

Abstract

# Identity Management through Privacy-Preserving Authentication

Ewa Syta

Yale University

2015

Maintaining privacy on the Internet is increasingly difficult in this ever-connected world. In most cases, our online interactions are a highly personalized experience and require some form of identity verification, most commonly, logging into an account. Unfortunately, people frequently give away a lot of information while obtaining accounts, reuse usernames and passwords across different services, or link their accounts to take advantage of single sign-on to avoid retyping passwords. This approach seriously blurs the line between different aspects of one's digital life, specifically personal and professional, as services dedicated for personal use (e.g., Facebook, bank accounts) and professional use (e.g. LinkedIn, corporate email account) become intertwined.

Identity management, the process of making decisions about online identities or accounts, is inherently linked to authentication, the process of creating and using online identities. However, the link between these two critical concepts is not always clear because of the lack of understanding of these terms as well as proper terminology to describe them. Identity management is further hindered by the lack of privacy-preserving authentication solutions that target specific applications and result in identities appropriate for those applications. Depending on the application, effective solutions to manage identities can be very diverse with unique or unexpected properties. In certain cases allowing users to hide their identity is as valuable

as providing unforgeable identities. Nonetheless, currently deployed authentication protocols do not reflect this approach.

In response, we make the following contributions. We carefully analyze the relationships between authentication, privacy and identity management and discover subtle yet important distinctions between the related concepts. As a result, we propose new terminology in order to clarify and draw distinctions between these critical concepts. We identify two distinct cases of authentication and propose privacy-preserving protocols to implement them. The protocols, PRIVATEEYES and DAGA, target different applications and produce identities that balance the requirements of their intended applications as well as their clients' privacy and security needs.

PRIVATEEYES is an efficient remote biometric identification protocol. It uses unique biometric characteristics in a privacy-preserving fashion for client verification, producing an identity that is suitable for applications requiring a high level of assurance of the client's real-world identity.

DAGA is a deniable anonymous authentication protocol. It offers four properties that give clients strong security and privacy protection, making it suitable for applications such as whistleblowing or access to sensitive resources. The properties are anonymity, proportionality, deniability, and forward anonymity. Anonymity and proportionality allow a client to authenticate as some group member without revealing exactly which one but only once per time period. Deniability makes it possible to deny ever participating in a protocol, while forward anonymity ensures protection even in case of a compromise of client's private key.

# Identity Management through Privacy-Preserving Authentication

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

by  
Ewa Syta

Dissertation Directors: Michael J. Fischer and Bryan A. Ford

December 2015

Copyright © 2015 by Ewa Syta  
All Rights Reserved

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contributions . . . . .	8
1.2	Analysis, Concepts and Terminology . . . . .	9
1.3	PRIVATEEYES: Secure Remote Biometric Identification . . . . .	10
1.4	DAGA: Deniable Anonymous Group Authentication . . . . .	11
<b>2</b>	<b>Identity Management and Authentication</b>	<b>13</b>
2.1	Overview of Authentication . . . . .	14
2.1.1	Types of Authentication Schemes . . . . .	15
2.1.2	Defining Authentication . . . . .	17
2.2	Authentication and Privacy . . . . .	19
2.3	Analysis, Concepts and Terminology . . . . .	23
<b>3</b>	<b>PrivateEyes: Secure Remote Biometric Identification</b>	<b>29</b>
3.1	Introduction . . . . .	29
3.2	Overview . . . . .	30
3.3	Biometric Identification . . . . .	33
3.3.1	Remote Biometric Identification . . . . .	34
3.3.2	Security and Privacy Issues . . . . .	35
3.4	Protocol Description . . . . .	37

3.4.1	Enrollment Phase . . . . .	38
3.4.2	Identification Phase . . . . .	39
3.4.3	Resynchronization . . . . .	41
3.5	Extensions . . . . .	42
3.5.1	Personas . . . . .	43
3.5.2	Digital Signatures . . . . .	44
3.5.3	Private Set Intersection . . . . .	44
3.6	Security Analysis . . . . .	45
3.6.1	Assumptions . . . . .	46
3.6.2	Security of Biometric Templates . . . . .	48
3.6.3	Impersonation . . . . .	49
3.6.4	Leakage of Information . . . . .	50
3.7	Practical Considerations . . . . .	51
3.7.1	Suitable Biometrics . . . . .	51
3.7.2	Enrollment Process and Tokens . . . . .	53
3.7.3	Template Generation . . . . .	54
3.7.4	Verification Decision . . . . .	55
3.8	Evaluation . . . . .	56
3.8.1	Implementation Details . . . . .	56
3.8.2	System Performance . . . . .	58
3.8.3	Feature Extraction Reliability . . . . .	59
3.8.4	Feature Extraction Timing . . . . .	60
<b>4</b>	<b>DAGA: Deniable Anonymous Group Authentication</b>	<b>62</b>
4.1	Introduction . . . . .	62
4.2	Overview . . . . .	66

4.2.1	Trust Model . . . . .	66
4.2.2	Protocol Overview . . . . .	67
4.2.3	Security Properties . . . . .	68
4.3	Protocol Description . . . . .	70
4.3.1	Notation . . . . .	70
4.3.2	Building Blocks . . . . .	71
4.3.3	Assumptions . . . . .	72
4.3.4	Authentication Context . . . . .	72
4.3.5	Client’s Protocol . . . . .	73
4.3.6	Servers’ Protocol . . . . .	75
4.3.7	Client’s Proof $PK^{client_i}$ . . . . .	77
4.3.8	Server’s Proof: Proving Correctness of its Work . . . . .	78
4.3.9	Server’s Proof: Exposing a misbehaving client . . . . .	79
4.4	Extensions . . . . .	80
4.4.1	Improving Efficiency . . . . .	81
4.4.2	Trading Deniability for Verifiability . . . . .	81
4.4.3	Optional Anonymity Revocation . . . . .	82
4.4.4	Secure on Full Key Exposure . . . . .	83
4.4.5	Delayed Revealing of Final Linkage Tags . . . . .	85
4.5	Applications . . . . .	87
4.5.1	Distributing Keys for Group Anonymity Systems . . . . .	87
4.5.2	Anonymous Voting with Deniability . . . . .	88
4.5.3	Secure Access to Sensitive Resources . . . . .	88
4.5.4	Server-provided Signatures . . . . .	89
4.5.5	Supporting Anonymous Federated Login . . . . .	89
4.6	Security Analysis . . . . .	90

4.6.1	Assumptions . . . . .	90
4.6.2	Properties of the Proofs of Knowledge . . . . .	90
4.6.3	Completeness . . . . .	97
4.6.4	Soundness . . . . .	99
4.6.5	Anonymity . . . . .	102
4.6.6	Forward anonymity. . . . .	105
4.6.7	Proportionality . . . . .	108
4.6.8	Deniability . . . . .	110
4.6.9	Forward Deniability . . . . .	113
4.7	Practical Considerations . . . . .	115
4.7.1	Servers' Liveness . . . . .	115
4.7.2	Dealing with Dishonest Servers . . . . .	115
4.7.3	Authentication Context . . . . .	116
4.7.4	Challenge Generation . . . . .	117
4.7.5	Per-Round Generators . . . . .	118
4.8	Evaluation . . . . .	119
4.8.1	Implementation . . . . .	119
4.8.2	Micro benchmarks . . . . .	119
<b>5</b>	<b>Related Work</b>	<b>122</b>
5.1	PRIVATEEYES . . . . .	122
5.2	DAGA . . . . .	125
<b>6</b>	<b>Conclusions</b>	<b>127</b>

# List of Figures

3.1	CPU time for enrollment in PRIVATEEYES . . . . .	58
3.2	CPU time for identification in PRIVATEEYES . . . . .	59
3.3	Difference scores using different template extraction libraries . . . . .	60
3.4	Time for feature extraction in PRIVATEEYES . . . . .	60
4.1	Conceptual model of DAGA . . . . .	68
4.2	Time and traffic evaluation results for DAGA . . . . .	120

# List of Tables

- 3.1 The relation between the difference score and odds of a false match [54] 52
- 5.1 A comparison of different biometric template protection schemes . . . 124

*To my loved ones*

# Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisors, Bryan A. Ford and Michael J. Fischer, for their guidance, support, patience, and encouragement. I thank Bryan for teaching me how diverse and fun research endeavors can be. I thank Mike for our endless discussions that have had a profound impact on my professional and personal lives. I sincerely thank both of them for making my PhD as enjoyable and as challenging as this process ought to be.

I would like to thank my internal committee members, Joan Feigenbaum and Avi Silberschatz, for their continuous support throughout my PhD. I also thank my external committee member, Vitaly Shmatikov, for his unique approach to life and research.

My PhD experience would have not been the same without David Wolinsky. As a collaborator, he was instrumental to all of my projects, including but definitely not limited to implementation and evaluation components. As a friend, he was always there to celebrate my success and share a cupcake when things seemed grim.

There are many other people to whom I owe a sincere thank you. Jerzy Gawinecki, for creating a stimulating learning environment at Military University of Technology that set me off on a path to an academic career. Janusz Szmidt, for introducing me to the world of cryptology. Neli Zlatareva, for being a wonderful role model. Bradley Kjell, for reminding me that there is more to life than just work. Neva Deutsch, for

believing in me and supporting my dream of teaching and research.

Finally, I would like to thank my friends and family. Celeste Ford, for bringing pure joy and happiness to my life. Malgorzata Misztal, for cheering me on during my PhD but most importantly, for all the unique experiences we have had during our studies at MUT and beyond. My parents, for instilling in me ambition and perseverance. And lastly, I sincerely thank my wonderful sister and husband for their love and endless support. I will never be able to repay them for everything they have done for me.

# Chapter 1

## Introduction

“Privacy is dead – get over it!” announced in 1999 Scott McNealy, the Chief Executive Officer of Sun Microsystems. Yet 16 years later users’ expectations of privacy, greatly fueled in the last year by the NSA’s mass surveillance activities revealed by Edward Snowden, are stronger than ever [113]. This phenomenon dubbed by the media as “the Snowden effect” [2, 68], has had a profound effect on people’s attitude towards privacy. Finally privacy is becoming a priority and people want to be proactive about it [2, 68]. A recent ESET-commissioned Harris Interactive study [48, 49] found that two thirds of people embrace individual responsibility for their privacy. This fact is supported by a finding from the same study that four out of five people have changed the privacy settings of their social media accounts and most have made changes in the last six months. This is a drastic change in user’s behavior. Previous studies found that only 16% of Internet users claimed to read privacy policies of the sites and online services with which they share their private information [135]. This new paradigm shift in online privacy shows that users want privacy, focus more on managing their online profiles, the information they share online and on the benefits and risks that come from it, and are willing to adjust their online behavior in hopes

of better privacy protection [73].

Maintaining privacy on the Internet is increasingly difficult in this ever-connected world. Today people are more connected to one another than ever before due to the explosion in the usage of digital devices coupled with the widespread accessibility to the Internet: 2.7 billion people – almost 40% of the world’s population and 77% of the developed countries’ population – are online [186]. Access to a computer and a high-speed Internet connection have become a commodity people have grown to rely on. The amount of global digital information created and shared online have surpassed 2 zetabytes<sup>1</sup> [75] and the value of online-accessible resources continuously increases.

Consequently, people use the Internet, often on a daily basis, to perform a variety of tasks, including tasks that used to be reserved for in-person interactions [90]. Social activities such as e-mail, real-time video and audio communications, various social media as well as entertainment such as gaming, movies, and music have been driving forces behind today’s Internet frenzy [11, 112]. However, the Internet is no longer a medium reserved for entertainment and social activities only; people use it to perform tasks encompassing all aspects of life. Many people have accepted the Internet as means to access more sensitive resources such as financial, medical or otherwise private records. We routinely make financial transactions, access our health records and communicate with health care providers, file taxes and interact with government agencies, or even obtain degrees through online educational programs [45, 93].

In most cases, our online interactions are a highly personalized experience [98]. We expect to access *our* own email or bank account, interact with *our* friends on social media or even see personalized content on websites or obtain e-commerce recommendations. In order to deliver these services, providers need (and want)

---

<sup>1</sup>1 zetabyte = 1 trillion gigabytes

to know who is accessing their resources. We are almost constantly prompted by websites we visit to use or create a new account, or sign-in with a third party account in order to see the content, or unlock all features and benefits a particular website provides. As a result, almost every transaction a client performs online results in some form of identity verification. Service providers perform identity verification for reasons such as ensuring legitimate access to protected resources (e.g., email or bank account, subscription services), enhancing client experience by providing personalized content (e.g., news portals, recommendation system) or even trying to combat spam or inhibit inappropriate behavior (e.g., discussion forums).

As a consequence, clients are expected to create and maintain multiple accounts, often one per each service provider they use. Unfortunately, out of convenience and without giving it proper consideration, people frequently give away a lot of information while obtaining those accounts, reuse usernames and e-mail addresses across different online services, or link their accounts to take advantage of single sign-on to avoid retyping passwords. This approach, however, seriously blurs the line between different aspects of one's digital life, specifically personal and professional, as services dedicated for personal use (e.g., Facebook, bank accounts) and professional use (e.g. LinkedIn, corporate email account) become intertwined. This might have serious consequences when personal information inevitably surfaces in a professional context. A barista from Washington lost his job when his boss found his blog containing snarky comments about the coffee shop and customers, and a teacher from Florida was asked to resign when her principal found her modeling photos published under a different name [29]. Moreover, hiring managers routinely use social media to screen job candidates and make hiring decisions [170] making it almost impossible to keep personal information private once shared online.

Therefore, people are charged with a daunting task — a task of maintaining their

online accounts in the context of their personal (or private) and professional (or public) lives. With the growing awareness of the relationship between privacy and digital footprint, many people wish to keep their personal and professional online activities separate and actively try to ensure that the corresponding online profiles are kept separate [171, 173]. This approach mirrors our behavior in the real world. We share certain information with our family members while keeping it away from professional contacts. Conversely, we tend to or are even obligated to keep work-related information within a certain circle of individuals.

There are some ways to limit the exposure of personal information and linkability of accounts meant for different purposes. They range from relatively easy options such as avoiding single sign-on, limiting the use of professional email address in personal contexts, to more restrictive options such as abstaining from social networks, switching from credit cards and other identifiable forms of payment to gift cards or anonymous currencies for all online purchases, and finally using secure browsing and anonymous communication technologies for all Internet activities. Interestingly, the recent story of Janet Vertesi, an associate professor of sociology at Princeton who went to the extreme to hide her pregnancy from big data, shows that the use of privacy-preserving technologies make people more likely to stand out and even be tagged as potentially engaged in criminal activity [147].

These examples offer only a small glimpse of how little privacy we can expect on the Internet, and how intertwined our personal and professional information have become. Unfortunately, identity management, the process of making decisions about online accounts or identities, is not an easy task. People are often ill-prepared to make those decisions. This stems from the lack of understanding of these complex notions and a clear terminology to describe them as well as the lack of control over mechanisms employed to create, verify and use online identities.

Service providers struggle to provide clients with adequate tools to manage identities. In an absence of the before-mentioned understanding and terminology, service providers may use subjective criteria, such as their own security and usability goals, to decide on specific authentication solutions, without taking into account the privacy needs of their clients.

Certainly, from the clients' point of view, using online identities should be convenient, secure and privacy-enhancing. Especially important to the service providers is to have a high level of assurance that clients are indeed who they claim to be. However, privacy aspects of online identities are not clearly defined. Intuitively, personal information clients disclose in order to create an online identity must be protected and preferably limited, and transactions clients perform with multiple service providers should remain separate so clients' actions cannot be tracked and later on linked together revealing personal information and possibly compromising their privacy.

Giving people control over their identities and how their identities are used is essential to privacy protection and effective identity management. User should have the ability to establish independent identities for specific uses, personal, professional or other, and maintain them in a way that preserves independence and specific properties of each identity.

Depending on the application, effective solutions to manage identities can be very diverse with unique or unexpected properties. In certain cases allowing users to hide their identity is as valuable as providing unforgeable identities. In applications such as electronic voting, an identity needs to be reliably verified whereas in applications such as anonymous data collection, an identity needs to be concealed, but at the same time it might be desirable to enforce certain rules, such as "one user, one use" or "one user, one account". In the case of anonymous communication, the identity

might be known but its link to certain actions performed using this identity must be hidden. More interestingly, accountable anonymity systems, in which misbehaving members are held responsible for their actions, often rely on the ability to uniquely bind a user to his identity, so that user's misbehavior can to be effectively punished.

Identity management is inherently linked to authentication — the process of creating and using online identities. The notion of *identity* is critical to both concepts. Identity management is concerned with making decisions regarding identities as to achieve the goals set out by a client while meeting the needs of service providers' who rely on those identities to control access to their resources. On the other hand, the specific properties of an authentication protocol define and carry over to the resulting identity. Therefore, identity management can accomplished through appropriate authentication mechanisms.

However, the link between these two critical concepts is not always clear because of the lack of the before-mentioned terminology to describe them. A client can only manage (reason and make decisions) about his identities, if he understands the life cycle of each identity, from its conception, through usage, and finally removal in different applications. A service provider needs the same understanding to decide on the most suitable solution.

Moreover, authentication is not a monolithic or one-dimensional concept; although it has been often treated as such. In most commonly employed authentication protocols, a client has to declare his identity and only once that identity is sufficiently verified is he allowed to obtain access under that identity. This approach allows each action the client performs to be linked back to that identity, which is appropriate for *some* applications but not for all. For instance, a health care provider needs to grant access to medical information only to authorized clients — e.g., patients, insurance companies, physicians — but also to keep a detailed record of who

accessed particular information and when. On the other hand, a subscription service — e.g., Netflix or Hulu <sup>2</sup> — does not necessarily need to know *which* specific client is requesting access, so long as it is a paying client, in order to fulfill its primary role of providing content.

An important observation is the fact that an identity is never static. It is not limited to the information revealed during the enrollment process, the process of establishing the identity. Each time an identity is used — a client logs into his account — the service provider learns more about the client and therefore the associated identity evolves. In the Netflix example, an identity is no longer “John Smith” but “John Smith” who watched certain movies, wrote certain reviews, and always logs in at a certain time. The client’s actions, in fact, become new attributes associated with his identity and in some cases might be enough to uniquely identify the client in a completely different context.

The idea that client’s actions indeed are a form of identifying information was demonstrated in 2007 by a group of researchers who were able to de-anonymize a set of clients based on their actions [140,162]. As part of a challenge for a better recommendation system, Netflix published 10 million movie rankings by 500,000 customers, whose data was anonymized by replacing personal details with random identifiers. Nonetheless, the clients were de-anonymized by comparing rankings and timestamps with public information in the Internet Movie Database (IMDb)<sup>3</sup>.

The above examples clearly show that we need to empower clients to manage their identities and also to support service providers in responding to their clients’ privacy needs. We no longer can take the “one-size-fits-all” approach to authentication if we hope for an effective identity management solution. We need proper yet simple ter-

---

<sup>2</sup>Hulu ([www.hulu.com](http://www.hulu.com)) and Netflix ([www.netflix.com](http://www.netflix.com)) are providers of on-demand Internet streaming media.

<sup>3</sup>[www.imdb.com](http://www.imdb.com)

minology for authentication that is general enough to reflect the diverse requirements of online identities. We also need specific authentication solutions that reflect the needs of diverse applications. Specifically, we need solutions for unforgeable identities but also less demanding solutions in which clients might dissociate their unique identity from their actions while still allowing service providers to adequately control access to their resources.

## 1.1 Contributions

The goal of this thesis is explore the idea of effective identity management through appropriate authentication mechanisms. This thesis makes the following three major contributions.

1. Provides a careful analysis of the relationships between authentication, privacy and identity management.
2. Proposes new terminology that draws distinctions between concepts related to authentication.
3. Proposes privacy-preserving protocols for two distinct cases of authentication.
  - PRIVATEEYES, a secure biometric identification protocol for privacy-preserving identification.
  - DAGA, a deniable anonymous authentication protocol for privacy-preserving authentication.

## 1.2 Analysis, Concepts and Terminology

Building upon the findings of our analysis of concepts related to authentication, privacy, and identity management, we propose and clarify terminology for concepts related to authentication. Specifically, we propose a new definition of authentication that is general enough to support the concept of *group* or *anonymous* authentication. Group authentication [36, 42, 155], the process of authenticating as one of many eligible clients without revealing exactly which one, has been receiving more attention as an alternative to identity verification. It offers clear privacy benefits as client's actions are no longer linked to her but rather to a well-defined group of people. Our definition clarifies this easily misunderstood concept.

We define an *identity* with respect to a certain set of attributes that belong to a particular *entity*. We then differentiate between the terms *identity* and *group identity*, depending on the size of the set of entities associated with it. The distinctions we draw between these terms allows us to identify two cases of authentication and in turn, we are able to define the term *identification*.

We separate privacy issues related to the two distinct phases of authentication: the creation of identities (the enrollment process) and the use of identities (the authentication process). We point out that while an identity is created during enrollment, it is continuously expanded during authentication by adding new attributes which result from actions performed under that identity. This observation is important to identity management. The newly-acquired attributes might define a new identity of the same entity, which might become linkable to other identities of that entity across different and often unrelated systems, which is detrimental to clients' privacy.

### 1.3 PrivateEyes: Secure Remote Biometric Identification

Biometrics has been long recognized as an excellent building block for identification protocols. Biometrics offer multiple benefits including non-repudiation and ease of use, and biometric data cannot be lost or forgotten and is constantly available. However, using biometric data raises privacy issues. Firstly, a biometric template carries a considerable amount of personal information, which often includes race, gender and certain medical conditions. Secondly, a biometric template can be used to identify an individual and successfully track and link her activities performed using the same biometric identity.

We propose PRIVATEEYES, a secure remote biometric identification protocol that gives strong protection to the user’s biometric data in case of two common kinds of security breaches: a full client compromise or a full server compromise.

The novelty of our approach lies in the way we handle biometric templates. The templates are never directly stored, transmitted during the protocol or made available to the verifier. A client uses a token, possibly with a built-in sensor (e.g., a smart card or a mobile device), to store a securely blinded biometric template. The secured template changes with each attempt, rendering the information stored on the token useless if stolen. A client is successfully identified if the verifier confirms that the difference between the blinded template and a fresh template as computed by the token is sufficiently close to 0. Because a client creates his identity using his biometric template but with respect to a special blinding factor shared with the verifier, many independent identities can be securely created using the same biometric data.

Our two-factor protocol can be combined with a broad class of existing biometric schemes to protect the privacy of the user’s biometric data and the template derived

from it. Our approach offers benefits such as protection of biometric data, revocability of templates, and privacy-protection with respect to users' biometric identities as well as actions performed using those identities. Furthermore, our protocol adds negligible overhead and maintains the recognition performance of the underlying biometric recognition algorithm.

PRIVATEEYES demonstrates an effective identification protocol that results in strong identities yet offers privacy-preservation in the context of biometrics without a significant performance penalty.

## 1.4 DAGA: Deniable Anonymous Group Authentication

Authentication is used to ensure that only legitimate users are granted access to protected resources or services. However, while it is important to limit access only to authorized users, it may not always be needed to verify their identity and instead it may be sufficient to allow users to use a well-defined group identity. This unique property is desirable in many systems such as systems providing access to information considered sensitive, online subscriptions, discussion forums, anonymous data collection, and many other application that necessarily require a client to be identified as long as certain properties of the client can be ensured in order to grant access.

We propose DAGA, a deniable anonymous group authentication (DAGA) protocol, which illustrates a new approach to privacy-preserving authentication. To the best of our knowledge, DAGA is the first protocol to provide these four security properties: anonymity, proportionality, deniability, and forward anonymity. The anonymity and proportionality properties allow a client to authenticate as some

group member (using a group identity) without revealing exactly which one but only once per time period or even at all. Deniability makes it possible to deny ever participating in a protocol, while forward anonymity is a stronger property that offers protection of user's identity and the ability to deny participation even in case of a compromise of user's private key.

We have built a working proof-of-concept implementation of DAGA to validate its performance and practical usability. A proof-of-concept prototype validates DAGA's practicality, authenticating a client into a 32-member group in one second, or into a 2048-member group in two minutes. Our evaluation suggests that DAGA compares reasonably well to anonymous and non-anonymous authentication given the security and privacy gains.

DAGA demonstrates an effective authentication protocol built on the concept of group authentication that offers clients strong security and privacy protection while preserving the ability to control access to resources, a feature important to service providers.

## Organization

This thesis is organized as follows. Chapter 2 overviews the process of authentication, and provides an analysis of concepts related to authentication, its finding and the new terminology. Chapter 3 provides a description, analysis, discussion, and evaluation of PRIVATEEYES while Chapter 4 does so for DAGA. Chapter 5 overviews related work. Chapter 6 concludes.

## Chapter 2

# Identity Management and Authentication

Identity management is a process of controlling *who* has access to online identities of a particular client or entity, *when*, and *why*. The goal of this process is to achieve the privacy goals established by a client while meeting the needs of a service provider, who relies on these identities to control access to its own resources. The process of authentication is fundamental to managing identities. In its most common form, authentication allows verifying one's identity in order to confirm if a user is who he claims to be, after such an identity is established. Hence, authentication encompasses two main aspects of identities: their creation and usage.

Depending on the properties of a particular authentication protocol, the client may or may not be able to manage her identities. This is because the properties of the authentication protocol carry over to the identities it establishes. On the one hand, if in order to authenticate, a client needs to reveal a significant amount of personal information and this information is not afforded proper protection, the client's options are extremely limited, beyond refusing to participate. On the other

hand, if authentication is accomplished through a proper, transparent process whose goal is to minimize the exposure of client's information, then the client can weigh the risks and benefits of using such an identity and make an informed decision.

Authentication, however, is a process between two parties whose goals might be competing. A client wishes to receive access to some resources while *minimizing* disclosure of her information. A service provider wishes to provide access to his resources while *maximizing* his confidence that only authorized clients will be granted access. Intuitively, the more information the client reveals, the higher the provider's confidence but the greater effect on the client's privacy. However, these seemingly contradictory needs may be addressed through appropriate privacy-preserving authentication mechanisms that target specific applications. Some applications call for highly reliable and verified identities while other can accommodate even strong protection to clients and their actions. All too often, however, an authentication solution is chosen as a "one-size-fits-all" approach without considering the specific needs of a particular application and balancing it against clients' privacy needs.

In this chapter, we explore different aspects of authentication and identity management, carefully analyze the relationships of the related concepts, and finally propose new terminology as a result of distinctions of these concepts we are able to provide.

## 2.1 Overview of Authentication

Informally, the process of verifying a client's identity is called *authentication*. This process consists of two independent steps: the *enrollment* process and the *authentication* process [188].

The goal of authentication is to link the originator of a transaction to an au-

thorized entity for that transaction. The linkage between the originating client and authorized entity is established using one or more authentication factors — pieces of evidence a client needs to present — depending on the type of the authentication process employed.

The authorized entity and the corresponding authentication factor (or factors) are established during the *enrollment* process. The enrollment process precedes any authentication attempts by a client. The process is performed between a client and a server and results in a new account (identity) created for the client. During the *authentication* process, a client makes a claim of identity he wishes to authenticate as and provides the corresponding authentication factor. The server verifies this claim through a protocol, which normally in some fashion compares the authentication factor presented during authentication to the one established during enrollment. If the factors match or are sufficiently similar, the server accepts the client's claim and the requested transaction is performed. Otherwise, the server rejects the client's claim, which concludes this process.

In describing the details of authentication, we will interchangeably use the terms *client*, *user*, *person* and *prover* to indicate an entity that requests access to some resources and whose identity is verified during the authentication process. We use the terms *server*, *service*, *service provider*, and *verifier* to indicate the other party to the authentication protocol that performs the verification of a claim made by a client.

### **2.1.1 Types of Authentication Schemes**

A number of authentication solutions have been proposed. They can be categorized as knowledge-based, possession-based, or biometrics, depending on the kind of factor they use to verify the claim of identity [19, 172, 174].

Knowledge-based authentication (“something you know”) is the most popular kind of authentication based on a shared secret (e.g., a password, PIN or a passphrase) that a client needs to provide in order to prove his claim. Passwords are convenient to use and virtually free to deploy in practice. However, the many drawbacks of passwords are well known [100, 110, 146]. Weak passwords are easily guessed; strong passwords are difficult for clients to remember and to supply when required [74]. In addition, username and password data must be somehow encrypted when sent over the network since once intercepted, they are sufficient to impersonate the legitimate client. Passwords are also often directly stored by a verifier for the purpose of comparison during authentication. Even if the chosen password is of a proper strength, this approach still enables an attacker to impersonate the client and gain an unauthorized access to multiple services as clients frequently reuse their passwords [50, 153, 165]. It has been long known that weak passwords are one of the leading causes for system break-ins [44].

Possession-based authentication (“something you have”) requires a client to demonstrate possession of a certain device (e.g., hardware token, smart card, or a mobile device) [92, 158, 180] in order to authenticate. This method is convenient as it relieves clients from remembering passwords and allows for random passwords or cryptographic techniques to be easily used. However, it imposes certain costs on clients to obtain and maintain devices. This method fails if the device is lost or stolen. Clients can then take action to revoke the compromised token to limit possible damage since, unlike passwords, it not likely to go unnoticed.

Biometric-based authentication (“something you are”) uses biometric characteristics (e.g., a fingerprint, voiceprint, or hand geometry) for identity verification [17, 56, 103, 105]. Biometrics offer multiple benefits including non-repudiation and ease of use. Biometric data cannot be lost or forgotten and is constantly available. Certain

biometric characteristics never change and can be measured quickly and unobtrusively. On the one hand, biometric data is unique to a person and therefore is an excellent way to define client's identity. On the other hand, the unchangability of a biometric datum poses a serious risk if it is ever compromised.

All three categories of authentication methods have their own advantages and disadvantages, and all have been extensively employed in different authentication systems [88]. Passwords have been the method of choice for client authentication for many years [122]. Even though they do not provide sufficient security guarantees, passwords are used to grant access even to sensitive services such as online banking or medical records. The following two examples abundantly illustrate the implications of using password-based authentication. The compromise of LinkedIn resulted in 6.5 million passwords were leaked online [50, 164] while the outage of the Sony Playstation Network put at risk personally identifiable information from over 77 million accounts [154].

Recently, in an attempt to improve this bleak situation, companies (e.g., Twitter, Google, Microsoft [87, 116]) offer and in some cases mandate *two-factor* or even *multi-factor* authentication [30], which requires a client to present two or more authentication factors. This, however, does not represent a fundamental change in how authentication is used to provide online identities [161]. Instead, the goal is to strengthen the current process in response to the vulnerabilities in single factor authentication. Fortunately, this new trend shows that companies understand the need for better authentication and are willing to deploy them in practice.

### **2.1.2 Defining Authentication**

In our daily online interactions, we are requested to prove our identity multiple times. *Entity authentication* is a basic primitive that is employed to accomplish this task

and is often used as a building block to provide security of larger and more complex access control systems.

Below we present several such definitions proposed by others.

- *Authentication* is the process of establishing confidence in the identity of users or information systems. *Authentication protocol* is a well specified message exchange process that verifies possession of a token to remotely authenticate a claimant. [30]
- *Authentication* is the process of determining whether someone or something is, in fact, who or what it is declared to be. [143]
- *Authentication* is the process by which one entity (the verifier) is assured of the identity of a second entity (the claimant) that is participating in a protocol (Identity Verification Protocol). [187]
- *Authentication* is the process whereby one party is assured (through the acquisition of corroborative evidence) of the identity of the second party involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired). [132]
- *Authentication* is a communication process (i.e., a protocol) by which a principal establishes a *lively correspondence* with a second principal whose claimed identity should meet what is sought by the first. [130]
- The goal of an *authentication scheme* is to allow someone's identity to be confirmed. [174]

As shown above, there is no uniform and widely accepted definition of authentication. Moreover, the existing definitions are often either overly vague or overly

specific to a particular scenario where authentication might be employed. Some definitions focus on select *goals* of authentication while others focus on select *ways* to accomplish authentication.

Furthermore, authentication is sometimes mistakenly referred to as *identification* [183]. Many definitions consider a case where authentication can be reduced to identification, that is, the authenticating client must make a claim of and prove his real-world identity. This approach has clear privacy implications: if a client must always be *identified* before obtaining access to some resources, then his actions are always attributed to him, preventing any form of unlinkability or separation of actions and identities, basic building blocks for privacy-preserving authentication solutions.

Consequently, the current understanding of authentication is clearly not sufficient and requires careful treatment in order to design authentication solutions that fit specific needs of clients and service providers.

## 2.2 Authentication and Privacy

The process of authentication relates to two main concepts, *establishing* and *using* identities in the online space, both of which give rise to different privacy issues.

When a client creates a new identity (often referred to an account), she has to supply certain information about herself. The type of information requested to establish an identity depends on the application; however, increasingly often people are asked to provide information regarding their *real* (or offline) identity in order to establish a new online identity. In addition to the usual email address and password, this information may include phone number, physical address, credit card number, date of birth, etc. While for many applications this kind of information is relevant and often necessary, people are still expected to provide it even for applications that

do not require the new identity to be linked to an actual human being. Rather, the information is unnecessarily collected and often used to create a “barrier to entry” in order to solve problems orthogonal to authentication and authorization, such as spam or misbehavior [80].

A client establishes a new identity during the enrollment process, when certain information associated with a client is transferred to the newly created identity. Some, but not all, of this information is later used to verify the client during her attempts to gain access to a system or resource. For instance, a service that requires a name, username, physical address, phone number, email address and password during enrollment will typically only use the email and password for verification. This, however, does not mean that there is a separation between the online and offline identities of the client, based on the limited information used for verification of the identity. On the contrary, each action a client performed using an authenticated online identity is linked to and can be attributed to the offline identity.

An *identity* is not a static concept. Rather, an identity is an ever-evolving set of information about the client the identity belongs to. A newly-established identity consists of the information revealed during the enrollment process. However, each time a client uses such an identity, the service provider learns additional information and often associates it with the identity, such as how often then client uses a particular service, client’s specific actions or preferences.

We identify two main sources of information associated with an identity: the enrollment process and the subsequent authentication process.

1. *Enrollment Privacy*. Information revealed during the enrollment process. This refers to the information a client provides in order to establish a new identity and information the provider gathers during this process.

2. *Authentication Privacy*. Information revealed during the authentication process. This refers to information acquired through the use of an identity. More specifically, *explicitly revealed information* refers to new information knowingly provided by the client. On the other hand, *implicitly revealed information* is gathered from actions performed by a client under a specific identity, and may or may not be known to the client.

Service providers choose to authenticate users for many reasons and have different expectations about the outcome. Each application has its specific requirements as to what the authentication process is supposed to accomplish, beyond the high-level goal of “providing access to authorized users”. Specifically, different applications will differently define “authorized users” depending on the level of verification or “realness” (a correspondence to an actual human being) needed.

In some applications, checking the “realness” of an identity and then tracking the activities performed by that identity is not only desirable but often required to adequately provide the specific services. This includes applications providing access to private information, such as a bank account or medical history. In this case, both, the service provider and the client, wish to limit access only to specific individuals and to know exactly who obtains access, what actions are performed, and when. This class of applications requires a client to establish and use an online identity which corresponds to a unique offline identity.

In other applications, service providers can still perform their primary duty without obtaining a lot of information about their clients. This is especially applicable to cases where a client may need to establish an identity related to an offline identity during enrollment but confirming the linkage is not necessary when the client authenticates and uses the online identity. For example, subscription services such as Netflix or Hulu want to provide access only to their subscribers (paying customers)

and perhaps track individual viewer's preferences. However, this does not necessarily imply that a client must always identify himself in order to obtain the service. In this case, identification is a sufficient mechanism to accomplish the service provider's goal but not always necessary. In fact, what the service provider needs is to know that a client requesting access to the service is a paying customer *and* perhaps a viewer with browsing history labeled 1234. While the service providers does not learn John's or Jane's individual preferences, he can tell them apart as a paying customer 1's preferences and paying customer's 2 preferences. This approach protects clients by allowing them to keep their actions private without adversely affecting service providers.

In yet another class of applications, it is not only undesirable but also not necessary to link the established identity to each specific action performed under an authenticated identity. Many applications use authentication to tackle problems orthogonal to but easily addressed using authentication. For example, discussion forums struggle with Sybil attacks [63,175] (people obtaining multiple accounts), spam (people posting irrelevant messages), and misbehavior (people posting inappropriate content). As a counter measure, often referred to as "real name policy", many service providers require clients to create an identity and insist that the identity must be reflective of an actual human being. However, in most cases a service provider does not need to know specific offline identities of clients wishing to participate but rather that the online identities created within the system will "behave" in a specific way or possess specific characteristics (e.g., will obtain only one account, does not have history of spamming message boards, has good reputation for posting good or appropriate content).

Users who wish to lessen the effects on their privacy of creating and using online identities, either minimize information provided during enrollment or even provide

false information [118, 184]. However, this approach severely impacts the value of such an identity from the provider’s point of view as it reduces the provider’s ability to rely on it. Still, the benefits to the clients are not clear since provider’s still have some abilities to track the client’s activity. Hence, neither the client nor service provider receives full utility from authenticating online identities.

Perhaps a middle ground is needed, where authentication does not always result in verifying an offline identity but in some cases only ensures that the client requesting access possesses characteristics of an authorized user as defined by a service provider. This approach, if implemented properly, would allow the provider’s needs to be met while reducing information collected from clients.

## 2.3 Analysis, Concepts and Terminology

In this section, we provide analysis of relationships and definitions of terms related to authentication and therefore identity management. Specifically, we define the following terms: *attribute*, *group*, *group identity*, *identity*, *enrollment*, *authentication*, *identification*, and *identity management*. Our goal is to provide clear and meaningful definitions that are useful to clients for identity management and may serve as a foundation for formal definitions needed for rigorous security analysis.

The main observation from the previous section is the fact that authentication need not deal with verifying specific identities of clients or *identification*, but rather verifying specific, desirable properties of clients.

The terms authentication and identification are often mistakenly used as synonyms. Identification is also sometimes used to describe the process of a client stating his identity, a step that precedes authentication defined as verifying this identity. Intuitively, however, identification must mean more than merely declaring

an identity and perhaps something more than what authentication is defined to be.

In order to tackle this issue, we start by defining basic terms. First, we will refer to clients as *entities*. We treat an entity as an atomic term following an informal definition that it is “something that exists by itself”. An entity might refer to subjects (e.g., people, organizations), objects (e.g., computers, servers), or more abstractly any resources. There should not be presumption of animation, although entities will mostly refer to humans.

An entity may be described using one or more attributes. An *attribute* is a property or a characteristic of an entity. Attributes may refer to the physical characteristics of an entity but also any other (even intangible) characteristic such as a possession of a certain password or a set of preferences. Some attributes may be easily changeable (e.g., an address, affiliation) while other may be difficult (e.g., a name, driver’s license number) or impossible (e.g., DNA) to change.

**Definition 1.** *An attribute is a property of an entity.*

An entity is described by one or more attributes. Because an entity can have multiple attributes, those attributes may be grouped together to describe different aspects of an entity. Entities may be grouped together too. A *universe* is a group of entities and represents a specific application. For this reason, there might be many universes and each entity might belong to more than one universe.

**Definition 2.** *A group is a subset of a universe.*

Some sets of attributes will uniquely define an entity within some universe while others will not. However, each set of attributes specifies a possibly empty *group* within a universe, namely the set of entities that have those attributes in common.

**Definition 3.** *A group identity is a set of attributes.*

A group identity specifies the group consisting of exactly those entities satisfying all attributes the in group identity. Some sets of attributes will specify a singleton, a group with exactly one element. For this reason, this special set of attributes can be viewed as a unique identifier or *identity* for that entity as it allows to distinguish that entity from every other entity in the universe. We say that an entity is uniquely specified by that identity.

**Definition 4.** *An identity is a group identity that specifies a group with exactly one entity.*

While an entity is unique to an identity, each entity can have more than one identity, depending on a specific set of attributes used to describe it. While it is easy to mistakenly believe that we each have only one identity, in fact, we can be distinguished from others using different sets of attributes, depending on the set of entities we are being compared to. For instance, a person can be identified by {name, driver's license number} on a state level (all other people from the same state) but by {name, passport number} or {name, driver's license number, state} on a federal level (all other people from the entire country).

Within a universe, each entity is associated with a certain set of attributes. The process of associating these attributes with an entity is performed during the enrollment process. The specifics of this process depend on the type of attribute and the application.

**Definition 5.** *An enrollment process is a process of establishing a set of attributes of an entity run between a verifier and that entity.*

In this context, we can view authentication as a process that deals with attributes. In fact, this process *verifies* that some entity or entities indeed possess a certain attribute or a set of attributes. Above, we already distinguished between different

sets of attributes: those that uniquely define an entity (identities) and those that define groups of entities (group identities). Intuitively, the process of proving an identity carries a different meaning than a process of proving a group identity.

If an entity wants to prove a specific identity, it must prove it possesses all attributes of that identity. There are some attributes we often view as associated with identities more than others. That is, attributes that are more likely to uniquely identify an entity. Name or social security number are examples of these attributes. We may call this special type of attribute an *identifying attribute* or *identifier*, which can serve as a label for some identity but may or may not be sufficient to define it.

In general, authentication is not limited to verifying attributes of an identity. Rather, authentication verifies *some* attributes of an entity, which may or may not define an identity. Therefore, we can view authentication as the process of verifying one or more attributes of an entity.

If the set of attributes does specify an identity, then we call the process of verifying these attributes *identification*. If these attributes do not specify an identity, it means that there may be more than one entity that possesses these attributes. Because these entities share these attributes, they also are associated with the same group identity. Therefore, either a set of attributes points to a single entity, in which case this set forms an identity and the process of verifying these attributes results in identification, or a set of attributes points to a group of entities and the process of verifying these attributes results in authentication. This gives us the following definition.

**Definition 6.** Authentication *is a process, run between an entity (a prover) and a verifier, of verifying membership in a group.*

In practice, a prover is a person. However, we do not distinguish between persons and devices as mentioned before. The process of verifying membership in a

group refers to any process that allows a verifier to confirm that an entity indeed possesses attributes that specify a group identity. Depending on the application, this process may even be implemented using the standard approaches to authentication, as described in Section 2.1.1.

Hence, a successful verification process puts the verifier into a knowledge state, in which it is true that a client is a member of a group or the process fails. This gives the verifier the ability to distinguish every entity that belongs to a group from any entity *not* in that group. If the process fails, the verifier does not know if a particular entity belongs to a group or not because of insufficient evidence presented to the contrary.

Since identification, unlike authentication, is strictly concerned with verifying identities (sets of attributes that form them), it becomes a special case of authentication. Specifically, authenticating into a group of entities of size 1 results in *identification* of a particular entity through claiming a particular identity and proving the possession of attributes that specify it.

**Definition 7.** Identification *is a process, run between an entity (a prover) and a verifier, of verifying membership in a group that has exactly one element.*

Given the above definitions, authentication refers to claiming a group identity while identification refers to claiming an identity. This approach offers immediate privacy and identity management benefits but also allows us to provide a more concrete definition of identity management.

**Definition 8.** Identity management *is process performed by an entity of managing attributes defining identities and group identities of that entity across different universes.*

Privacy issues arise when a service provider uses identification as means to ac-

compish authentication. Often, authentication as outlined above — ensuring properties of authorized clients without uniquely identifying them — sufficiently meets the needs of the provider. On the other hand, there are many applications where identification is needed. It is critical for effective identity management to distinguish between these two cases and apply authentication and identification appropriately.

Identification prevents unlinkability of actions performed under some identity and the corresponding entity since identification creates a link (*action — identity — entity*). If a client’s authentication results in her identification, then each action she performs is linked directly to her. Similarly, authentication provides unlinkability of actions and a specific entity because authentication results in a link (*action — group identity — group of entities*).

If a client authenticates and therefore *assumes* a group identity, then the actions of a particular client are linked to the group identity since the client is indistinguishable from other clients that use the same group identity. Moreover, authentication still ensures authorized access but defines authorized clients in terms of specific attributes they need to possess as opposed to specific identities.

The benefits for identity management are clear. A client can consider an authentication scheme with respect to the privacy protection it provides during the enrollment (which attributes are required) and the authentication phases (the ability to use identities and group identities).

In the next two chapters, we provide two protocols for authentication. One that implements identification (PRIVATEEYES, Chapter 3), a special case of authentication, and one that implements authentication (DAGA, Chapter 4). Both protocols, however, implement their respective goals in a privacy-preserving manner.

# Chapter 3

## PrivateEyes: Secure Remote Biometric Identification<sup>1</sup>

This chapter presents PRIVATEEYES, a secure remote identification protocol.

### 3.1 Introduction

A major problem facing the Internet is “the reliance on passwords to authenticate users” [70]. The drawbacks of passwords have long been known [110]. Weak passwords are easily guessed; strong passwords are difficult for users to remember and to supply when required. In addition, username and password data must be somehow protected when sent over the network and stored on a server since once compromised, they are sufficient to impersonate the legitimate user.

Combining biometrics with cryptographic authentication schemes is an attractive alternative to password authentication [71], an approach supported by the FIDO (Fast IDentity Online) Alliance, a non-profit organization formed to promote easier

---

<sup>1</sup>Portions of the research reported in this chapter were done in collaboration with Michael J Fischer, David I Wolinsky, Abraham Silberschatz, Gina Gallegos-Garcia, and Bryan Ford. Preliminary versions of this work have appeared in [178] and [179].

to use and stronger authentication. FIDO works closely with dozens of prominent industry partners (e.g., Bank of America, Google, Visa, RSA) and strives to reflect the current needs and expectations of the authentication process, from clients and services alike. In their approach, a secret key, stored on the user’s local device, is used with a challenge-response protocol to authenticate securely to the server. Biometrics are used to prevent the local device from being activated by any but the legitimate user. However, the user’s device stores secret information, which becomes problematic when the device is compromised.

Especially sensitive in any biometric scheme is the user’s biometric data which, if compromised, can subsequently be used by an attacker to impersonate the individual to whom it belongs. Unlike passwords, biometric data cannot be changed, so once compromised, it becomes useless as an authentication factor.

Many techniques exist for extracting data stored on a device’s internal memory, even from so-called “tamper-resistant” devices [8, 9]. One must assume that an attacker who steals or otherwise gains physical possession of the user’s device also obtains the entire contents of the device’s internal memory, including any secret keys and biometric data stored therein. Therefore, the user’s biometric data must not be stored on the user’s device in any form that would allow an attacker to reconstruct it. Similar considerations apply to the server, which also must protect the user’s biometric data even in the face of a total compromise.

## 3.2 Overview

We propose an efficient remote biometric identification<sup>2</sup> protocol that gives strong protection to the user’s biometric data in case of two common kinds of security

---

<sup>2</sup> Following Definition 7, we use the more precise term *identification* instead of commonly (mis-)used term *authentication*. See Chapter 2 for a detailed discussion of this issue.

breaches: a full client compromise or a full server compromise. Our scheme also allows the creation of multiple unlinkable personas in much the same way as with passwords. (See Section 3.5.1.)

Our two-factor protocol can be combined with a broad class of existing biometric identification schemes to protect the privacy of the user's biometric data and the template derived from it. It works with any biometric scheme where the result of feature extraction can be represented by a binary feature vector, and the matching criterion for two feature vectors is based on their Hamming distance.<sup>3</sup> The user's device (or *token*) can insist that a scan of a biometric feature be performed each time before beginning an identification round, ensuring that the presented biometric template is fresh.

The token stores only an encrypted form of the reference biometric template, which we call the *blinded template*. This keeps the biometric data safe even if the token is compromised. The encryption is performed by computing the XOR of the biometric template with a random *blinding factor* that is stored only on the server. How this is accomplished is the heart of our protocol and is described in Section 3.4. Since the blinding factor is random and carries no information about the actual biometric template, the biometric data is safe even if the server is compromised. Only if *both* token *and* server are simultaneously compromised is the user's biometric data vulnerable to exposure.

In order to perform the identification protocol, the user provides a fresh biometric sample. A new template is constructed immediately from the fresh scan and then XORed with the blinded template. The result is a blinded difference vector, that is, a vector that is the XOR of the original template, the fresh template, and the

---

<sup>3</sup>The Hamming distance between two bit vectors is the number of indices in which the two vectors differ.

blinding factor. The blinded template is then sent to the server, which unblinds it to reveal the difference vector between the two templates. The server then counts the number of “1” bits in the difference vector to obtain the Hamming distance between the two templates, which it then compares against the acceptance threshold of the underlying biometric scheme in order to establish the user’s claim of identity.

Note that the blinded template itself never leaves the token. It is used only to construct the blinded difference vector, which is sent to the server. Thus, even a compromised server that watches a valid identification attempt can learn only the difference between the two templates; not either of the templates themselves.

The scheme just described is not secure against replay attacks. An eavesdropper who records the blinded difference vector could subsequently present it to the server and have it accepted as valid.

To overcome this problem, our protocol changes the blinding factor after each successful identification, similar to the way a rolling code used to protect some keyless entry systems from replay attacks [191]. The idea is that both token and server use synchronized pseudorandom number generators to produce a sequence of blinding factors that changes after each identification round. Only if the current blinding factors match can the identification succeed.

The rest of this chapter is structured as follows. Section 3.3 discusses biometric identification. Section 3.4 presents our protocol and Section 3.5 provides several extensions. Section 3.6 analyzes its security properties. Section 3.7 discusses issues that arise in deploying our protocol. Section 3.8 describes a prototype implementation and a performance evaluation of our protocol.

### 3.3 Biometric Identification

Biometrics have been long recognized as an excellent building block for identification protocols. Biometric identification offers a higher level of confidence that users are who they claim to be as well as great convenience and usability as it relieves users from the need to remember multiple user names and passwords. Biometric identification uses unique characteristics of a human body to verify the identity of a person. There are two types of biometric characteristics suitable for biometric identification: physiological (face geometry, fingerprint, iris pattern, etc.) and behavioral (voice print, keystroke characteristics, gait, etc.) [129]. Biometric characteristics suitable for identification purposes should be universal (every person should have it), unique (it should be different for every person), permanent (it should not significantly change with time), and collectable (it must be possible to quantitatively measure it) [47]. Such characteristics are exceptionally suitable for identification purposes as they uniquely identify people and persist over time. Additionally, biometric data is constantly available and it also cannot be lost or forgotten. However, the uniqueness of biometric data is also a source of security and privacy concerns. Unlike passwords and other knowledge-based factors, biometric data cannot be reset or changed. Therefore, if compromised, it is potentially unusable for identification purposes and can be used to successfully impersonate an individual. Additionally, user's unique identity is embedded into a biometric template and if used across different service providers, it may allow to track and linked together user's actions over different transactions. For these reasons, the security of biometric data has become an important issue.

### 3.3.1 Remote Biometric Identification

A biometric system typically consists of five components: a sensor, feature extractor, template database, matcher, and a decision module. A sensor obtains a biometric sample, which the feature extractor uses to derive a biometric template. During the enrollment phase, a client enrolls into a system by creating a reference biometric template, a template that will be used during subsequent identification to verify the client's identity. The reference template is stored in a template database. During the identification phase, the client obtains a fresh biometric template that the matcher compares to the reference template. The decision module uses the output of the matcher to decide whether to accept or reject the client's claim of identity.

Each component has its specific vulnerabilities that can affect the security of the system. A sensing device might be fooled to read a biometric sample not from a person but an artifact, such as a fake finger. If a feature extractor does not properly work, it can generate very similar biometric templates for different users despite the differences in their biometric characteristics. A matcher might calculate a high match score for two very different templates or a decision module might not be sensitive enough to reject templates which are not sufficiently similar to the reference template. Lastly, if a template database is not secured, then biometric templates might be compromised. [105]

An attacker may choose to exploit individual system components or the communication channels between them [105,151]. Such attacks affect the security of a system to a different degree, depending on the way biometric identification is performed: remotely or on a stand-alone system (in a so called kiosk setting)

Biometric identification in a kiosk setting is performed in person. All or most system components are in same physical location and the process is often supervised

by humans. The biometric data is obtained and verified from the user on the spot. This protects the biometric data from an unauthorized use and exposure, however, it is only suitable for limited applications, such as entry systems. Disney World has been using hand-geometry to prevent customers from reusing the same season ticket for many years. The United Arab Emirates have deployed iris-based verification in all ports of entry for immigration purposes. India's national biometric ID program has so far has enrolled about 550 million residents and aims to cover entire population of 1.25 billion in a few years. Interestingly, using biometrics for identity verification is not a new idea: iris-scan technology has been piloted in ATM environments in the US, England, Japan and Germany since 1997. Unfortunately, we cannot simply apply the same techniques in the remote scenario, where the user and the server are in a different physical location.

Remote biometric identification refers to the process of performing a biometric identification protocol over a network. This approach makes the identification process more universal and flexible as the parties can be in different physical locations. However, system components are typically distributed between the proving and verifying party. In order to perform remote identification, biometric data must be transported over the network and biometric templates must be kept by the verifying party, which raises several security and privacy issues.

### **3.3.2 Security and Privacy Issues**

The perception and acceptance of biometric systems significantly depends on the security of biometric data [7, 66]. However, the uniqueness of biometric data, a cherished feature of biometrics, is also a source of security and privacy concerns.

A biometric template derives from characteristics, which uniquely identify an individual, and as mentioned before cannot be changed. The template has the user's

identity “embedded” into it and therefore there are limited defenses in case of compromise [148]. Typically, the proving party makes the biometric template available to the verifying party for the purpose of comparison. In case of remote identification, this poses a risk of serious attacks in which biometric data is intercepted during transmission, stolen from the verifying party or even misused by the verifying party.

From the security point of view, once compromised, biometric data has limited utility for identification purposes as it is sufficient to impersonate an individual. The privacy issues are two fold. Firstly, a biometric template, in addition to defining a user’s identity, carries considerable personal information, which often include race, gender and certain medical conditions [134]. Secondly, a biometric template can be used to identify an individual and successfully track and link his or her activities performed using the same biometric identity across different verifying parties.

For these reasons, biometric templates security has become a crucial issue resulting in a high level of awareness and concern [141]. Users expect that the verifying parties protect their biometric data and use them only for the purpose provided [17], in order to prevent identity theft, information linkage across different providers, and secondary uses of supplied information [133].

There are two major factors that play an important role to template security: the way a biometric template is generated from a biometric sample and the location where the template is stored. Attacks on templates can lead to many vulnerabilities: a template can be replaced by an attacker’s template to gain unauthorized access, biometric data can be retrieved from the template, or the stolen template can be replayed to the matcher [104].

There are four main locations for storing biometric templates: portable tokens, central databases, sensors, and individual workstations [144], with the two former being the most popular options. A portable token, for example, a smart card or

mobile device, allows users to secure their biometric templates and gives them a sense of control over their personal data. However, issues arise when tokens are lost as their content is usually unsecured. A central database makes it possible for users to easily authenticate from multiple locations because the templates are constantly available for verification. Such a database, however, needs to be kept secure and may become a target of attacks because of its valuable content. Furthermore, central storage of templates causes privacy concerns because all identification attempts go through a single point, potentially revealing users' actions. Storing templates directly on a sensor used to obtain biometric samples provides for quick responses during identification attempts. However, each sensor needs to store a copy of templates to allow multiple location access. Individual workstations offers the benefits of storing templates on tokens and sensors. Users are still in charge of their own personal data and templates are available where the user would mostly use them. The downside is the security of the workstation which is typically much lower than what a central database would offer.

### 3.4 Protocol Description

In this section, we give a full description of the protocol that is described informally in the previous section. In particular, we describe the enrollment phase, the identification phases, and the resynchronization method that is needed to recover from failed identification attempts.

The identification process is performed over a network between an authenticating party (Peggy, the user) and a verifying party (Victor, the server).

The protocol requires a pseudorandom number generator  $G = (m, S, \iota, \delta, \rho, n)$  whose arguments are the length of the seed  $m$ , a finite set of internal states  $S$ , an

initial-state function  $\iota$  which maps a seed  $z$  to a state  $s_0 \in S$ , a next-state function  $\delta$ , and an output function  $\rho$ , which produces  $n$ -bit values. We assume that  $\delta$  is a permutation on  $S$ .  $G$  is used during the protocol to generate a sequence  $s_0, s_1, s_2, \dots$  of states and a corresponding sequence  $r_0, r_1, \dots$  of pseudorandom numbers. These in turn are used to generate a sequence of blinding factors  $R_0, R_1, R_2, \dots$ . More formally,  $s_0 = \iota(z)$ , and for each  $k \geq 0$ ,  $r_k = \rho(s_k)$ ,  $R_k = \bigoplus_{j=0}^k r_j$ , and  $s_{k+1} = \delta(s_k)$ .

### 3.4.1 Enrollment Phase

During the enrollment phase, Peggy and Victor cooperate to create Peggy's credentials and to establish the shared protocol information.

The public information includes the choice of a biometric characteristic, a feature extractor that produces a biometric template, an appropriate matching metric on templates, and an appropriate pseudorandom number generator  $G$ . Our protocol assumes the template is described by a Boolean vector, and the matching metric is a function of the difference between templates. It assumes that  $G$  is cryptographically secure and backtracking resistant. (See Section 3.6.1 for definitions.)

Since our protocol is a two-factor scheme, Peggy needs to obtain a token on which to store her blinded biometric template and the state of the pseudorandom number generator. Section 3.7.2 discusses the issue of tokens and obtaining them.

Peggy and Victor also need to obtain a shared secret  $z$  to be used as the seed for  $G$ . The seed needs to be established in a secure manner in order to keep the sequences of blinding factors secret and the blinded template secure. How the seed is obtained will depend on the enrollment method. With face-to-face enrollment, Victor can generate the seed and give it to Peggy. For remote enrollment, the secret seed can be exchanged using one of the schemes to establish a shared secret, for example a key agreement protocol [84]. Alternatively, Victor can generate the seed

and send it to Peggy through a secure channel.

To continue the enrollment, Peggy and Victor initialize  $G$  using seed  $z$ . Peggy obtains a biometric sample using an external sensor or a sensor built into the token depending on the kind of token she chose as described in Section 3.7.2. She next generates a reference template  $P_{\text{ref}}$ . She then blinds  $P_{\text{ref}}$  with the first blinding factor  $R_0 = r_0$  generated using  $G$  to produce the blinded template  $T_0 = P_{\text{ref}} \oplus R_0$ . This process binds Peggy's biometric identity to the secret seed  $z$  established with Victor.

Victor meanwhile uses  $G$  to generate the blinding factor  $R_0$ , which he stores for future use. Both Peggy and Victor store the next state  $s_1$  of  $G$ . Finally, Peggy securely erases her raw biometric data, the unprotected template, the secret  $z$ , the first blinding factor  $r_0$ , and the first state  $s_0$  of  $G$ . Similarly, Victor securely erases the secret  $z$  and the first state  $s_0$  of  $G$ .

Algorithm 1 shows the steps of the enrollment process in detail.

### 3.4.2 Identification Phase

Peggy and Victor run the identification phase each time Peggy wishes to prove her identity to Victor. We number the identification phases in sequence, starting with 1, and we denote the current phase number by  $k$ . Neither Peggy nor Victor need to know the current phase number in order to carry out the protocol, but we use the phase number to distinguish the values available to Peggy and Victor during the phase. Thus, at the start of phase  $k$ ,

- Peggy's token stores  $T_{k-1} = P_{\text{ref}} \oplus R_{k-1}$  and  $s_k$ .
- Victor stores  $R_{k-1} = \bigoplus_{j=0}^{k-1} r_j$  and  $s_k$ .

In order to authenticate, Peggy obtains a fresh biometric sample and generates a

---

**Algorithm 1** Enrollment Phase
 

---

1. Peggy obtains a token. Peggy and Victor agree on non-secret identification information: the biometric recognition protocol and the choice of  $G = (m, S, \iota, \delta, \rho, n)$ , where  $m$  defines the length of the seed,  $S$  is the finite set of states of the generator,  $\iota$  is the initial-state function,  $\delta$  is the next-state function,  $\rho$  is the output function, and  $n$  is the length of biometric templates.
2. Peggy and Victor securely exchange a random seed  $z \in \{0, 1\}^m$ .
3. Peggy and Victor both initialize their generator  $G$  to the initial state  $s_0 = \iota(z)$ . They use  $G$  to generate the first random number  $r_0 = \rho(s_0)$  and the next state  $s_1 = \delta(s_0)$  of  $G$ .
4. Peggy obtains a biometric template  $P_{\text{ref}}$ , computes the first blinding factor  $R_0 = r_0$ , and creates a blinded template  $T_0 = P_{\text{ref}} \oplus R_0$ , where  $\oplus$  is the bit-wise exclusive-OR operation. She securely erases  $z$ ,  $P_{\text{ref}}$ ,  $R_0$ ,  $r_0$ , and  $s_0$ . She keeps  $T_0$  and  $s_1$  on her token.
5. Victor computes the first blinding factor  $R_0 = r_0$ . He securely erases  $z$ ,  $r_0$ , and  $s_0$ . He retains  $R_0$  and  $s_1$  in private storage.

To summarize, after the enrollment phase:

- Peggy's token stores  $T_0 = P_{\text{ref}} \oplus R_0$  and  $s_1$ .
  - Victor retains  $R_0 = r_0$  and  $s_1$ .
- 

new template  $P_k$  from it. She then calculates an identification message

$$W_k = P_k \oplus T_{k-1} = (P_k \oplus P_{\text{ref}}) \oplus R_{k-1}.$$

Peggy sends  $W_i$  to Victor for verification.

Without waiting for a response from Victor, she immediately updates her token.

She uses  $G$  to compute  $r_k = \rho(s_k)$  and  $s_{k+1} = \delta(s_k)$ . She then computes

$$T_k = T_{k-1} \oplus r_k = P_{\text{ref}} \oplus R_{k-1} \oplus r_k = P_{\text{ref}} \oplus R_k.$$

Finally, she securely replaces  $T_{k-1}$  with  $T_k$  and  $s_k$  with  $s_{k+1}$ , and she securely erases

$W_k$  and all other temporary data from memory. By updating after each identification attempt, successful or not, she ensures that the same blinding factor is never used more than once.<sup>4</sup>

Upon receiving  $W_k$  from Peggy, Victor removes the blinding factor  $R_{k-1}$  to obtain the difference vector  $V_k = P_k \oplus P_{\text{ref}}$ . He applies the matching metric of the underlying biometric system to  $V_k$  in order to decide whether or not to accept Peggy's identification attempt as valid.

If valid, he updates his stored information. Using  $G$ , he computes  $r_k = \rho(s_k)$  and  $s_{k+1} = \delta(s_k)$ . He then computes  $R_k = R_{k-1} \oplus r_k$ . Finally, he securely replaces  $R_{k-1}$  with  $R_k$  and  $s_k$  with  $s_{k+1}$ , and he securely erases all temporary data from memory.

### 3.4.3 Resynchronization

A legitimate identification attempt might fail for many reasons, for example, because Peggy's message never reaches Victor, or because of poor feature extraction by Peggy, or because of Victor's not storing the updated blinding factor before going offline, or because of intentional malicious identification attempts by an adversary. In such cases, Peggy advances her generator but Victor does not. This will leave Victor unable to unblind Peggy's future messages.

Peggy's and Victor's generators must be resynchronized in order for identifications to continue. Because Peggy updates her blinded template  $T_k$  after each identification attempt and Victor does so only after a successful identification, if the generators are out of sync, Peggy's generator will be ahead of Victor's. A simple solution is for Victor to search forward in the sequence produced by  $G$  for some limited predefined distance looking for a blinding factor  $R_{k'}$  that leads to a successful identification

---

<sup>4</sup>This prevents an attacker from obtaining useful information from differencing two identification messages  $W_i$  and  $W_j$ . If they both used the same blinding factor  $T$ , the blinding factor would cancel out, and an attacker could compute  $W_i \oplus W_j = (P_i \oplus T) \oplus (P_j \oplus T) = P_i \oplus P_j$ .

using Peggy’s current identification message  $W_k$ . After finding the correct value of  $T_k$ , both generators will again be in sync. Such a scheme is called a rolling code and is widely used in keyless entry systems [191]. Alternatively, Peggy and Victor can both keep track of the current stage of their generators, and Peggy can send it to Victor along with her identification message. Both of these schemes can be exploited by an adversary if the distance that Victor is allowed to advance during resynchronization is not reasonably limited.

Algorithm 2 shows the steps of the identification process in detail.

---

**Algorithm 2** Identification Phase

---

1. Peggy obtains a biometric sample and generates a fresh biometric template  $P_k$ .
2. Peggy calculates  $W_k = P_k \oplus T_{k-1}$  and sends  $W_k$  to Victor.
3. Peggy uses  $G$  to compute  $r_k = \rho(s_k)$  and  $s_{k+1} = \delta(s_k)$ . She then computes  $T_k = T_{k-1} \oplus r_k$ . She securely replaces  $T_{k-1}$  on her token with  $T_k$  and  $s_k$  with  $s_{k+1}$ , and she securely erases  $P_k$ ,  $W_k$ ,  $r_k$ , and  $s_k$  from her token.
4. Victor, upon receiving  $W_k$ , computes the difference vector  $V_k = W_k \oplus R_k$ . He passes  $V_k$  to the matching algorithm and accepts Peggy’s identification attempt if the match is sufficiently good.
5. If the identification attempt succeeds, Victor uses  $G$  to compute  $r_k = \rho(s_k)$  and  $s_{k+1} = \delta(s_k)$ . He then computes  $R_k = R_{k-1} \oplus r_k$ . He securely replaces  $R_{k-1}$  with  $R_k$  and  $s_k$  with  $s_{k+1}$  in memory, and he securely erases  $r_k$ , and  $s_k$ .

To summarize, at the end of identification phase  $k$ :

- Peggy’s token stores  $T_k = P_{\text{ref}} \oplus R_k$  and  $s_{k+1}$ .
  - Victor retains  $R_k = \bigoplus_{j=0}^k r_j$  and  $s_{k+1}$ .
- 

## 3.5 Extensions

In this section we describe three possible extensions of our main protocol. First, we discuss how our protocol can be used to establish independent identities (per-

sonas). Second, we explore using digital signatures as a defense against a possible impersonation attack in case of a server compromise. Finally, we discuss further minimizing information leakage during the verification process through the use of privacy-preserving private set intersection techniques.

### 3.5.1 Personas

In case of password authentication, during enrollment, the user creates a username and password, with the username as the unique user identifier and the password as the authentication factor. To authenticate, the user presents his credentials and the server verifies that they match with the data presented at enrollment time. Passwords permit the user to create multiple unlinkable personas, or identities, that be used with different services. One simply chooses a different username and password for each service.

Many biometric identification protocols create user's identification credentials directly based on a biometric template. As a result, if a user chooses to enroll with multiple verifying parties using the same biometrics, then these verifying parties can identify the user and successfully track his activities performed using the same biometric credentials or credentials based on the same biometric features.

In our protocol, credentials are based on a user's unprotected biometric template but with respect to the blinding factors known to the verifying party. Therefore, a user can create multiple, fully independent personas. Each persona is based on the same biometric data but on a different secret shared with a verifying party. Therefore, a persona represents a user's unique identity as seen by the verifying party. Users can create different personas to deal with multiple verifying parties, or use personas for different transactions with the same verifying party. This creates a separation and *unlinkability* of biometric identities and transactions performed using

those identities. The user must perform the enrollment process once for each persona, choosing a new secret for each. Policies and procedures controlling the enrollment process would determine how many and what types of personas a user may acquire.

### 3.5.2 Digital Signatures

We consider the previously described case when Mallory compromises Victor and therefore obtains his entire secret state. Specifically, Mallory gets the sequence of all blinding factors needed to unblind Peggy’s next identification message. This gives Mallory enough information to prepare a message on Peggy’s behalf and effectively impersonate her.

This attack is easily prevented by the use of digital signatures. During the enrollment phase, Victor obtains Peggy’s public verification key  $K_{pub}$  which corresponds to a private signing key  $K_{sec}$ . During the identification phase, Peggy signs her identification message  $W_i$  using  $K_{sec}$ . Victor only processes messages that are properly signed. If Mallory chooses to send an identification message without a valid signature, Victor will reject it and decline Mallory’s request since Mallory cannot produce valid signatures without  $K_{sec}$ .

### 3.5.3 Private Set Intersection

Our protocol allows Victor to make an identification decision based on a match function calculated from the difference vector between Peggy’s reference template and a fresh template. Peggy securely computes the blinded difference vector and sends it to Victor who decrypts it and uses to make an identification decision. This approach leaks the additional information of which positions of the difference vector differ—information that could conceivably be useful to an attacker.

We consider the use of private set intersection (PSI) techniques [72] for directly calculating the output of the match function. Private set intersection techniques allow two parties holding private input sets to calculate the intersection of their sets (elements they have in common), or the intersection set cardinality (the number of common elements), or simply whether the intersection set cardinality exceeds a fixed threshold. All this can be done without revealing anything else about their private inputs.

These techniques are directly applicable to our verification process. We envision using the PSI cardinality protocol [72] enriched with an encoding mechanism to calculate the difference score without giving Victor access to the binary difference vector. Using a PSI protocol would offer a greater protection against even minimal information leakage but it would come at a computational cost at least  $\mathcal{O}(k \log \log k)$  and communication cost  $\mathcal{O}(k)$ , where  $k$  is the size of the template in bits. Trading performance for enhanced protection may be justified in situations which require preventing even minimal information leakage.

### 3.6 Security Analysis

Our main goal and concern is the security of biometric data, not only under normal use of the protocol, but also in case of complete compromise of either Peggy’s token or Victor’s entire internal state. In addition, our protocol prevents an attacker who compromises Peggy’s token from impersonating her to Victor.

We note that if an attacker compromises both Peggy and Victor, then Peggy’s biometric template is easily obtained. Peggy’s token contains her blinded template; Victor has the blinding factor. It is needed to unblind the difference vector, but it will also unblind the template stored on the token.

### 3.6.1 Assumptions

Peggy and Victor interact over a network, possibly in the presence of a computationally bounded adversary (Mallory, the malicious adversary). In addition to eavesdropping on all communication between Peggy and Victor, we assume that Mallory can talk directly to Victor in an attempt to impersonate Peggy. In addition, Mallory might actively attack either Peggy or Victor but not both.

In an attack on Peggy, Mallory takes possession of Peggy's token and obtains access to all of the data stored on it. Peggy detects the attack since her token is physically gone. Nevertheless, our protocol guarantees that Peggy's biometric data remains secret and Mallory cannot impersonate Peggy to Victor.

In an attack on Victor, Mallory compromises Victor and gains access to all of his data. Victor does not necessarily detect the intrusion. He continues processing identification requests, and Mallory sees everything that happens on the server. In this case, Peggy's biometric data still remains secret, but Mallory can easily impersonate Peggy to Victor. This can be prevented by using digital signatures as described in Section 3.5.2.

We assume that all communication occurs over an unsecured channel, so after  $k$  identification attempts, Mallory knows the identification messages  $W_1, \dots, W_k$ , which are blinded differences between pairs of biometric templates. Thus, Mallory has the following information.

$$\begin{aligned}
 W_1 &= P_1 \oplus T_0 &= P_1 \oplus P_{\text{ref}} \oplus r_0 \\
 W_2 &= P_2 \oplus T_1 &= P_2 \oplus P_{\text{ref}} \oplus r_0 \oplus r_1 \\
 W_3 &= P_3 \oplus T_2 &= P_3 \oplus P_{\text{ref}} \oplus r_0 \oplus r_1 \oplus r_2 \\
 &\dots &\dots \\
 W_k &= P_k \oplus T_{k-1} &= P_k \oplus P_{\text{ref}} \oplus r_0 \oplus \dots \oplus r_{k-1}
 \end{aligned}$$

Furthermore, we assume that the sensor Peggy uses to obtain biometric samples does not directly reveal her biometric data to Mallory, prior to his possible compromise of Peggy’s token. We also assume that Peggy does not use her token after it has been compromised. Similarly, we assume that the communication channel between the sensor and the token is trusted.

The security of our protocol depends critically on the pseudorandom number generator  $G$ , which we assume is cryptographically secure and backtracking resistant. We also assume that the secret seed  $z$  and the unprotected reference template  $P_{\text{ref}}$  are securely erased after enrollment and are not available to Mallory.

To be *cryptographically secure* means that the sequence of outputs are computationally indistinguishable from a similar sequence of truly random numbers. The notion of computational indistinguishability, introduced by Yao [194], means that any probabilistic polynomial-time algorithm behaves essentially the same whether supplied with pseudorandom inputs or truly random inputs. See Goldreich [83] for further details.

To be *backtracking resistant* means that it is not feasible to run  $G$  backwards from a given state to recover previously-generated values.<sup>5</sup> More formally, it means that the sequence  $r_0, \dots, r_k$  is computationally indistinguishable from a truly random sequence of the same form, where the distinguishing judge also has access to  $s_{k+1}$ . Cryptographically strong pseudorandom number generators that are resistant to a previous-outputs backtracking attack exist and are proven to be secure [14].

For our protocol, backtracking resistance protects the biometric reference template from an attack where Mallory obtains the token and has access to the blinded reference template  $T_k$  and the state  $s_{k+1}$  of  $G$ . Backtracking resistance prevents

---

<sup>5</sup>Note that the previous values are well defined since we assume the next-state function  $\delta$  is a permutation on  $S$ . Among other things, backtracking resistance implies that  $\delta$  is a one-way permutation.

Mallory from running the generator backwards from  $s_{k+1}$  to obtain  $r_k, r_{k-1}, \dots, r_0$  from which the blinding factor  $R_k$  and the reference template  $P_{\text{ref}} = T_k \oplus R_k$  could be computed. The same protection holds even if Mallory has prior knowledge of  $r_0, r_1, \dots, r_{k-1}$  (which she might) but not  $r_k$ .

### 3.6.2 Security of Biometric Templates

#### Mallory compromises Victor

We assume that Mallory can compromise Victor at any time and remain undetected. If she compromises him at phase  $k$ , she learns the current blinding factor  $R_k$  and the next state of the generator  $s_{k+1}$ . This enables her to compute the future random numbers  $r_{k+1}, r_{k+2}, \dots$  and the future blinding factors  $R_{k+1}, R_{k+2}, \dots$ . Clearly, she learns the most by compromising Victor at the very beginning, in which case she recovers the exact same information that Victor receives from Peggy. From this, she can learn all of the difference vectors,  $V_1, V_2, \dots$ .

The usefulness of the difference vectors depends on the underlying feature extractor. Ideally, we want a feature extractor that produces templates on repeated scans of the same biometric that lead to small false rejection rates. When the match function is based on the Hamming distance between two templates, small false rejection rates imply that most difference vectors approximate the zero vector. Hence, Mallory only learns vectors in the neighborhood of 0. In any case, we can say that Mallory has no other information about the template since Peggy sends nothing else besides the blinded difference vectors.

### Mallory compromises Peggy

When Mallory obtains access to Peggy's token, she learns the current blinded reference template

$$T_k = P \oplus r_0 \oplus r_1 \oplus \cdots \oplus r_{k-1} \oplus r_k$$

and the next state of the generator  $s_{k+1}$ . This new information is in addition to all identification messages  $W_1, \dots, W_k$  sent up to that point which we assume she already knew.

$T_k$  looks random to Mallory because of the blinding factor  $r_k$ , which she does not know. It was securely erased from the token when  $T_k$  was updated, and it was never included in any of the messages sent. Additionally,  $r_k$  cannot be recovered using the stored state  $s_{k+1}$  of  $G$  and  $r_0, \dots, r_{k-1}$  (assuming they are known) since  $G$  is backtracking resistant and cryptographically secure. Thus, Mallory cannot obtain any information about the blinding factor  $R_k$ , so neither  $T_k$  nor  $s_{k+1}$  give Mallory any information about  $P_{\text{ref}}$ .

We assume that Peggy's token is not compromised at the moment she is using it, since for a brief interval, the token contains her unprotected biometric template as well as data from both stage  $k - 1$  and stage  $k$ .

### 3.6.3 Impersonation

#### Mallory compromises Victor

As before, when Mallory compromises Victor, she gets  $R_k$  and  $s_{k+1}$ .  $R_k$  is the blinding factor needed to unblind the next identification message. Therefore, Mallory might be able to prepare a fake message  $W'_k$  so that verification will succeed from Victor's point of view. Section 3.5.2 discusses a practical defense against this attack.

## Mallory compromises Peggy

As before, when Mallory compromises Peggy, she gets  $T_k$  and  $s_{k+1}$ . We argued in Section 3.6.2 that her compromise of Peggy’s token does not give her any information about  $P_{\text{ref}}$  or  $R_k$ . Hence, she gets no information that would allow her to impersonate Peggy.

### 3.6.4 Leakage of Information

During each identification attempt, Victor receives a difference between two biometric templates. If he has been compromised, then Mallory also receives this information. Unlike the case of a compromise of Peggy, we assume that the compromise of Victor might be undetected so that Mallory can collect data over time from legitimate identification requests.

After a number of such identifications, Mallory has a set of differences between Peggy’s templates. Those differences are binary vectors of differences between Peggy’s reference template and the sample template used on a given identification. A difference bit of 1 indicates a discrepancy between the reference and the sample templates.

The frequency of 1’s in any given bit position represents the unreliability in that position of the template. A low-frequency position indicates a reliable bit; a high frequency position means that little useful information is being carried by that bit. Mallory can compute these frequencies and thereby learn about the reliability of each bit in the template. What information these frequencies carry about the actual reference template or Peggy’s raw biometric data depends in detailed properties of the sensor as well as the feature extraction algorithm. Although analyzing this kind of information leakage for any particular sensor and feature extractor is beyond the scope of this paper, it is well to keep in mind this possibility in designing biometric

systems.

Low-frequency bits can arise equally well from 0's in both reference and sample templates or from 1's in both, so knowing that it is low frequency says little about the actual template bit. With a good feature extractor, we expect most difference bits to be low-frequency, so information leakage would seem to be minimized with good quality biometric systems. Section 3.5.3 discusses an approach to further minimize any information leakage.

## 3.7 Practical Considerations

### 3.7.1 Suitable Biometrics

There are two main categories of biometric characteristics used in biometric systems: physiological (e.g., a fingerprint or iris pattern) and behavioral (e.g., voice print or signature). Characteristics must be universal (everyone has it), unique (different for every person), permanent (it does not change with time), and collectable (it can be quantitatively measured) [47]. In practice, fingerprints, face geometry, and iris patterns have been popular choices as they can be obtained easily and non-intrusively using a simple camera. Fingerprints tend to be prone to spoofing, however, and the accuracy of facial recognition may be impacted by pose, expression, or lighting [56, 103]. An iris, on the other hand, exhibits many highly desirable properties. Its pattern varies greatly among different people, even identical twins, and persists over a lifetime. Iris-based recognition systems have been widely deployed by many organizations including British Telecom, Panasonic, LG and IBM Schiphol Group [25, 54].

For these reasons, we chose to use an iris-based template for our implementation, described in Section 3.8. These templates typically consist of 2048 bits to represent

the iris pattern with any bit equally likely to be either 1 or 0. On average half of all the bits will disagree between the templates of two different people. A study [54] based on 9.1 million comparisons between different pairings of iris images concluded that it is extremely improbable that two different irises might disagree in fewer than a third of all bits. Consequently, if a difference score is less than 0.32, then a positive match is statistically guaranteed.

Difference Score	False Match
0.26	1 in $10^{13}$
0.27	1 in $10^{12}$
0.28	1 in $10^{11}$
0.29	1 in 13 billion
0.30	1 in 1.5 billion
0.31	1 in 185 million
0.32	1 in 26 million
0.33	1 in 4 million
0.34	1 in 690,000
0.35	1 in 133,000

Table 3.1: The relation between the difference score and odds of a false match [54]

An iris-based template encodes an iris pattern as a binary vector. Fingerprint templates use the fingerprint texture as a real-valued fixed length vector. Finally face-based templates use facial features represented again as a real-valued fixed length vector. A match can be performed by calculating the Hamming distance (or alternatively a fractional Hamming distance) for binary vectors, while a Euclidian distance for real-valued vectors with the points defined by the set difference. Our current protocol assumes binary biometric templates, which are suitable for all iris-based templates, however, a binarization technique [150] can convert other types of templates into a binary vector [43, 111, 150] and make it usable in our protocol but likely trading off some recognition performance for the ability to use diverse types of templates. Another promising approach is the use of private set intersection techniques,

as briefly outlined in Section 3.5.3.

### 3.7.2 Enrollment Process and Tokens

The main drawback of possession-based authentication is the need to obtain and manage tokens. Eddie, an enrolling agent, can be responsible for issuing tokens and performing the enrollment phase, ensuring a successful bond between a token and a biometric identity. Depending on the application-specific security requirements, Eddie can be an independent, trusted enrollment center, Victor can assume Eddie's role, or Eddie's role can be delegated to users. In the first scenario, Eddie's services can be offered by an organization such as VeriSign [190]. This approach would provide a good way to issue and manage a variety of tokens. VeriSign already provides similar services and issues security credentials (VIP Security Token or Card [189]).

Tokens and central databases are two most popular locations for storing biometric templates [103, 144]. A token allows users to physically secure their biometric templates and gives them a sense of control over their personal data. However, issues arise when tokens are lost or stolen. A central database makes it possible for users to authenticate from multiple locations easily as templates are constantly available for verification. On the other hand, the database may become a target of attacks because of its valuable content and central storage of templates raises privacy concerns because all authentication attempts go through a single point.

Our protocol has been designed with security of biometric data in mind, we use tokens to store biometric templates but ensure that their content is protected in case of loss or theft.

There are two different approaches to utilizing tokens depending on the token's ability to obtain biometric samples.

Using a token with a built-in sensor removes security concerns related to the

sensor and the communication channel between the token and the sensor. Mobile devices are an obvious choice for such tokens; they are equipped with a high resolution camera capable of capturing images suitable for identification using several biometric characteristics such as a fingerprint, facial geometry, or iris pattern [103]. Additionally, mobile devices make it possible to take advantage of less frequently utilized characteristics like voiceprint, keystroke or handwriting patterns, service utilization [46] or even gait [57].

A token without a sensor is only used to store identification information and to perform computations. It must be paired with an external sensor to obtain a biometric sample. This implies certain level of trust that the sensor is not compromised and the channel between the token and sensor is secure. However, such tokens are inexpensive and make it possible to utilize virtually any biometric characteristic. Smart cards are the most obvious choice for such tokens. They have been extensively used for authentication or identification as they offer enough computational power and are relatively cheap, small, and convenient to use [169].

### 3.7.3 Template Generation

A biometric template is a representation of the features from a biometric sample. A feature extractor is a component of a biometric system responsible for generating templates. During the feature extraction process, key features of the biometric sample are located, selected, measured, encoded and then stored in form of a template. The template quality directly impacts the performance of a biometric system. We require that a feature extractor produces templates of high quality. More specifically, we assume that two templates created based on a biometric sample from the same user are “sufficiently” similar to be suitable for identification purposes. Similarly, we require that two templates created based on biometric samples from different users

are “sufficiently” different. The goal is to achieve an acceptable false rejection rate (FRR) and more importantly a low false acceptance rate (FAR).

### 3.7.4 Verification Decision

In biometric systems, a matcher and decision module are the two components directly involved in making the verification decision.

A matcher takes two biometric templates, the reference template created in the enrollment phase and the freshly obtain template from the verification phase, as input. Then, it calculates a *match* score which shows how similar the two templates are [103]. In case of our protocol, the matcher functionality is embedded into the protocol. The verifying party calculates  $V_i = T_{s_{i-1}} \oplus W_i$  which defines  $\Delta(P, P'_i)$ , the difference between two biometric templates. Therefore, the value of  $V_i$  defines the *difference score*. In other biometric identification protocols, the identification decision is based on the match score while in our it is based on the difference score. To express the difference score in terms of the match score we can say that the smaller the difference  $V_i$ , the higher the match score is.

The decision module takes a match score (in our case a difference score) as input and based on a predefined threshold parameter  $\tau$  decides whether the two templates were created based on biometric samples from the same person. If the match score is greater than a predefined threshold  $\tau$ , user’s identity is verified. In our protocol, if the difference score is lower than  $\tau$ , then the two templates are accepted as coming from the same user.

Choosing a proper value for  $\tau$  is a challenging task. To have a high level of confidence that two templates were created based on samples from the same user, the difference should be very low. Hence, the value chosen for  $\tau$  should reflect the desired level of security as well as the sensor and feature extractor’s capabilities

to create accurate templates. The goal is to balance the false rejection and false acceptance rates while ensuring a proper level of security.

## 3.8 Evaluation

We evaluate our prototype implementation to observe the performance characteristics of our protocol in comparison to using unprotected templates. We then analyze the behavior of the feature extraction libraries to determine their usefulness and potential overheads in this scheme. We used the CASIA Iris Image Database [102] as input into our system.

### 3.8.1 Implementation Details

We have implemented our biometric identification system in C++ using the Qt framework and Crypto++ cryptographic libraries. For feature extraction, we have employed two different iris recognition libraries: Project Iris [24] and Libor Masek’s Iris Recognition [131], both of which utilize John Daugman’s approach [54] to produce an iris template. Project Iris uses C++ and the Qt framework; however, for Masek’s library, we constructed a C++ to Octave<sup>6</sup> interface. In evaluation figures, we denote Project Iris [24] feature extraction library as C++ and Masek’s library [131] as Octave.

We ran the evaluations on a workstation computer equipped with an Intel Core i7-2600 processor with 4 cores, 16 GB of memory, and a Crucial 256GB SSD hard drive. Our software ran in single threaded mode and never exceeded 12 MB of used memory, the typical amount for an application utilizing Qt. This shows a very

---

<sup>6</sup>Open-source Matlab compatible system

modest memory requirement of our implementation.

We use a Diffie-Hellman Key Agreement [84] to agree on a common key. We then use the agreed on key to seed a provably secure Blum-Blum-Shub generator [21] (PRBG). We used a SQLite database to store enrollment information. We set a minimal difference score of 0.32 to ensure a low probability of a false match (1 in 26 million) [54]. In their evaluations, Masek's scheme uses a modified hamming distance scheme that depends on a comparison between two unencrypted templates. Our system encrypts the templates, making use of this scheme incompatible and hence we use traditional hamming distance.

The two feature extraction libraries we employ use a technique to further boost the recognition performance. A single template may need to be rotated up to  $8^\circ$  in both directions to achieve better results. In an unprotected biometric system, the server can manipulate the template itself because it gets full access to the client's biometric template. In our system, we preprocess templates to produce these rotations and store the resulting blinded templates on the client's side. We do so because the server never gets access to unprotected biometric data and therefore cannot manipulate the templates itself. Then, during identification, the client uses all of the saved templates as input to the protocol and forwards the result to the server. Therefore, our protocol preserves the same recognition performance as schemes employing this recognition-enhancing technique.

All CASIA database images have been converted to gray scale images. CASIA database version 1 contains 108 individuals with 7 images each. Project Iris only supports version 1 of the CASIA database, in which images have been preprocessed by replacing the pupils with a black (constant intensity) circle. Masek's iris recognition library handles CASIA database version 2, however, had trouble parsing approximately 4% of the images, though had no issue in database version 1. The libraries

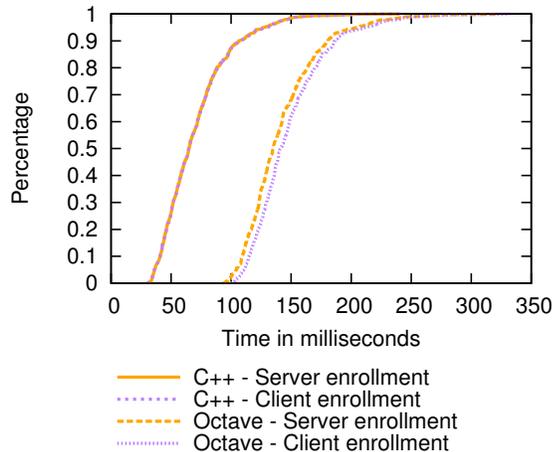


Figure 3.1: CPU time for enrollment

also differ in the resolution of their extracted features. While Project Iris extracts a 2048-bit template like Daugman [54], Masek extracts a 9600-bit template.

### 3.8.2 System Performance

To evaluate the enrollment phase, we created a client (Peggy) for each image in the CASIA database version 1 and used a single server (Victor). Clients, in no particular order, enrolled one after another. The enrollment occurred within the same process, as a result the evaluation focuses on data processing and message serialization, i.e., CPU time. Our results can be found in Figure 3.1.

The clients enrollment time includes both the initial enrollment request and the subsequent processing for a successful enrollment, both represented as a single, summed value. The client enrollment time is negligibly larger than the server enrollment time. The major factor in performance appears to be the size of the stored template(s) and the need to generate an appropriate amount of random blinding factors. While Octave, Masek’s library, uses 17 9600-bit templates with 17 masks resulting in 40.8 KBs of PRNG work, C++, Project Iris, uses only 8.7 KBs.

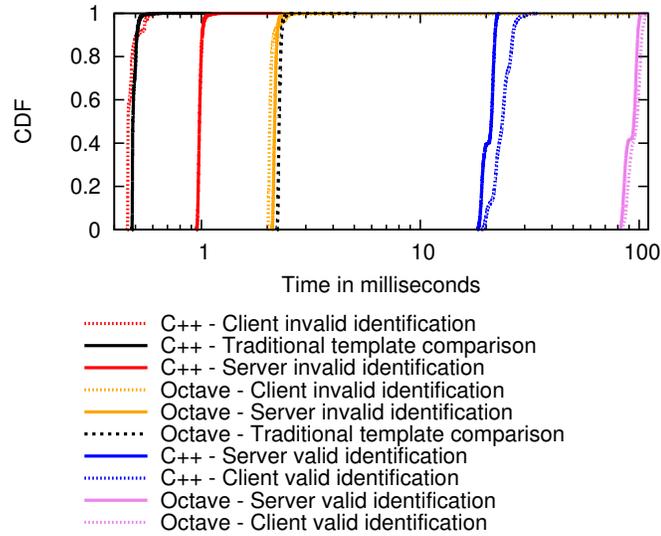


Figure 3.2: CPU time for identification

To evaluate identification time, we had each image in the database tested against every client for a total of 571,536 identification attempts or 756 attempts per client. We separated the results, in Figure 3.2, into valid and invalid client and server identifications, those that our system processed, and compared them against the time a traditional template comparison would take.

### 3.8.3 Feature Extraction Reliability

To evaluate the ability of the readily available feature extraction libraries, we computed the difference scores for two images extracted from the same individual as well as different individuals and then processed them using our system. The results, as expected, were identical, though the time to do so was different, as shown earlier in Figure 3.2. Therefore, in this section, the evaluation primarily focuses on the recognition performance of the feature extraction libraries, as shown in Figure 3.3.

While both libraries were able to identify common individuals relatively easily (low false rejection rate—FRR), we were surprised to see different participants had

such low difference scores in both systems, in particular in Masek's. Regardless of the reason for the discrepancy, we are satisfied that our system works equally well as the underlying feature extraction and unprotected matching scheme.

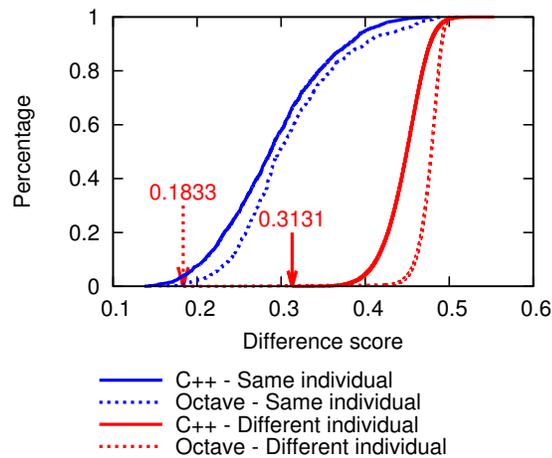


Figure 3.3: Difference scores using different template extraction libraries

### 3.8.4 Feature Extraction Timing

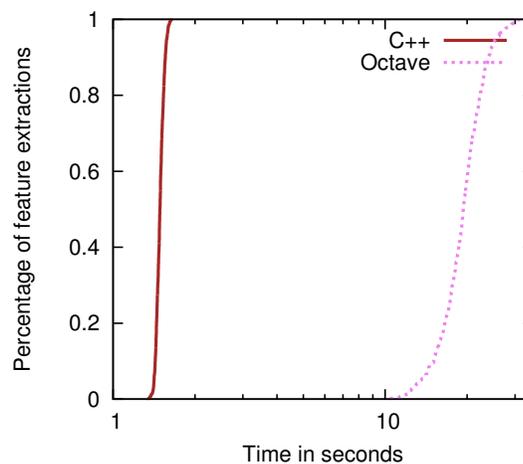


Figure 3.4: Time for feature extraction

While our system has good response time on the orders of 10s to 100s of mil-

liseconds, depending on the template size, we discovered that feature extraction has significant performance overheads as shown in Figure 3.4. This, along with the ability to capture the image, plays a critical role during the clients' identification process.

On average, the Project Iris processes images in less than 2 seconds; however, the Masek's library took an order of magnitude longer. Considering these costs, our protocol has nearly negligible overhead in comparison to the cost of feature extraction alone. Therefore, the performance of our system is very comparable to an identification system that offers no biometric data protection.

# Chapter 4

## DAGA: Deniable Anonymous Group Authentication<sup>1</sup>

This chapter presents DAGA, a deniable anonymous group authentication protocol.

### 4.1 Introduction

In privacy-sensitive communications, one user sometimes needs to prove to be a member of some explicit, well-defined group, without revealing his individual identity. Consider for example a whistleblower who wishes to leak evidence of corporate or government wrongdoing to a journalist, via an anonymous electronic “drop box” [55]. The journalist needs to validate the source’s trustworthiness, but the whistleblower is reluctant to reveal his identity for fear their communications might be compromised [89], or that the journalist will be coerced into testifying against the source [159]. The whistleblower thus wishes to *authenticate anonymously* as a member of some authoritative circle who plausibly has knowledge of and access to the

---

<sup>1</sup>Portions of the research reported in this chapter were done in collaboration with Benjamin Peterson, David I Wolinsky, Michael J Fischer, and Bryan Ford. The preliminary version of this work has appeared in [181].

leaked information, such as a corporate board member or executive, or a government official of a given rank.

Even if the whistleblower convinces the journalist of his authority, the journalist may also require *corroboration*: e.g., confirmation by one or more *other* members of this authoritative circle that the leaked information is genuine. Other members of this authoritative circle may be just as reluctant to communicate with the journalist, however. If a potential corroborator also demands anonymity, how can the journalist (or the public) know that the corroborator is indeed a *second* independent source, and not just the original source wearing a second guise? In general, if the journalist knows  $k$  pseudonymous group members, how can he know that these pseudonyms *proportionally* represent  $k$  real, distinct group members, and are not just  $k$  Sybil identities [64]?

Finally, the whistleblower is concerned that once the leak becomes public, he may be placed under suspicion – perhaps merely for being in the relevant authoritative circle – and any of his computing devices may be confiscated or compromised along with his private keys. Even if his keys are compromised, the whistleblower needs his anonymity *forward* protected, against both the journalist and any third-parties who might have observed their communications. Further, the whistleblower wishes to be able to *deny* having even participated in any sensitive communication, including the fact of having authenticated at all (even anonymously) to the journalist.

We present *deniable anonymous group authentication* (DAGA), the first protocol we are aware of satisfying the above requirements, which we term *anonymity*, *proportionality*, *forward anonymity*, and *deniability*. Like ring signatures [155], DAGA allows a user to authenticate as an anonymous member of an ad hoc group or ring, defined by an arbitrary list of public keys. The user can *conscript* other users into a group without their participation, consent, or even knowledge. Neither ring sig-

natures nor deniable ring authentication [139] offer proportionality, however: a verifier cannot tell whether several authentications were by the same or distinct group members. Linkable ring signatures [125] include a *tag* enabling a verifier to check distinctness, but anyone who later compromises the user’s private key can reproduce the linkage tags in all past signatures, violating forward anonymity and deniability. It appears likely that no purely offline anonymous signature scheme can offer both proportionality (corroboration capability), forward anonymity, and deniability in combination.

To resolve these apparently conflicting requirements, DAGA relies on a federation of independently operated servers that are *collectively* but not individually trusted. DAGA’s security property properties are ensured as long as *at least one* server operates correctly and honestly during an authentication process, even if the client does not know which server is honest. The servers divide authentication activity into *epochs*, choosing a set of fresh server-side secrets for each epoch. These secrets collectively protect the relationship between a client’s private key and the epoch-specific tags that DAGA produces to offer proportionality and corroboration capability. After each epoch, the honest server(s) securely erase their secrets, preventing anyone from compromising any client’s anonymity in past authentication epochs – even if the attacker later compromises the long-term private keys of *all* clients and *all* servers. Finally, the authentication process offers deniability by employing only interactive zero-knowledge proofs, ensuring that any valid DAGA communication transcript could have been synthesized independently by anyone.

We have analyzed and verified DAGA’s four key security properties of anonymity, proportionality, forward anonymity, and deniability.

We have also built a working proof-of-concept implementation of DAGA to validate its performance and practical usability. Using 2048-bit DSA keys, our DAGA

prototype can authenticate as a member of a 32-member group to 2 servers in about one second after consuming less than 1KB of total messaging bandwidth. Authenticating in a 2048-member group takes about two minutes and consumes about 100KB of bandwidth. Our initial prototype is currently unoptimized, and we expect its performance and efficiency can be improved in many ways. Nevertheless, our results suggest that DAGA is already practical for sensitive anonymous interactions requiring maximum security, and we believe DAGA's unique combination of proportionality (corroboration), forward anonymity, and deniability features can justify this cost in such scenarios.

We make the following key contributions:

1. Propose a new authentication scheme that offers anonymity, deniability, and proportionality even in the case of a full compromise of private keys.
2. Propose an authentication scheme that supports evolving groups *while* preserving proportionality.
3. Separate the notions of deniability, anonymity and forward anonymity, and analyze these security properties.
4. Evaluate DAGA and compares it to non-anonymous and anonymous authentication methods.

Section 4.2 offers an overview of DAGA's trust model, operation, and security properties. Section 4.3 presents the details of the DAGA's protocol and Section 4.4 outlines potentially useful extensions to the basic protocol. Section 4.5 outlines several applications for which DAGA's might be suited. Section 4.6 provides a security analysis of DAGA's properties. Section 4.7 outlines practical implementation and deployment considerations. Section 4.8 presents our prototype implementation and experimental results.

## 4.2 Overview

### 4.2.1 Trust Model

We assume an *anytrust* [192, 193] model, where there is a large set of  $n$  clients and a smaller set of  $m$  reliable servers, which includes at least one honest server that runs the prescribed protocols and does not collude with dishonest entities. The clients do not need to assume that any particular server is trustworthy; they need only trust that *some* honest server exists. We further assume that there are always at least two honest clients; anonymity is trivially impossible if  $n - 1$  clients choose to collude against only one honest client.

We assume that each anytrust server is run by a respected, reliable, and independently managed organization, each responsible for ensuring that its server remains online and uncompromised. We envision these anytrust servers being deployed by a federation of organizations wishing to support responsible forms of anonymous participation: e.g., providers of online services such as Wikipedia or Twitter, anonymity system providers such as the Tor project, non-profit organizations whose aim is to further online privacy and anonymity, or even for-profit organization desiring strong guarantees and large anonymity sets for their clients.

In such a deployment scenario, we expect the servers to offer high reliability and to offer clients with a high level of confidence that at least one honest server exists. In practice, we hope and expect a majority of the servers to be honest, allowing for an efficient resolution of issues related to the servers' performance or availability, should they arise, but we leave such availability issues outside the scope of this paper.

## 4.2.2 Protocol Overview

The main idea underlying DAGA is to allow a client to authenticate anonymously, and at the same time enforce proportionality, by enabling the servers to link authentications of the same client. To achieve this goal, we use a combination of proofs of knowledge to prove membership to a particular group and per-client *linkage tags* that effectively become clients' anonymous IDs.

Each client  $i$  authenticates using a publicly available *authentication context*  $C$ , which consists of a group definition  $G$  and other per-round authentication information. A client  $i$  prepares and sends his authentication message to an arbitrarily chosen server who starts the collective process of producing the client's final linkage tag by all servers, and upon its completions responds to the client with an authentication decision as shown in Figure 4.1.

To produce an authentication message, a client  $i$  generates an *initial linkage tag*  $T_0^i = h_i^{\prod_{k=1}^m s_k}$ , where  $h_i$  is the client's per-round generator assigned by the servers and  $s_k$  is a shared secret for every server  $k$  that a client generates in a way that each server is able to independently reconstruct it. In addition to creating the tag, the client proves in zero-knowledge that he correctly computed  $T_0^i$  and that he is a member of the group  $G$  and therefore he knows a private key  $x_i$  that corresponds to one of the public keys included in the group definition. A client  $i$  executes the following interactive "OR" proof [37, 52]:

$$\text{PK} = \left\{ \bigvee_{i=1}^n (\text{I know private key } x_i \wedge T_0 \text{ is correctly based on } h_i) \right\}$$

After completing these steps, client  $i$  securely erases his private ephemeral state and sends to some server  $j$  his tag, proof, and all other information needed by the servers to process his authentication request. The server who receives  $i$ 's authentication

request, verifies the attached proof, and processes the initial tag by scrubbing from  $T_0$  the secret  $s_j$  it shares with  $i$  and adding his own per-round secret  $r_j$ . Finally, server  $j$  proves in zero-knowledge that he correctly performed these steps and generates the following proof using a standard proof of knowledge about discrete logarithms [41, 69, 163]:

$$\text{PK} = \{(\text{Tag } T_j \text{ is correct} \wedge \text{I know my secret } r_j)\}$$

The remaining servers repeat this process, however, also verifying that the proof coming from the previous server is valid. Provided that the client  $i$  and the servers correctly follow the protocol, it yields a final linkage tag  $T_f^i = h_i^{\prod_{k=1}^m r_k}$ . Each final linkage tag  $T_f$  is unique to a client and remains the same for each authentication within the same context  $C$  as the tag depends on a client's generator and a product of all servers' secrets which remain the same.

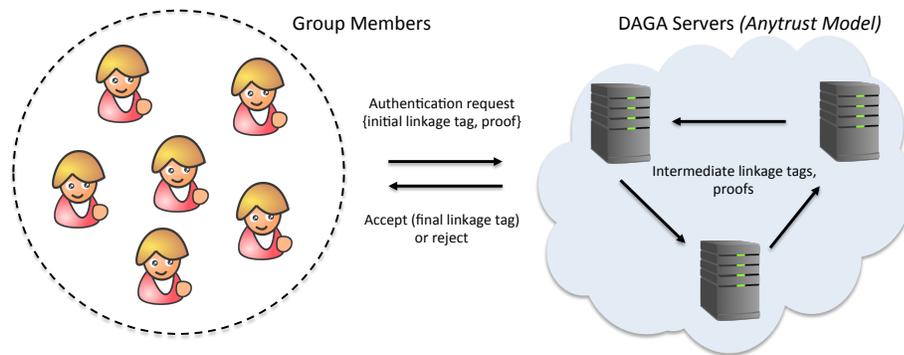


Figure 4.1: Conceptual model of DAGA

### 4.2.3 Security Properties

DAGA provides for a deniable and anonymous authentication scheme that maintains its properties even if a client's private key is compromised. A client should be able to convince the servers that he is a *unique* member of a *particular* group, without

disclosing his non-anonymous identity and without leaving any evidence that can be used later on to link him to his well known identity. Anonymity and deniability should persist even in the case of a compromise of a client’s private key after the round completion. More specifically, DAGA maintains anonymity and deniability as long as the private keys of at least two honest members and a private key of at least one honest server are not compromised. To maintain forward security, the basic DAGA protocol assumes that at least one honest server’s private key remains secure. Section 4.4.4 proposes an extension that relaxes this requirement, however, preserving forward secrecy even if *all* servers are eventually compromised.

In addition to *completeness* and *soundness*, DAGA offers four security properties: *anonymity*, *deniability*, *forward anonymity*, and *proportionality*.

*Soundness:* Under the Discrete Logarithm assumption, servers only accept authentication requests coming from a client who is a member of a group  $G$  specified in his authentication context.

*Anonymity:* Informally speaking, we want to ensure that after a complete protocol run, an adversary cannot guess which group member has been authenticated with a probability greater than random guessing. DAGA provides anonymity under the DDH assumption in the random oracle model.

*Forward Anonymity:* We extend the anonymity property to situations in which an adversary obtains a client’s private key but only after a protocol run has completed, and ensure that the knowledge of even all but the honest server’s key does allow an adversary to break any client’s anonymity.

*Deniability:* We want to ensure that the protocol does not leave a “paper trail” that an adversary could use to link a client to his authentication requests based on intercepted authentication transcripts. This guarantee persists even in the case of a compromise of private keys yielding an interesting notion of *forward deniability*.

*Proportionality:* We enforce that a client can authenticate as a unique member only once given a particular authentication context and each subsequent authentication request within the same context is recognized as coming from that client. At the same time, we ensure that client’s authentications made within two different authentication contexts are unlinkable. Additionally, proportionality persists even when new clients are added to a group because proportionality is independent of the group membership.

## 4.3 Protocol Description

### 4.3.1 Notation

We denote the client  $i$ ’s proof of correctness as  $\text{PK}^{client_i}$ , the server  $j$ ’s proof of correctness as  $\text{PK}_1^{server_j}$ , and the server  $j$ ’s proof of a client  $i$ ’s misbehavior as  $\text{PK}_2^{server_j(i)}$ . We denote the client  $i$ ’s initial linkage tag as  $T_0^i$ , the intermediate linkage tag created by a server  $j$  as  $T_j$ , and the client  $i$ ’s final tag as  $T_f^i$ . We will omit the client’s ID from  $T_0^i$  ( $T_f^i$ ) and write  $T_0$  ( $T_f$ ) when it is clear from the context which client the tag belongs to. We denote a client  $i$ ’s authentication message as  $M_0^i$  and a server’s  $j$  message as  $M_j$ .

To simplify notation, we will omit “mod  $p$ ” when performing computation on elements of  $\mathbb{Z}_p$  and “mod  $q$ ” when performing computation on exponents. We will denote choosing a random element  $x$  from  $\mathbb{Z}_q^*$  as  $x \in_R \mathbb{Z}_q^*$

### 4.3.2 Building Blocks

#### $\Sigma$ -Protocols

Zero-knowledge proofs of knowledge [85] are proofs that yield nothing beyond of the validity of the assertion a prover  $P$  wants to convince a verifier  $V$  about. However, such proofs normally require a large number of interactions between the prover and verifier. A  $\Sigma$ -protocol [53] is a special type of an interactive zero-knowledge proof of knowledge that requires only one interaction and always consists of exactly three moves: given common input  $I$  (1)  $P$  sends a commitment  $t$  to  $V$ , (2)  $V$  responds with a random  $\ell$ -bit challenge  $c$ , and (3)  $P$  sends back a response  $r$ .  $V$  makes a decision based on  $(I, t, c, r)$ . By definition [53], a  $\Sigma$ -protocol has the properties of completeness, special soundness and special honest-verifier zero-knowledge. The client's and servers' proofs instantiated in DAGA have these properties.

#### “OR” Proofs

An “OR” proof of knowledge is an example of a  $\Sigma$ -protocol and allows a prover to convince a verifier that he knows a secret  $x$  that corresponds to one out of two assertions without the verifier learning which one. An “OR” proof can be easily generalized to proving the knowledge of a witness to one of many assertions (“1-out-of- $n$ ”) or even multiple witnesses (“ $k$ -out-of- $n$ ”).

DAGA makes use of interactive and non-interactive proofs. The client's proof is an interactive protocol instantiated using the techniques of Camenisch and Stadler [37] which are an extension of the previous works on proof of knowledge [52,69,163]. The server's proofs uses non-interactive protocols based on Schnorr's proof of knowledge of discrete logarithms [163], proof of equality of discrete logarithms [41], While  $\Sigma$ -protocols are interactive by nature, a heuristic proposed in [69] allows to replace the

interaction with a verifier with a hash function modeled as a random oracle. For simplicity, we write “proof” or “proof of knowledge” for “three-move honest-verifier computationally zero-knowledge proof of knowledge”.

### 4.3.3 Assumptions

We assume that communication channels exist between all parties and a client has an authenticated channel with every server. We assume an adversary that is polynomial-time limited, can control a colluding subset of up to  $n - 2$  clients and up to  $m - 1$  servers, and can observe and record all network messages.

We assume that each client has a long-lived non-anonymous identity associated with a public-private key pair. We define a client’s identity as his associated key pair; therefore, a client  $i$  represents a client who owns a public key  $X_i$ . Specifically, each client  $i$  has a long-term Diffie-Hellman (DH) key pair consisting of a private key  $x_i$  and public key  $X_i = g^{x_i}$  and each server  $j$  has a corresponding private/public key pair  $(y_j, Y_j = g^{y_j})$ . We assume that there is a readily available group definition  $G = (\vec{X}, \vec{Y})$  listing clients and servers and their long-term public keys,  $X_i$  and  $Y_j$  respectively. The author of a group definition may conscript arbitrary clients knowing only their public keys. Some of the clients listed in the group definition need not ever participate in the protocol or even be aware that they are included. The group definition is a part of an authentication context which defines all constants for each authentication round.

### 4.3.4 Authentication Context

In DAGA, a client  $i$  anonymously authenticates as a member of a particular group  $G$  with the help of a set of *anytrust* servers using a publicly available *authentic-*

*tion context C.* An authentication context  $C = (G, \vec{R}, \vec{H}, p, g)$  consists of a group definition  $G$ , a set  $\vec{R}$  of each server's commitment to a per-round secret, a set  $\vec{H}$  of each client's per-round generators, a safe prime  $p = 2q + 1$  where  $q$  is a sufficiently large prime, and a generator  $g$  of the order  $q$  subgroup  $\mathcal{G}$  of  $\mathbb{Z}_p^*$ . We define a *group G* as a tuple  $(\vec{X}, \vec{Y})$  where  $\vec{X}$  is a set of the  $n$  clients' public keys and  $\vec{Y}$  is a set of the  $m$  servers' public keys. To generate  $\vec{R} = (R_1, \dots, R_m)$ , each server chooses a secret  $r_j \in_R \mathbb{Z}_q^*$  and publishes a commitment  $R_j = g^{r_j}$ .  $\vec{H} = (h_1, \dots, h_n)$  consists of  $n$  unique per-round generators of  $\mathcal{G}$ , one for each client  $i$ , such that no one knows the logarithmic relationship between any  $h_i$  and  $g$  or between  $h_i$  and  $h_{i'}$  for any pair of clients  $i \neq i'$ . Section 4.7.5 describes how to find these generators and Section 4.7.3 further discusses issues related to creating and using an authentication context.

### 4.3.5 Client's Protocol

A client  $i$  wishing to authenticate, obtains an authentication context  $C$ , uses it to produce an authentication message  $M_i^0$ , and sends it to one arbitrarily chosen server listed in  $\vec{Y}$ . Upon receiving the client's message, all servers collectively process  $M_i^0$  and either accept or reject  $i$ 's authentication request. If  $i$ 's request is accepted, then it results in a final linkage tag  $T_f^i$ . If it is rejected, however, then the client's proof  $\text{PK}^{client_i}$  is invalid, at least one server produces a proof  $\text{PK}_2^{server_{j(i)}}$  of the client's misbehavior, or some server produces an invalid proof  $\text{PK}_1^{server_j}$ .

We define an *authentication round* with respect to a particular authentication context  $C$ . Each authentication request, regardless of the identity of the originating client, belongs to the same round if it is made with respect to  $C$ . All requests within the same round are linkable, that is, each time a client  $i$  authenticates, the servers will be able to link these requests as coming from *some* client from  $G$ .

A client  $i$  performs the following steps to create  $M_i^0$ .

**Step 1:** Client  $i$  first picks an ephemeral private DH key  $z_i \in_R \mathbb{Z}_q^*$  and computes a public key  $Z_i = g^{z_i}$ . Client  $i$  keeps  $z_i$  secret.

**Step 2:** For each server  $j$ ,  $i$  computes a shared secret exponent  $s_j = \mathcal{H}_1(Y_j^{z_i}) = \mathcal{H}_1(g^{y_j z_i})$ , where  $\mathcal{H}_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$  is a hash function and  $Y_j$  is the sever  $j$ 's public key as listed in  $\vec{Y}$ .

**Step 3:** Client  $i$  computes his initial linkage tag  $T_0^i = h_i^{\prod_{k=1}^m s_k}$  using his per-round generator  $h_i$  as listed in  $\vec{H}$  and the secret exponents shared with all servers. Then, for each server  $1 \leq j \leq m$ ,  $i$  computes  $\vec{S} = (S_0, \dots, S_m)$ , a set of commitments to a secret  $s_j$  he shares with each server  $j$ :  $S_j = g^{\prod_{k=1}^j s_k}$  such that  $S_0 = g$ ,  $S_1 = g^{s_1}$ ,  $\dots$ ,  $S_m = g^{\prod_{k=1}^m s_k}$ . Finally, client  $i$  sets  $S = (Z_i, \vec{S})$ .

**Step 4:** Now, client  $i$  proves that (i) he correctly followed the protocol, that is his initial linkage tag  $T_0^i$  is correctly constructed using  $h_i$  and  $s = \prod_{k=1}^m s_k$ , and (ii) he belongs to the group  $G$  because he knows *some* private key  $x$  that corresponds to *some* public key  $X \in \vec{X}$ . To do so,  $i$  runs an interactive proof of knowledge as described in Section 4.3.7. The client's proof  $\text{PK}^{\text{client}_i}$  looks as follows.

$$\text{PK}\{(x_i, s) : \{\vee_{k=1}^n (X_k = g^{x_k} \wedge S_m = g^s \wedge T_0 = h_k^s)\}\}$$

**Step 5:** Client  $i$  securely erases each secret  $s_j$  and  $z_i$ . Finally, client  $i$  creates and sends to an arbitrarily chosen server  $j$  his message  $M_i^0 = (C, S, T_0, P_0)$ , where  $C = (G, \vec{R}, \vec{H}, p, g)$  is the authentication context  $i$  used,  $S = (Z_i, S_0, \dots, S_m)$  is the client's ephemeral public key and the set of client's commitments,  $T_0^i$  is the initial linkage tag,  $P_0$  is the client's proof.

### 4.3.6 Servers' Protocol

All servers collectively process a client  $i$ 's authentication message  $M_i^0$  and at the end of this process either reject  $i$ 's authentication request or accepts it as output  $i$ 's final linkage tag  $T_f^i$ . A client  $i$  arbitrarily chooses some server  $j$  to whom he sends  $M_i^0$ . We denote that server  $j$  as server 1, since  $j$  is the first server to process  $i$ 's request, and denote server 2 as  $j + 1$  and finally the last server  $m$  as  $j - 1$ . This defines a unique order based on the list of server in  $\vec{Y}$  in which each server processes  $i$ 's message.

The first server to process the client's authentication requests receives the message  $M_i^0 = (C, S, T_0, P_0)$ . Then, each server  $j$  creates a message  $M_j = (M_{j-1}, T_j, P_j)$  to pass to the next server in sequence, where  $M_{j-1}$  is the authentication message  $M_i^0$  (if  $j = 1$ ) and the message received from the previous server (if  $j > 1$ ),  $T_j$  is the linkage tag produced by  $j$  and  $P_j$  is the proof produced by  $j$ . Each  $M_j$  for  $j > 1$  consists of all previous messages such that each server  $j$  can verify all messages produced thus far (including the client's original message  $M_i^0$ ).

Each server  $j$  performs the followings steps to create  $M_j$ .

**Step 1:** Server  $j$  checks the incoming message  $M_{j-1}$  and rejects it the message is valid. Then,  $j$  checks the proof of correctness of the previous servers' computations (unless  $j = 1$ ) as well as the client's proof  $P_0$ . Server  $j$  proceeds only if all proofs are valid, and aborts otherwise.

**Step 2:** First, server  $j$  reconstructs the secret  $s_j$  he shares with the client as  $s_j = \mathcal{H}_1(Z^{y_j}) = \mathcal{H}_1(g^{y_j z})$ . Then,  $j$  verifies the client's commitments  $S_{j-1}$  and  $S_j$  against  $s_j$ . That is, the server checks that  $S_j = S_{j-1}^{s_j}$ . If yes, then  $j$  proceeds to Step 3 and if not, then  $j$  reveals  $s_j$  together with a proof that he computed it

correctly based on the client's commitment  $Z$  and his public key  $Y_j$  as described in Section 4.3.9. In such a case, server  $j$  produces the following proof  $\text{PK}_2^{\text{server}}$ .

$$\text{PK}\{(y_j) : (Z_{s_j} = Z^{y_j} \wedge Y_j = g^{y_j})\}.$$

Server  $j$  creates and sends to the next server his message  $M_j = M_{j-1}, T_j = 0, P_j = \text{PK}_2^{\text{server}}$ .

**Step 3:** Server  $j$  computes his intermediate linkage tag  $T_j = (T_{j-1})^{(r_j)(s_j^{-1})}$  using his per-round secret  $r_j$  and a multiplicative inverse of the shared secret  $s_j$  which results in a tag  $T_j = h_i^{\prod_{k=1}^j r_k \prod_{k=j+1}^m s_k}$ . Now,  $j$  produces a non-interactive proof  $P_j$  of correctness as described in Section 4.3.8. The server proves that he correctly computed the new tag  $T_j$  with respect to the server's per-round commitment  $R_j$  and the shared secret  $s_j$ . Server  $j$  produces  $\text{PK}_1^{\text{server}}$  as follows.

$$\text{PK}\{(r_j, s_j) : T_{j-1}^{r_j} = T_j^{s_j} \wedge R_j = g^{r_j} \wedge S_j = S_{j-1}^{s_j}\}$$

**Step 4:** Finally, server  $j$  securely erases  $s_j$ , forms his outgoing message  $M_j = (M_{j-1}T_j, P_j = \text{PK}_1^{\text{server}})$ , and sends  $M_j$  to server  $j + 1$  if  $j < m$ , or to all servers if  $j = m$ .

**Step 5:** Server  $j$  securely erases his per-round secrets  $r_j$  upon a completion of a round, that is when the authentication context  $C$  expires.

After a successful completion of the protocol, all servers learn a final linkage tag  $T_f = h_i^{\prod_{k=1}^m r_k}$ . The tag only depends on the client's per round generator  $h_i$  and a product of all servers' per-round secrets  $r_j$ , regardless of the initial linkage tag  $T_0$ . Thus, a client can obtain only one linkage tag per round.

### 4.3.7 Client's Proof $PK^{client_i}$

Each clients  $\hat{i}$ 's authentication message  $M_{\hat{i}}^0$  includes the following proof of knowledge  $P_0$ :

$$PK\{(x_i, s) : \{\vee_{k=1}^n (X_k = g^{x_k} \wedge S_m = g^s \wedge T_0 = h_k^s)\}\}$$

In this proof, the client  $\hat{i}$  proves that he either knows a private key  $x_1$  and his tag  $T_0$  is correct, or that he knows  $x_2$  and  $T_0$  is correct, including an “OR” statement for each private key included in  $\vec{X}$ . Because  $\hat{i}$  knows only one private key, namely  $x_i$ , he simulates the “OR” statements for all other private keys in a way that will convince the servers that the authenticating client knows one private key and the tag is properly formed. More specifically, client  $\hat{i}$  proves that that (i) client  $\hat{i}$ 's linkage tag  $T_0$  is created with respect to his per-round generator  $h_i$ , (ii)  $S_m$  is a proper commitment to  $s = \prod_{k=1}^m s_k$ , the product of all secrets that  $\hat{i}$  shares with the servers, and (iii) client  $\hat{i}$ 's private key  $x_i$  corresponds to one of the public keys included in the group definition  $G$ .

**Prover's Steps** The prover, a client  $\hat{i}$  holding private key  $x_i$  and  $s$  performs the following step to calculate  $P_0 = P$ :

1. Choose  $w_1, \dots, w_n$  such that  $w_i = 0$  and  $w_i \in_R \mathbb{Z}_q^*$  for  $i \neq \hat{i}$ , and choose  $v_{1.0}, v_{1.1}, \dots, v_{n.0}, v_{n.1} \in_R \mathbb{Z}_q^*$ . For each client  $i \in G$ , compute commitments  $t_{i.0} = X_i^{w_i} g^{v_{i.0}}$ ,  $t_{i.10} = S_m^{w_i} g^{v_{i.1}}$ , and  $t_{i.11} = T_0^{w_i} h_i^{v_{i.1}}$ . Set  $t = (t_{1.0}, t_{1.10}, t_{1.11}, \dots, t_{n.0}, t_{n.10}, t_{n.11})$  and send it to an arbitrarily chosen server.
2. Upon receiving the client's commitments, the servers collectively generate a random challenge  $c_s$  (as described in Section 4.7.4) and send  $c_s$  back to the client.

3. Compute  $c = (c_1, \dots, c_n)$  as:

$$c_i = \begin{cases} c_s - \sum_{k=1}^n w_k & \text{for } i = \hat{i} \\ w_i & \text{otherwise} \end{cases}$$

Compute responses  $r = (r_{1.0}, r_{1.1}, \dots, r_{i.0}, r_{i.1})$  as follows. Let  $x_{i.0} = x_{i.1} = 0$  for all  $i \neq \hat{i}$ , let  $x_{\hat{i}.0} = x_{\hat{i}}$ , and let  $x_{\hat{i}.1} = s$ . Compute  $r_{i.k} = v_{i.k} - c_i x_{i.k}$  for all  $1 \leq i \leq n$  and  $k \in \{0, 1\}$ . Set  $P = (c_s, t, c, r)$ .

**Verifier's Steps** The verifier, one of the servers, performs the following steps to verify the proof.

1. Check the commitments  $t_{i.0} \stackrel{?}{=} X_i^{c_i} g^{r_{i.0}}$ ,  $t_{i.10} \stackrel{?}{=} S_m^{c_i} g^{r_{i.1}}$ , and  $t_{i.11} \stackrel{?}{=} T_0^{c_i} h_i^{r_{i.1}}$ , for all  $1 \leq i \leq n$ .
2. Check the challenge  $c_s \stackrel{?}{=} \sum_{i=1}^n c_i$ .

#### 4.3.8 Server's Proof: Proving Correctness of its Work

After processing an incoming tag  $T_{j-1}$ , each server  $j$  must prove the correctness of its computations. That is, a server produces a proof of knowledge  $\text{PK}_1^{\text{server}_j}$  that he created the tag  $T_j$  according to the protocol specification. That is,  $j$  proves that he (i) correctly recovered the shared secret  $s_j$ , (ii) used the correct per-round secret  $r_j$  with respect to  $R_j \in \vec{R}$ , and (iii) correctly removed  $s_j$  and added  $r_j$  to the tag.

$$\text{PK}\{(r_j, s_j) : T_{j-1}^{r_j} = T_j^{s_j} \wedge R_j = g^{r_j} \wedge S_j = S_{j-1}^{s_j}\}$$

Server  $j$  can generate such a proof if it knows  $r_j$  and  $s_j$ . Each honest server knows its own per-round secret  $r_j$ , and the secret  $s_j$  that relates  $S_j$  to  $S_{j-1}$ , otherwise if  $j$

were unable to reconstruct a correct  $s_j$ , then he would have exposed the client by producing a proof  $\text{PK}_2^{\text{server}_j}$  and would have never produced his tag  $T_j$ .

**Prover's Steps** The prover, server  $j$  holding  $s_j$  and  $r_j$ , performs the following steps to create  $P_j = P$ .

1. Choose  $v_1, v_2 \in_R \mathbb{Z}_q^*$ . Calculate  $t_1 = T_{j-1}^{v_1} T_j^{-v_2}$ ,  $t_2 = g^{v_1}$ ,  $t_3 = S_{j-1}^{v_2}$ .
2. Calculate  $c = \mathcal{H}_2(T_{j-1}, T_j, R_j, g, S_j, S_{j-1}, t_1, t_2, t_3)$ , where  $\mathcal{H}_2 : \{0, 1\}^* \rightarrow \mathbb{Z}_p$  is a hash function.
3. Calculate  $r_1 = v_1 - cr_j$  and  $r_2 = v_2 - cs_j$ .
4. Set  $P = (t_1, t_2, t_3, c, r_1, r_2)$ .

**Verifier's Step** The verifier, another server, upon receiving  $P_j$  can verify the proof as follows.

1. Reconstruct commitments  $t'_1 = T_{j-1}^{r_1} T_j^{-r_2}$ ,  $t'_2 = g^{r_1} R_j^c$ ,  $t'_3 = S_{j-1}^{r_2} S_j^c$ .
2. Check  $c \stackrel{?}{=} \mathcal{H}_2(T_{j-1}, T_j, R_j, g, S_j, S_{j-1}, t'_1, t'_2, t'_3)$ .

### 4.3.9 Server's Proof: Exposing a misbehaving client

To create a tag  $T_j$ , server  $j$  needs to reconstruct and then remove the secret  $s_j$  it shares with the client from the incoming tag  $T_{j-1}$ . Server  $j$  calculates  $s_j = \mathcal{H}_1(Z^{y_j})$  using the client's commitment  $Z$  and its own private key  $y_j$ , and verifies that the recovered secret is correct by checking  $S_j = S_{j-1}^{s_j}$ . If the recovered secret is not correct, then  $j$  exposed the client as dishonest by providing a proof  $\text{PK}_2^{\text{server}_j}$  to other servers. To do so, the server reveals the secret  $s_j$  it computed and  $Z_{s_j} = Z^{y_j}$ , the preimage of  $s_j$  under  $\mathcal{H}_1$ . Then, server  $j$  prepares a proof that he (*i*) used his

private key  $y_j$  that corresponds to a public key  $Y_j \in \vec{Y}$ , and (ii) correctly computed  $Z_{s_j}$  by raising the client's commitment  $Z$  to his private key  $y_j$ . Server  $j$  prepares the following proof of knowledge:

$$PK\{(y_j) : (Z_{s_j} = Z^{y_j} \wedge Y_j = g^{y_j})\}.$$

After receiving and verifying  $PK_2^{server_j}$ , each server can reconstruct  $s_j = \mathcal{H}_1(Z_{s_j})$ , check that indeed  $S_j \neq S_{j-1}^{s_j}$ , and

**Prover's Steps** The prover, server  $j$  holding a private key  $y_j$ , performs the following steps and obtains  $P_j = (Z_{s_j}, P)$ :

1. Choose  $v \in_R \mathbb{Z}_q^*$ . Calculate  $t_1 = Z^v$  and  $t_2 = g^v$ .
2. Calculate  $c = \mathcal{H}_2(Z_{s_j}, Z, Y_j, g, t_1, t_2)$ .
3. Calculate  $r = v - cy_j$
4. Set  $P = (t_1, t_2, c, r)$ .

**Verifier's Steps** The verifier, either a server or the client, upon receiving  $P_j$  can verify the proof as follows:

1. Reconstruct commitments  $t'_1 = Z^r Z_{s_j}^c$  and  $t'_2 = g^r Y_j^c$ .
2. Check  $c \stackrel{?}{=} \mathcal{H}_2(Z_{s_j}, Z, Y_j, g, t'_1, t'_2)$ .

## 4.4 Extensions

In this section we describe several possible extensions of our main protocol. First, we discuss ideas for improving DAGA's performance, then we discuss trading deniability

for verifiability, show how to give the servers the ability to collectively revoke a client’s anonymity, show how to make DAGA secure on full clients’ and servers’ key exposure, and lastly we present a variant of DAGA in which the client has a chance to inspect his linkage tag before it is revealed to the servers.

#### 4.4.1 Improving Efficiency

Currently, the computation and communication overhead of DAGA grows linearly in the number of members in a group  $G$ . Ideally, we would like to improve the efficiency from  $\mathcal{O}(n)$  to  $\mathcal{O}(1)$  to make it independent from the group size  $n$ .

One possibility is to use a cryptographic accumulator [67] (or a dynamic accumulator [34] to retain support for evolving groups) that makes it possible to accumulate multiple values into a single one such that for each accumulated value there is a proof that the value was correctly incorporated. Therefore, instead of using a 1-out-of- $n$  “OR” proof, we could first accumulate all public keys and then prove that a client’s public key is indeed a part of the the resulting short accumulator. Similar ideas were used to design a short linkage ring signature scheme [10], for example. Another possibility is to use more efficient proofs of knowledge [36] and new, efficient batching verification techniques for proofs of partial knowledge [94, 95, 145]. We fully expect to obtain a much efficient protocol using the outlined ideas.

#### 4.4.2 Trading Deniability for Verifiability

DAGA offers a strong zero-knowledge notion of deniability; the protocol does not leave a ‘paper trail’ that one could use to prove that *some*, and therefore at least one, member participated in the protocol. DAGA achieves deniability by using an interactive rather than non-interactive zero-knowledge proof on the client’s side.

An interactive proof is not transferable and only “convinces” the party directly involved in the proof. In a non-interactive proof, the verifier is replaced with a hash function [69] to create an unpredictable challenge that a prover cannot anticipate in advance. Therefore, anyone can verify the non-interactive proof, even after the protocol’s completion. This property, while useful, goes against the notion of deniability we set out to achieve. However, certain applications might benefit from the transferability of the proof that would allow for a *third-party verifiability* that some user or a certain number of users indeed authenticated.

Consider an anonymous voting scenario, where voters want to remain anonymous but wish for a third-party verifiable proof (independently of the election results) that a specific number of voters participated. A small change to DAGA, changing the client’s proof from interactive to non-interactive, easily achieves this goal and each voter’s authentication message  $M_0$  becomes such a proof.

Interestingly, trading deniability for verifiability does not affect other properties, specifically forward anonymity and proportionality. Moreover, DAGA still retains a weaker notion of plausible deniability: since DAGA is anonymous and a group  $G$  can be created without the listed members’ participation or knowledge, any member can plausibly deny participating in the protocol.

### 4.4.3 Optional Anonymity Revocation

DAGA provides clients with a strong notion of anonymity. However, the ability to revoke a client’s anonymity might be a desirable feature but only if it is done carefully so that the client’s anonymity is not inadvertently or maliciously compromised.

Any client’s anonymity can be revoked if each server  $j$  reveals his per-round secret  $r_j$ , in which case the anonymity of all clients is compromised, or each server reveals his secret  $s_j$  shared with a client in question, breaking the rule of retaining private

input secret, however. Therefore, we wish for a protocol which explicitly allows for anonymity revocation.

To achieve this goal, we use a threshold version of the ElGamal encryption scheme to encrypt the client’s ephemeral private key  $z_i$  under a public key that is a product of all servers’ commitments to their per-round secrets  $r$  (if we want to limit anonymity revocation to the lifetime of an authentication context) or under a public key that is a product of all servers’ long-term public keys (if we want the ability to revoke clients’ anonymity at any point).

After encrypting his ephemeral key  $z_i$  under a shared public key  $K_{all}$  of all servers, a client  $i$  produces a modified version of the  $\text{PK}^{client_i}$  proof which includes a proof that  $E_{K_{all}}(z_i)$  is an encryption of an element committed to as  $Z_i$ , using a standard technique of proving a property of a ciphertext from [35]. To reveal a client’s identity, all servers collectively decrypt  $E_{K_{all}}(z_i)$ , retrieve  $z_i$  and use it to recover all secrets the client shares with the servers’ finally recovering a per-round generator, which corresponds to a unique client  $i$  as defined by  $\vec{H}$ .

This modification does not affect the properties offered by DAGA. Specifically, the anonymity and forward anonymity properties are still guaranteed, unless explicitly revoked, assuming that there is always one honest and never compromised server.

#### 4.4.4 Secure on Full Key Exposure

Currently, the forward anonymity property holds as long as the honest server’s private key is protected. If the long term private key  $z_h$  of the honest server is known, an adversary who controls all other servers can recover the ephemeral secret shared with a client  $i$  and calculate  $s_h$ . Then, if the adversary has access to the previous authentication messages that include the initial linkage tags, the adversary can triv-

ially calculate  $T_0'^i = h_i^{s_1 \dots s_m}$  for every  $h_i \in \vec{H}$  and compare with the initial linkage tags. Knowing the association of a client's identity with a per-round generator, the adversary breaks the anonymity and forward anonymity of every client for whom he finds a matching tag.

We can avoid this (rather unlikely but not impossible) attack by adding a server-side per-round randomness into the secret a client shares with each server. This way even if the adversary compromises the server's private key, the additional secret included in  $s_h$  has been forgotten.

To do so, we extend the authentication context  $C$  to include an additional a vector  $\vec{A} = A_1, \dots, A_m$ , where  $A_j = g^{a_j}$  and modify Step 2 of the client's protocol described in Section 4.3.5 as follows. For each server  $j$ ,  $i$  uses both  $A_j$  and  $Y_j$  to compute a shared secret exponent  $s_j = \mathcal{H}_1(A_j^{z_i}, Y_j^{z_i}) = \mathcal{H}_1(g^{a_j z_i}, g^{y_j z_i})$ . Then, each server  $j$  recovers  $s_j$  as  $\mathcal{H}_1((Z^{a_j}, Z^{y_j}) = \mathcal{H}_1(g^{a_j z_i}, g^{y_j z_i})$ .

The protocol works as follows.

1. Client  $i$  encrypts his ephemeral key  $z_i$  under  $R_S = \prod_{j=1}^m R_m$ : as follows:  $i$  chooses  $\ell \in_R \mathbb{Z}_q^*$  and calculates  $E_{R_S}(z_i) = (A = g^\ell, B = z_i R_S^\ell)$ .
2. Then, client  $i$  creates a modified version of the  $\text{PK}^{client} i$  proof appending to it a proof that  $E_{R_S}(z_i)$  is an encryption of an element committed to as  $Z_i$  using a technique of proving a property of a ciphertext from [35].

In order to reveal an identity of a client, the servers perform the following steps.

1. Each server  $j$  calculates and publishes  $A_j = A^{r_j} = g^{\ell r_j}$  as well as a proof of knowledge that  $DL(A_j) = DL(R_j)$ , that is  $j$  correctly computed  $A_j$  by raising it to its private key  $y_j$ .
2. All servers retrieve  $z_i = B(\prod_{j=1}^m A_m)^{-1}$ .

3. For each server  $i \in G$ ,  $j$  calculates  $s_i = \mathcal{H}_i(Y)x^{z_i}$ .
4. The client's identity is revealed by removing all secrets the client shares with the servers from a particular tag was created to:  $(T_0)^{\prod_{i=1}^m s_i^{-1}}$  and deciding which generator the result is equal to.

$$PK\{(x_i, s, z_i) : \{\bigvee_{k=1}^n (X_k = g^{x_k} \wedge S_m = g^s \wedge T_0 = h_k^s) \wedge B = z_i g^{\sum_{j=0}^m r_j}\}$$

While we recognize that the client's identity can be revealed if each server reveals their secret shared with the client, we wish to be able to do so in a way that guarantees that a client is aware of this possibility (by creating a modified proof of knowledge). This modification does not affect the properties offered by DAGA. Specifically, the anonymity and forward anonymity properties are still guaranteed assuming that there is always one honest and never compromised server.

#### 4.4.5 Delayed Revealing of Final Linkage Tags

After a successful authentication, the servers immediately learn the client's final linkage tag. However, the client might want to have an opportunity to "inspect" the tag first before finishing authentication. This way a client could check if the servers already seen such a tag by looking it up in a server-published list of the linkage tags seen in a particular authentication context  $C$  thereby avoiding the potential risk of unintentionally trying to authenticate twice in a linkage context, for example.

This can be easily accomplished by delaying the removal of the client-side secret  $s$  from the linkage tag until the client can verify  $T_0 = h_i^{sr}$ . That is, a client allows the servers to incorporate their per-round secrets  $r$ , then verifies the resulting tag, and if the tag is correct, he removes his secret  $s$ , proves in zero-knowledge that he did so correctly and sends the final tag back to the servers.

To do so, a client creates the initial linkage tag as before,  $T_0 = h_i^s$ , however now  $s$  is a single secret known by the client, not a product of all secrets  $i$  assigns to the servers, and  $i$  produces a proof  $PK^{client_i}$  as before.

Upon receiving the client's message  $M_0$ , the servers iteratively incorporate their per-round secrets  $r_j$  as before, but this time *without* removing the client's secret, finally yielding a final linkage tag  $T_f = h_i^{s \prod_{k=1}^m r_k}$ . Each server  $j$  prepares a simplified proof of its correctness:

$$PK\{(r_j) : T_j = T_{j-1}^{r_j} \wedge R_j = g^{r_j}\}$$

After all servers process the tag, the last server sends  $T_j$  and the proofs of its correctness back to the client, who can verify the proofs and calculate the client's final linkage  $T_f = (T_j)^{-s}$ . This way the client has a chance to inspect the tag *before* making it available to the servers. If the client decides to complete the authentication, he can prove the correctness of  $T_f$  with respect to  $T_j$  as follows

$$PK\{(s) : T_f = T_j^s \wedge S = g^s\}$$

While this approach gives the client more control over his authentication requests, it requires an additional communication round, but might be suitable for applications where clients are limited to a certain number of authentications within a certain authentication context and authenticating more than the allowed number of times has negative consequences.

## 4.5 Applications

DAGA may be useful in many conventional applications such as online surveys, voting, subscription services but also whistleblowing and access to sensitive resources. DAGA is most suitable for authentication into well-defined, closed groups of manageable size, and when guarantees of deniability and forward security are needed. Below we overview several non-conventional applications we envision for DAGA.

### 4.5.1 Distributing Keys for Group Anonymity Systems

Most anonymity systems fall into two categories: mix networks [40] (mix-nets) mask the identity of the sender by forwarding messages through multiple relays, and Dining Cryptographers networks [39] (DC-nets) leverage secrets exchanged within a well-defined group of members to anonymize messages. While mix-nets based systems (e.g., Tor [58]) are efficient, they do not provide unconditional anonymity and traffic analysis resistance as DC-nets based systems (e.g., Dissent [51, 192], Herbivore [79, 168]) do.

To provide *accountability* – the ability to identify and expel members that attempt to disrupt group communication – Dissent requires each member to have a long-term signing key. This key, if well-known, links the intermediate output of the protocol to the key’s owner, and links the protocol’s entire output to a particular group of keys. If a client’s identity is defined by a long-term non-anonymous key pair, a compromise of the client’s private key could retroactively compromise the user’s anonymity in all past exchanges.

Therefore, an anonymity system such as Dissent can leverage DAGA to set up ephemeral pseudonyms (signing keys) for participating group members, breaking the link between the client’s long-term key (and identity) and the anonymous exchanges,

while ensuring fairness via DAGA’s proportionality property. Additionally, we can achieve a larger anonymity set and plausible deniability for the anonymous communication if we draw ephemeral pseudonyms from a much larger group of members than those who actively participate: e.g., many “members” may be conscripted into the group without their participation or knowledge.

### 4.5.2 Anonymous Voting with Deniability

DAGA may lend itself to certain forms of anonymous voting. Anonymity (to preserve voter’s privacy) and proportionality (to enforce one-voter one-vote rule) is generally required in any anonymous voting scheme. Many anonymous e-voting schemes [3, 97, 107, 119] provide additional properties such as coercion-resistant and receipt-freeness, which offer a weaker notion of deniability. DAGA’s deniability property ensures that once an election has ended, the resulting communication transcript leaves no verifiable proof that the vote even occurred. DAGA may thus be attractive for voting in a “dissident forum” under repression from an authoritarian regime, for example.

### 4.5.3 Secure Access to Sensitive Resources

We can envision using DAGA to distribute access tokens to resources, in particular to sensitive resources, in a way that gives access to a certain group of clients while providing deniability of ever requesting those sensitive resources.

Additionally, because DAGA provides proportionality, the servers can keep track of requests made by a particular anonymous user based on his final linkage tag. This gives the servers the ability to limit access to resources as desired (to one or  $k$  times) without exposing a client who inadvertently makes  $k + 1$  requests as done in many

*k*-show credential systems [117]

#### 4.5.4 Server-provided Signatures

After a client has been successfully authenticated as a unique group member, he might request that the servers collectively perform a specific action on his behalf, for example to sign a message anonymously and deniably.

This might be accomplished in several ways. Servers might sequentially sign a message as they process the client's tag provided that the client's proof is valid. At the end of a successful authentication, the servers might endorse a collective ephemeral signing key and produce a signature on the provided message. Alternatively, a client-defined subset of servers might issue a threshold DSS signature [76,77].

#### 4.5.5 Supporting Anonymous Federated Login

Crypto-Book [128] provides for a privacy-preserving and accountable digital identities. It leverages the existing digital identity providers, such as Facebook or Twitter, and the use of public-key encryption and linkable ring signatures. Linkable ring signatures [125,126] allow a group member to anonymously sign a message in a way that hides his identity but allows others to verify that the signature was produced by a group member and to link all future signatures as coming from the same, anonymous member.

DAGA can be used in place of any linkable ring signature scheme as it provides the same functionality (anonymity and linkability) while adding deniability and forward security.

## 4.6 Security Analysis

This section describes and analyzes DAGA's security properties.

### 4.6.1 Assumptions

We assume that the Discrete Logarithm (DL) and Decisional Diffie-Hellman (DDH) assumptions hold, that is, any probabilistic polynomial algorithm solves the DL problem and the DDH problem respectively only with a negligible probability [22]. DAGA assumes a cyclic multiplicative group  $\mathcal{G}$  of prime order  $q$ , where  $p = 2q + 1$ , where the Discrete Logarithm and Decisional Diffie-Hellman assumptions hold in  $\mathcal{G}$  [22].

### 4.6.2 Properties of the Proofs of Knowledge

In this section we show that the client's and server's proofs of knowledge have the properties of completeness, special soundness and special honest-verifier zero-knowledge [53]. Note that  $\mathcal{H}$  is modeled as a random oracle.

**Definition 9** ( $\Sigma$ -protocol [53]). *A protocol  $\mathcal{P}$  is said to be a  $\Sigma$ -protocol for relation  $R$  if:*

- *$P$  is of the 3-move form, and if  $P, V$  follow the protocol, the verifier always accepts (completeness).*
- *From any  $x$  and any pair of accepting conversations on input  $x$ ,  $(t, c, r)$ ,  $(t, c', r')$  where  $c \neq c'$ , one can efficiently compute  $w$  such that  $(x, w) \in R$  (special soundness).*
- *There exists a polynomial time simulator  $S_{zk}$ , which on input  $x$  and a random  $e$  outputs an accepting conversation of the form  $(t, c, r)$ , with the same proba-*

bility distribution as conversations between the honest  $P, V$  on input  $x$  (special honest-verifier zero-knowledge).

### **PK<sup>client</sup>: Client's Proof of Knowledge**

*Completeness.* If a prover and verifier faithfully follow the protocol on common input  $C$  and prover's private input  $x$ , then the verifier always accepts the proof generated by the prover. Assume that the proof  $P_0$  is generated by a client  $i$  who knows a solution  $x_i$ , his private key, and  $s$ , the product of all secrets shared with the servers. The verifier checks the commitment  $t$  and the challenge  $c_s$ . For client  $i$ , the commitment verification proceeds as follows.

$$\begin{aligned}
 t_{i,0} &\stackrel{?}{=} X_i^{c_i} g^{r_{i,0}} & t_{i,10} &\stackrel{?}{=} S_m^{c_i} g^{r_{i,1}} \\
 g^{v_{i,0}} &= X_i^{c_i} g^{r_{i,0}} & g^{v_{i,1}} &= S_m^{c_i} g^{r_{i,1}} \\
 g^{v_{i,0}} &= g^{c_i x_i} g^{v_{i,0} - c_i x_i} & g^{v_{i,1}} &= h_i^{c_i s} g^{v_{i,1} - c_i s} \\
 g^{v_{i,0}} &= g^{c_i x_i} g^{v_{i,0}} g^{-c_i x_i} & g^{v_{i,1}} &= g^{c_i s} g^{v_{i,1}} g^{-c_i s} \\
 g^{v_{i,0}} &= g^{v_{i,0}} & g^{v_{i,1}} &= g^{v_{i,1}}
 \end{aligned}$$

$$\begin{aligned}
 t_{i,11} &\stackrel{?}{=} T_0^{c_i} h_i^{r_{i,1}} \\
 h_i^{v_{i,1}} &= T_0^{c_i} h_i^{r_{i,1}} \\
 h_i^{v_{i,1}} &= h_i^{c_i s} h_i^{v_{i,1} - c_i s} \\
 h_i^{v_{i,1}} &= h_i^{c_i s} h_i^{v_{i,1}} h_i^{-c_i s} \\
 h_i^{v_{i,1}} &= h_i^{v_{i,1}}
 \end{aligned}$$

For every client  $j \neq i$ , the commitment verification proceeds as follows.

$$\begin{aligned} t_{j,0} &\stackrel{?}{=} X_j^{c_j} g^{r_{j,0}} & t_{j,10} &\stackrel{?}{=} S_m^{c_j} g^{r_{j,1}} \\ X_j^{w_j} g^{v_{j,0}} &= X_j^{c_j} g^{r_{j,0}} & S_m^{w_j} g^{v_{j,1}} &= S_m^{c_j} g^{r_{j,1}} \\ X_j^{w_j} g^{v_{j,0}} &= X_j^{w_j} g^{v_{j,0}} & S_m^{w_j} g^{v_{j,1}} &= S_m^{w_j} g^{v_{j,1}} \end{aligned}$$

$$\begin{aligned} t_{j,11} &\stackrel{?}{=} T_0^{c_j} h_j^{r_{j,1}} \\ T_0^{w_j} h_j^{v_{j,1}} &= T_0^{c_j} h_j^{r_{j,1}} \\ T_0^{w_j} h_j^{v_{j,1}} &= T_0^{w_j} h_j^{v_{j,1}} \end{aligned}$$

The challenge verification proceeds as follows.

$$\begin{aligned} c_s &\stackrel{?}{=} c \\ c_s &= \sum_{i=1}^n c_i \\ c_s &= \sum_{i=1}^k w_k + c_s - \sum_{i=1}^k w_k \\ c_s &= c_s \end{aligned}$$

As shown above, the verifier will be able to successfully verify the commitments  $t$  and  $c$  based on the challenge  $c_s$ , hence, the proof is complete.

*Special Soundness.* Given common input  $i$  and two transcripts of successful conversations  $(t, c = (c_1, \dots, c_n), r = (r_1, \dots, r_n))$  and  $(t, c' = (c'_1, \dots, c'_n), r' = (r'_1, \dots, r'_n))$ , where  $c \neq c'$ , client  $i$ 's private input  $x_i$  and  $s$  can be successfully computed as follows:

$$\begin{aligned} r_{i,0} &= v_{i,0} - c_i x_i & r_{i,1} &= v_{i,1} - c_i s \\ r'_{i,0} &= v_{i,0} - c'_i x_i & r'_{i,1} &= v_{i,1} - c'_i s \\ x_i &= \frac{r_{i,0} - r'_{i,0}}{c_i - c'_i} & s &= \frac{r_{i,1} - r'_{i,1}}{c_i - c'_i} \end{aligned}$$

*Special Honest Verifier Zero-Knowledge.* There exists a polynomial time simulator  $S_{zk}$ , which on common input  $I$  generates a conversation transcript that is computationally indistinguishable from a transcript generated by a prover. The simulator  $S_{zk}$  works as follows:

1. Choose  $w_1, \dots, w_n \in_R \mathbb{Z}_q^*$  for all  $i$  and  $v_{1,0}, v_{1,1}, \dots, v_{n,0}, v_{n,1} \in_R \mathbb{Z}_q^*$  for all  $i$ .  
 Compute commitments  $t_{i,0} = X_i^{w_i} g^{v_{i,0}}$ ,  $t_{i,10} = S_m^{w_i} g^{v_{i,1}}$ , and  $t_{i,11} = T_0^{w_i} h_i^{v_{i,1}}$  for each  $i$ .  
 Set  $t = (t_{1,0}, t_{1,10}, t_{1,11}, \dots, t_{n,0}, t_{n,10}, t_{n,11})$ ,
2. Compute  $c_s = \sum_{k=1}^n w_k$ . Set  $c_i = w_i$  for each  $i$  and set  $c = (c_1, \dots, c_n)$ .
3. Compute responses  $r = (r_{1,0}, r_{1,1}, \dots, r_{i,0}, r_{i,1})$  using  $r_{i,k} = v_{i,k}$  for all  $1 \leq i \leq n$  and  $k \in \{0, 1\}$ .
4. Set  $P = (c_s, t, c, r)$ .

The verification of the proof works as follows for every  $i$ :

$$\begin{aligned}
 t_{i,0} &\stackrel{?}{=} X_i^{c_i} g^{r_{i,0}} & t_{i,10} &\stackrel{?}{=} S_m^{c_i} g^{r_{i,1}} \\
 X_i^{w_i} g^{v_{i,0}} &= X_i^{c_i} g^{r_{i,0}} & S_m^{w_i} g^{v_{i,1}} &= S_m^{c_i} g^{r_{i,1}} \\
 X_i^{w_i} g^{v_{i,0}} &= X_i^{w_i} g^{v_{i,0}} & S_m^{w_i} g^{v_{i,1}} &= S_m^{w_i} g^{v_{i,1}} \\
 t_{i,11} &\stackrel{?}{=} T_0^{c_i} h_i^{r_{i,1}} & c_s &\stackrel{?}{=} \sum_{k=1}^n c_k \\
 T_0^{w_i} h_i^{v_{i,1}} &= T_0^{c_i} h_i^{r_{i,1}} & & \\
 T_0^{w_i} h_i^{v_{i,1}} &= T_0^{w_i} h_i^{v_{i,1}} & \sum_{k=1}^n w_k &= \sum_{k=1}^n w_k
 \end{aligned}$$

**PK<sub>1</sub><sup>server</sup>: Proving correctness of its work**

*Completeness.* The verifier reconstructs the commitments as follows:

$$\begin{aligned}
t'_1 &= T_{j-1}^{r_1} T_j^{-r_2} \\
&= T_{j-1}^{v_1 - cr_j} T_j^{-(v_2 - cs_j)} \\
&= T_{j-1}^{v_1} T_{j-1}^{-cr_j} T_j^{-v_2} T_j^{cs_j} \\
&= T_{j-1}^{v_1} T_j^{-v_2} \\
&= t_1
\end{aligned}$$

$$\begin{aligned}
t'_2 &= g^{r_1} R_j^c & t'_3 &= S_{j-1}^{r_2} S_j^c \\
&= g^{v_1 - cr_j} g^{cr_j} & &= S_{j-1}^{v_2 - cs_j} S_{j-1}^{cs_j} \\
&= g^{v_1} g^{-cr_j} g^{cr_j} & &= S_{j-1}^{v_2} S_{j-1}^{-cs_j} S_{j-1}^{cs_j} \\
&= g^{v_1} & &= S_{j-1}^{v_2} \\
&= t_2 & &= t_3
\end{aligned}$$

Given that  $t'_1 = t_1$ ,  $t'_2 = t_2$  and  $t'_3 = t_3$ , we have

$$\begin{aligned}
c &\stackrel{?}{=} \mathcal{H}(T_{j-1}, T_j, R_j, g, S_j, S_{j-1}, t'_1, t'_2, t'_3) \\
c &= c
\end{aligned}$$

*Special Soundness.* Given common input  $i$  and two transcripts of successful conversations  $(t_1, t_2, t_3, c, r_1, r_2)$  and  $(t_1, t_2, t_3, c', r'_1, r'_2)$ , where  $c \neq c'$ , server  $j$ 's private

input  $r_j$  and  $s_j$  can be successfully computed as follows:

$$\begin{aligned} r_1 &= v_1 - cy_j & r_2 &= v_2 - c_i s_j \\ r'_1 &= v_1 - c'y_j & r'_2 &= v_2 - c'_i s_j \\ r_j &= \frac{r_1 - r'_1}{c - c'} & s_j &= \frac{r_2 - r'_2}{c - c'} \end{aligned}$$

*Special Honest Verifier Zero-Knowledge.* The simulator  $S_{zk}$  accepts  $h$  as input and performs the following steps to produce  $P_j = P$ :

1. Choose  $v_1, v_2 \in_R \mathbb{Z}_q^*$ .
2. Set  $c = h$ .
3. Calculate  $t_1 = T_{j-1}^{v_1} T_j^{-v_2}$ ,  $t_2 = g^{v_1} R_j^c$ ,  $t_3 = S_{j-1}^{v_2} S_j^c$ .
4. Set  $r_1 = v_1$  and  $r_2 = v_2$ .
5. Set  $P = (t_1, t_2, t_3, c, r_1, r_2)$ .

The verification of the proof works as follows:

$$\begin{aligned} t'_1 &= T_{j-1}^{r_1} T_j^{-r_2} & t'_2 &= g^{r_1} R_j^c & t'_3 &= S_{j-1}^{r_2} S_j^c \\ &= T_{j-1}^{v_1} T_j^{-v_2} & &= g^{v_1} R_j^c & &= S_{j-1}^{v_2} S_j^c \\ &= t_1 & &= t_2 & &= t_3 \end{aligned}$$

Then, we verify the challenge  $c \stackrel{?}{=} h$ , which gives  $h = h$ .

**PK<sub>2</sub><sup>server</sup>: Exposing a misbehaving client**

*Completeness.* The verifier reconstructs the commitments as follows:

$$\begin{aligned}
t'_1 &= Z^r Z_{s_j}^c & t'_2 &= g^r Y_j^c \\
&= Z^{v-cy_j} Z^{cy_j} & &= g^{v-cy_j} g^{cy_j} \\
&= Z^v Z^{-cy_j} Z^{cy_j} & &= g^v g^{-cy_j} g^{cy_j} \\
&= Z^v & &= g^v \\
&= t_1 & &= t_2
\end{aligned}$$

Given that  $t'_1 = t_1$  and  $t'_2 = t_2$ ,

$$\begin{aligned}
c &\stackrel{?}{=} \mathcal{H}(Z_{s_j}, Z, Y_j, g, t'_1, t'_2) \\
\mathcal{H}(Z_{s_j}, Z, Y_j, g, t_1, t_2) &= \mathcal{H}(Z_{s_j}, Z, Y_j, g, t'_1, t'_2)
\end{aligned}$$

*Special Soundness.* Given common input  $i$  and two transcripts of successful conversations  $(t_1, t_2, c, r)$  and  $(t_1, t_2, c', r')$ , where  $c \neq c'$ , server  $j$ 's private input  $y_j$  can be successfully computed as follows:

$$\begin{aligned}
r &= v - c_i y_j \\
r' &= v - c'_i y_j \\
y_j &= \frac{r - r'}{c - c'}
\end{aligned}$$

*Special Honest Verifier Zero-Knowledge.* The simulator  $S_{zk}$  accepts  $h$  as input and performs the following steps to produce  $P_j = (Z_{s_j}, P)$ :

1. Choose  $v \in_R \mathbb{Z}_q^*$ . Calculate  $c = h$ .

2. Calculate  $t_1 = Z^v Z_{s_j}^c$  and  $t_2 = g^v Y_j^c$ .
3. Set  $r = v$
4. Set  $P = (t_1, t_2, c, r)$ .

The verification of the proof works as follows:

$$\begin{aligned}
 t'_1 &= Z^r Z_{s_j}^c & t'_2 &= g^r Y_j^c \\
 &= Z^v Z_{s_j}^c & &= g^v Y_j^c \\
 &= t_1 & &= t_2
 \end{aligned}$$

Then, we verify the challenge as follows:

$$\begin{aligned}
 c &\stackrel{?}{=} h \\
 h &= h
 \end{aligned}$$

### 4.6.3 Completeness

We require that servers accept a properly formed authentication request from every honest client  $i$  who belongs to a group  $G$  defined by a particular authentication context  $C$ , unless the protocol is aborted because of a discovered misbehavior of some server. A client  $i$  *belongs* to a group  $G$  if he knows a private key  $x_i$  such that  $X_i = g^{x_i} \in \vec{X}$ .

**Definition 10.** *An authentication protocol offers the completeness property, if for any client  $i \in G = (\vec{X}, \vec{Y})$  who correctly follows the prescribed protocol, the servers accept  $i$ 's authentication request with an overwhelming probability.*

**Theorem 1.** *DAGA offers the completeness property.*

*Proof.* Under our assumptions, a client  $i$  belongs to a group  $G$ , is in a possession of a private key  $x_i$  such that  $X_i = g^{x_i}$  and  $X_i \in \vec{X}$ . Further, we assume that  $i$  is in possession of a well-formed authentication context  $C = (G, \vec{R}, \vec{H}, p, g)$  where  $G = (\vec{X}, \vec{Y})$  is the group definition,  $H = (h_1, \dots, h_n)$  is a set of per-round generators for clients, and  $R = (R_1, \dots, R_m)$  is server-published randomness. From the trust model it follows that all servers participate in the round and all server's but one can behave arbitrarily dishonestly.

In order to make an authentication request, a client  $i$  must prepare an authentication message  $M_i^0 = (C, S, T_0, P_0)$ , where  $C$  is the authentication context,  $S = (Z, \vec{S} = S_0, \dots, S_m)$  consists of the client's ephemeral public key and the set of client's commitments,  $T_0$  is the initial linkage tag, and  $P_0$  is a proof of correctness for  $T_0$ . We will show that client  $i$  is able to produce a valid message  $M_i^0$  and that this message will be accepted by the servers and therefore result in an accepted authentication request.

To produce a valid message  $M_i^0$ , client  $i$  needs to produce all of its components.

- *Authentication context  $C$ .* Client  $i$  obtains  $C$  before making an authentication request.
- *Ephemeral key and commitments  $S$ .* Client  $i$  can produce  $S$  since it depends on  $i$ 's randomly chosen key and information included in  $C$ .
- *Linkage tag  $T_0$ .* The tag  $T_0 = h_i^{s_1 s_2 \dots s_m}$  depends on  $i$ 's publicly available per-round generator  $h_i$  and secrets  $s_1, s_2, \dots, s_m$  created in the previous step.
- *Proof  $P_0$ .* The proof  $P_0$  depends on the knowledge of  $x_i$  and  $s$ . Based on the completeness property of the underlying proof of knowledge, a client  $i$  can always produce a valid proof if he's in possession of the private information

he tries to prove knowledge of. In our case,  $i$  creates a proof of knowledge  $PK(x_i, s)$ , where  $x_i$  is  $i$ 's private key and  $s = s_1 s_2 \dots s_m$ .

Therefore,  $i$  can produce a valid message  $M_i^0 = (C, S, T_0, P_0)$ . Now, we will show that this message will be accepted by the servers with an overwhelming probability and therefore result in an accepted authentication request.

After creating  $M_i^0$ ,  $i$  sends it to an arbitrarily chosen server  $j$ . Each server  $j$  verifies the message  $M_i^0$  and either (i) accepts it and produces an outgoing tag  $T_j$  and a proof  $PK_1^{server_j}$  of correctness of its own work or (ii) rejects it and produces a proof  $PK_2^{server_j(i)}$  of the client  $i$ 's misbehavior. By the soundness property of the underlying proof of knowledge  $PK_2^{server}$ , none of the servers can expose  $i$  as dishonest and therefore reject  $i$ 's authentication request, except with a negligible probability, given that  $i$ 's  $M_i^0$  is valid. By the soundness property of  $PK_1^{server}$ , none of the servers can produce a valid proof of correctness if they did not follow the protocol, except with a negligible probability. Therefore, if the protocol is not terminated and a faulty server discovered, each server produces a valid intermediate tag  $T_j$ , which results in a valid final tag  $T_j^i$ . Consequently,  $i$ 's authentication request is accepted with with an overwhelming probability.  $\square$

#### 4.6.4 Soundness

We require that DAGA is a sound authentication protocol, that is, servers only accept properly formed authentication requests from members who belong to a particular group  $G$  as defined in an authentication context  $C$ .

This means that it should offer soundness and not allow forgeries, where the notions of forgeability from [86] apply.

Typically, the soundness property of a non-anonymous authentication protocol

is defined with respect to “legitimate” users that have established credentials with a verifier. In case of DAGA, we define legitimate users as users who belong to a particular group  $G$ . Any client that possesses a correct long-lived well-known key pair associated with a non-anonymous identity can belong to any  $G$ , even without their knowledge. This is true because any member  $i$  can be listed in  $G$  if their public key is publicly available since there is no action required on the client’s side in order to be added to  $G$ .

We assume that honest clients keep their private keys secret and dishonest clients can arbitrarily share their private keys. In such a case, we note that if  $i$  is in possession of a private key  $\hat{i}$  such that  $X_{\hat{i}} = g^{\hat{i}}$  and  $X_{\hat{i}} \in \vec{X}$ , then  $i$  can successfully impersonate  $\hat{i}$  by authenticating as a legitimate client and does not violate the soundness property.

**Definition 11.** *An authentication protocol offers the soundness property if an authentication request from any client  $i \notin G$  is rejected with an overwhelming probability.*

**Theorem 2.** *DAGA offers the soundness property.*

*Proof.* Assume that a client  $i$  does not belong to a group  $G$ , that is,  $i$ ’s public key  $X_i$  is not included in  $\vec{X}$ . Therefore,  $i$  does not know any  $x_i$  such that  $X_i = g^{x_i}$  and  $X_i \in \vec{X}$ . To successfully authenticate as a member of  $G$ ,  $i$  needs to prepare a message  $M_i^0$  on behalf of some  $\hat{i} \in G$ . To do so,  $i$  must prepare a valid message  $M_i^0 = (C, S, T_0, P_0)$  without the knowledge of  $x_i$ , such that each server accepts  $i$ ’s authentication request.

We will show that  $i$  cannot produce a valid message  $M_i^0$ , except with a negligible probability, and each message  $M_i^0$   $i$  can produce will be rejected by the servers because of the invalid proof. Hence,  $i$ ’s authentication requests will be denied with an overwhelming probability.

In order to forge a message  $M_i^0$ , a client  $i$  must forge its individual elements, that is  $C, S, T_0$  and  $P_0$ .

- *Authentication context  $C$ .*  $C$  is publicly available and so  $i$  has access to a valid authentication context.
- *Ephemeral key and commitments  $S$ .* Client  $i$  can produce a valid  $S$  since it does not depend on  $x_i$ . To do so,  $i$  chooses  $z \in_R \mathbb{Z}_q^*$  and calculates  $S = (Z, \vec{S})$ , where  $\vec{S} = S_0, \dots, S_m$ , since it only depends on the knowledge of  $z$ , a generator  $g$  and the set of servers' public keys  $\vec{Y}$  included in  $C$ .
- *Linkage tag  $T_0$ .* To calculate the tag  $T_0 = h_i^{s_1 s_2 \dots s_m}$ ,  $i$  uses  $h_i$  included in  $C$  and and secrets  $s_1, s_2, \dots, s_m$  created in the previous step.
- *Proof  $P_0$ .* The last piece  $i$  must produce is the proof  $PK^{client}$  which depends on the knowledge of  $x_i$  and  $s$ .  $i$  knows one of the secrets, namely  $s$ , but he does not know client  $i$ 's private key  $x_i$ .

By the soundness property of the underlying proof of knowledge  $PK^{client_i}$ ,  $i$  cannot produce a valid proof  $P_0$  on behalf of some  $\hat{i} \in G$ , except with a negligible probability. Therefore,  $i$  must forge  $P_0$  and create an authentication message  $M_i^0$  that includes the forged proof. By the *anytrust* assumption, there exists at least one honest server and therefore  $i$ 's message will be eventually rejected. If  $i$  can forge a proof  $P_0$ , however, such that each honest server  $j$  accepts the proof as coming from a member  $\hat{i}$  with non-negligible probability, then  $i$  must be able to find  $x_i'$  such that  $X_i = g^{x_i'}$ . This, however, is impossible under the Discrete Logarithm assumption, except with a negligible probability. Therefore, each authentication request from  $i \notin G$  is rejected with an overwhelming probability.  $\square$

### 4.6.5 Anonymity

Informally, we want to ensure that an adversary cannot guess which member has been authenticated with a probability greater than random guessing. We will show that DAGA provides anonymity under the Decisional Diffie-Hellman assumption in the random oracle model [13].

We argue that in order to break a client  $i$ 's anonymity, an adversary must leverage the linkage tags because he cannot infer the client's identity based on a proof  $\text{PK}^{\text{client}}$  under the zero-knowledge property of the proof.

An adversary cannot infer the identity of a client based on a proof of knowledge  $\text{PK}^{\text{client}}$  because the proof is zero-knowledge and does not leak the identity of its creator. Consequently, the adversary must focus on the initial, intermediate or final tags of a particular client in hopes of discovering the client's identity. After observing a protocol run, the adversary sees the initial tag  $T_0$ , all the intermediate linkage tags  $T_j$ , a final linkage tag  $T_f$ , and all partial intermediate tags of the servers he controls. By using per-round secrets  $r_j$  and  $s_j$  of every dishonest server  $j$ , the adversary obtains  $T_0 = h_i^{s_h}$  and  $T_f = h_i^{r_h}$ , two tags protected by the per-round secrets  $r_h$  and  $s_h$  of an honest server. For simplicity, assume that there are only two clients  $\{i, j\} \in G$ , hence, the adversary's goal is to decide whether  $\hat{i} = i$  or  $\hat{i} = j$  based on the tags he obtained but *without* the knowledge of either  $s_h$  or  $r_h$ . Because both  $h_i$  and  $h_j$  are generators of  $\mathcal{G}$ , then both generators generate the entire group  $\mathcal{G}$  when raised to  $x \in \{1, \dots, q\}$ . Therefore,  $h_i^{s_h}, h_j^{s_h} \in \mathcal{G}$  as well as  $h_i^{r_h}, h_j^{r_h} \in \mathcal{G}$ , and each element is a random and indistinguishable element of  $\mathcal{G}$ . Moreover, by the properties of  $\mathcal{G}$ , there exists  $s_1, s_2, r_1, r_2, h_1, h_2 \in \mathcal{G}$  such that  $h_i^{s_1} = h_j^{s_2}$  and  $h_i^{r_1} = h_j^{r_2}$ , hence, the tag can be created with respect to  $h_i$  and  $h_j$  equally likely.

**Definition 12.** *An authentication protocol is anonymous if for any probabilistic*

polynomial time adversary  $\mathcal{A}$ , the probability  $p(n)$  that  $\mathcal{A}$  wins the anonymity game is negligible s.t.  $|p(n) - \frac{1}{2}| = \text{negl}(n)$ .

The following *anonymity* game is played between the adversary  $\mathcal{A}$  and the challenger  $\mathcal{C}$ .

1. The challenger  $\mathcal{C}$  randomly generates all private and public keys for every client  $i \in G$  and for every server  $j \in G$ ,  $(X_i = g^{x_i}, x_i)$  and  $(Y_j = g^{y_j}, y_j)$  respectively.
2. The adversary chooses two honest members  $i$  and  $j$ , both of which belong to  $G$ .
3. The challenger gives the adversary the public keys of all clients and all servers, and the private keys of the  $n-2$  dishonest clients ( $G \setminus \{i, j\}$ ) and  $m-1$  dishonest servers.
4. The adversary is allowed to run the protocol polynomially-many times for any member  $k \in G \setminus \{i, j\}$ .
5. The challenger chooses a bit  $b \in \{0, 1\}$  uniformly at random. If  $b = 0$ , then the challenger chooses member  $i$  to participate in the protocol and chooses member  $j$  otherwise.
6. The challenger participates in the authentication protocol playing the role of all honest servers and the chosen honest member. The adversary participates in the authentication protocol playing the role of the dishonest members and the dishonest servers.
7. After the challenge protocol run, the adversary is allowed to run the protocol polynomially many times for any member  $k \in G \setminus \{i, j\}$ . Finally, the adversary outputs his guess  $b' \in \{0, 1\}$ . The adversary wins the anonymity game if  $b' = b$ .

**Theorem 3.** *DAGA offers the anonymity property.*

We will show that if there exists a polynomial time adversary  $\mathcal{A}_{\text{ANON}}$  that breaks the anonymity property with non-negligible probability, then we can use this adversary to create an adversary  $\mathcal{A}_{\text{DDH}}$  that solves the Decisional Diffie-Hellman problem with non-negligible probability.

*Proof.* An adversary  $\mathcal{A}$  has two choices for his behavior: (1)  $\mathcal{A}$  can play the role of the dishonest entities and deviate from the protocol in an arbitrary way, or (2)  $\mathcal{A}$  can follow the prescribed protocols and try to break the anonymity property by observing the protocol runs. Briefly, DAGA requires that each entity produces a proof of correctness, hence, if the adversary does not follow the protocol, he will fail to produce the required output by the soundness property of the underlying proofs of knowledge, and the protocol will abort. Therefore, in order to break the anonymity property,  $\mathcal{A}$  follows the prescribed protocols.

Assume that there exists a probabilistic polynomial time  $\mathcal{A}_{\text{ANON}}$  that breaks the anonymity property with a non-negligible probability and therefore has a non-negligible advantage  $\epsilon_{\text{ANON}}$  in the anonymity game. We will show that if  $\mathcal{A}_{\text{ANON}}$  exists, then we can use  $\mathcal{A}_{\text{ANON}}$  as a subroutine to another probabilistic polynomial time adversary  $\mathcal{A}_{\text{DDH}}$  that solves the DDH problem with non-negligible probability.

$\mathcal{A}_{\text{DDH}}$  plays the DDH game, receives a challenge tuple  $(g, g^a, g^b, g^c)$  from the DDH challenger, and outputs 0 if  $(g^a, g^b, g^c)$  is a Diffie-Hellman tuple, that is  $c = ab$ , otherwise  $\mathcal{A}_{\text{DDH}}$  outputs 1.  $\mathcal{A}_{\text{DDH}}$  uses  $\mathcal{A}_{\text{ANON}}$  as a subroutine and therefore must simulate the  $\mathcal{A}_{\text{ANON}}$ 's view of its interaction with the challenger in the anonymity game. Because all the client's and server's proof of knowledge are zero-knowledge,  $\mathcal{A}_{\text{DDH}}$  can efficiently simulate all proofs in the random oracle model [13]. Therefore, we omit the proofs in the description below keeping in mind that they can be simulated

and correctly verified.  $\mathcal{A}_{\text{DDH}}$  simulates the view of  $\mathcal{A}_{\text{ANON}}$  as follows.

*Step 1:*  $\mathcal{A}_{\text{DDH}}$  creates an authentication context  $C$  as prescribed in the protocol except that he uses  $g^a$  from the DDH challenge tuple as the public key  $Y_h$  of the honest server  $h$ .  $\mathcal{A}_{\text{DDH}}$  sets the generator  $g$  of  $C$  to be the same as  $g$  from the DDH tuple.

*Step 2:*  $\mathcal{A}_{\text{DDH}}$  proceeds to simulate the initial linkage tag  $T_0^i$  by first setting  $g^b$  from the DDH tuple as the client's ephemeral key  $Z_i$ . Now  $\mathcal{A}_{\text{DDH}}$  generates an ephemeral secret  $s_k$  for every server  $k \neq h$  as follows:  $s_k = \mathcal{H}((g^b)^{y_k})$ , which he can do because he possesses all private keys of dishonest servers, and  $\mathcal{A}_{\text{DDH}}$  uses  $g^c$  for the ephemeral secret  $s_h$  client  $i$  shares with the honest server  $h$ . Then,  $\mathcal{A}_{\text{DDH}}$  generates the initial linkage tag  $T_0^i = h_i^{\prod_{k=1}^m s_k}$ .

*Step 3:* For every server  $k \neq h$ ,  $\mathcal{A}_{\text{DDH}}$  processes the tag  $T_0^i$  as prescribed in the protocol. For server  $h$ ,  $\mathcal{A}_{\text{DDH}}$  uses  $g^c$  for  $s_h$  and  $T_h = (T_{h-1})^{(-s_h)(r_h)}$ , finally outputting a final linkage tag  $T_f^i$ .

Now, that  $\mathcal{A}_{\text{DDH}}$  correctly simulated the view of  $\mathcal{A}_{\text{ANON}}$ ,  $\mathcal{A}_{\text{ANON}}$  outputs his guess  $b'_{\text{ANON}} \in \{0, 1\}$ , which  $\mathcal{A}_{\text{DDH}}$  copies and outputs as his own guess  $b'_{\text{DDH}} = b'_{\text{ANON}}$ .

$\mathcal{A}_{\text{DDH}}$  correctly simulates the view of the challenger in the anonymity game with probability  $\frac{1}{2}$  when the challenge tuple is a Diffie-Hellman tuple. Hence,  $\mathcal{A}_{\text{DDH}}$ 's advantage is  $\frac{1}{2}$  of the advantage of  $\mathcal{A}_{\text{ANON}}$ . Following our assumption,  $\mathcal{A}_{\text{ANON}}$  has a non-negligible advantage  $\epsilon_{\text{ANON}}$  in the anonymity game and therefore  $\mathcal{A}_{\text{DDH}}$ 's advantage  $\epsilon_{\text{DDH}} = \frac{\epsilon_{\text{ANON}}}{2}$ , which is also non-negligible. Hence, a contradiction.  $\square$

#### 4.6.6 Forward anonymity.

Informally, an authentication protocol is *forward anonymous* if an adversary cannot break any client's anonymity even if the adversary is in possession of some (or even all) group members' private keys obtained after the protocol round completed. Recall

that a protocol run is defined in terms of an authentication context. The reason that we can only ensure forward anonymity after the protocol round has ended is because an adversary who possesses the private keys of the clients can run the protocol himself using some private key  $x_i$ , successfully impersonating a client  $i$ . After a successful authentication request, the adversary would learn the final linkage tag  $T_f^i$  that would allow him to distinguish all previous authentication requests made by  $i$  as the linkage tag persists throughout the protocol round.

**Definition 13.** *An authentication protocol is forward anonymous if for any probabilistic polynomial time adversary  $\mathcal{A}$ , the probability  $p(n)$  that  $\mathcal{A}$  succeeds at the forward anonymity game is negligible s.t.  $|p(n) - \frac{1}{2}| = \text{negl}(n)$ .*

The *forward anonymity* game is played between the adversary and the challenger and is exactly as the anonymity game defined in the previous section except that in Step 7 the adversary is given the private keys  $(x_i, x_j)$  of both honest members.

**Theorem 4.** *DAGA offers the forward anonymity property.*

*Proof.* Following that DAGA offers anonymity, we know that an adversary  $\mathcal{A}_{\text{ANON}}$  has a negligible advantage in the anonymity game. The only difference between the anonymity and forward anonymity games is the fact that  $\mathcal{A}_{\text{FA}}$  receives the clients' private keys. Because the linkage tags and per-round generators are independent of the private keys,  $\mathcal{A}_{\text{FA}}$  can at most do as well as  $\mathcal{A}_{\text{ANON}}$  by using  $\mathcal{A}_{\text{ANON}}$  as a subroutine and simply not using the private keys. Hence,  $\mathcal{A}_{\text{FA}}$  advantage  $\epsilon_{\text{FA}} = \epsilon_{\text{ANON}}$ , which is negligible.

The only element of any authentication message  $M_0$  that depends on the private key is the proof  $\text{PK}^{\text{client}}$ . This is because each client would use the same authentication context  $C$ ,  $\vec{S}$  is generated based on  $z \in_R \mathbb{Z}_q^*$  and  $\vec{Y} \in C$ . We can easily show

that a proof  $P_0$  from  $M_0$  could have been produced using any private key in question ( $x_i$  or  $x_j$ ), and the knowledge of both keys does not aid  $\mathcal{A}_{\text{FA}}$ .

Recall the prover's steps to prepare  $\text{PK}^{\text{client}}$  described in Section 4.3.7.  $P_0 = (c_s, t, c, r)$ , where  $c_s$  is the random challenge  $t$  and  $c$  are the sets of commitments, and  $r$  is the set of responses as follows

1.  $t = (t_{1.0}, t_{1.10}, t_{1.11}, \dots, t_{n.0}, t_{n.10}, t_{n.11})$  for each  $i \in G$ , where  $t_{i.0} = X_i^{w_i} g^{v_{i.0}}$ ,  $t_{i.10} = S_m^{w_i} g^{v_{i.1}}$ , and  $t_{i.11} = T_0^{w_i} h_i^{v_{i.1}}$ .
2.  $c = (c_1, \dots, c_n)$ , where

$$c_i = \begin{cases} c_s - \sum_{k=1}^n w_k & \text{for } i = \hat{i} \\ w_i & \text{otherwise} \end{cases}$$

3.  $r = (r_{1.0}, r_{1.1}, \dots, r_{i.0}, r_{i.1})$ , where  $r_{i.k} = v_{i.k} - c_i x_{i.k}$  for all  $1 \leq i \leq n$  and  $k \in \{0, 1\}$ , and  $x_{i.0} = x_{i.1} = 0$  for all  $i \neq \hat{i}$ ,  $x_{\hat{i}.0} = x_{\hat{i}}$ , and  $x_{\hat{i}.1} = s$ .

We observe that a private key  $x_{\hat{i}}$  of some prover  $\hat{i}$  is only used once to calculate  $r_{\hat{i}.0} = v_{\hat{i}.0} - c_{\hat{i}} x_{\hat{i}}$  since for each  $i \neq \hat{i}$ ,  $r_{i.0} = v_{i.0}$ . Hence, in order to decide the value of  $b$ ,  $\mathcal{A}_{\text{FA}}$  needs to decide that  $r_{k.0}$  for some position  $k$  is equal to  $r_{\hat{i}.0}$ , where  $\hat{i} \in \{i, j\}$ , in which case the adversary would have to distinguish an element of form  $v_{\hat{i}.0} - c_{\hat{i}} x_{\hat{i}}$  from  $v_{k.0}$ .

However, both  $v_{\hat{i}.0}$  and  $c_{\hat{i}}$  are random and unknown to the adversary since they were securely deleted. Thus, even with the knowledge of  $x_{\hat{i}}$ , all elements are indistinguishable. Alternatively,  $\mathcal{A}_{\text{FA}}$  can solve each  $r_{\hat{i}.0} = v_{\hat{i}.0} - c_{\hat{i}} x_{\hat{i}}$ , using both  $x_i$  and  $x_j$ . In this case, however,  $\mathcal{A}_{\text{FA}}$  obtains two sets of valid, and therefore indistinguishable, solutions. Hence, the knowledge of the private keys does not aid the adversary.  $\square$

### 4.6.7 Proportionality

Intuitively, the proportionality property ensures that within an authentication context  $C$  each client  $i$  can authenticate only once as a particular anonymous client and each subsequent authentication request within the same context will be recognized as coming from that client. Therefore, the verifier will be able to recognize when the same client authenticates but without knowing that client's identity. We achieve this property by assigning a unique linkage tag  $T_i = h_i^{\prod_{j=1}^m r_j}$  to each client  $i$  in a way that ensures that the tag is always the same for each authentication request within the same context  $C$ .

The linkage tags enjoy an additional property of unlinkability between different authentication contexts. That is, the same client  $i$  receives a different and unlinkable tag  $T_f$  within some context  $C_2$  as long as  $C_1 \neq C_2$  such that  $\vec{R} \in C_1 \neq \vec{R} \in C_2$ . This property is important to ensure that clients remain anonymous and unlinkable even after performing authentications within different authentication contexts. It is straightforward to see that two linkage tags of client  $i$  from two different contexts are two independent elements of the underlying group  $\mathcal{G}$ .

**Definition 14.** *An authentication protocol offers the proportionality property if each member  $i$  receives exactly one unique final linkage tag  $T_f^i$  within the same authentication context  $C$ .*

**Theorem 5.** *DAGA offers the proportionality property.*

*Proof.* We will show that each client  $i$ 's final linkage tag is unique, and that during each authentication within the same authentication context  $C$ ,  $i$ 's final linkage tag is the same.

First, however, we will consider the constraints placed on the behavior of each client  $i$  and each server  $j$  by the fact that they need to produce a proof of correctness

of their work and other assumptions. By the soundness property of the underlying proof of knowledge  $\text{PK}^{\text{client}}$  and the assumption that no client knows  $x$  such that  $g_j = g_i^x$  for any  $i, j$ , any client  $i$  must generate his initial linkage tag  $T_0$  with respect to his per-round generator  $h_i$ . That is,  $T_0 = h_i^s$  for some  $s$ . By the soundness property of the underlying proof of knowledge  $\text{PK}_1^{\text{server}}$ , each server  $j$  must correctly remove the ephemeral secret  $s_j$  assigned by a client and add the correct server's ephemeral secret  $r_j$ . That is, each server  $j$  calculates  $T_j = T_{j-1}^{s_j^{-1}r_j}$ , where  $s_j^{-1}$  is a multiplicative inverse (mod  $q$ ) of  $s_j = H(Z^{y_j})$  and  $r_j = \log_g(R_j)$ .

*Each client's final linkage tag is unique.* The fact that each client's final linkage tag is unique, that is,  $T_f^i \neq T_f^j$  for any  $j \neq i \in G$ , follows from the basic number theoretic properties of  $\mathcal{G}$ . Assume the contrary, that is for  $i, j$  such that  $i \neq j$   $T_f^i = T_f^j$ . Then it must be so that  $h_i^r = h_j^r$  where  $h_i \neq h_j$ , and  $r = \prod_{j=1}^m r_j$ . Because  $h_i$  is a generator of  $\mathcal{G}$ , then by definition every element of  $\mathcal{G}$  can be expressed as  $h_i^x$  for some  $x \in \{0, \dots, q-1\}$ , where  $q$  is the order of  $\mathcal{G}$ . Then, we have  $h_j^r = h_i^{xr}$ . Consequently,  $h_i^r = h_i^{xr}$  only if  $r = xr \pmod{q}$ . Because  $|h_i| = q$ , then it must be so that  $x = q$ . Then,  $r = qr \pmod{q}$  and  $r = r \pmod{q}$ . This contradicts the assumption that  $h_i \neq h_j$ .

*Each client  $i$ 's final linkage tag is the same for each accepted authentication request.* The fact that each client  $i$ 's final linkage tag is the same is straightforward. Assume two distinct authentication requests from  $i$  using the same  $C$  that result in two initial linkage tags  $T_0'$  and  $T_0''$  such that  $T_0' = h_i^{s'}$  and  $T_0'' = h_i^{s''}$ , where each  $s' \neq s''$ . The servers will collectively process both tags as follows:  $T_f' = (T_0')^{s'^{-1}r} = (h_i^{s'})^{s'^{-1}r} = h_i^r$  and  $T_f'' = (T_0'')^{s''^{-1}r} = (h_i^{s''})^{s''^{-1}r} = h_i^r$ .

Therefore, each client  $i$  receives exactly one unique final linkage tag for each accepted authentication request.  $\square$

### 4.6.8 Deniability

The deniability notion that DAGA provides follows the zero-knowledge notion of deniability first formalized in the context of authentication in [65]. Informally, we can say that an authentication protocol is *deniable* if after a complete protocol run there is no proof that any client participated in the protocol given an authentication transcript of a protocol run and all public information.

The notion of deniability is closely related to anonymity, however, the subtle differences between these two properties might make a significant difference and make a protocol that provides both properties more suitable for certain situations where the mere fact that *some* client from a particular group authenticated anonymously reveals useful information. In case of anonymity, an adversary should not be able to tell *which* member authenticated while in case of deniability the adversary should not be able to tell whether *any* member authenticated based on the authentication transcripts. We can achieve anonymity by ensuring that two valid transcripts  $T_i$  and  $T_j$  of members  $i, j \in G$  respectively are indistinguishable from one another. On the other hand, we achieve deniability by ensuring that a valid transcript  $T_i$  of client  $i$  is indistinguishable from a simulated transcript  $T_s$  that was computed without the help of *any* member  $i \in G$ .

DAGA inherently offers a weak notion of deniability in the sense that any member listed in  $G$  can plausibly deny being an active client because anyone can conscript an arbitrary group  $G$  using publicly available public keys and without any help or knowledge of the listed members. However, as pointed out above, this might not be sufficient because the fact that a valid authentication transcript exists implies that at least one of the clients in  $G$  authenticated.

DAGA achieves deniability by using an interactive rather than non-interactive

zero-knowledge proof. This is because an interactive proof is not transferable and only “convinces” the party involved in the proof. In a non-interactive proof the verifier is replaced with a hash function to create an unpredictable challenge that a prover cannot anticipate in advance. Therefore, anyone at any point, even much later, can verify the non-interactive proof and be convinced (or not) that the prover knows his secret. This property, while useful, goes against the notion of deniability.

**Definition 15.** *An authentication protocol is deniable if for any client  $i \in G$  there exists a simulator  $S_D$  produces a transcript  $TR_{sim}$  of a protocol run such that  $TR_{sim}$  is indistinguishable from a real transcript  $TR_i$  that resulted from  $i$ 's run of the protocol.*

**Theorem 6.** *DAGA offers the deniability property.*

*Proof.* We will show that there exists a polynomial-time simulator  $S_D$  that produces a transcript  $TR_{S_D}$  that is computationally indistinguishable from a client generated transcript. We assume that  $S_D$  produces a transcript “on behalf” of some client  $i$  using an authentication context  $C = (G, \vec{R}, \vec{H}, p, g)$  without the knowledge of any private key  $x_i$  that corresponds to some public key  $X_i \in \vec{X}$ . The simulator  $S_D$  works as follows. First,  $S_D$  produces a linkage tag  $T_0$  using the client  $i$ 's prescribed per-round generator  $h_i \in \vec{H}$  exactly as client  $i$  would. Since  $S_D$  has all required information (the per-round generator  $h_i$  and the product of all ephemeral secrets shared with the servers) the simulator reproduces the exact tag  $T_0$  as follows.

1. Choose  $z \in_R \mathbb{Z}_q^*$  and compute  $Z = g^z$ .
2. For each server  $j$ , compute  $s_j = H(Y_j^z)$ .
3. Compute  $T_0 = h_i^{s_1 s_2 \dots s_m}$ . Then, for each  $1 \leq j \leq m$  compute  $S_j = g^{s_1 s_2 \dots s_j}$  such that  $S_0 = g$ ,  $S_1 = g^{s_1}$ ,  $\dots$ ,  $S_m = g^{s_1 s_2 \dots s_m}$ . Set  $\vec{S} = (Z, S_0, \dots, S_m)$ .

At this point  $S_D$  has computed  $\vec{S}$  and  $T_0$ . Next,  $S$  must produce a proof  $P_0$ , however, *without* the knowledge of  $i$ 's private key  $x_i$  (or any other key). To do so, we leverage the fact that  $\text{PK}^{client_i}$  is zero-knowledge and therefore there exists a polynomial-time simulator  $S_{zk}$  that produces computationally indistinguishable transcripts of  $\text{PK}^{client_i}$ .  $S_D$  uses  $S_{zk}$  as a subroutine to produce  $P_0$ .  $S_{zk}$  takes as input the authentication context  $C$  and works as follows.

1. Choose  $w_1, \dots, w_n \in_R \mathbb{Z}_q^*$  for all  $i$ .
2. Choose  $v_{1,0}, v_{1,1}, \dots, v_{n,0}, v_{n,1} \in_R \mathbb{Z}_q^*$  for all  $i$ .
3. Compute commitments  $t_{i,0} = X_i^{w_i} g^{v_{i,0}}$ ,  $t_{i,10} = S_m^{w_i} g^{v_{i,1}}$ , and  $t_{i,11} = T_0^{w_i} h_i^{v_{i,1}}$  for each  $i$ .  
Set  $t = (t_{1,0}, t_{1,10}, t_{1,11}, \dots, t_{n,0}, t_{n,10}, t_{n,11})$ ,
4. Compute  $c_s = \sum_{k=1}^n w_k$ .
5. Set  $c_i = w_i$  for each  $i$  and set  $C = (c_1, \dots, c_n)$ .
6. Compute responses  $r = (r_{1,0}, r_{1,1}, \dots, r_{i,0}, r_{i,1})$  using  $r_{i,k} = v_{i,k}$  for all  $1 \leq i \leq n$  and  $k \in \{0, 1\}$ .
7. Output  $P = (C, R)$ .

After obtaining  $P$ ,  $S_D$  sets  $P_0 = P$ .

It is straightforward to see that the tag produced by  $S_D$  is identical to a tag a client  $i$  would have produced, hence, it will be accepted by servers. Also, the proof  $P_0$  is correct and can be successfully verified with respect to the simulated challenge.

$$\begin{array}{ll}
T_{i,0} \stackrel{?}{=} X_i^{c_i} g^{r_{i,0}} & T_{i,10} \stackrel{?}{=} S_m^{c_i} g^{r_{i,1}} \\
X_i^{w_i} g^{v_{i,0}} = X_i^{c_i} g^{r_{i,0}} & S_m^{w_i} g^{v_{i,1}} = S_m^{c_i} g^{r_{i,1}} \\
X_i^{w_i} g^{v_{i,0}} = X_i^{w_i} g^{v_{i,0}} & S_m^{w_i} g^{v_{i,1}} = S_m^{w_i} g^{v_{i,1}}
\end{array}$$

$$\begin{aligned}
T_{i.11} &\stackrel{?}{=} T_0^{c_i} h_i^{r_{i.1}} & c_s &\stackrel{?}{=} \sum_{k=1}^n c_k \\
T_0^{w_i} h_i^{v_{i.1}} &= T_0^{c_i} h_i^{r_{i.1}} & \sum_{k=1}^n w_k &= \sum_{k=1}^n w_k \\
T_0^{w_i} h_i^{v_{i.1}} &= T_0^{w_i} h_i^{v_{i.1}} & &
\end{aligned}$$

Finally,  $S_D$  prepares an authentication message  $M_0 = (C, S, T_0, P_0)$  and sets  $TR_{S_D} = M_0$ .

Now we argue that the transcript  $TR_{S_D} = (C, S, T_0, P_0)$  is computationally indistinguishable from a client generated one.

- $C$ , the authentication context, has an identical distribution,
- $\vec{S}$ , client's ephemeral key and commitments, has an identical distribution,
- $T_0$ , the linkage tag, has an identical distribution,
- $P_0$  is produced by a simulator  $S_{zk}$  that produces proofs that are computationally indistinguishable from a client generated one as the underlying proof of knowledge is honest-verifier zero-knowledge.

□

#### 4.6.9 Forward Deniability

One of the goals of DAGA is to retain anonymity even under the exposure of the clients' long term private keys. This raises an interesting idea to apply the same requirement of forward security to the deniability property. That is, we would like to ensure that a pair of transcripts  $TR_{sim}$  and  $TR_{real}$  generated using an authentication context  $C$ , remains indistinguishable even given the additional knowledge of the compromised private keys. We call this notion of deniability *forward deniability*. Intuitively, forward deniability should hold given that deniability holds as we were

able to show that we can generate an indistinguishable transcript  $T_{sim}$  *without* the knowledge of any private key. The proof of forward deniability follows similarly to the proof of forward anonymity where we argue that the additional knowledge of the private key does not aid the adversary in distinguishing the transcripts.

**Definition 16.** *An authentication protocol is forward deniable if for any client  $i$  a simulated transcript  $TR_{S_D}$  remains (computationally) indistinguishable from a real transcript  $TR_i$  that resulted from  $i$ 's run of that protocol even given a private key  $x_j$  of every client  $j \in G$ .*

**Theorem 7.** *DAGA offers the forward deniability property.*

*Proof.* Assume we have an authentication context  $C$  and two transcripts  $TR_i$  and  $TR_{S_D}$  where each transcript consists of an authentication message  $M_0^i = C, \vec{S}, T_0, P_0$  and  $M_0^{S_D} = C', \vec{S}', T'_0, P'_0$  respectively created using  $C$ . We previously argued, in the proof of deniability, that these two transcripts are (computationally) indistinguishable. Now we wish to revisit this claim and verify if the knowledge of all private keys  $x_i$  aids to distinguish the two transcripts.

- $C = C'$  are identical and therefore have an identical distribution.
- $\vec{S}$  and  $\vec{S}'$  are randomly generated based on  $z, z' \in_R \mathbb{Z}_q^*$  and have an identical distribution, and do not depend on a client's private key.
- $T_0$  and  $T'_0$  have an identical distribution and also do not depend on a client's private key.
- $P_0$  is produced by a simulator  $S_{zk}$  without the knowledge of any private key and  $P'_0$  is a client generated transcript using his private key  $x_i$ .

Therefore, the only element of the transcript that could be affected by the knowledge of the private keys is the proof of knowledge  $P_0$  as  $P_0$  is created using  $x_i$  and  $P'_0$  without  $x_i$ . Therefore, we observe the following differences in the set of responses of  $r$  and  $r'$ :  $r_{i,0} = v_{i,0} - c_i x_i$  and  $r'_{i,0} = v'_{i,0}$ . In order to distinguish between  $P_0$  and  $P'_0$ , it must be possible to distinguish between  $r_{i,0}$  and  $r'_{i,0}$ . However, both  $v_{i,0}$  and  $c_i$  are random and unknown and therefore even with the knowledge of  $x_i$   $r_{i,0}$  and  $r'_{i,0}$  are indistinguishable.  $\square$

## 4.7 Practical Considerations

### 4.7.1 Servers' Liveness

DAGA depends on a set of servers to process each authentication request. Therefore, if a server goes offline or refuses to process a message, the protocol stalls or aborts. While we cannot guarantee that DAGA terminates if one of the above happens, we can employ a wrapper protocol that uses gossip techniques such as those used in PeerReview [91] to ensure liveness.

### 4.7.2 Dealing with Dishonest Servers

Before processing an incoming authentication message, each server  $j$  verifies all proofs of correctness of every server that comes before  $j$ . If an invalid proof  $\text{PK}_1^{\text{server}_k}$  or  $\text{PK}_2^{\text{server}_{k(i)}}$  for some server  $k$  is discovered, the authentication must be aborted and the client cannot be authenticated. We assume that the issue of dealing with dishonest servers within the anytrust set is done administratively [192, 193].

### 4.7.3 Authentication Context

**Generating an authentication context** In order to establish a new authentication context, the servers need to define the clients who belong to a group  $G$  and establish servers' per-round secrets and clients' per-round generators.

*Step 1:* First, the servers choose a safe prime  $p = 2q + 1$  where  $q$  is a sufficiently large prime a generator  $g$  of a prime order  $q$  group  $\mathcal{G}$ .

*Step 2:* Each server  $j$  picks a per-round secret  $r_j \in_R \mathbb{Z}_q^*$ , which is kept secret, and then  $j$  sends to other servers a commitment  $R_j = g^{r_j}$ .

*Step 3:* Servers collectively establish a random per-round generator  $h_i$  for each client  $i$  such that no one knows the logarithmic relationship between  $h_i$  and  $g$ , or between  $h_i$  and  $h_{i'}$  for any pair of clients  $i \neq i'$ , for example using a technique described in Section 4.7.5.

*Step 4:* Servers create a set of the servers' commitments  $\vec{R} = (R_1, \dots, R_m)$  and clients' generators  $\vec{H} = (h_1, \dots, h_n)$ . Then, the servers publish an authentication context  $C = (G, \vec{R}, \vec{H}, p, g)$ .

**Validity of an authentication context** An authentication context might be one time, where each client is expected to make exactly one authentication request or a context may remain valid for certain period of time or some maximum number of authentications made by a single clients or all of clients in  $G$ . Since the servers can keep track of each anonymous client's authentication request, a client may be allowed to make up to  $k$  requests so that each request beyond that is rejected regardless of the validity of the supplied authentication message. After a context expires, all servers securely erase their per-round secrets  $r$  making it impossible to process authentication messages within this context.

**Updating an authentication context** DAGA supports the evolution of the clients is a particular group  $G$  included a context  $C$  in a way that preserves the proportionality property within that context. A new client  $k$  may be efficiently added to  $G$ , by simply adding his public key  $X_k$  to  $\vec{X}$  and adding a new generator  $h_k$  to  $\vec{H}$ . The proportionality property is preserved, because each client's linkage tag only depends on the client's generator and the servers' per-round secrets making it independent of the membership of  $G$ . After the context is updated, each client would create  $\text{PK}^{\text{client}_i}$  with respect to the new group  $G$ . Care needs to be taken to propagate the updated context to all clients to avoid accidentally compromising the identity of the newly added client as he would be the only one using the updated context.

#### 4.7.4 Challenge Generation

A client  $i$  produces a proof of knowledge using an interactive honest-verifier zero-knowledge proof of knowledge as described in Section 4.3.7. Because the proof is interactive, a client  $i$  must obtain a random challenge  $c_s$  from the servers after submitting his commitments. Additionally, because the proof is honest-verifier, the challenge must be indeed randomly chosen. This can be ensured by requiring all servers to collectively generate  $c_s$  so that each server, which would include at least one honest server, contributes its randomness towards  $c_s$ .

One approach to collectively establish  $c_s$  is as follows.

*Step 1:* Upon receiving a client's request, server  $j$  assumes the role of a leader and requests that the other servers generate a new challenge  $c_s$  for client  $i$ .

*Step 2:* Each server  $i$  chooses  $c_i \in_R \mathbb{Z}_q^*$  and then calculates a commitment  $C_i$ . Server  $i$  signs and publishes  $C_i$ .

*Step 3:* Upon receiving  $C_i$  from every other server  $i$ , server  $j$  verifies if all  $C_i$  are

of valid form and properly signed, and if yes server  $j$  publishes an opening  $c_j$  of his commitment  $C_j$  and requests other servers to open their commitments.

*Step 4:* Upon receiving an opening  $c_i$  from every other server  $i$ , server  $j$  verifies if every  $c_i$  is indeed a valid opening of  $C_j$ . If yes, server  $j$  calculates  $c_s = c_1 + \dots + c_m$ . Server  $j$  collects all commitments  $C_i$ , openings  $c_i$ , and the calculated challenge  $c_s$  and forwards to server  $j + 1$  who signs  $c_s$  after verifying that it was correctly calculated.

*Step 5:* Upon receiving  $c_s$  signed by every other server, server  $j$  forwards  $c_s$  for the client along with a proof that every other server calculated the same value.

Under our assumption, there is at least one honest server  $h$  who will randomly choose his  $c_j$  and therefore guarantee that the collective challenge  $c_s$  is properly generated.

#### 4.7.5 Per-Round Generators

For each protocol round, defined by the same context  $C$ , we require that there is a set  $\vec{H} = (h_1, \dots, h_n)$  of  $n$  per-round generators of  $\mathcal{G}$ , where there is one unique generator  $h_i$  for each client  $i$ . The proportionality property depends on the uniqueness of the final linkage tags. Each client  $i$ 's linkage tag  $T_f^i$  is unique and remains fixed within the same  $C$ , precisely because each client  $i$  creates the initial tag  $T_i^0$  with respect to the same but unique per-round generator  $h_i$ .

As defined in Section 4.3.4,  $\mathcal{G}$  is a multiplicative cyclic group of prime order  $q$ . Therefore, all elements of  $\mathcal{G}$ , except for the identity element, are generators of  $\mathcal{G}$  so generating  $\vec{H}$  reduces to choosing  $n$  random elements of  $\mathcal{G}$ .

However, it is important that  $\vec{H}$  is chosen randomly to ensure that the assumption no one knows the logarithmic relationship between any  $h_i$  and  $g$  or between  $h_i$  and  $h_{i'}$  for any pair of clients  $i \neq i'$  holds. Therefore, the anytrust servers must collectively choose  $\vec{H}$  in a way that ensures that none of the servers know the aforementioned

logarithmic relationships. An efficient method to find generators is to use a hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$  to map a per-round fixed string  $(i, \vec{R})$  into each client  $i$ 's generator.

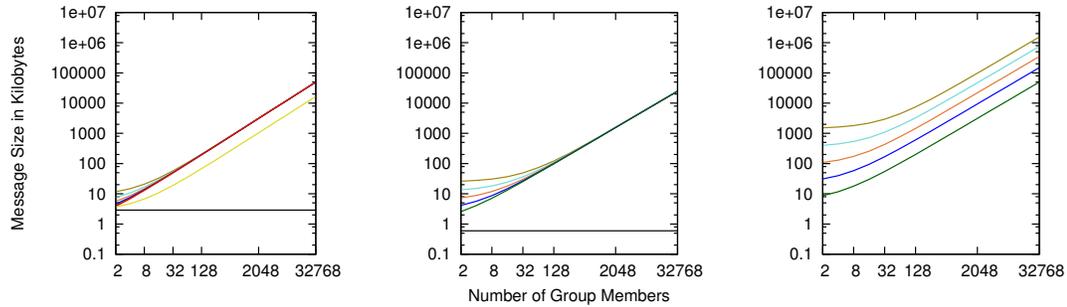
## 4.8 Evaluation

### 4.8.1 Implementation

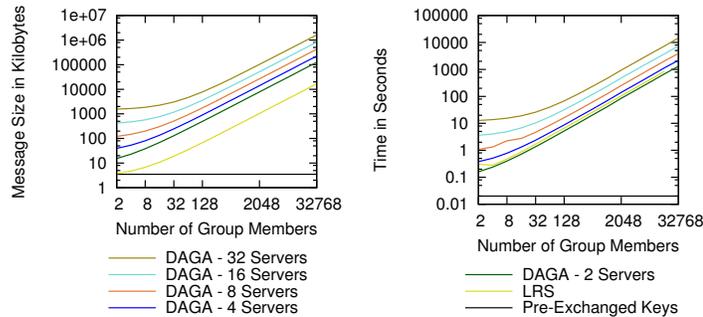
We have implemented DAGA within the context of Dissent [192] using C++ with the Qt framework and the CryptoPP cryptography library. The prototype implements both the client and server aspects of DAGA, but currently does not support exposing misbehaving clients nor any of the extensions to DAGA, discussed in Section 4.3.9 and Section 4.4, respectively. The prototype assumes that all keys derive from the same modulus and subgroup, and that all participants have used an outside channel to agree upon a common set of authentication servers and an authentication context. With the introduction of DAGA, Dissent now includes a modular authentication framework that supports pre-exchanged keys using Stinson's two-way authentication protocol [174] (Protocol 9.6), linkable ring signatures (LRS) [125], and DAGA.

### 4.8.2 Micro benchmarks

We evaluate DAGA in comparison to pre-exchanged keys and LRS. The evaluations were performed on a 64-bit x86 machine running Ubuntu 12.04. This evaluation simulates the authentication of a client to one or more servers within a single process. All communication between parties occurs through bytestreams as if they were sent over the network. Both the authentication time for a single client and the amount of data transmitted during this authentication were recorded. All client and servers



(a) Client to Server Traffic (b) Server to Client Traffic (c) Server to Server Traffic



(d) System Traffic (e) Authentication Time

Figure 4.2: Time and traffic comparison among DAGA, LRS, and pre-exchanged key authentication

keys derive from a common 2048-bit DSA key. For both DAGA and LRS, the number of clients varied from 2 to 32768 by powers of 2. Only DAGA depends on more than one server for authentication. Both the LRS and DAGA depend on a linkage context. For this evaluation, we assume that the administrators of the authentication systems have agreed upon and distributed the linkage context along with the set of the group’s public keys.

Figure 4.2d shows the total system traffic during a single client authentication for the various forms of authentication and group configurations. The traffic results have been broken down into client to server, server to client, and server to server traffic in figures 4.2a, 4.2b, and 4.2c, respectively. As expected, pre-exchanged key authentication does not depend on the number of clients in the group. DAGA authentication transfers more data in all three cases and uniquely has the requirement that servers

communicate with each other during an authentication. LRS authentication involves a non-interactive zero knowledge proof, therefore has constant traffic from server to client. Finally, all forms of DAGA traffic grow linearly in the number of clients and nearly linearly in the number of servers, while only LRS client to server traffic grows linearly.

The time for authentication, Figure 4.2e, exhibits similar characteristics to that of the traffic for the respective authentication techniques. In this scenario, however, unlike traffic, DAGA and LRS computation time remain competitive, particularly when using 4 or less DAGA servers.

While DAGA compares well with other anonymous authentication schemes, like LRS, the performance concerns remain. In order to remain anonymous among  $k$  individuals, anonymous authentication systems traditionally require linear computation. Using more efficient DSA keys, such as, those derived from elliptic curves, would reduce computation and traffic load for these style of protocols including DAGA and LRS.

# Chapter 5

## Related Work

This chapter provides related work for both protocols described in this thesis: PRIVATEEYES and DAGA.

### 5.1 PrivateEyes

Many biometric authentication protocols offer protection of biometric data. Unlike our protocol, however, those protocols frequently protect the biometric data at the cost of a degraded recognition performance, higher complexity, or a lack of mechanisms to create unlinkable personas.

Standard cryptographic solutions for protecting passwords or other secrets, such as encryption or hashing, are difficult to use for protecting biometric templates because even if two templates are generated using two samples of the same biometric characteristic, they are never exactly the same. Homomorphic encryption [78] and secure two-party computation techniques [121] offer good security and privacy guarantees but they normally come at a high performance cost.

There are two main categories of schemes for protecting templates: biometric cryptosystems (BC), and template transformation [104, 137].

Biometric cryptosystems such as fuzzy extractors [59, 61], fuzzy vaults [108] and fuzzy commitments [109], use a template as well as helper data to extract a cryptographic key, with the resulting key validated by verifying its correctness. Helper data generally consists of a biometric template (secure sketches and fuzzy extractors [59, 61]) and optionally an external key (fuzzy vaults [108] and fuzzy commitments [109]). The helper data in BC systems, however, unavoidably leaks data [62, 101]. While BC offers additional features such as reliable cryptographic key generation, they come at the cost of performance and complexity. They heavily rely on error correction codes, which limits their recognition performance to the error-correcting capability of the employed code [104, 152]. Furthermore, BC has not been designed with reusability and revocability in mind [104, 137]. Attacks on multiple records in BC may lead to a full recovery of the secret key and/or the biometric template [26, 160, 167]. To achieve reusability and unlinkable personas, BC schemes must be strengthened by adding auxiliary information, for example passwords [12, 138]. This adds to their complexity, limits user convenience, and in some cases may still be insufficient [99].

Template transformation schemes use a transformation function, either invertible (BioHashing [106]) or non-invertible (cancelable biometrics [151]), and apply it to biometric data during the enrollment phase. For the authentication phase, they apply the same transformation and compare the resulting template against the reference template. In case of invertible transformations, users need to supply, and therefore remember or keep secure, a password or a key, which impacts their convenience. A compromise of this additional information can yield further vulnerabilities [115, 127]. This is in contrast to our protocol where a compromise of the user's token does not expose her biometric data. In the case of non-invertible transformations, the recognition performance is affected because the matching is applied to degraded

transformed templates [152]. However, unlinkable personas can be achieved [104,137]. Finally, it has been shown that in some cases it is possible to recover biometric data from transformed biometric templates [4, 5, 96, 157]. Additionally, both schemes are vulnerable to intrusion and linkage attacks using information recovered from transformed templates [136].

Table 5.1 provides a comparison of our scheme and other template protection systems with respect to personas (providing unlinkability, reusability, renewability) [104, 129, 152], recognition performance, secret information required in addition to biometrics, type of the resulting authentication protocol (single request as opposed to more complex challenge-response like designs) and known attacks.

Scheme	Personas	Recognition performance	Secret information required	Single request authentication	Known attacks
PRIVATEEYES	Yes	Preserved	No	Yes	
Fuzzy vault [108]	No	Affected	No	No	Attack via record multiplicity [20, 167]; brute force attack [6].
Fuzzy commitment [109]	No	Affected	No	No	Leakage of information [62, 101]; Attacks via record multiplicity [20, 167].
Secure sketch and fuzzy extractor [59, 61]	No	Affected	No	No	Attacks via record multiplicity, attacks via key compromise [20, 26, 160, 167]
Hardened fuzzy extractors [12, 20, 138]	Yes	Affected	Yes	No	Attacks via record multiplicity (select schemes) [99]
Biohashing [106]	Yes	Preserved	Yes	Yes	Intrusion and linkage attacks, recovery of the original features [4, 5, 96, 127, 136, 157]; Attack via compromised secret information [115, 127].
Cancellable biometrics [151]	Yes	Affected	No	Yes	Intrusion and linkage attacks, recovery of the original template [4, 5, 96, 136, 149, 157]

Table 5.1: A comparison of different biometric template protection schemes

## 5.2 DAGA

There are many approaches to anonymous and deniable authentication, a broad class of schemes offering varying sets of properties. Some focus on providing properties, such as unlinkability or anonymity revocation by a third party, that contradict to the properties DAGA is designed to achieve.

*Deniable schemes* Deniable authentication [65] defines the idea of deniability in the context of authentication. Their notion of deniability assures that the protocol does not leave any paper trail, however, the scheme is not anonymous. Deniable Ring Authentication [139] combines deniable authentication with ring signatures [155]. While it offers protection against compromised private keys, it still lacks proportionality. [176, 177] makes the protocol of [139] non-interactive. [114] proposes another protocol to achieve deniable ring signature, however, the deniability property is viewed as non-frameability of honest client. Off-the-record [23, 81] (OTR) messaging is a two-party communication protocol that allows confidentiality, repudiability, and forward security. OTR does not aim to provide anonymity and its repudiability property is weaker than deniability we set out to achieve. In fact, OTR is not deniable as users must use their long-term keys for one-time authentication of session keys. OTR for group conversation [18] is an early attempt at OTR, however, it requires a trusted peer who acts as a virtual server responsible for processing and delivering all messages. Multi-party off-the-record messaging [82] (mpOTR) extends OTR to a true group setting. mpOTR leverages a two-party authentication protocol. Consequently, in order to offer plausible deniability, a subset of users must collude which might be problematic if the group consists of users unlikely to do so. Group off-the-record messaging [123] (GOTR) furthers the concept of OTR to provide two strong notions of online and offline deniability, however, it still lacks, even optional,

anonymity.

*Group and ring signatures* Group signatures [36, 42] allow a member to anonymously sign a message on behalf of a pre-defined group. However, user's anonymity is revocable by a group manager. Ring signatures [1, 15, 16, 28, 60, 155, 156] offer greater flexibility by allowing ad-hoc group creation thereby supporting a weaker notion of deniability. Linkable ring signatures [125, 126] and short linkable signatures [10, 185] further improve upon ring signatures by adding linkability. The schemes of [1, 155] support heterogeneous keys. Threshold signatures [166] are useful for collective (“ $k$  out of  $n$ ” members) corroboration.

*E-cash* E-cash schemes [27, 32, 38, 40] are designed with anonymity (also referred to as untraceability) in mind and often achieve deniability as well. However, normally these schemes prevent double-spending as using a coin twice reveals the owner's identity, rendering the schemes useable for one-time authentication only. Many anonymous e-voting schemes [3, 97, 107, 119] provide coercion-resistant and receipt-freeness which offer a weaker notion of deniability.

*Anonymous credentials* The credential system proposed in [33] allows users to obtain credentials from organizations and later demonstrate their possession in an anonymous and unlinkable way as many times as desired. The lack of linkability was later addressed by one-time credentials [124] in form of coins that if spent twice would reveal user's identity. The schemes proposed in [31, 117, 142, 182] bridge this gap and offer credentials that a user can show up to  $k$  times, offering limited linkability which in case of [31] applies to certain periods of time. AnonPass [120] is a system that builds upon blacklistable anonymous credentials that allow to enforce a form of proportionality. The system, however, is not concerned with deniability and forward security as these properties are not critical to its usage model.

# Chapter 6

## Conclusions

This thesis makes three major contributions in response to the critical need for better privacy online and approaches for managing digital footprint. We carefully analyze the complex and often misunderstood relationships between authentication, privacy, and identity management. We then propose a better terminology and clarifications of concepts related to authentication. Specifically, we identify two distinct cases of authentication that are critical to effective identity management. We provide users with two privacy-preserving approaches, PRIVATEEYES and DAGA, to implement these two cases of authentication. These approaches, which we summarize below, target different applications and allow to produce online identities in a privacy-preserving fashion.

Our goal is to better equip clients to actively engage in managing their online presence through a better understanding of this process and ways to accomplish it. We hope that the contributions of this thesis will enable people to be better prepared to take control of their digital footprint and achieve the privacy protection they desire.

**PrivateEyes.** Protecting sensitive biometric data is critically important to re-

note biometric identification, because once compromised, the biometric data becomes unusable for identification purposes. PRIVATEEYES offers a new and secure approach to using biometrics for identification, making them an attractive alternative to passwords and other currently deployed methods. In particular, biometric data is never stored on the server, only on the user's token, and then only in encrypted form. The token itself requires no secure storage; the biometric data cannot be recovered even if an attacker has full and complete access to everything stored on the token. A lost token also cannot be used to impersonate its owner.

Our method is computationally efficient and has the same recognition performance as the underlying feature extraction scheme. It also allows the creation of independent identities to provide enhanced privacy of users' actions across different verifying parties.

**DAGA.** In our online interactions, we need not always reveal our identity in order to obtain access to some resources or services. DAGA is a new anonymous group authentication protocol that offers a unique set of properties: anonymity, deniability, proportionality, and forward anonymity. The anonymity and proportionality properties allow a client to authenticate as some group member (using a group identity) without revealing exactly which one but only once per time period or even at all. Deniability makes it possible to deny ever participating in a protocol, while forward anonymity is a stronger property that offers protection of user's identity and the ability to deny participation even in case of a compromise of user's private key.

To resolve these apparently conflicting requirements, DAGA relies on a federation of independently operated servers that are *collectively* but not individually trusted. DAGA's security property properties are ensured as long as *at least one* server operates correctly and honestly during an authentication process, even if the client does not know which server is honest.

We have analyzed and verified DAGA's four key security properties and have also built a working proof-of-concept implementation of DAGA to validate its performance and practical usability. Our evaluation suggests that DAGA compares reasonably well to LRS and non-anonymous authentication given the functionality gain.

# Bibliography

- [1] Masayuki Abe, Miyako Ohkubo, and Koutarou Suzuki. 1-out-of-n signatures from a variety of keys. In *Advances in Cryptology - Asiacrypt 2002*, pages 415–432. Springer, 2002.
- [2] Byron Acohido. Snowden effect: Young people now care about privacy. USA TODAY, March 13 2014. <http://www.usatoday.com/story/cybertruth/2013/11/13/snowden-effect-young-people-now-care-about-privacy/3517919/>.
- [3] Riza Aditya, Byoungcheon Lee, Colin Boyd, and Ed Dawson. An efficient mixnet-based voting scheme providing receipt-freeness. In *Trust and Privacy in Digital Business*, pages 152–161. Springer, 2004.
- [4] Andy Adler. Can images be regenerated from biometric templates? In *Biometrics Consortium Conference*, 2003.
- [5] Andy Adler. Sample images can be independently restored from face recognition templates. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 2, pages 1163–1166. IEEE, 2003.
- [6] Preda Mih Ailescu. The fuzzy vault for fingerprints is vulnerable to brute force attack. *CoRR*, 2007.

- [7] Astrid Albrecht. Understanding the issues behind user acceptance. *Biometric Technology Today*, 9(1):7–8, 2001.
- [8] Ross Anderson and Markus Kuhn. Tamper resistance-a cautionary note. In *Proceedings of the second Usenix workshop on electronic commerce*, volume 2, pages 1–11, 1996.
- [9] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols*, pages 125–136. Springer, 1998.
- [10] Man Ho Au, Sherman SM Chow, Willy Susilo, and Patrick P Tsang. Short linkable ring signatures revisited. In *Public Key Infrastructure*, pages 101–115. Springer, 2006.
- [11] Tom P Bakker and Claes H de Vreese. Good news for the future? young people, internet use, and political participation. *Communication Research*, page 0093650210381738, 2011.
- [12] Lucas Ballard, Seny Kamara, Fabian Monrose, and Michael K Reiter. Towards practical biometric key generation with randomized biometric templates. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 235–244. ACM, 2008.
- [13] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
- [14] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In *Topics in Cryptology - CT-RSA 2003*, pages 1–18. Springer, 2003.

- [15] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Theory of Cryptography*, pages 60–79. Springer, 2006.
- [16] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 22(1):114–138, 2009.
- [17] Otto Bernecker. Biometrics: Security: An end user perspective. *Information security technical report*, 11(3):111–118, 2006.
- [18] Jiang Bian, Remzi Seker, and Umit Topaloglu. Off-the-record instant messaging for group conversation. In *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*, pages 79–84. IEEE, 2007.
- [19] Matt Bishop. *Introduction to computer security*. Addison-Wesley Professional, 2004.
- [20] Marina Blanton and Mehrdad Aliasgari. On the (non-)reusability of fuzzy sketches and extractors and security in the computational setting. In *SECRYPT*, 2011.
- [21] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on computing*, 15(2):364–383, 1986.
- [22] Dan Boneh. The decision diffie-hellman problem. In *Algorithmic number theory*, pages 48–63. Springer, 1998.
- [23] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, pages 77–84. ACM, 2004.

- [24] Michael Boyd, Dragos Carmaciu, Francis Giannaros, Thomas Payne, and William Snell. Iris recognition (project iris). <http://projectiris.co.uk/>, March 2010.
- [25] Michael Boyd, Dragos Carmaciu, Francis Giannaros, Thomas Payne, William Snell, and Duncan Gillies. Iris recognition. *Imperial College London, Inglaterra*, 2010.
- [26] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 82–91. ACM, 2004.
- [27] Stefan Brands. Untraceable off-line cash in wallet with observers. In *Advances in Cryptology - CRYPTO 1993*, pages 302–318. Springer, 1994.
- [28] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Advances in Cryptology - Crypto 2002*, pages 465–480. Springer, 2002.
- [29] Ryan Broderick and Emanuella Grinberg. 10 people who learned social media can get you fired. *Forbes*, June 6 2013. <http://www.cnn.com/2013/06/06/living/buzzfeed-social-media-fired/>.
- [30] William E Burr, Donna F Dodson, and W Timothy Polk. Nist special publication 800-63. *Electronic Authentication Guideline, ? Version*, 1, 2004.
- [31] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.

- [32] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *Advances in Cryptology - EUROCRYPT 2005*, pages 302–321. Springer, 2005.
- [33] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology - EUROCRYPT 2001*, pages 93–118. Springer, 2001.
- [34] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - CRYPTO 2002*, pages 61–76. Springer, 2002.
- [35] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology - CRYPTO 2003*, pages 126–144. Springer, 2003.
- [36] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology - CRYPTO 1997*, pages 410–424. Springer, 1997.
- [37] Jan Camenisch and Markus Stadler. Proof systems for general statements about discrete logarithms. Technical Report 260, Dept. of Computer Science, ETH Zurich, March 1997.
- [38] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology*, pages 199–203. Springer, 1983.
- [39] David Chaum. The Dining Cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

- [40] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology*, pages 319–327. Springer-Verlag New York, Inc., 1990.
- [41] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology - CRYPTO 1992*, pages 89–105. Springer, 1993.
- [42] David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology - EUROCRYPT 1991*, pages 257–265. Springer, 1991.
- [43] Chun Chen and Raymond Veldhuis. Binary biometric representation through pairwise polar quantization. In *Advances in Biometrics*. Springer Verlag, 2009.
- [44] William R Cheswick, Steven M Bellovin, and Aviel D Rubin. *Firewalls and Internet security: repelling the wily hacker*. Addison-Wesley Longman Publishing Co., Inc., 2003.
- [45] Clayton M. Christensen and Michael B. Horn. The rise of on-line education. The Washington Post, October 11 2011. [https://www.washingtonpost.com/national/on-innovations/the-rise-of-online-education/2011/09/14/gIQA8e2AdL\\_story.html](https://www.washingtonpost.com/national/on-innovations/the-rise-of-online-education/2011/09/14/gIQA8e2AdL_story.html).
- [46] N.L. Clarke and S.M. Furnell. Authentication of users on mobile telephones - a survey of attitudes and practices. *Computers and Security*, 2005.
- [47] Roger Clarke. Human identification in information systems: Management challenges and public policy issues. *Information Technology & People*, 1994.
- [48] Stephen Cobb. Do consumers pass the buck on online safety? new survey reveals mixed messages. We Live Security, November 15

2013. <http://www.welivesecurity.com/2013/11/13/do-consumers-pass-the-buck-on-online-safety-new-survey-reveals-mixed-messages/>.
- [49] Stephen Cobb. Survey finds social media privacy surprises. ESET, November 19 2013. <http://blog.eset.ie/tag/privacy/>.
- [50] Reuven Cohen. LinkedIn hacked: A few app suggestions for protecting your online passwords. <http://www.forbes.com/sites/reuvencohen/2012/06/06/linkedin-hacked-a-few-apps-suggestions-for-protecting-your-online-passwords/>, June 6 2012.
- [51] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pages 340–350. ACM, 2010.
- [52] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology - CRYPTO 1994*, pages 174–187. Springer, 1994.
- [53] Ivan Damgard. Lecture notes on cryptographic protocol theory: On sigma protocols, 2004.
- [54] John Daugman. How iris recognition works. *IEEE Trans. on Circuits and Systems for Video Technology*, 2002.
- [55] Amy Davidson. Introducing strongbox. *The New Yorker*, May 2013.
- [56] K. Delac and M. Grgic. A survey of biometric recognition methods. In *International Symposium on Electronics in Marine*, 2004.

- [57] M.O. Derawi, C. Nickel, P. Bours, and C. Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing*, 2010.
- [58] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *12th USENIX Security Symposium*, August 2004.
- [59] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, 2004.
- [60] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology - EUROCRYPT 2004*, pages 609–626. Springer, 2004.
- [61] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 2008.
- [62] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In *ACM Symposium on Theory of Computing*, 2005.
- [63] John R. Douceur. The Sybil attack. In *1st International Workshop on Peer-to-Peer Systems*, 2002.
- [64] John R Douceur. The Sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.
- [65] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 409–418. ACM, 1998.

- [66] M. El-Abed, R. Giot, B. Hemery, and C. Rosenberger. A study of users' acceptance and satisfaction of biometric systems. In *IEEE International Carnahan Conference on Security Technology*, 2010.
- [67] Nelly Fazio and Antonio Nicolosi. Cryptographic accumulators: Definitions, constructions and applications. *Paper written for course at New York University: [www.cs.nyu.edu/nicolosi/papers/accumulators.pdf](http://www.cs.nyu.edu/nicolosi/papers/accumulators.pdf)*, 2002.
- [68] Matthew F. Ferraro and Daniel Dolgin. The snowden effect, how americas spy agencies can fix their millennials problem. USA POLITICO, March 13 2014. [www.politico.com/magazine/story/2014/03/the-snowden-effect-104642.html](http://www.politico.com/magazine/story/2014/03/the-snowden-effect-104642.html).
- [69] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO 1986*, pages 186–194. Springer, 1987.
- [70] FIDO alliance: Mission. <http://fidoalliance.org/about>, September 2014.
- [71] FIDO alliance: Specifications overview. <http://fidoalliance.org/specifications>, September 2014.
- [72] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.
- [73] Jeff Fromm. Top 5 millennial trends of 2014. The Business Journals, April 7 2014. [www.bizjournals.com/bizjournals/how-to/marketing/2014/04/top-5-millennial-trends-of-2014.html](http://www.bizjournals.com/bizjournals/how-to/marketing/2014/04/top-5-millennial-trends-of-2014.html).

- [74] Steven M. Furnell, PS Dowland, HM Illingworth, and Paul L. Reynolds. Authentication and supervision: A survey of user attitudes. *Computers & Security*, 19(6):529–539, 2000.
- [75] John Gantz and David Reinsel. Extracting value from chaos. *IDC iview*, 1(1142):9–10, 2011.
- [76] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. In *Advances in Cryptology - EUROCRYPT 1996*, pages 354–371. Springer, 1996.
- [77] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. *Information and Computation*, 164(1):54–84, 2001.
- [78] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.
- [79] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, Ithaca, NY, February 2003.
- [80] Tal Golan. Sender address verification: Solving the spam crisis, July 26 2004. [http://www.circleid.com/posts/sender\\_address\\_verification\\_solving\\_the\\_spam\\_crisis/](http://www.circleid.com/posts/sender_address_verification_solving_the_spam_crisis/).
- [81] Ian Goldberg and Nikita Borisov. Off-the-record messaging, 2007.
- [82] Ian Goldberg, Berkant Ustaoglu, Matthew D Van Gundy, and Hao Chen. Multi-party off-the-record messaging. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 358–368. ACM, 2009.

- [83] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.
- [84] Shafi Goldwasser and Mihir Bellare. Lecture notes in cryptography, 2001.
- [85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 291–304. ACM, 1985.
- [86] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
- [87] Whitson Gordon. Here’s everywhere you should enable two-factor authentication right now. Life Hacker, December 10 2013. <http://lifehacker.com/5938565/heres-everywhere-you-should-enable-two-factor-authentication-right-now>.
- [88] Lawrence O Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.
- [89] David Greene. EFF files 22 firsthand accounts of how NSA surveillance chilled the right to association, November 2013. <https://www.eff.org/press/releases/eff-files-22-firsthand-accounts-how-nsa-surveillance-chilled-right-association>.
- [90] Robyn Greenspan. Users connect online with offline. ClickZ, August 24 2004. <http://www.clickz.com/clickz/news/1700363/users-connect-online-offline>.

- [91] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical accountability for distributed systems. In *ACM SIGOPS Operating Systems Review*, volume 41, pages 175–188. ACM, 2007.
- [92] Steffen Hallsteinsen, Ivar Jorstad, et al. Using the mobile phone as a security token for unified authentication. In *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, pages 68–68. IEEE, 2007.
- [93] John Heggestuen. The future of mobile and online banking. Business Insider, October 6 2014. <http://www.businessinsider.com/the-future-of-mobile-and-online-banking-2014-slide-deck-2014-10>.
- [94] Ryan Henry and Ian Goldberg. Batch proofs of partial knowledge. *University of Waterloo, Technical report CACR*, 2012.
- [95] Ryan Henry and Ian Goldberg. Thinking inside the black box: smarter protocols for faster anonymous blacklisting. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society*, pages 71–82. ACM, 2013.
- [96] C.J. Hill. Risk of masquerade arising from the storage of biometrics. *Master's thesis, Australian National University*, 2001.
- [97] Martin Hirt and Kazuo Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology - EUROCRYPT 2000*, pages 539–556. Springer, 2000.
- [98] Shayna Hodkin. The internet of me: Creating a personalized web experience. Wired, November 25 2014. <http://www.wired.com/insights/2014/11/the-internet-of-me/>.

- [99] Sumin Hong, Woongryul Jeon, Seungjoo Kim, Dongho Won, and Choonsik Park. The vulnerabilities analysis of fuzzy vault using password. In *International Conference on Future Generation Communication and Networking*, 2008.
- [100] Hedley Hurwitz. Authentication critical in cyber security war. [http://www.itweb.co.za/index.php?option=com\\_content&view=article&id=61800:authentication-critical-in-cyber-security-war](http://www.itweb.co.za/index.php?option=com_content&view=article&id=61800:authentication-critical-in-cyber-security-war), February 18 2013.
- [101] T. Ignatenko and F.M.J. Willems. Information leakage in fuzzy commitment schemes. *IEEE Trans. on Information Forensics and Security*, 2010.
- [102] Institute of Automation, Chinese Academy of Sciences CASIA. Iris image databases. <http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp>, 11 2012.
- [103] A.K. Jain, A.A. Ross, and K. Nandakumar. *Introduction to Biometrics*. Springer, 2011.
- [104] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar. Biometric template security. *EURASIP J. Adv. Signal Process*, 2008.
- [105] Anil K. Jain, Arun Ross, and Sharath Pankanti. Biometrics: A tool for information security. *IEEE Trans. On Information Forensics and Security*, 2006.
- [106] A.T.B. Jin, D.N.C. Ling, and A. Goh. Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognition*, 2004.

- [107] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 61–70. ACM, 2005.
- [108] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 2006.
- [109] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *ACM Conference on Computer and Communications Security*, 1999.
- [110] Gary C. Kessler. Passwords – strengths and weaknesses. <http://www.garykessler.net/library/password.html>, January 1996.
- [111] T.A.M. Kevenaar, G.J. Schrijen, M. van der Veen, A.H.M. Akkermans, and F. Zuo. Face recognition with renewable and privacy preserving binary templates. In *IEEE Workshop on Automatic Identification Advanced Technologies*, 2005.
- [112] Sara Kiesler. *Culture of the Internet*. Psychology Press, 2014.
- [113] Kristina Knight. TRUSTe: Most people concerned about online privacy, September 19 2013. <http://www.bizreport.com/2013/09/truste-most-people-concerned-about-online-privacy.html>.
- [114] Yuichi Komano, Kazuo Ohta, Atsushi Shimbo, and Shinichi Kawamura. Toward the fair anonymous signatures: Deniable ring signatures. In *Topics in Cryptology - CT-RSA 2006*, pages 174–191. Springer, 2006.
- [115] Adams Kong, King-Hong Cheung, David Zhang, Mohamed Kamel, and Jane You. An analysis of biohashing and its variants. *Pattern Recognition*, 2006.

- [116] Greg Kumparak. Here are all the sites you should enable two factor authentication on. Tech Crunch, March 17 2014. <http://techcrunch.com/2014/03/17/here-are-all-the-sites-you-should-enable-two-factor-authentication-on-and-the-ones-you-should-yell-a>.
- [117] Mohamed Layouni and Hans Vangheluwe. Anonymous k-show credentials. In *Public Key Infrastructure*, pages 181–192. Springer, 2007.
- [118] Adriana Lee. Do you use your real name online? TechnoBuffalo, August 7 2011. <http://www.technobuffalo.com/2011/08/07/do-you-use-your-real-name-online/>.
- [119] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Information Security and Cryptology - ICISC 2003*, pages 245–258. Springer, 2004.
- [120] Michael Z Lee, Alan M Dunn, Brent Waters, Emmett Witchel, and Jonathan Katz. Anon-pass: Practical anonymous subscriptions. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 319–333. IEEE, 2013.
- [121] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology-EUROCRYPT 2007*. Springer, 2007.
- [122] Jing-Chiou Liou and Sujith Bhashyam. On improving feasibility and security measures of online authentication. *Int. Journal of Information and Communication Technology*, 2010.

- [123] Hong Liu, Eugene Y Vasserman, and Nicholas Hopper. Improved group off-the-record messaging. In *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society*, pages 249–254. ACM, 2013.
- [124] Joseph K Liu, Patrick P Tsang, and Duncan S Wong. Recoverable and untraceable e-cash. In *Public Key Infrastructure*, pages 206–214. Springer, 2005.
- [125] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Information Security and Privacy*, pages 325–335. Springer, 2004.
- [126] Joseph K Liu and Duncan S Wong. Linkable ring signatures: Security models and new schemes. In *Computational Science and Its Applications - ICCSA 2005*, pages 614–623. Springer, 2005.
- [127] Alessandra Lumini and Loris Nanni. An improved biohashing for human authentication. *Pattern Recognition*, 2007.
- [128] John Maheswaran, David Isaac Wolinsky, and Bryan Ford. Crypto-book: an architecture for privacy preserving online identities. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, page 14. ACM, 2013.
- [129] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer-Verlag New York, Inc., 2003.
- [130] Wenbo Mao. *Modern cryptography: theory and practice*. Prentice Hall Professional Technical Reference, 2003.
- [131] Libor Masek and Peter Kovesi. Matlab source code for a biometric identification system based on iris patterns. Technical report, University of Western Australia, 2003.

- [132] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.
- [133] L.I. Millett and S.H. Holden. Authentication and its privacy effects. *IEEE Internet Computing*, 2003.
- [134] E. Mordini and S. Massari. Body, biometrics and identity. *Bioethics*, 2008.
- [135] Chris Morran. 1-in-5 internet users always read privacy policies, but that doesn't mean they understand what they're reading. Consumer Media LLC, November 28 2012. <http://consumerist.com/2012/11/28/1-in-5-internet-users-always-read-privacy-policies-but-that-doesnt-mean-they-understand-what-theyre-reading/>.
- [136] A. Nagar, K. Nandakumar, and A.K. Jain. Biometric template transformation: a security analysis. *SPIE, Electronic Imaging, Media Forensics and Security*, 2010.
- [137] Abhishek Nagar. *Biometric Template Security*. Dissertation, Michigan State University, 2012.
- [138] Karthik Nandakumar, Abhishek Nagar, and Anil K. Jain. Hardening fingerprint fuzzy vault using password. In *International Conference on Advances in Biometrics*, 2007.
- [139] Moni Naor. Deniable ring authentication. In *Advances in Cryptology - Crypto 2002*, pages 481–498. Springer, 2002.
- [140] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125. IEEE, 2008.

- [141] Singer Natasha. When a palm reader knows more than your life line. *The New York Times*, 2012.
- [142] Lan Nguyen and Rei Safavi-Naini. Dynamic k-times anonymous authentication. In *Applied Cryptography and Network Security*, pages 318–333. Springer, 2005.
- [143] VK Pachghare. *Cryptography and information security*. PHI Learning Pvt. Ltd., 2008.
- [144] Andrew Patric. Usability and acceptability of biometric security systems. In *Proceedings of the Financial Cryptography Conference (FC04)*, 2004.
- [145] Kun Peng, Colin Boyd, and Ed Dawson. Batch zero-knowledge proof and verification and its applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(2):6, 2007.
- [146] Adrian Perrig. Shortcomings of password-based authentication, 2000.
- [147] Matt Petronzio. How one woman hid her pregnancy from big data. Mashable, April 26 2014. <http://mashable.com/2014/04/26/big-data-pregnancy/>.
- [148] Salil Prabhakar, Sharath Pankanti, and Anil K. Jain. Biometric recognition: Security and privacy concerns. *IEEE Security and Privacy*, 2003.
- [149] Feng Quan, Su Fei, Cai Anni, and Zhao Feifei. Cracking cancelable fingerprint template of ratha. In *Computer Science and Computational Technology, 2008. ISCSCT'08. International Symposium on*, volume 2, pages 572–575. IEEE, 2008.
- [150] S. Rane, A. Nagar, and A. Vetro. Method and system for binarization of biometric data, 2010. US Patent App. 12/688,089.

- [151] N. K. Ratha, J. H. Connell, and R. M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Syst. J.*, 2001.
- [152] Christian Rathgeb and Andreas Uhl. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP Journal on Information Security*, 2011.
- [153] Mudassar Raza, Muhammad Iqbal, Muhammad Sharif, and Waqas Haider. A survey of password attacks and comparative analysis on methods for secure authentication. *World Applied Sciences Journal*, 2012.
- [154] Shane Richmond and Christopher Williams. Millions of internet users hit by massive sony playstation data theft. <http://www.telegraph.co.uk/technology/news/8475728/Millions-of-internet-users-hit-by-massive-Sony-PlayStation-data-theft.html/>, April 26 2011.
- [155] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology - ASIACRYPT 2001*, pages 552–565. Springer, 2001.
- [156] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret: Theory and applications of ring signatures. In *Theoretical Computer Science*, pages 164–186. Springer, 2006.
- [157] A. Ross, J. Shah, and A.K. Jain. From template to image: Reconstructing fingerprints from minutiae points. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2007.
- [158] Ravi Sandhu and Pierangela Samarati. Authentication, access control, and audit. *ACM Computing Surveys (CSUR)*, 28(1):241–243, 1996.
- [159] Charlie Savage. Court rejects appeal bid by writer in leak case. *New York Times*, October 2013.

- [160] Walter J. Scheirer and Terrance E. Boult. Cracking fuzzy vaults and biometric encryption. In *Biometrics Symposium*, 2007.
- [161] Bruce Schneier. Two-factor authentication: too little, too late. *Commun. ACM*, 48(4):136, 2005.
- [162] Bruce Schneier. Why anonymous data sometimes isn't. *Wired*, December 13 2007. [http://archive.wired.com/politics/security/commentary/securitymatters/2007/12/securitymatters\\_1213](http://archive.wired.com/politics/security/commentary/securitymatters/2007/12/securitymatters_1213).
- [163] Claus P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
- [164] Stan Schroeder. 6.5 million encrypted linkedin passwords leaked online. <http://mashable.com/2012/06/06/6-5-million-linkedin-passwords/>, June 6 2012.
- [165] Anand Sharma, V Ojha, RC Belwal, and G Agarwal. Password based authentication: Philosophical survey. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, 2010.
- [166] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology - EUROCRYPT 2000*, pages 207–220. Springer, 2000.
- [167] Koen Simoens, Pim Tuyls, and Bart Preneel. Privacy weaknesses in biometric sketches. In *IEEE Symposium on Security and Privacy*, 2009.
- [168] Emin Gün Sirer, Sharad Goel, Mark Robson, and Do?an Engin. Eluding carnivores: File sharing with strong anonymity. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 19. ACM, 2004.

- [169] Smart cards and biometrics. A Smart Card Alliance Physical Access Council White Paper, March 2011. Publication Number: PAC-11002.
- [170] Jacquelyn Smith. How social media can help (or hurt) you in your job search. CNN, April 16 2013. <http://www.forbes.com/sites/jacquelynsmith/2013/04/16/how-social-media-can-help-or-hurt-your-job-search/>.
- [171] Daniel J. Solove. People care about privacy despite their behavior, November 19 2014. <https://www.teachprivacy.com/people-care-privacy-despite-behavior/>.
- [172] Mark Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011.
- [173] Jay Stanley. People care about privacy despite their behavior, November 19 2014. <https://www.aclu.org/blog/do-young-people-care-about-privacy>.
- [174] Douglas R Stinson. *Cryptography: theory and practice*. CRC press, 2005.
- [175] Brad Stone and Matt Richtel. The hand that controls the sock puppet could get slapped. *New York Times*, 2007.
- [176] Willy Susilo and Yi Mu. Deniable ring authentication revisited. In *Applied Cryptography and Network Security*, pages 149–163. Springer, 2004.
- [177] Willy Susilo and Yi Mu. Non-interactive deniable ring authentication. In *Information Security and Cryptology - ICISC 2003*, pages 386–401. Springer, 2004.
- [178] Ewa Syta, Michael J. Fischer, , David Wolinsky, Abraham Silberschatz, Gina Gallegos García, and Bryan Ford. Private Eyes: Secure Remote Bio-

- metric Authentication (Extended Version). Technical Report TR1510, Yale University, May 2015.
- [179] Ewa Syta, Michael J Fischer, David I Wolinsky, Abraham Silberschatz, Gina Gallegos-Garcia, and Bryan Ford. Private eyes: Secure remote biometric authentication. In *Proceedings of the 12th International Conference on Security and Cryptography*, July 2015.
- [180] Ewa Syta, Stan Kurkovsky, and Bernardo Casano. Rfid-based authentication middleware for mobile devices. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–10. IEEE, 2010.
- [181] Ewa Syta, Benjamin Peterson, David Isaac Wolinsky, Michael Fischer, and Bryan Ford. Deniable anonymous group authentication. Technical Report 1486, Department of Computer Science, Yale University, February 2014.
- [182] Isamu Teranishi, Jun Furukawa, and Kazue Sako. K-times anonymous authentication. In *Advances in Cryptology - ASIACRYPT 2004*, pages 308–322. Springer, 2004.
- [183] Dobromir Todorov. *Mechanics of user identification and authentication: Fundamentals of identity management*. CRC Press, 2007.
- [184] Alice Truing. This is when online commenters are most likely to use a fake name. Quartz, December 15 2014. <http://qz.com/310963/disqus-this-is-when-online-commenters-are-most-likely-to-use-a-fake-name/>.
- [185] Patrick P Tsang and Victor K Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *Information Security Practice and Experience*, pages 48–60. Springer, 2005.

- [186] International Telecommunication Union. The world in 2013: ICT facts and figures, February 27 2013. <http://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx>.
- [187] Henk CA Van Tilborg and Sushil Jajodia. *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2011.
- [188] Mark Vandenwauver, René Govaerts, and Joos Vandewalle. Overview of authentication protocols. In *The Institute of Electrical and Electronics Engineers 31st Annual 1997 International Carnahan Conference on Security Technology*, pages 108–113, 1997.
- [189] VeriSign. Symantec Corporation. Verisign ID protection center: Validation & ID protection (VIP). <http://idprotect.verisign.com>, November 2012.
- [190] VeriSign. Symantec Corporation web site. <http://www.verisign.com/>, November 2012.
- [191] Thomas J Waraksa, Paul A Michaels, Sherri A Slaughter, James A Poirier, and Irvin B Rea. Rolling code for a keyless entry system, 1995. US Patent 5,412,379.
- [192] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 179–182, 2012.
- [193] David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. Scalable anonymous group communication in the anytrust model. In *EuroSec*, 2012.

- [194] Andrew C. Yao. Theory and application of trapdoor functions. In *Annual Symposium on Foundations of Computer Science*, 1982.