanath, B., Mislove, A., Cha, M., and Gummadi, K. P. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks* (2009), ACM, pp. 37–42.

[109] Viswanath, B., Mondal, M., Clement, A., Druschel, P., Gummadi, K. P., Mislove, A., and Post, A. Exploring the design space of social network-based sybil defenses. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on* (2012), IEEE, pp. 1–8.

[110] Viswanath, B., Mondal, M., Gummadi, K. P., Mislove, A., and Post, A. Canal: Scaling social network-based sybil tolerance schemes. In *Proceedings of the 7th ACM european conference on Computer Systems* (2012), ACM, pp. 309–322.

[111] Viswanath, B., Post, A., Gummadi, K. P., and Mislove, A. An analysis of social network-based sybil defenses. *ACM SIGCOMM Computer Communication Review 41*, 4 (2011), 363–374.

[112] Watanabe, R., and Miyake, Y. Account management method with blind signature scheme. *Engineering and Technology, World of Science*, 59 (2011), 2069–2073.

[113] Whitten, A., and Tygar, J. D. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *Usenix Security* (1999), vol. 1999.

[114] Wolinsky, D., Corrigan-Gibbs, H., Ford, B., and Johnson, A. Scalable anonymous group communication in the anytrust model. In *EuroSec* (Apr. 2012).

[115] Wolinsky, D. I., Corrigan-Gibbs, H., Ford, B., and Johnson, A. Dissent in numbers: Making strong anonymity scale. *10th OSDI* (2012).

[116] Yee, K.-P., and Sitaker, K. Passpet: convenient password management and phishing protection. In *Proceedings of the second symposium on Usable privacy and security* (2006), ACM, pp. 32–43.

[117] Yu, H., Gibbons, P. B., Kaminsky, M., and Xiao, F. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 3–17.

[118] Yu, H., Kaminsky, M., Gibbons, P. B., and Flaxman, A. Sybilguard: defending against sybil attacks via social networks. In *ACM SIGCOMM Computer Communication Review* (2006), vol. 36, ACM, pp. 267–278.

[119] Yue, C., and Wang, H. Bogusbiter: A transparent protection against phishing attacks. *ACM Transactions on Internet Technology (TOIT) 10*, 2 (2010), 6.