Secure, Private, and Personal: Advancing Digital Identity

Author(s):
Hammann, Sven

Publication Date:
2021

Permanent Link:

https://doi.org/10.3929/ethz-b-000494496 →

Diss. ETH. No. 27125

Secure, Private, and Personal: Advancing Digital Identity

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

Sven Hammann

MSc ETH in Computer Science, ETH Zurich
born on 28.04.1990
citizen of Bassersdorf ZH

accepted on the recommendation of

Prof. Dr. David Basin, examiner
Prof. Dr. Srdjan Čapkun, co-examiner
Prof. Dr. Sjouke Mauw, co-examiner
Dr. Saša Radomirović, co-examiner
Dr. Ralf Sasse, co-examiner

2021

Abstract

Internet users' digital identities consist of accounts for numerous services. These accounts can typically be accessed by providing a credential, most commonly a password. However, this primary authentication method is rarely the only way to access an account. Accounts are connected to other accounts, for example through recovery methods, password managers, or single sign-on. Accounts are also connected with the user's electronic devices and with other aspects of the physical world. For example, users frequently have open sessions or saved passwords on their devices and write down passwords physically. This complex web of connections gives rise to a wide range of potential security weaknesses in users' account setups. However, from the user perspective, these connections are often necessary to reduce the risk of being locked out of one's own accounts.

In the first part of this thesis, we introduce *account access graphs*, the first formalism that enables a comprehensive modeling and analysis of the user's entire setup of interconnected accounts, credentials, devices, keys, and documents. Account access graphs support systematically identifying both security vulnerabilities and lockout risks in a user's accounts. We employ the methodology associated with this formalism in a *qualitative user study* where we obtain the account access graphs of twenty participants. In the study, we obtain detailed insights on how users' personal setup choices and behaviors affect their overall account security.

In the second part, we focus on one particularly important kind of connection between users' account setups: *single sign-on* using OpenID Connect. This protocol allows a user to leverage her account with an identity provider (IdP) to log in to other services, called relying parties (RPs). However, OpenID Connect comes with a significant privacy trade-off: Whenever the user logs in to an RP using the protocol, the IdP learns to which RP the user logged in. This is especially problematic when using a particular RP may reveal sensitive information about the user. We present a protocol design that solves this privacy issue, called *Privacy-preserving OpenID Connect*

(POIDC). We have formally modeled our protocol design in the symbolic protocol model, and obtained machine-checked security proofs using the protocol verification tool Tamarin. Thus, POIDC improves users' privacy while obtaining the same security guarantees as the unmodified OpenID Connect protocol.

Zusammenfassung

Die digitale Identität von Internetnutzern umfasst Benutzerkonten für verschiedene Dienste. Auf diese Benutzerkonten kann üblicherweise zugegriffen werden, indem Zugangsdaten eingegeben werden, meistens ein Passwort. Diese primäre Methode zur Authentisierung ist allerdings selten die einzige Art, auf ein Benutzerkonto zuzugreifen. Benutzerkonten sind mit anderen Benutzerkonten verbunden, zum Beispiel durch Wiederherstellungsmethoden, Passwortmanager, oder Einmalanmeldung (Single sign-on). Benutzerkonten sind ausserdem mit den elektronischen Geräten des Nutzers sowie mit anderen Aspekten der physikalischen Welt verbunden. Beispielsweise sind Nutzer häufig längerfristig auf ihren Geräten bei Diensten angemeldet, und sie speichern Passwörter auf ihren Geräten oder schreiben sie auf. Dieses komplexe Netz an Verbindungen kann zu verschiedensten Sicherheitsproblemen und Schwachstellen in Benutzerkonten führen, sind allerdings aus der Sicht der Nutzer häufig notwendig, um das Risiko zu verringern, dass diese aus ihren Konten ausgesperrt werden.

Im ersten Teil dieser Doktorarbeit führen wir *Kontenzugriffsgraphen* (account access graphs) ein. Dieser Formalismus erlaubt erstmals die ganzheitliche Modellierung und Analyse der gesamten Einrichtung eines Benutzers, bestehend aus verknüpften Benutzerkonten, Zugangsdaten, elektronischen Geräten, sowie physischen Schlüsseln und Dokumenten. Mit Hilfe dieses Formalismus können auf systematische Art und Weise sowohl Sicherheitslücken als auch Risiken bezüglich Aussperrung aus Benutzerkonten erkannt werden. Wir wenden diese Methodologie in einer *qualitativen Benutzerstudie* an, in welcher wir die Kontenzugriffsgraphen von zwanzig Teilnehmern erlangen. Die Studie gibt detaillierte Einsichten in die persönlichen Entscheidungen, die Benutzer bezüglich ihrer Einrichtung treffen, ihre Verhaltensweisen diesbezüglich, und wie sich dies im Allgemeinen auf die Sicherheit ihrer Benutzerkonten auswirkt.

Im zweiten Teil fokussieren wir uns auf eine bestimmte, wichtige, Art von Verbindung zwischen Benutzerkonten: Einmalanmeldung (single sign-on) mit Hilfe von OpenID

Connect. Dieses Protokoll ermöglicht es Nutzern, ihre Konten bei einem Identitätsanbieter (identity provider, IdP) zu verwenden, um sich bei anderen Diensten, vertrauende Parteien (relying parties, RPs) genannt, anzumelden. OpenID Connect weist einen deutlichen Nachteil bezüglich der Privatsphäre von Nutzern auf: Jedes mal, wenn sich ein Nutzer bei einem RP mit diesem Protokoll anmeldet, lernt der IdP, bei welchem RP diese Anmeldung stattgefunden hat. Dies ist besonders problematisch, wenn die Tatsache, dass der Nutzer einen bestimmten RP verwendet, heikle Informationen über den Nutzer preisgibt. Wir präsentieren einen Protokollentwurf, der dieses Problem bezüglich der Privatsphäre löst, nämlich *Privacy-preserving OpenID Connect* (POIDC). Wir haben unseren Protokollentwurf im symbolischen Protokollmodell formal modelliert und geben automatisch überprüfte Sicherheitsbeweise mit dem Sicherheitsprotokollverifizierungswerkzeug Tamarin. Somit haben wir gezeigt, dass POIDC die Privatsphäre von Nutzern verbessert und gleichzeitig die gleiche Sicherheit garantiert wie das unmodifizierte OpenID Connect.

Acknowledgements

I would like to thank Prof. Dr. David Basin for giving me the opportunity to perform my doctorate in his group. Even with his extremely busy schedule as Head of Department, David always prioritized giving me feedback. He taught me how to iterate on my work to have it become the best it can be rather than simply good enough. I would like to thank him for teaching me the importance of presenting the contributions of my work in a convincing manner. The valuable advice on this topic allowed me to co-author papers published at great conferences.

Next, I would like to thank Dr. Ralf Sasse for his mentorship. Especially in my first two years in the group, our regular meetings helped guide my work and greatly improved my productivity. Ralf also taught me the importance of presenting my ideas clearly, and that even if definitions are correct mathematically, they must also be unambiguous and understandable to others. Thank you also for taking the time to give me feedback for the introduction and conclusion of this thesis, and taking the time to be on my thesis committee. Finally, thank you for all the fun non work-related discussions, and for having become a good friend.

My appreciation and thanks also goes to Prof. Dr. Saša Radomirović. Saša, together with Ralf, introduced me to the initial ideas that started our development of account access graphs, the part of this thesis that I am the most proud of. Thank you for many fruitful and friendly discussions. In particular, I appreciated our critical discussion culture of openly stating our disagreements. I strongly believe that our discussions led to stronger results. Thank you also for spending the time to be on my thesis committee.

I would also like to thank my other two committee members, Prof. Dr. Sjouke Mauw and Prof. Dr. Srdjan Čapkun. It is a great opportunity to

have people on my committee that are not yet as familiar with my work, and may provide novel perspectives.

Thanks to Dr. Michael Crabb for all his valuable work on our user study. When I was struggling on how to properly analyze the data from our interview participants, your guidance was extremely helpful. I greatly enjoyed working with you on our user study paper, and learned a lot from you.

I would like to express my thanks to Barbara Pfändner, Saskia Wolf, and Vivien Klomp for dealing with my many administrative requests over the years in a helpful and efficient manner. Thank you also to Bernadette Gianesi for quickly and unbureaucratically answering all of my requests about the administrative process for defending my thesis.

Thanks to my office mates over the years, Patrick Leu, Damien Desfontaines, Taeho Lee, Enis Ulqinaku, and Lara Schmid for providing a great work environment. In particular, thanks to Enis for helping me in the initial steps of conducting our user study. Thanks to Lara for our many interesting discussions, both work and non-work related. Thanks to Carlos Cotrini for showing our group that all work and no play makes Johnny a dull boy - thanks for organizing the game evenings, and for becoming a good friend.

I would also like to thank all the other members of the Information Security Group. In particular, thanks to all the people who have taken time to give us feedback on our various paper drafts over the years. Also thanks to all people I worked together during teaching. It has been a pleasant experience to work with all of you. Overall, I have greatly enjoyed my time here, and, while I am excited for my next step in life, I am sad to leave the group.

Finally, I would like to express my gratitude to my friends and family, who have supported me throughout my doctorate. Thank you to my parents, Ute and Peter Hammann, for always showing me that you are proud of me for pursuing this endeavor, and for listening when times were difficult. Thank you to my amazing wife, Esther Tschanen-Hammann, for your love and listening, and for putting me back on track when I was at risk of straying too far from my goals. Thank you for reminding me to stay diligent and to keep my goal in mind even when things were not going smoothly.

Contents

1

Introduction

Every user's digital identity on the Internet is comprised of numerous accounts with services, credentials for those accounts, and their electronic devices that can both hold credentials and have open sessions with accounts. All the user's accounts, credentials, and devices are interconnected in a complex web that spans both the digital and the physical world. There are many different kinds of connections between accounts; we next give two prominent examples. First, account recovery methods commonly connect email accounts with other accounts, designating the email account as a recipient for password reset links. Second, single sign-on using delegated authentication protocols, such as OpenID Connect [1], enable a user to use their account with an identity provider to log in to other services.

Accounts and devices are also connected in different ways. Users are commonly logged in to various services on their devices. These open sessions can be long-lived; for example, email apps on mobile phones only rarely require the user to enter login data. Furthermore, users commonly save passwords on their devices, either using built-in browser features or dedicated password managers. Thus, a user's devices are often directly or indirectly linked to many accounts. Consider for example a phone that has an open session with an email account that can in turn be used to recover most of the user's other accounts. In this scenario, the phone indirectly provides access to those accounts, and is a central element of the user's setup.

These connections also extend into other aspects of the physical world, which we also illustrate with a few examples. Banks send activation codes for online banking accounts by physical mail. Service desks of companies or universities can help users

recover their accounts if they identify themselves in person. In many countries, mobile phone providers require users that ask for a new SIM card to authenticate themselves physically by providing an identification card or passport. As SIM swap attacks [2, 3] and social engineering attacks more generally show, these physical connections can also be exploited by attackers. Finally, the physical location of a device and how this location is secured physically is also relevant. This is particularly important for devices that are not digitally locked, e.g., with a password or PIN. In this case, an attacker who could physically access the device would obtain access to all credentials saved on it and all accounts with an open session on it.

The connections between each user's accounts, credentials, and devices, as well as physical keys and documents are individual for each user and stretch across many different service providers, kinds of devices, and authentication methods. Security weaknesses in a user's setup therefore arise not necessarily from an isolated flaw in a particular service or device, but due to the complex connections in the setup. The individual nature of these connections is a crucial observation; it implies that detecting and amending security weaknesses in a user's setup necessarily require an individual analysis of this user's personal setup.

Furthermore, security is not the only relevant aspect of a user's account setup. If this were the case, then the best way to optimize a setup might be to minimize the number of connections, for example by removing recovery methods. However, recovery methods exist for a good reason: they reduce users' risk of being locked out of their own accounts. Any additional method of accessing an account represents a potential trade-off between security and lockout risk. Each method providing an additional option of accessing an account could be used both by a potential attacker and by the legitimate account owner.

Other aspects that users may optimize their setup for are usability and convenience. While having many open sessions or saved passwords on devices may constitute a security risk, this risk may be considered an acceptable trade-off with respect to the convenience gain. However, there are also scenarios where the security risk outweighs the convenience gain. Consider for example a phone not protected with a locking mechanism, and a user who does not pay close attention to this phone. In this case, the convenience gained by not having to unlock the phone is likely not worth the considerable increase in security risk.

Another important aspect is that of privacy and the trust required with respect to service providers, which we illustrate with two examples. First, a password manager that is synchronized on a service provider's cloud infrastructure may require trusting that service provider with your passwords. While this trust requirement may be lessened by encrypting the passwords with a user-provided master password, this still requires trusting that the service provider does not perform off-line guessing attacks against that master password. Second, using single sign-on with OpenID Connect requires trusting the identity provider and leaks sensitive information since the identity provider learns to which relying parties the user logs in.

To summarize, the following aspects are all relevant concerns with respect to a user's account setup: *Security risk, lockout risk, usability, convenience, and privacy.* This list is likely not exhaustive, but demonstrates that security is not the only concern, and thus users' account setups are likely not optimized for security only. Nevertheless, some connections in a user's setup may lead to security risks that are too high and should not be accepted. However, due to the complexity of these connections, it is unclear how such risks can be detected systematically. Detecting these risks precisely is critical for helping users make meaningful, informed, trade-offs between security and the other mentioned aspects, thereby improving their account setup.

To perform such a systematic analysis, and help users better understand the risks arising from their account setup, a necessary first step is to *model* a user's setup with all its connections. In particular, such a model should enable an automated analysis that can handle even large and complex setups with many different kinds of connections. The need for such a model leads to the first open question that we address in this thesis:

How can we systematically model users' setups of connected accounts, credentials, devices, and other aspects of the physical world, in a way that allows for an automated analysis of security and lockout risks?

Other mentioned aspects, in particular usability and convenience, are more subjective and less suited for an automated analysis. Instead, these aspects relate to the field of *usable security*, which lends itself to qualitative user studies. Indeed, to properly assess the value of a model, it is necessary to apply this model to real user's setups. Different users consciously or subconsciously make different choices and have different preferences with respect to trade-offs between all the mentioned aspects. Thus, we also

address the following questions, which relate our theoretical modeling to the real world, and real users' experiences:

How do real users' account setups look, and what aspects do users prioritize when configuring these setups? Are they aware of security weaknesses in their setups, are they willing to change them, and what threat models do they consider when thinking about their setups' security?

We finally focus on the *privacy* aspect of users' account setups. As mentioned previously, privacy is important when the account connections involve a third party, such as a cloud service provider or an identity provider for single sign-on. In the last part of the thesis, we therefore *zoom in from the big picture view we have taken with respect to the previous questions, and focus on the privacy of single sign-on procedures, specifically using OpenID Connect.*

There is a good reason for focusing on this topic specifically for the last part of this thesis, which lies in the aforementioned trade-offs: Single sign-on using OpenID Connect fares well with respect to usability and convenience, since users who have an open session with their identity provider (IdP) can log in to relying parties (RPs) with the click of a button. It also fares well with respect to security, since it reduces the number of passwords users have, which in turn reduces their likelihood of reusing passwords or picking weak passwords due to memory issues. However, it fares poorly with respect to privacy, since with every login, the IdP learns to which RP the user logs in. That is, users currently have to trade off privacy against security and usability by using this feature. Clearly, it would be desirable to get "the best of all worlds" rather than accepting this trade-off.

This privacy issue is especially problematic when using a particular RP reveals sensitive information about the user. For example, consider a platform for people suffering from substance abuse disorder. While one could argue that such platforms should simply not offer single sign-on, this would mean that these platforms could not reap the benefits provided by the technology. In particular, they would have to manage user credentials themselves, exposing themselves and their users to security risk.

Furthermore, the user's trace of logins to RPs may be particularly sensitive to share with an IdP who already possesses user profiles containing verified personal information. This allows the IdP to link the data about which RPs the user visits to a uniquely

identified person rather than just a pseudonym. For example, the *VERIMI* IdP service [4] performs extensive verification procedures, such as video identification for ID cards and passports.

The OpenID Connect protocol is designed in a way that reveals the RP's identity to the IdP during each protocol run, even if its *implicit flow* is used, a protocol mode where the RP and IdP do not communicate directly with each other, but only through redirects in the user's browser. In particular, the identifier for the RP is a required parameter in a request sent to the IdP. While this is problematic for privacy, there are good security reasons for this design. In particular, simply removing this parameter would result in an insecure protocol. However, if we applied more complex changes to the protocol to improve its privacy, it may still be unclear if the resulting protocol is secure. In particular, OpenID Connect has previously been proven secure [5], and a modified protocol would also require security proofs to be a worthwhile option. This leads into the third topic we address in this thesis:

How can OpenID Connect be modified such that it no longer reveals the RP's identity to the IdP with each login, while provably maintaining all of its security guarantees?

We next explain in more detail how, in this thesis, we address the open questions presented above.

1.1 Contributions

This thesis makes three major contributions, each relating to one of the questions we posed. We next explain each contribution in more detail.

1.1.1 User Account Access Graphs: Formalism and methodology

The first contribution is the *account access graphs* formalism, which for the first time enables a systematic modeling and analysis of a user's entire setup of interconnected accounts, credentials, devices, keys, and documents. We present an associated methodology and algorithms that enable an automated analysis of security and lockout risks in users' setups. This personalized security analysis of users' individual setups can detect subtle security risks that arise from the interplay between many different account access

methods. Analyzing these methods in isolation only would be insufficient to discover such risks.

An account access graph's vertices model the entities comprising a users' setup, such as accounts, credentials, and devices, and its edges model their connections in terms of providing access to each other. We next outline the main ingredients that collectively enable the systematic analysis of security and lockout risks.

We define an account's *access base* as the set of minimal vertex sets such that anyone who initially has access to all vertices in such a set can leverage this to access the account, possibly transitively through accessing connected accounts. This definition can be adjusted to model different scenarios. For example, one can either model that accounts can only be compromised by obtaining access to a credential or connected account, or also model that an account could be compromised directly due to a vulnerability in the service provider's software.

We analogously define an account's *lockout base* as the set of minimal vertex sets such that, if the user lost access to all of the vertices in such a set, this would cause her to be locked out of the account. This definition provides a similar flexibility as that of access bases. For example, one can model that lockout can only occur when the user is lacking a required credential or connected account for each possible access method, or one can also model inherent unavailability of accounts, e.g., due to service providers going out of business.

We give efficient algorithms to compute access and lockout bases. The algorithms also handle cycles; account access graphs can and do contain cycles, for example, when two email accounts are used directly or indirectly to recover each other.

On top of these fundamental concepts, we define *scoring schemes* and *predicates*. These definitions enable analysis of different kinds of security or lockout risks. Scoring schemes assign security or recoverability scores to vertices. These scores can also be non-numerical and can express a wide range of different properties. For example, a scoring scheme could be used to express different attacker capabilities; an account's score then expresses which capabilities would be necessary and sufficient to compromise the account. Predicates can be used to identify different security weaknesses in the graph. For example, our *backdoor* predicate definition expresses that an account can be accessed more easily by a recovery method than by its primary authentication method.

That is, the user may inadvertently have introduced a possibility for an attacker to compromise the account more easily than expected.

Overall, our definitions comprise a flexible *framework* that is independent of any particular security risk or attacker model. Instead, people using the framework for security analysis can design custom scoring schemes and predicates that express exactly the kinds of threats that they consider relevant. Thus, our methodology provides a powerful threat modeling tool that can be used in various contexts.

1.1.2 User Account Access Graphs: Qualitative interview user study

The second contribution is a *qualitative user study* that employs account access graphs to model and analyze the account setups of twenty participants. We also performed semi-structured interviews with the participants discussing their account setups, their rationale for the decisions leading to their setup, and their beliefs about its security. The rich context provided by our participants' real account setups allows us to obtain more detailed insights into users' behaviors affecting their account security than studies that operate without such a context.

Our study is the first to investigate in detail *how users' account setups look*, including the connections between their accounts and devices. Our participants' setups cover a range of different structures, depending on how they use their devices, whether they save passwords, and what kind of recovery methods they have configured. Modeling the setups as account access graphs allows us to analyze their structural features in terms of graph properties. For example, we analyzed which vertices are the most central in our participants' graphs, whether they contain multiple connected components and cycles, and how different password management strategies impact the overall setup.

We have obtained *highly contextualized interview data* about the participants' thought processes leading to their account setup. Our data provides insights into our participants' awareness and understanding of security problems and solutions, their willingness to improve their security, and the threat models they consider relevant when securing their accounts, devices, and credentials. For example, we found that our participants were willing to put effort into securing their important accounts, but did not necessarily have an accurate understanding of which accounts are the most central in their setup, and therefore likely of great importance.

We also give *recommendations to service providers and security experts* for improving users' account setup security. Our recommendations are based on concrete obstacles that prevented our participants from adopting security best practices. For example, some of our participants had usability and privacy concerns with text-message-based two-factor authentication yet were unaware of alternatives, such as dedicated authenticator apps. This observation yields actionable advice for service providers: they should inform users better about the different options for two-factor authentication.

The study shows that our formalism can be effectively used as part of a qualitative user study. It shows how account access graphs can be used to model participants' setups, not only to understand how these setups look, but also to serve as context for in-depth, personal, discussions about each participant's setup.

1.1.3 Privacy-preserving OpenID Connect

The third contribution is the *Privacy-preserving OpenID Connect* (POIDC) protocol, which modifies OpenID Connect such that the identity provider (IdP) no longer learns to which relying party (RP) the user logs in, while provably preserving the security guarantees provided by standard OpenID Connect.

We also further extend POIDC in two different ways. First, we enable additional functional properties that seemingly conflict with user privacy. In particular, we allow the user to request logs to which RPs her account was used to log in. We also allow an IdP to obtain aggregated information about the total number of logins to each RP by all of its users over a fixed time period. Second, we present *Pairwise Privacy-preserving OpenID Connect* (pairwise POIDC), a modification that also prevents colluding RPs from tracking users. It does so by enabling pairwise subject identifiers, where each user is assigned a different identifier at each RP. Standard OpenID Connect offers this feature, but requires the IdP to look up the user's pairwise identifier matching the RP she wishes to log in to. This directly contradicts our desired privacy property with respect to the IdP. Thus a different, more sophisticated, design is necessary to enable pairwise subject identifiers for pairwise POIDC.

An important feature of our proposed extensions is that they provide the same security guarantees as standard OpenID Connect. To show this, we formalize the *security properties* of standard OpenID Connect, POIDC, and pairwise POIDC as trace properties and prove them using the security protocol verification tool Tamarin [6, 7].

For all three protocols, we show the following authentication property: a successful user login at an RP requires that the user has started the protocol and has given consent for logging in to that RP. Previous security proofs for OpenID Connect have been manual [5]. Hence we provide the first machine-checked security proofs for OpenID Connect. Furthermore, our models use a new precise way to symbolically model digital signatures, introduced recently by Jackson et al. [8]. Thus, our models capture a wider range of attacker capabilities with respect to digital signatures than has been traditionally considered for symbolic protocol verification.

We prove the *privacy properties* of all provided POIDC and pairwise POIDC variants using game-based cryptographic reductions. Our protocol variants make use of different *masking functions* that replace the RP's identity with a one-time pseudonym. The concrete masking function used differs according to the desired functional properties. In general, it is instantiated either with a cryptographic hash function or a public-key encryption function. Our cryptographic proofs then reduce the security of the cryptographic primitives used to instantiate this masking function to the privacy of the overall protocol with respect to an honest-but-curious IdP.

1.2 Publications

This thesis is based on the following articles, for which I am the first and main author.

- Sven Hammann, Saša Radomirović, Ralf Sasse, and David Basin, *"User Account Access Graphs"*, in Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (**CCS 2019**)
- Sven Hammann, Michael Crabb, Saša Radomirović, Ralf Sasse, and David Basin, *"Making Security Personal: A Study on Users' Individual Account Connections"*
- Sven Hammann, Ralf Sasse, and David Basin *"Privacy-preserving OpenID Connect"*, in Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (**ASIACCS 2020**)

An initial version of the front-end tool for account access graphs described in Section 2.5 was provided in the following Master's thesis: Jan Falk, *"A frontend for account access graphs"*, 2019, which I co-supervised with Ralf Sasse.

The TAMARIN models for privacy-preserving OpenID Connect described in Section 4.6 are partially based on the following Bachelor's thesis: Xenia Hofmeier, *"Formal Analysis of Web Single-Sign On Protocols using Tamarin"*, 2019, which I co-supervised with Ralf Sasse.

The following new content is not part of the mentioned articles or theses. In Chapter 2, we provide more efficient algorithms for access and lockout base computation than those presented in *"User Account Access Graphs"*. We also provide additional scoring scheme and predicate definitions, as well as a formal definition of the centrality notion used in Chapter 3. In Chapter 4, we present a more flexible protocol definition for POIDC, based on a generic masking function, than that presented in *"Privacy-preserving OpenID Connect"*, and additional protocol variants based on this definition. Furthermore, we provide detailed cryptographic definitions for our privacy properties and cryptographic privacy proofs for all protocol variants.

1.3 Organization

The remainder of this thesis is structured as follows. In Chapter 2, we present the *account access graphs* formalism as well as associated algorithms and tools. In Chapter 3, we present the methodology and results of our *qualitative user study* on users' personal account setups. In Chapter 4, we present privacy-preserving OpenID Connect (POIDC) and its variants, as well as associated security and privacy proofs. Related work is discussed separately for each chapter. In Chapter 5, we draw conclusions from the overall work and discuss directions for future work.

2

User Account Access Graphs: Formal Model

In the introduction, we have motivated the need for a systematic modeling and analysis of users' account setups. These setups consist of entities, such as accounts, credentials, devices, physical keys, and documents that are connected in the sense of providing access to each other. These connections are usually unidirectional: A source credential provides access to a target account, for example. Therefore, a *directed graph* is a natural way of modeling these setups. This chapter introduces *account access graphs*, our modeling formalism, and presents a methodology that leverages this formalism to enable an automated analysis of security and lockout risks.

This chapter is organized as follows. In Section 2.1, we describe the basic aspects of our model, namely accounts, credentials, and connections between them. In Section 2.2, we define account *access sets* and *lockout sets*, the foundational notions that our methodology relies on, and give algorithms to compute them efficiently. In Section 2.3, we define security scoring schemes, based on access sets, and recoverability scoring schemes, based on lockout sets. In Section 2.4, we define *predicates* based on scoring schemes that identify weaknesses in an account connection setup. Section 2.5 describes software we developed to help employ our formalism in practice. We compare with related work in Section 2.6.

2.1 System model

We explain the concepts that we model, and how we model them.

2.1.1 Basic concepts

Users are persons that interact with services and have *accounts* with these services. An account can be accessed by providing *credentials*, or by providing proof of access to other accounts, e.g., in the example of account recovery or single sign-on. We shall take a broad view of these concepts to also model the physical world. An account provides access to a service or resource, which can be digital or physical. Credentials can also be digital or physical, and must be presented to access, or unlock, an account. For example, we can model a physical key unlocking a room by modeling the key as a credential, and physical access to the room as an account.

A set S of credentials and accounts *provides access* to an account a if presenting all credentials in S and proving access to all accounts in S is sufficient to access account a. Losing access to all credentials and accounts in a set S *locks* a user *out* of an account a if at least one of these credentials or accounts is necessary to access account a.

2.1.2 Graph model

We model a user's account setup as an *account access graph*. Vertices represent credentials or accounts. We model the relation of providing access to a vertex v as follows. When v represents an account, we consider the access mechanisms of v. We draw edges of the *same color* from vertices (credentials or accounts) that are *all* needed to access the account (i.e., multi-factor authentication). We draw edges of *different colors* from vertices that represent *alternative* access methods. When v represents a credential, we draw an edge to v when an account provides access to that credential. For example, a password manager is an account that provides access to its stored passwords.

Note that the semantics of colors is local to each target vertex: whether two edges have the same color is only relevant if they are edges to the same target vertex. That is, the colors are simply our way of encoding the difference between *and* and *or* semantics. Encoding these semantics as colors allows easily visualizing account access graphs in a way that directly represent the formalism.

We reuse colors for simplicity in our examples for different target vertices. We use different types of lines (dashed, dotted, and solid) for different colors to distinguish them also in the black-and-white version of this thesis.

Definition 1. *An* account access graph *is a directed graph* $G = (V_G, E_G, C_G)$*, where* V_G *are vertices,* C_G *are colors, and* $E_G \subseteq V_G \times V_G \times C_G$ *are directed colored edges.*

Example 1. *We introduce our running example in Figure 2.1. There is a webshop account* acc_{shop} *that can be accessed with password* pwd_{shop} *or recovered from the e-mail account* acc_{mail}*. The e-mail account requires two-factor authentication with password* pwd_{mail} *and a* code*. The* code *is generated by an authentication app on a (mobile)* device*. The* device *can be unlocked using either a fingerprint (*finger*) or a* PIN*.*

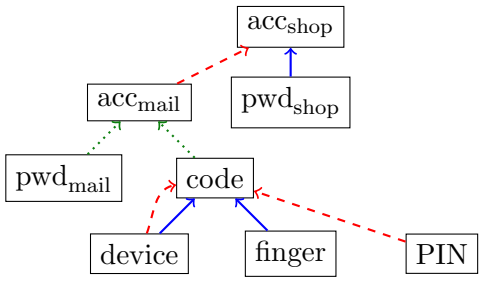

Figure 2.1: Same colors denote conjunction (all necessary for access), different colors denote disjunction (alternative).

Note that our graph model is *untyped*, i.e., we do not formally distinguish between accounts, credentials, devices, and other vertices. If we wish to distinguish between credentials and accounts, we can use the set of *initial vertices* to do so, which we define in the next section.

2.2 Access and lockout sets

In this section, we first provide definitions for access and lockout sets, and then give algorithms to compute them efficiently. These sets provide the basis to analyze security and lockout risks that arise with respect to an account access graph.

2.2.1 Definitions

We first define an account's *access sets*, which denote the minimal sets of credentials or other accounts that are sufficient to provide (possibly transitively) access to the

account. An account's access sets model the possibilities an attacker has to compromise the account. In Section 2.3, we use an account's access base to compute a security score for that account, enabling a detailed security evaluation.

However, if security were our only concern, then accounts should not contain any recovery methods at all. In reality, security must be balanced against the risk of locking users out of their accounts. Thus, we also introduce a way to analyze *lockout sets*. Lockout sets are analogous to access sets, and the main idea of the analogy is the following. To obtain access, an attacker must compromise *all* the necessary credentials, or other accounts, for *one* access method. To be locked out, a user must lose access to *one* of the necessary credentials, or other accounts, for *each* access method.

2.2.1.1 Access sets

We first define what it means that a set of vertices *provides access* to another vertex. A set of vertices V directly provides access to a vertex v if it includes v or includes, for some color c, all vertices with a c-colored edge to v.

Definition 2. *For an account access graph,* $\mathrm{In}_c(v)$ *is the set of all vertices that have a* c*-colored edge to* v*. Formally, for an account access graph* $G = (V_G, E_G, C_G)$*,*

$$\mathrm{In}_c(v) := \{v' \in V_G \mid e = (v', v, c) \in E_G\}.$$

We next define accessFrom(V) as the set of vertices that can directly or transitively be accessed from a set of vertices V.

Definition 3. *For an account access graph* G*, the set* accessFrom(V) *for a vertex set* $V \subseteq V_G$ *is the smallest set that satisfies* $V \subseteq \mathrm{accessFrom}(V)$ *and is closed under the rule*

$$\frac{\exists c \in C_G : \ \emptyset \subsetneq \mathrm{In}_c(v) \subseteq \mathrm{accessFrom}(V)}{v \in \mathrm{accessFrom}(V)} \quad .$$

Example 2. *In the graph from Figure 2.1,*

$$\mathrm{accessFrom}(\{\mathrm{pwd}_{\mathrm{mail}},\ \mathrm{device},\ \mathrm{PIN}\}) = \{\mathrm{pwd}_{\mathrm{mail}},\ \mathrm{device},\ \mathrm{PIN},\ \mathrm{code},\ \mathrm{acc}_{\mathrm{mail}},\ \mathrm{acc}_{\mathrm{shop}}\}\ .$$

We now use accessFrom(V) to define AccessTo(v) as the sets of vertices that provide access to a vertex v.

Definition 4. $\text{AccessTo}(v)$ *is the set of all sets of vertices that provide access to a vertex* v*, that is,* $\text{AccessTo}(v) := \{V \subseteq V_G \mid v \in \text{accessFrom}(V)\}$ *. We call an element of* $\text{AccessTo}(v)$ *an* access set *of* v*.*

We use uppercase names for sets of sets, such as $\text{AccessTo}(v)$, and lowercase names for sets, such as $\text{accessFrom}(v)$. Constants, e.g., acc_{mail}, use lowercase except for acronyms, e.g., PIN.

Example 3. *For the graph in Figure 2.1,* $\text{AccessTo}(\text{acc}_{\text{mail}})$ *contains these sets, and all of their supersets:*

$$\{\text{acc}_{\text{mail}}\}, \{\text{pwd}_{\text{mail}},\ \text{code}\}, \{\text{pwd}_{\text{mail}},\ \text{device},\ \text{finger}\}, \{\text{pwd}_{\text{mail}},\ \text{device},\ \text{PIN}\}\ .$$

As this example illustrates, $\text{AccessTo}(v)$ may contain many elements, but we are often only interested in the *minimal* sets that provide access to a vertex.

Definition 5. *The* minimal access sets *of a vertex* v *are all minimal sets that provide access to* v*. Formally,*

$$\text{MinAccessTo}(v) =$$
$$\{V \subseteq V_G \mid V \in \text{AccessTo}(v) \wedge (\forall\ V' \subsetneq V : V' \notin \text{AccessTo}(v))\}\ .$$

For the graph in Figure 2.1, $\text{MinAccessTo}(\text{acc}_{\text{mail}})$ contains the sets listed in Example 3, but not any of their supersets.

We are particularly interested in those minimal access sets that suffice to gain access to a target account v and only contain vertices from a given set of initial vertices V_{init}. This set contains all vertices that a user may initially have access to, or that an adversary may compromise directly. That is, we assume that an adversary can only directly compromise the vertices in V_{init}, and must derive access to any other vertex using the interconnections modeled in the account access graph. In most of our examples with acyclic graphs, V_{init} is the set of all leaves. However, we show in Section 2.3.3 that V_{init} can also include intermediate vertices to model that an adversary could compromise accounts directly, i.e., without presenting any credentials, by exploiting a vulnerability in the underlying service.

Definition 6. *The* access base $\text{AccessBase}(v, V_{\text{init}})$ *of a vertex* v *with respect to a set of initial vertices* V_{init} *consists of the minimal access sets* V *that only contain vertices from* V_{init}*.*

$$\text{AccessBase}(v, V_{\text{init}}) := \{V \in \text{MinAccessTo}(v) \mid V \subseteq V_{\text{init}}\}\ .$$

An account's access base models all possibilities that an adversary has to compromise that account by compromising credentials or accounts in V_{init}. Each access set contained in the access base models one such possibility.

Example 4. *Let V_{init} be the set of all leaves in the graph from Figure 2.1.*

$$\text{AccessBase}(\text{acc}_{\text{shop}}, V_{\text{init}}) =$$
$$\{\ \{\text{pwd}_{\text{shop}}\}, \{\text{pwd}_{\text{mail}},\ \text{device},\ \text{finger}\}, \{\text{pwd}_{\text{mail}},\ \text{device},\ \text{PIN}\}\ \}\ .$$

Note that account access graphs may have cycles. We illustrate this with the following example.

Example 5. *The account access graph from Figure 2.2 contains an account* $\text{acc}_{\text{backup}}$ *on a backup platform. This account can be accessed using the password* $\text{pwd}_{\text{backup}}$ *and a* code *generated by a two-factor authentication app on a* device*. The account* $\text{acc}_{\text{backup}}$ *then provides access to the password* $\text{pwd}_{\text{manager}}$ *for a password* manager*, and the* seed *to reset the two-factor authentication app, usable to obtain a* code *without access to the* device*. The password* manager *provides access to the password* $\text{pwd}_{\text{backup}}$*.*

Note that the graph contains only a single leaf, device*. Setting V_{init} to all leaves would result in an empty access base for* $\text{acc}_{\text{backup}}$*: it is impossible to obtain access to that account starting from* device *only. A more natural initial set to consider is*

$$V_{\text{init}} := \{\text{pwd}_{\text{backup}}, \text{pwd}_{\text{manager}}, \text{device}, \text{seed}\}\ .$$

It contains all vertices that represent credentials in V_{init} but does not include vertices that represent accounts. Then,

$$\text{AccessBase}(\text{acc}_{\text{backup}}, V_{\text{init}}) =$$
$$\{\{\text{pwd}_{\text{backup}},\ \text{device}\}, \{\text{pwd}_{\text{backup}},\ \text{seed}\}, \{\text{pwd}_{\text{manager}},\ \text{device}\}, \{\text{pwd}_{\text{manager}},\ \text{seed}\}\ \}\ .$$

This example illustrates how V_{init} can be used to distinguish between credentials and accounts without explicitly introducing vertex types into our formalism.

2.2.1.2 Lockout sets

We define the *minimal lockout sets* for a vertex. A lockout set for v is a set of vertices V such that, if the user cannot access any credential in V and is locked out of all accounts in V, the user is locked out of v. We define lockoutFrom(V) for a set of vertices V analogous to accessFrom(V). We assume that the user is initially locked out

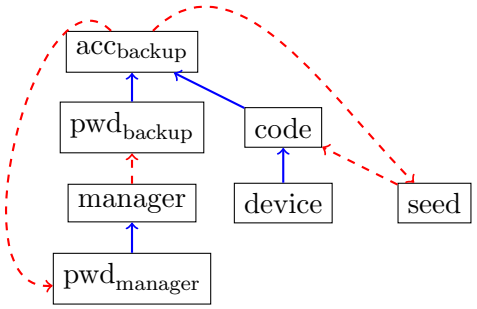

Figure 2.2

of all vertices in V. Then, lockoutFrom(V) also contains all vertices that the user is transitively locked out of.

We construct lockoutFrom(V) as follows. For a vertex v, consider all colors for which v has at least one incoming edge. When lockoutFrom(V) contains the source vertex of an edge for each such color, then this means that the user lacks one vertex for each possible access mechanism of v. Thus, she cannot access v, so $v \in$ lockoutFrom(V).

Definition 7. *For an account access graph G, the set* lockoutFrom(V) *for a vertex set V is the smallest set that satisfies $V \subseteq$ lockoutFrom(V) and is closed under the following rule:*

$$\frac{\exists c \in C_G : \mathrm{In}_c(v) \neq \emptyset \wedge \qquad \forall c \in C_G : (\mathrm{In}_c(v) \neq \emptyset) \rightarrow (\mathrm{In}_c(v) \cap \text{lockoutFrom}(V) \neq \emptyset)}{v \in \text{lockoutFrom}(V)}$$

Example 6. *In Figure 2.1,*

$$\text{lockoutFrom}(\{\mathrm{pwd}_{\mathrm{shop}}, \text{device}\}) = \{\mathrm{pwd}_{\mathrm{shop}}, \text{device}, \text{code}, \mathrm{acc}_{\mathrm{mail}}, \mathrm{acc}_{\mathrm{shop}}\} \ .$$

We next define the set Lockout(v) in terms of lockoutFrom(V) analogous to the relation between AccessTo(v) and accessFrom(V), and then define minimal lockout sets and lockout bases.

Definition 8. *The set of* lockout sets *of a vertex v,* Lockout(v)*, is defined as* Lockout(v) := $\{V \subseteq V_G \mid v \in \text{lockoutFrom}(V)\}$.

Note that the definition for Lockout(v) does not necessarily relate to that of AccessTo(v) in the intuitive, expected way. For example, consider the singleton set $\{v\}$ that is part

of both sets. Its interpretation for AccessTo(v) is the following: An attacker might obtain direct access to v, *without* presenting appropriate credentials. For Lockout(v), the interpretation of $\{v\}$ is that a user might directly be locked out of v, *despite* presenting appropriate credentials.

That is, being locked out of an account is not necessarily the same as being unable to derive access to said account. We caution the reader against trying to interpret the meaning of AccessTo(v) and Lockout(v) at this point; they are meant mostly as building blocks to construct the *access base* and *lockout base*, which we define next. The choice of initial vertices, V_{init}, provides additional semantics. Section 2.3.3 further discusses cases that may at this stage seem unintuitive, and discusses the subtle differences between access and lockout analysis in much more detail.

Definition 9. *We define the set of* minimal lockout sets *of a vertex v as*

$$\text{MinLockout}(v) :=$$
$$\{V \subseteq V_G \mid V \in \text{Lockout}(v) \wedge (\forall\, V' \subsetneq V : V' \notin \text{Lockout}(v))\}\ .$$

Definition 10. *The* lockout base $\text{LockoutBase}(v, V_{\text{init}})$ *of a vertex v with respect to a set of initial vertices V_{init} consists of the minimal lockout sets that only contain vertices from V_{init}.*

$$\text{LockoutBase}(v, V_{\text{init}}) := \{V \in \text{MinLockout}(v) \mid V \subseteq V_{\text{init}}\}\ .$$

For lockout analysis, V_{init} denotes the vertices that a user might directly get locked out of. It should contain any credential that the user could lose or forget. If V_{init} also contains accounts, this models accounts that could be unavailable even if the user has all required credentials. This models that the service provider shut down or is unavailable.

It is important to note that the semantics of V_{init} differ between access and lockout analysis. Thus, the set can and should be chosen differently depending on the analysis that should be performed. Nevertheless, including all leaves of an account access graph that is a tree, or including all vertices that represent credentials, are useful baselines for both kinds of analyses. The differences become more pronounced when vertices that represent accounts are also included in V_{init}, as discussed in more detail in Section 2.3.3.

Example 7. *Let V_{init} be the set of all leaves in the graph from Figure 2.1.*

$$\text{LockoutBase}(\text{acc}_{\text{shop}}, V_{\text{init}}) =$$
$$\{\ \{\text{pwd}_{\text{shop}}, \text{pwd}_{\text{mail}}\}, \{\text{pwd}_{\text{shop}}, \text{device}\}, \{\text{pwd}_{\text{shop}}, \text{finger}, \text{PIN}\}\ \}\ .$$

2.2.2 Algorithms

We give different algorithms for computing each accounts' access and lockout base in an account access graph.

2.2.2.1 Direct application of definitions

The sets can be computed by applying the definitions directly as follows.

The set accessFrom(V) can be computed by starting from V and computing the least fixpoint of the rule from Definition 3. For a query $v \in^? \text{accessFrom}(V)$, computation can stop if $v \in \text{accessFrom}(V)$ is deduced before a fixpoint is reached.

In more detail, the function for computing whether $v \in^? \text{accessFrom}(V)$ for a set of vertices V is described in Algorithm 1, where

$$\text{accessFromStep}(V) := V\ \cup \{v \mid \exists c \in C_G :\ \emptyset \subsetneq \text{In}_c(v) \subseteq V\}$$

applies a single step of the rule given in Definition 3. Since accessFrom(V) is defined as the smallest set closed under that rule, it can be computed as the least fixpoint of iterating this stepwise function.

This algorithm runs in $\mathcal{O}(n^2)$, where $n = |V_G|$, as accessFromStep(V) must consider all vertices, and $\mathcal{O}(n)$ calls to accessFromStep(V) are necessary in the worst case. $v \in^? \text{lockoutFrom}(V)$ is computed in an analogous way, using a single step of the rule given in Definition 7.

We then compute AccessBase(v, V_{init}) as follows. We compute access sets of increasing size, skipping supersets of access sets computed previously. That is, we only perform a query of the form $v \in^? \text{accessFrom}(V)$ if there is no previously computed access set V' such that $V \supseteq V'$. Algorithm 2 shows the details. Lockout bases are computed analogously by replacing $v \in \text{accessFrom}(V)$ with $v \in \text{lockoutFrom}(V)$.

Note that this algorithm may have to perform an exponential number of $v \in^? \text{accessFrom}(V)$ queries, since the number of supersets it may have to consider can be exponential. While it is still feasible to use for small graphs, such as the examples

Data: $G, V \subseteq V_G, v \in V_G$
Result: $v \in^? \text{accessFrom}(V)$
if $v \in V$ **then**
  return true ;
end
$V_{\text{next}} := V$;
repeat
  $V_{\text{prev}} := V_{\text{next}}$;
  $V_{\text{next}} := \text{accessFromStep}(V_{\text{prev}})$;
  if $v \in V_{\text{next}}$ **then**
    return true ;
  end
until $V_{\text{prev}} = V_{\text{next}}$;
return false ;

Algorithm 1: Algorithm for answering a query $v \in^? \text{accessFrom}(V)$

given in this chapter of the thesis, it is generally not an efficient method to compute the access bases of all accounts in an account access graph.

It is more efficient to reuse access or lockout bases for accounts in computing the access or lockout bases of their connected accounts. We use this idea to develop recursive algorithms, which we describe next.

2.2.2.2 Recursive computation of access and lockout bases

The main idea is to recursively compute an access or lockout base from the bases of its incoming vertices until the algorithm reaches leaves in the graph. We first give an algorithm to compute access bases for acyclic graphs, and then generalize it to work with general, possibly cyclic, graphs. We then describe the corresponding algorithm for computing lockout bases as well.

These algorithms first compute supersets of the access and lockout bases that may not be minimal. We denote these sets by AccessBase$'$ or LockoutBase$'$. As a last step before returning the access or lockout base, non-minimal access or lockout sets must be removed.

We require the following auxiliary definitions for defining our recursive algorithms.

Data: $G, V_{\text{init}} \subseteq V_G, v \in V_G$
Result: AccessBase(v, V_{init})
AccessBase $:= \emptyset$;
for $v' \in V_{\text{init}}$ **do**
  if $v \in$ accessFrom$(\{v'\})$ **then**
    AccessBase $:=$ AccessBase $\cup \{\{v'\}\}$;
  end
end
for $i := 2$ **to** $|V_{\text{init}}|$ **do**
  for $V \subseteq V_{\text{init}}$ *with* $|V| = i$ **do**
    for $V' \in$ AccessBase **do**
      if $V \supsetneq V'$ **then**
        skip V ;
      end
    end
    if $v \in$ accessFrom(V) **then**
      AccessBase $:=$ AccessBase $\cup \{V\}$;
    end
  end
end
return AccessBase ;

Algorithm 2: Algorithm for computing an access base

Definition 11. *Let* $\mathrm{In}(v)_G$ *be the set of incoming edges to* v *in graph* $G = (V, E, C)$*, regardless of color. That is,*

$$\mathrm{In}(v)_G := \{e \in E \mid \exists v', c : e = (v', v, c)\} \ .$$

Definition 12. *Let* $\mathrm{Out}(v)_G$ *be the set of all outgoing edges of* v *in graph* $G = (V, E, C)$*, regardless of color. That is,*

$$\mathrm{Out}(v)_G = \{e \in E \mid \exists v', c : e = (v, v', c)\} \ .$$

Access bases for acyclic graphs only The two base cases of the algorithm are the following: A vertex v with no incoming edges, i.e., a leaf, has an access base of $\{v\}$ if $v \in V_{\text{init}}$ and $\emptyset$ otherwise. That is, we cannot access a leaf vertex that is not in V_{init}. Formally, for $\mathrm{In}(v)_G = \emptyset$,

$$\mathrm{AccessBase}'(v, V_{\text{init}})_G = \begin{cases} \{v\} & \text{, if } v \in V_{\text{init}} \ . \\ \emptyset & \text{, if } v \notin V_{\text{init}} \ . \end{cases}$$

For the recursive case, we consider a vertex v with at least one incoming edge. For such a vertex, we consider each color of incoming edges separately. The vertex's access base is the union of each partial access base that only considers edges of one color. It contains the singleton set $\{v\}$ if and only if $v \in V_{\text{init}}$. We first define the partial access base that only considers edges of color c as follows.

$$\mathrm{AccessBasePart}(v, c, V_{\text{init}})_G := \bigcup_{v' \in \mathrm{In}_c(v)} \mathrm{AccessBase}'(v', V_{\text{init}})_G \ .$$

Then, we define the recursive case for $\mathrm{In}(v)_G \neq \emptyset$ as follows.

$$\mathrm{AccessBase}'(v, V_{\text{init}})_G = \begin{cases} \bigcup_{c \in C_G, \mathrm{In}_c(v)_G \neq \emptyset} \mathrm{AccessBasePart}(v, c, V_{\text{init}})_G \cup \{v\} & \text{, if } v \in V_{\text{init}} \ . \\ \bigcup_{c \in C_G, \mathrm{In}_c(v)_G \neq \emptyset} \mathrm{AccessBasePart}(v, c, V_{\text{init}})_G & \text{, if } v \notin V_{\text{init}} \ . \end{cases}$$

This recursion does not terminate on cyclic graphs, so it cannot be applied to those directly. We next give a modified algorithm that works for cyclic graphs as well.

Access bases for general graphs We explain the main idea for transforming the above algorithm into one that also terminates on cyclic graphs. Let us informally call a set of incoming edges to v' for some color c a *method to access* v'. We make the following observation. Any method to access an account v' that contains an outgoing edge from v is not useful in computing the access base of v. Using a method containing such an outgoing edge would already require access to v. Thus, any access set computed using such a method must necessarily be a superset of another access set that provides access to v. It is therefore either already part of the access base or is not minimal.

This insight can be used to modify the algorithm for general, possibly cyclic, graphs as follows: The recursive steps to compute an access for a vertex v are not performed on the original graph, but on the graph where the following edges have been removed:

- All outgoing edges of v.
- All edges that have the same color and target as an outgoing edge of v.

Formally, we define $G' := (V_G, E'_G, C_G)$, where

$$E'_G := E_G \setminus \{e = (v_1, v_2, c) \mid \exists e' = (v, v_2, c) \in \mathrm{Out}(v)_G\} \ .$$

The excluded edges include those in $\mathrm{Out}(v)_G$ itself since it is possible that $v_1 = v$.

We then perform the recursive steps on G' rather than G. That is, we consider the following AccessBasePart definition with respect to G'.

$$\mathrm{AccessBasePart}(v, c, V_{\mathrm{init}})_{G'} := \bigcup_{v' \in \mathrm{In}_c(v)} \mathrm{AccessBase}'(v', V_{\mathrm{init}})_{G'} \ .$$

Then, we define the recursive case for $In(v)_G \neq \emptyset$ as follows.

$$\mathrm{AccessBase}'(v, V_{\mathrm{init}})_G := \bigcup_{c \in C_G, \mathrm{In}_c(v)_{G'} \neq \emptyset} \mathrm{AccessBasePart}(v, c, V_{\mathrm{init}})_{G'} \cup \{v\} \quad , \text{if } v \in V_{\mathrm{init}} \ .$$
$$\bigcup_{c \in C_G, \mathrm{In}_c(v)_{G'} \neq \emptyset} \mathrm{AccessBasePart}(v, c, V_{\mathrm{init}})_{G'} \quad , \text{if } v \notin V_{\mathrm{init}} \ .$$

Lockout bases We can compute lockout bases with a similar algorithm. The main difference is the partial lockout bases that we consider in the recursive case. Rather than considering a partial lockout base for each color, we instead consider all different sets of incoming edges that each contain exactly one edge for each color (among the

colors of incoming edges). We then map these edge sets to sets of their source vertices, and compute our partial lockout bases on these sets of source vertices. The idea can be explained as follows. If an access method requires multiple factors, represented by edges of the same color, then these factors represent different lockout possibilities. Losing any one of them leads to lockout, and thus each must be contained in a different lockout set. For each color c, each of our set combinations contains exactly one source vertex of an edge with color c. Thus, each of the multiple factors is part of a different such source vertex set.

The base case looks the same as for access bases, that is, for $\mathrm{In}(v)_G = \emptyset$,

$$\mathrm{LockoutBase}'(v, V_{\mathrm{init}})_G = \begin{cases} \{v\} & \text{, if } v \in V_{\mathrm{init}} \ . \\ \emptyset & \text{, if } v \notin V_{\mathrm{init}} \ . \end{cases}$$

For the recursive case, we define the sets of source vertices we described as follows.

Definition 13. *Let $c_1, \ldots, c_n$ be the colors for which $In(v)_c \neq \emptyset$. Then,* $\mathrm{OneOfEach}(v)_G$ *is the set of all source vertices that contain exactly one such vertex for each incoming color $c_1, \ldots, c_n$. Formally,*

$$\mathrm{OneOfEach}(v)_G := \{\{v_1, \ldots, v_n\} \mid (v_1, v, c_1) \in E_G, \ldots, (v_n, v, c_n) \in E_G\} \ .$$

We can then define the recursive steps as follows. To handle cycles, we can use the same mechanism as for access bases, so G' is defined the same way. We first define

$$\mathrm{LBasePart}(v, S, V_{\mathrm{init}})_{G'} := \bigcup_{v' \in S} \mathrm{LockoutBase}'(v', V_{\mathrm{init}})_{G'} \ .$$

Then, we define for $In(v)_G \neq \emptyset$,

$$\mathrm{LockoutBase}'(v, V_{\mathrm{init}})_G := \begin{cases} \bigcup_{S \in \mathrm{OneOfEach}(v)_G} \mathrm{LBasePart}(v, S, V_{\mathrm{init}})_{G'} \cup \{v\} & \text{, if } v \in V_{\mathrm{init}} \ . \\ \bigcup_{S \in \mathrm{OneOfEach}(v)_G} \mathrm{LBasePart}(v, S, V_{\mathrm{init}})_{G'} & \text{, if } v \notin V_{\mathrm{init}} \ . \end{cases}$$

2.2.2.3 Translation into Horn clauses

The next algorithms we describe are particularly interesting to optimize the efficiency of $v \in^? \mathrm{accessFrom}(V)$ and $v \in^? \mathrm{lockoutFrom}(V)$ queries, but require an initial pre-computation. The idea is to translate the account access graph into a set of Horn

clauses to leverage Horn clause resolution algorithms for answering these queries efficiently. Different sets of Horn clauses are required for access and lockout computation, respectively.

Definition 14. *Let G be an account access graph. $\mathcal{V}(G)$ is called the set of G's* equivalent variables*, and is given as follows. For each vertex $v \in V_G$, we introduce a variable $v \in \mathcal{V}(G)$ and an additional auxiliary variable $v_c \in \mathcal{V}(G)$ for each color for which v has at least one incoming edge. That is, the set of variables is*

$$\mathcal{V}(G) := V_G \cup \{v_c \mid v \in V_G \wedge c \in C_G \wedge \mathrm{In}_c(v) \neq \emptyset\} \ .$$

Definition 15. *Let G be an account access graph. $\mathcal{C}_{\mathrm{Access}}(G)$ is called the set of G's* equivalent access-clauses*, and is given as follows over the variables in $\mathcal{V}(G)$.*

$$\mathcal{C}_{\mathrm{Access}}(G) := \{v_c \to v \mid v \in V_G,\ \mathrm{In}_c(v) \neq \emptyset\} \ \cup \ \{\bigwedge_{v' \in \mathrm{In}_c(v)} v' \to v_c\} \ .$$

Theorem 1. *Let G be an account access graph. In the set of Horn clauses $\mathcal{C}_{\mathrm{Access}}(G)$, a variable v that corresponds to a vertex is entailed by the variables V corresponding to vertices, written $V \to v$, if and only if $v \in \mathrm{accessFrom}(V)$ in G.*

Proof. We first show that $v \in \mathrm{accessFrom}(V)$ implies $V \to v$ for the Horn clause variables. We show this by structural induction on the derivation tree of $v \in \mathrm{accessFrom}(V)$ using the rule from Definition 3. The base case is $v \in V$. In this case, $v \in V$ also over the variables $\mathcal{V}(G)$ and thus trivially $V \to v$.

For the induction step, consider an application of the rule with conclusion $v \in \mathrm{accessFrom}(V)$. From the induction hypothesis, for any previous deduction of $v' \in \mathrm{accessFrom}(V)$, $V \to v'$ for the Horn clause variables. Now, consider the current rule's premise $\exists c \in C_G : \ \emptyset \subsetneq \mathrm{In}_c(v) \subseteq \mathrm{accessFrom}(V)$. Let c' be a fixed color such that $\emptyset \subsetneq \mathrm{In}_{c'}(v) \subseteq \mathrm{accessFrom}(V)$. For any $v' \in \mathrm{In}_{c'}(v)$, $v' \in \mathrm{accessFrom}(V)$, and thus, due to the induction hypothesis, $V \to v'$. We can therefore apply the Horn clause $\bigwedge_{v' \in \mathrm{In}_{c'}(v)} v' \to v_{c'}$ to obtain $v_{c'}$, and then apply $v_{c'} \to v$ to obtain v.

We now show the converse by structural induction on the derivation of v by applying Horn clauses. The base case is again $v \in V$, in which case $v \in V$ also for the graph and thus $v \in \mathrm{accessFrom}(V)$.

For the induction step, consider the derivation step that yields v. This step must apply a clause of the form $v_c \to v$, since any other clauses only yield auxiliary variables of the form v_c. Note that $v_c \notin V$, since V does not contain any auxiliary variables. Thus, v_c must have been derived from a clause of the form $\bigwedge_{v' \in \mathrm{In}_{c'}(v)} v' \to v_{c'}$, where

$\text{In}_c(v) \neq \emptyset$. Thus, we must have $V \to v_i$ for all v_i on the left-hand side of that clause. Due to the induction hypothesis, also $v_i \in \text{accessFrom}(V)$ for all of these v_i. Thus, $\emptyset \subsetneq \text{In}_{c'}(v) \subseteq \text{accessFrom}(V)$, and therefore also $\exists c \in C_G : \emptyset \subsetneq \text{In}_c(v) \subseteq \text{accessFrom}(V)$. We apply the rule from Definition 3 and obtain $v \in accessFrom(V)$.

□

Definition 16. *Let G be an account access graph. $\mathcal{C}_{\text{Lockout}}(G)$ is called the set of G's* equivalent lockout-clauses*, and is given as follows over the variables in $\mathcal{V}(G)$.*

$$\mathcal{C}_{\text{Lockout}}(G) := \{ \bigwedge_{\text{In}_c(v)\neq\emptyset} v_c \to v \mid v \in V_G,\ \exists c \in C_G : \text{In}_c(v) \neq \emptyset\} \cup \{v' \to v_c \mid v' \in \text{In}_c(v)\} \ .$$

Theorem 2. *Let G be an account access graph. In the set of Horn clauses $\mathcal{C}_{\text{Lockout}}(G)$, a variable v that corresponds to a vertex is entailed by the variables V corresponding to vertices, written $V \to v$, if and only if $v \in \text{lockoutFrom}(V)$ in G.*

Proof. We first show that $v \in \text{lockoutFrom}(V)$ implies $V \to v$ for the Horn clause variables. We show this by structural induction on the derivation tree of $v \in \text{lockoutFrom}(V)$ using the rule from Definition 7. The base case is $v \in V$. In this case, $v \in V$ also over the variables $\mathcal{V}(G)$ and thus trivially $V \to v$.

For the induction step, consider an application of the rule with conclusion $v \in \text{lockoutFrom}(V)$. By the induction hypothesis, for any previous deduction of $v' \in \text{lockoutFrom}(V)$, $V \to v'$ for the Horn clause variables. Now, consider the current rule's premise

$$\exists c \in C_G : \text{In}_c(v) \neq \emptyset \ \wedge \forall c \in C_G : (\text{In}_c(v) \neq \emptyset) \to (\text{In}_c(v) \cap \text{lockoutFrom}(V) \neq \emptyset) \ .$$

As $\exists c \in C_G : \text{In}_c(v) \neq \emptyset$, there exists a clause of the form $\bigwedge_{\text{In}_c(v)\neq\emptyset} v_c \to v$. From the rule's premise, for all such c with $\text{In}_c(v) \neq \emptyset$, $\text{In}_c(v) \cap \text{lockoutFrom}(V) \neq \emptyset$. Thus, for each such c, there is a $v' \in \text{In}_c(v)$ with $v' \in \text{lockoutFrom}(V)$. By the induction hypothesis, $V \to v'$ for all of these v'. We can therefore apply the Horn clause $v' \to v_c$ to obtain v_c for each c with $\text{In}_c(v) \neq \emptyset$, and then apply $\bigwedge_{\text{In}_c(v)\neq\emptyset} v_c \to v$ to obtain v.

We now show the converse by structural induction on the derivation of v by applying Horn clauses. The base case is again $v \in V$, in which case $v \in V$ also for the graph and thus $v \in \text{lockoutFrom}(V)$.

For the induction step, consider the derivation step that yields v. This step must apply a clause of the form $\bigwedge_{\text{In}_c(v)\neq\emptyset} v_c \to v$ (with $\exists c : \text{In}_c(v) \neq \emptyset$), since any other clauses only yield auxiliary variables of the form v_c. Note that $v_c \notin V$, since V does not contain any auxiliary variables. Thus, all the v_c's must have been derived from clauses

of the form $v' \to v_c$, where $v' \in \text{In}_c(v)$. Thus, for each c with $\text{In}_c(v) \neq \emptyset$, $V \to v'$ for at least one such v' with $v' \in \text{In}_c(v)$. By the induction hypothesis, $v' \in \text{lockoutFrom}(V)$ for all of these v'. Thus, $\text{In}_c(v) \cap \text{lockoutFrom}(V) \neq \emptyset$ for each such c. Since also $\exists c : \text{In}_c(v) \neq \emptyset$, we obtain

$$\exists c \in C_G : \text{In}_c(v) \neq \emptyset \ \wedge \forall c \in C_G : (\text{In}_c(v) \neq \emptyset) \to (\text{In}_c(v) \cap \text{lockoutFrom}(V) \neq \emptyset) \ .$$

We apply the rule from Definition 7 to obtain $v \in \text{lockoutFrom}(V)$. □

These results allow for the use of efficient decision procedures for Horn clause entailment, such as the linear time algorithm by Dowling and Gallier [9], for answering $v \in^? \text{accessFrom}(V)$ and $v \in^? \text{lockoutFrom}(V)$ queries.

2.3 Scoring schemes

The security of a user's account setup depends on many factors. Not all accounts are equally valuable. Some users share their credentials with friends or family and other users are at particular risk of having their physical devices stolen. A security evaluation of a user's account setup should thus depend on the user's threat model. There can therefore be no single evaluation method that fits all use cases. For regular users, it may be sufficient to point out critical flaws in a setup that could easily be exploited. For users where the risk of targeted attacks is higher, such as reporters or politicians, analyzing security with respect to both local and remote attackers would be appropriate. Moreover, our account access graphs are general enough to be applicable not only to individual users but also to organizations. These organizations may want to perform an even more fine-grained analysis, incorporating our methods into their threat modeling process. They can also combine our methods with insights obtained from existing risk analysis.

To allow for a flexible yet expressive security analysis, we introduce *security scoring schemes*. These schemes provide a general method to evaluate an account's security using its access base. Recall that an account's access base consists of the credential sets that are sufficient to compromise that account. From these sets, security scoring schemes compute a security *score* for the account. A *score* is a general concept that can be defined over various domains. These domains are (partially) ordered, and a higher score denotes a more secure account.

We also introduce an analogous notion for analyzing lockout risk. We say that an account with low lockout risk achieves high *recoverability*. We can then analyze both security and recoverability of an account setup and understand whether the setup achieves a good balance between the two. The definitions for security scoring schemes can naturally be adapted to apply to recoverability instead by replacing the access base with the lockout base, yielding definitions for *recoverability scoring schemes*. A higher recoverability score for an account denotes that a user is less likely to get locked out of the account.

We first give the general definitions, then illustrate their flexibility with various examples. These examples include both simple and more expressive numerical scoring schemes, and show how non-numerical schemes can be used to model attackers with different capabilities.

2.3.1 General Definitions

2.3.1.1 Security scoring schemes

A security scoring scheme assigns a score to each vertex v based on that vertex's access base. Given initial scores of vertices in V_{init}, the scoring scheme evaluates each access set in v's access base, assigning it an intermediate score. It then combines the intermediate scores of all access sets in v's access base into a single score for v. We use $\mathcal{P}_{\mathcal{M}}(S)$ to denote the set of multisets over S, and $\mathcal{P}(S)$ to denote the powerset of S. Moreover, $\{\!\{\ \}\!\}$ denotes a multiset.

Definition 17. *A* security scoring scheme *for an account access graph G is a 6-tuple* $(D, \preceq, V_{\text{init}}, \text{Init}, \text{Eval}, \text{Combine})$*, where:*

- *D is the domain over which scores are defined.*
- *$\preceq$ is a partial order relating elements in D.*
- *$V_{\text{init}} \subseteq V_G$ is called the set of* initial vertices*.*
- *$\text{Init} : V_{\text{init}} \to D$ maps initial vertices to* initial scores*.*
- *$\text{Eval} : \mathcal{P}_{\mathcal{M}}(D) \to D$ maps a multiset of initial scores (of vertices in an access set) to an* intermediate score *(for that access set).*

- Combine : $\mathcal{P}(D) \to D$ *maps a set of intermediate scores (of access sets in a vertex's access base) to a* score *(for that vertex).*

Given Eval*, we define an auxiliary function* $\text{EvalSet} : \mathcal{P}(V_{\text{init}}) \to D$ *that directly maps an access set to its intermediate score by first computing the initial scores of its vertices and then applying* Eval*.*

$$\text{EvalSet}(S) := \text{Eval}(\{\!\{\text{Init}(v') \mid v' \in S\}\!\}) \ .$$

We then define the following score function $\text{Score} : V_G \to D$*, which directly maps a vertex to its (final) score:*

$$\text{Score}(v) := \text{Combine}(\{\text{EvalSet}(S) \mid S \in \text{AccessBase}(v, V_{\text{init}})\}) \ .$$

We usually instantiate Eval with a maximum or sum function, which captures an *and*-semantics: All credentials in an access set are necessary to access an account. We usually instantiate Combine with a minimum function, which captures an *or*-semantics: Any one access set is sufficient to access an account.

A security scoring scheme should meaningfully measure the security of accounts. In particular, whenever one account is at least as secure as another, a scoring scheme should assign at least as high a score to the first account as to the second. In what follows, we formalize this requirement.

We define the notion of an account being *at least as secure* as another by implication. If an attacker that can access an account vertex v_B can also always access another account vertex v_A, then we say that v_B is at least as secure as v_A.

Example 8. *In Figure 2.3, consider an account where some features are available after only logging in with a password* pwd_{acc}*. We denote this basic access to the account by* $\text{acc}_{\text{basic}}$*. For security-critical features, a second factor* code *generated by a* device*, is necessary. We denote access to these features by* acc_{full}*. For* $V_{\text{init}} = \{\text{pwd}_{\text{acc}},\ \text{device}\}$*,*

$$\begin{aligned} &\text{AccessBase}(\text{acc}_{\text{basic}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{acc}}\}\}, \ \textit{and} \\ &\text{AccessBase}(\text{acc}_{\text{full}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{acc}},\ \text{device}\}\} \ . \end{aligned}$$

Whenever we can access acc_{full}*, we can also access* $\text{acc}_{\text{basic}}$*. Therefore,* acc_{full} *is at least as secure as* $\text{acc}_{\text{basic}}$*.*

Based on this intuition, we next define a relation on access bases. We will then use this relation to define a soundness condition for scoring schemes.

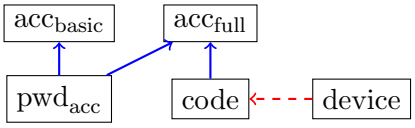

Figure 2.3

Definition 18. *Let v_A and v_B be vertices in an account access graph G. An access base for a vertex v_B is* at least as secure *as that for v_A if and only if any set of vertices V that provides access to v_B also provides access to v_A. Formally,*

$$\text{AccessBase}(v_A, V_{\text{init}}) \preceq \text{AccessBase}(v_B, V_{\text{init}}) :\Leftrightarrow$$
$$\forall\ V \subseteq V_{\text{init}} : v_B \in \text{accessFrom}(V) \rightarrow v_A \in \text{accessFrom}(V)\ .$$

We also write $A \prec B$ to denote $A \preceq B \wedge A \neq B$. We now state a necessary and sufficient condition for $A \preceq B$.

Theorem 3. *Let v_A and v_B be vertices in an account access graph G and let $V_{\text{init}} \subseteq V_G$ be given. Let $A := \text{AccessBase}(v_A, V_{\text{init}})$ and $B := \text{AccessBase}(v_B, V_{\text{init}})$. Then*

$$(\forall B_i \in B\ \exists A_j \in A : A_j \subseteq B_i) \Leftrightarrow A \preceq B\ .$$

Proof. We first show $\Rightarrow$: We want to show that, given the left-hand side, we have $A \preceq B$, that is, $\forall\ V \subseteq V_{\text{init}} : v_B \in \text{accessFrom}(V) \rightarrow v_A \in \text{accessFrom}(V)$.

Consider any $V \subseteq V_{\text{init}}$ such that $v_B \in \text{accessFrom}(V)$. Then, V must contain all elements of some set in the access base B, i.e., we must have $\exists B_k \in B : B_k \subseteq V$. Thus, from the left-hand side, we have $\exists A_j \in A : A_j \subseteq B_k \subseteq V \Rightarrow A_j \subseteq V$. Thus, V contains all elements of A_j, an element of the access base of A, giving us $v_A \in \text{accessFrom}(V)$.

To show $\Leftarrow$, consider $\neg(\forall B_i \in B\ \exists A_j \in A : A_j \subseteq B_i) \Rightarrow \neg(A \preceq B)$. The left-hand side can be rewritten as $\exists B_k \in B\ \forall A_j \in A : \neg(A_j \subseteq B_k)$. Consider such a B_k. We clearly have $v_B \in \text{accessFrom}(B_k)$ since B_k is a set in v_B's access base. We also have $\neg(v_A \in \text{accessFrom}(B_k))$ since B_k is not a superset of any set in v_A's access base.

Thus, $v_B \in \text{accessFrom}(B_k) \wedge \neg(v_A \in \text{accessFrom}(B_k))$ and hence

$$\exists V \subseteq V_{\text{init}} : v_B \in \text{accessFrom}(V) \wedge \neg(v_A \in \text{accessFrom}(V))\ ,$$

which is exactly $\neg(A \preceq B)$. □

Example 9. *In Example 8,* $\text{AccessBase}(\text{acc}_{\text{basic}}, V_{\text{init}}) \prec \text{AccessBase}(\text{acc}_{\text{full}}, V_{\text{init}})$ *, since* $\{\text{pwd}_{\text{acc}}\} \subseteq \{\text{pwd}_{\text{acc}}, \text{device}\}$ *, where* $\{\text{pwd}_{\text{acc}}, \text{device}\}$ *is the only element in* $\text{AccessBase}(\text{acc}_{\text{full}}, V_{\text{init}})$.

Based on this, we define soundness of a scoring scheme.

Definition 19. *A security scoring scheme* $(D, \preceq_S, V_{\text{init}}, \text{Init}, \text{Eval}, \text{Combine})$ *with score function* Score_S *is* sound *if, for any two vertices* v_A *and* v_B*,*

$$\text{AccessBase}(v_A, V_{\text{init}}) \preceq \text{AccessBase}(v_B, V_{\text{init}}) \Rightarrow \text{Score}_S(v_A) \preceq_S \text{Score}_S(v_B) \ .$$

We next give sufficient conditions for the Eval and Combine functions for a scoring scheme to be sound.

Theorem 4. *A security scoring scheme* $(D, \preceq_S, V_{\text{init}}, \text{Init}, \text{Eval}, \text{Combine})$ *is sound if the following two conditions hold.*

$$(1)\ \forall A, B \subseteq V_{\text{init}} : A \subseteq B \Rightarrow \text{EvalSet}(A) \preceq_S \text{EvalSet}(B) \ .$$

$$(2)\ \forall S, T \in \mathcal{P}(D) : (\forall T_i \in T\ \exists S_j \in S : S_j \preceq_S T_i) \Rightarrow \text{Combine}(S) \preceq_S \text{Combine}(T) \ .$$

Proof. Let v_A, v_B be two vertices in G. Let $A := \text{AccessBase}(v_A, V_{\text{init}})$ and $B := \text{AccessBase}(v_B, V_{\text{init}})$, and let $A \preceq B$. Let $S = \{\text{EvalSet}(A_i) \mid A_i \in A\}$ and $T = \{\text{EvalSet}(B_i) \mid B_i \in B\}$. We want to show

$$\text{Score}(v) = \text{Combine}(S) \preceq_S \text{Combine}(T) = \text{Score}(v') \ .$$

We show that, given conditions (1) and (2), as well as $A \preceq B$, that $\text{Combine}(S) \preceq \text{Combine}(T)$.

From $A \preceq B$ and Theorem 3, we have $\forall B_i \in B\ \exists A_j \in A : A_j \subseteq B_i$. Combining this with condition (1) yields $\forall T_i \in T\ \exists S_j \in S : S_j \preceq_S T_i$. From condition (2), this yields $\text{Combine}(S) \preceq_S \text{Combine}(T)$. □

2.3.1.2 Recoverability scoring schemes

We arrive at the definition for a recoverability scoring scheme directly by replacing the access base with the lockout base in Definition 17, and obtain analogous soundness definitions the same way.

Definition 20. *A* recoverability scoring scheme *is defined analogously to a security scoring scheme, with the score function operating on the lockout base rather than the access base:* $\text{Score}(v) := \text{Combine}(\{\text{EvalSet}(S) \mid S \in \text{LockoutBase}(v, V_{\text{init}})\})$.

Definition 21. *Let* v_A *and* v_B *be vertices in an account access graph* G*. A lockout base for* v_B *is* at least as recoverable *as that for* v_A *when being locked out of* v_B *implies being locked out of* v_A*.*

$$\text{LockoutBase}(v_A, V_{\text{init}}) \preceq \text{LockoutBase}(v_B, V_{\text{init}}) :\Leftrightarrow$$
$$\forall\ V \subseteq V_{\text{init}} : v_B \in \text{lockoutFrom}(V) \rightarrow v_A \in \text{lockoutFrom}(V) \ .$$

Definition 22. *A recoverability scoring scheme* $(D, \preceq_R, V_{\text{init}}, \text{Init}, \text{Eval}, \text{Combine})$ *with score function* Score_R *is* sound *if, for any two vertices* v_A *and* v_B*,*

$$\text{LockoutBase}(v_A, V_{\text{init}}) \preceq \text{LockoutBase}(v_B, V_{\text{init}}) \Rightarrow \text{Score}_R(v_A) \preceq_R \text{Score}_R(v_B) \ .$$

2.3.2 Concrete Instances

We give two numerical scoring schemes. The first assigns a single natural number to each vertex, while the second is based on multisets of numbers. In our examples, we give security scoring schemes, where a higher score denotes that an account is more resistant against compromise by an adversary. The scoring schemes can easily be adapted to recoverability scoring schemes operating on the lockout bases instead. In this case, a higher score denotes that an account is more resistant against accidental lockout in case the user loses credentials or access to other accounts. After presenting the two numerical scoring schemes, we demonstrate the usefulness of non-numerical scoring schemes to model different kinds of attackers.

2.3.2.1 Sum-then-min: A simple scoring scheme

We next give a simple scoring scheme that assigns a single natural number to each vertex.

Definition 23. *The* sum-then-min *scoring scheme is given as*

$$D = \mathbb{N},\ \preceq\ =\ \leq,\ \text{Eval}(M) = \sum_{m \in M} m,\ \text{Combine}(S) = \min_{s \in S}(s),$$

where $\leq$ *is standard comparison of natural numbers.*

Example 10. *We illustrate the scoring scheme with an example in Figure 2.4. We extend the running example introduced in Figure 2.1 by assigning initial security scores to the leaves.*

$$\begin{gathered}\text{Score}(\text{acc}_{\text{shop}}) = \text{Combine}(\text{EvalSet}(\{\text{pwd}_{\text{shop}}\}),\\ \text{EvalSet}(\{\text{pwd}_{\text{mail}},\ \text{device},\ \text{finger}\}), \text{EvalSet}(\{\text{pwd}_{\text{mail}},\ \text{device},\ \text{PIN}\})\ )\\ = \min(\{1, 1+2+2, 1+2+1\}) = \min(\{1, 5, 4\}) = 1\ .\end{gathered}$$

Unfortunately for many setups, this scoring scheme is too simplistic. We give an example that illustrates this.

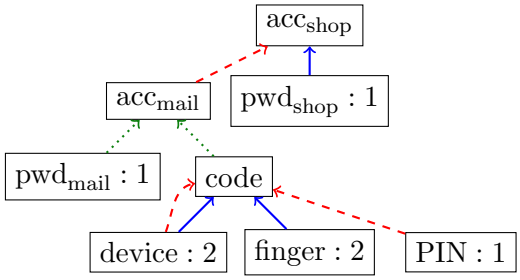

Figure 2.4

Example 11. *In the graph in Figure 2.5, account* acc_A *requires two passwords, while* acc_B *requires a* code *generated by a* device. *We assign initial security scores to the leaves, and get*

$$\text{Score}(\text{acc}_\text{A}) = \text{Score}(\text{acc}_\text{B}) = 2 \ .$$

Whether obtaining two passwords is more or less difficult than obtaining a device depends on many factors, and assigning the same score to both accounts oversimplifies the situation.

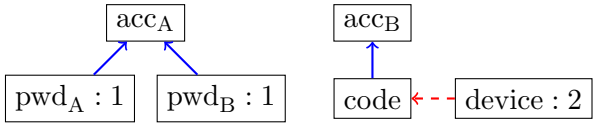

Figure 2.5

To solve this problem, we will give a more fine-grained scoring scheme next.

2.3.2.2 A multiset-based scoring scheme

We present a scoring scheme where each score is a set of multisets S. Each multiset in an account's score S represents an alternative access method for that account. Thus, this scoring scheme considers more details about an account's access sets than the previous scheme, which reduced the information from all different access sets to a single number.

Consider a multiset M that is associated with one access method. Each element in the multiset represents a credential as a single numeric value, where a higher number represents a credential that is more difficult to compromise. For example, $\{\!\{1, 1\}\!\}$ denotes an access method that requires two credentials with value 1 each. Then, the score S of an account represents all different access methods for that account, including

(transitively) the different access methods for accounts linked to S. For example, a score of $\{\{\!|1,1|\!\}, \{\!|2|\!\}\}$ denotes that the account can (transitively) be accessed either by two credentials with value 1 each, or a single credential with value 2.

For comparing scores, we first define a partial order on multisets. For two multisets M and N, $M \preceq N$ holds if and only if each number in M can be injectively mapped to a number in N that is at least as high. Then, M represents a less (or equally) secure access method than N.

Definition 24. *For two multisets M and N over $\mathbb{N}$, $M \preceq N$ if and only if $k = |M| \leq |N| = n$, and there exists an indexing of their elements such that $M = \{\!|m_1, \ldots, m_k|\!\}$, $N = \{\!|n_1, \ldots, n_n|\!\}$, and $\forall\ 1 \leq i \leq k : m_i \leq n_i$.*

Example 12. *$\{\!|1,1|\!\} \prec \{\!|1,2|\!\}$ and $\{\!|1,2|\!\} \prec \{\!|1,1,2|\!\}$, but $\{\!|1,2|\!\}$ and $\{\!|1,1,1|\!\}$ are incomparable.*

Note that this multiset ordering is specialized to our needs and differs from standard multiset orderings [10].

We next define a partial order on *sets of multisets* based on the same idea of implication for comparing access bases explained in Theorem 3. Consider a set of multisets S that represents a score for an account. Recall that each multiset in S is associated with an *alternative* access method. Thus, a set of multisets S_1 represents a lower score than another set S_2 if and only if, for every multiset in S_2, there is a multiset representing a less secure access method in S_1.

Definition 25. *For two sets of multisets S_1, S_2, $S_1 \preceq S_2$ if and only if $\forall N \in S_2\ \exists M \in S_1 : M \preceq N$.*

Note that the empty set is the highest possible score, since it denotes an account that cannot be accessed at all.

Example 13. *$\{\{\!|1,1|\!\}, \{\!|1,2|\!\}\} \prec \{\{\!|1,2|\!\}, \{\!|1,1,2|\!\}\}$, but $\{\{\!|1,2|\!\}\}$ and $\{\{\!|1,1|\!\}, \{\!|1,1,1|\!\}\}$ are incomparable.*

To define our scoring scheme, we will employ a definition of distributed product from [11] for Eval:

Definition 26. *The distributed product of two sets of multisets S_1 and S_2 is defined as*

$$S_1 \otimes S_2 = \{M \uplus N \mid M \in S_1, N \in S_2\}\ ,$$

where $\uplus$ *denotes multiset sum (union). Moreover, we define* $\bigotimes_{S_i \in S}$ *as the generalization of* $\otimes$ *with unit element* $\{\emptyset\}$*, where* S *is a multiset containing sets of multisets.*

Example 14.

$$\bigotimes \{\!\{\{\!\{1\}\!\}\}, \{\!\{\{\!\{2,2\}\!\}\}, \{\!\{\{\!\{3\}\!\}, \{\!\{4\}\!\}\}\}\!\} = \{\!\{\{\!\{1,2,2,3\}\!\}, \{\!\{1,2,2,4\}\!\}\}\!\} \ .$$

Definition 27. *The* sets of multisets *scoring scheme is given as*

- $D = \mathcal{P}_{\mathcal{M}}(\mathbb{N})$*, the sets of multisets over* $\mathbb{N}$*.*
- $\preceq$ *is given according to Definition 25.*
- $\text{Eval} = \bigotimes$*.*
- $\text{Combine}(S) = \text{minimal}(S_{\cup})$*, where* $S_{\cup} := \bigcup_{S_i \in S} S_i$*, and*

$$\text{minimal}(S_{\cup}) := \{M \mid M \in S_{\cup} \wedge \neg(\exists M' \in S_{\cup} : M' \prec M)\} \ .$$

Eval takes the distributed product of sets of multisets given in Definition 26. Combine then takes the minimal values of the union. For example, given a set of two multisets $\{M_1, M_2\}$ with $M_1 \prec M_2$, M_1 is a minimal value but M_2 is not, so $\text{minimal}(\{M_1, M_2\}) = \{M_1\}$.

This scoring scheme addresses the problem illustrated in Example 11, where two accounts with very different configurations were assigned the same score.

Example 15. *Consider the graph in Figure 2.6.*

$$\text{Score}(\text{acc}_\text{A}) = \text{Combine}(\text{EvalSet}(\{\text{pwd}_\text{A}, \text{pwd}_\text{B}\})) = \{\!\{1,1\}\!\}, \text{ and}$$
$$\text{Score}(\text{acc}_\text{B}) = \text{Combine}(\text{EvalSet}(\{\text{device}\})) = \{\!\{2\}\!\} \ .$$

Note that $\{\!\{1,1\}\!\}$ and $\{\!\{2\}\!\}$ are incomparable. Thus, the scoring scheme assigns incomparable scores to accounts whose security configurations should be considered incomparable.

While this scoring scheme is more fine-grained, it still requires assigning initial numerical scores to credentials. These numbers can be obtained from a risk analysis, but may not always be precise. We next present a scoring scheme with a non-numerical domain based on an attacker model, illustrating the flexibility of our formalism.

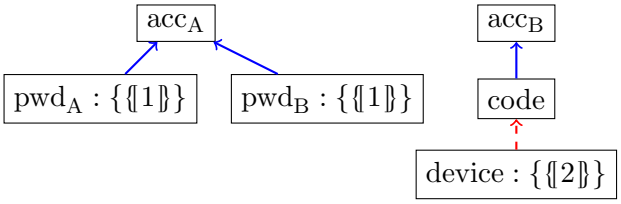

Figure 2.6

2.3.2.3 An attribute-based attacker model scoring scheme

An important measure of an account's security is which kinds of attackers could potentially compromise the account. We consider the *possibility* of compromise by different kinds of attackers. We associate each account with the weakest attacker that could potentially compromise that account. Note that possibility here only means that such a compromise could occur, but not necessarily that it is likely. For example, an account that is only protected by a password could be compromised by a *remote* attacker, while an account that requires authentication from a device not connected to the Internet could only be compromised by a *local* attacker.

We formalize these concepts with a security scoring scheme based on n attribute sets $A_1, \ldots, A_n$ modeling attacker capabilities. We model an attacker as an n-tuple $(a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$.

Example 16. *Consider the following attribute sets:*

- Location $= \{\text{rem}, \text{loc}\}$, *with* $\text{rem} < \text{loc}$.
- Skill $= \{\text{none}, \text{some}, \text{exp}\}$, *with* $\text{none} < \text{some} < \text{exp}$.

For Location, rem *means the attacker acts remotely, e.g., from another country, while* loc *means that the attacker is local, and may thus obtain access to physical devices owned by the user. The relation* $\text{rem} < \text{loc}$ *means that a local attacker has more capabilities than a remote one, that is, a local attacker could always also act like a remote attacker. For* Skill, none *means that the attacker has no special skills,* some *means that he has special skills, e.g., he can exploit known vulnerabilities, and* exp *denotes an expert hacker. For example, the tuple* $(\text{rem}, \text{some})$ *models a remote attacker with special skills.*

We next give an ordering on such attacker tuples.

Definition 28. *Let $A_1, \ldots, A_n$ be totally ordered attribute sets with order relations $\leq_1, \ldots, \leq_n$. Then, for $t, u \in A_1 \times \cdots \times A_n$, we define the following partial order $\preceq$ on tuples: $t \preceq u$ if and only if each component of t is less than or equal to the corresponding component in u. That is:*

$$t \preceq u :\Leftrightarrow \forall 1 \leq i \leq n : proj_i(t) \leq_i proj_i(u) \ ,$$

where $proj_i(t)$ maps t to its i-th component.

We will define a scoring scheme that denotes the minimal attribute values an attacker must have to compromise a credential or account. The values assigned to vertices by the scoring scheme are *sets of attribute tuples*. Each tuple denotes one kind of attacker who could compromise the account. An attacker who has only better attributes as denoted in a tuple could also compromise the account.

Example 17. *An account with a score of* $\{(\text{rem}, \text{some}), (\text{loc}, \text{none})\}$ *could be compromised by a remote attacker with special skills, represented by* $(\text{rem}, \text{some})$*, or by a local attacker without any special skills, represented by* $(\text{loc}, \text{none})$*. An attacker stronger than at least one of these options, for example a remote attacker with expert skills,* (rem, exp)*, or a local attacker with some skills,* $(\text{loc}, \text{some})$*, could also compromise the account. However, a remote attacker without special skills,* $(\text{rem}, \text{none})$*, cannot compromise the account.*

We can also compare sets of attribute tuples as follows. Consider a set of attacker tuples S_2. Another set S_1 is smaller or equal to S_2 if and only if for each attacker tuple in S_2, there is a tuple representing a weaker (or equal) attacker, in S_1. That is, any attacker that could compromise the account with score S_2 could also compromise the account with score S_1.

Definition 29. *For $S_1, S_2 \in \mathcal{P}(A_1 \times \cdots \times A_n)$, $S_1 \preceq S_2$ if and only if $\forall\ u \in S_2\ \exists\ t \in S_1 : t \preceq u$.*

This ordering is analogous to the ordering for sets of multisets. The empty set is the highest possible score, since it denotes that *no* attacker could ever compromise the account.

Definition 30. *Let $A_1, \ldots, A_n$ be totally ordered attribute sets. We define the* attacker attribute *security scoring scheme based on these attribute sets as follows:*

- $D = \mathcal{P}(A_1, \ldots, A_n)$, *sets of attribute tuples.*
- $\preceq$ *is given according to Definition 29.*
- $\text{Eval}(M) = \text{compMax}(M_\cup)$, *where* $M_\cup := \bigcup_{M_i \in M} M_i$, *and* compMax *is a singleton set that contains the component-wise maximum of tuple set* $M_\cup$.
- $\text{Combine}(S) = \text{minimal}(S_\cup)$, *where* $S_\cup := \bigcup_{S_i \in S} S_i$, *and*

$$\text{minimal}(S_\cup) := \{t \mid t \in S_\cup \wedge \neg(\exists t' \in S_\cup : t' \prec t)\} \ .$$

Note that minimal is defined analogous to the sets of multisets scoring scheme given in Definition 27.

Example 18. *In Figure 2.7, we extend the graph from Figure 2.1 by assigning attacker tuple security scores to the leaves.*

$$\begin{aligned}
&\text{Score}(\text{acc}_{\text{shop}}) = \text{Combine}(\ \text{EvalSet}(\{\text{pwd}_{\text{shop}}\}),\\
&\text{EvalSet}(\{\text{pwd}_{\text{mail}}, \text{device}, \text{finger}\}), \text{EvalSet}(\{\text{pwd}_{\text{mail}}, \text{device}, \text{PIN}\})\ ) =\\
&\text{Combine}(\{(\text{rem}, \text{some})\}, \{(\text{loc}, \text{exp})\}, \{(\text{loc}, \text{some})\}) =\\
&\text{minimal}(\{(\text{rem}, \text{some}), (\text{loc}, \text{exp}), (\text{loc}, \text{some})\}) = \{(\text{rem}, \text{some})\} \ .
\end{aligned}$$

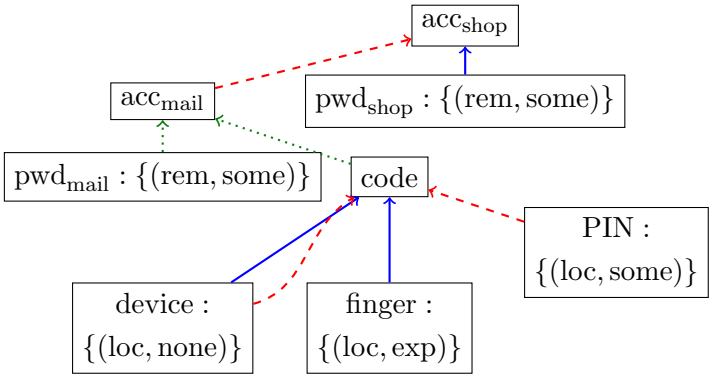

Figure 2.7

That is, a remote attacker with special skills could potentially compromise the acc_{shop} account.

2.3.2.4 A simplified attacker model scoring scheme

Our previous scoring scheme showed the flexibility of our definitions in theory, but the concrete example we gave is still not perfectly suited for practical applications. In particular, we may not always correctly assess an attacker's skill. We may be more interested in an attacker with a set of clearly defined abilities, such as an attacker who can intercept text messages, but do nothing else, or an attacker who can steal physical devices but not compromise secrets.

Such simplified attacker models are expressed by the following security scoring scheme.

Definition 31. *The* single attacker vulnerability *scoring scheme is given as*

$$D = \{\text{Vulnerable}, \text{NotVulnerable}\},\ \text{Eval}(M) = \max_{m \in M}(m),\ \text{Combine}(S) = \min_{s \in S}(s),$$

where Vulnerable $\prec$ NotVulnerable*, and* max *and* min *are defined according to this relation.*

This scoring scheme allows analyzing whether a credential or account is vulnerable against one particular kind of attacker. In our user study described in Chapter 3, we employed different instances of this scheme with respect to different attackers. We explain these instances in more detail in Section 3.1. The instances differ only in the definition of initial scores. For example, a scoring scheme that models an attacker who can compromise text messages but nothing else would assign an initial score of Vulnerable to text messages and a score of NotVulnerable to every other initial vertex.

2.3.3 Inherent account vulnerabilities and lockout risk

Our modeling and analysis so far was based on the assumption that an account can be accessed only by authenticating, i.e., by presenting credentials or proof of access to other accounts. In reality, services may have vulnerabilities that allow an attacker to circumvent a service's authentication methods. We can model such vulnerabilities by also including intermediate vertices representing accounts in V_{init}, and assigning initial scores to them. An account's initial score denotes how difficult it is for an attacker to compromise it directly, without compromising connected credentials or accounts.

Example 19. *In Figure 2.8, we extend the example given in Figure 2.4 by assigning initial security scores also to the intermediate vertices, setting* $V_{\text{init}} = V_G$. *Note that* $\text{AccessBase}(v, V_G) = \text{MinAccessTo}(v)$. *We assign an initial score of* 3 *to* acc_{shop} *and* acc_{mail}, *measuring the risk of direct account compromise through a service vulnerability. We assign an initial score of* 5 *to* code, *which means that we deem it difficult to compromise the* code *without access to the unlocked* device. *We then compute the access bases, which also contain intermediate vertices:*

$$\text{AccessBase}(\text{acc}_{\text{mail}}, V_G) =$$
$$\{\{\text{acc}_{\text{mail}}\}, \{\text{pwd}_{\text{mail}}, \text{code}\}, \{\text{pwd}_{\text{mail}}, \text{device}, \text{finger}\}, \{\text{pwd}_{\text{mail}}, \text{device}, \text{PIN}\}\ \}\ .$$
$$\text{AccessBase}(\text{acc}_{\text{shop}}, V_G) =$$
$$\{\{\text{acc}_{\text{shop}}\}, \{\text{acc}_{\text{mail}}\}, \{\text{pwd}_{\text{shop}}\}, \{\text{pwd}_{\text{mail}}, \text{code}\},$$
$$\{\text{pwd}_{\text{mail}}, \text{device}, \text{finger}\}, \{\text{pwd}_{\text{mail}}, \text{device}, \text{PIN}\}\ \}\ .$$

From these sets, we then compute the final security scores.

$$\text{Score}(\text{acc}_{\text{mail}}) = \min(\{3, 6, 5, 4\}) = 3\ .$$
$$\text{Score}(\text{acc}_{\text{shop}}) = \min(\{3, 3, 1, 6, 5, 4\}) = 1\ .$$

The score of 3 *for* acc_{mail} *is less than the score of* 4 *obtained previously.* $\text{Score}(\text{acc}_{\text{mail}}) = \text{Init}(\text{acc}_{\text{mail}})$, *i.e., the final score computed for* acc_{mail} *is equal to its initial score. This shows that the highest risk for the e-mail account is direct compromise, without compromising the user's credentials.*

The score of 1 *for* acc_{shop} *is the same as the score obtained previously.* $\text{Score}(\text{acc}_{\text{shop}}) < \text{Init}(\text{acc}_{\text{shop}})$, *i.e., the final score computed for* acc_{shop} *is lower than its initial score. This shows that compromise of the credentials is the highest risk for the web shop account (in particular, the credential* pwd_{shop}*).*

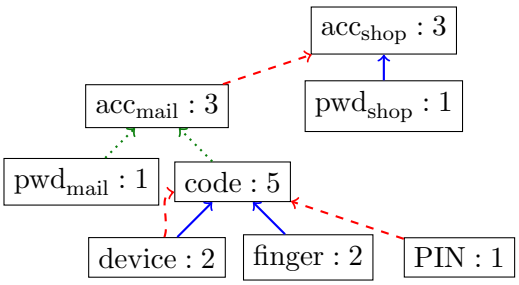

Figure 2.8

We can similarly model inherent lockout risk with recoverability scoring schemes by including intermediate vertices in V_{init}. For inherent account vulnerabilities, we considered additional *compromise* possibilities by giving the attacker more options: he might compromise an account without possessing sufficient credentials. For recoverability, we instead consider additional *lockout* possibilities by giving the user fewer options: she might not be able to access an account despite having sufficient credentials. The difference can be subtle, and we illustrate it with the following examples.

Example 20. *In the graph from Figure 2.9, there is an account* acc_{SSO} *with a service that provides single sign-on, and an account* acc_{shop} *with a web shop using this single sign-on. For* $V_{\text{init}} := \{\text{pwd}_{\text{SSO}}, \text{acc}_{\text{SSO}}\}$*,*

$$\text{AccessBase}(\text{acc}_{\text{shop}}, V_{\text{init}}) = \text{LockoutBase}(\text{acc}_{\text{shop}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{SSO}}\}, \{\text{acc}_{\text{SSO}}\}\ \}\ .$$

Note the different interpretation of the set $\{\text{acc}_{\text{SSO}}\}$ *in the access base and the lockout base. In the access base, the set denotes that an attacker who directly compromises* acc_{SSO} *can access* acc_{shop}*. In the lockout base, it denotes that, when* acc_{SSO} *is (inherently) unavailable, the user is locked out of* acc_{shop}*.*

We apply the sum-then-min scoring scheme as a recoverability scoring scheme, and assign a score of 1 *to* pwd_{SSO}*, and a score of* 2 *to* acc_{SSO}*. Then, the final score for* acc_{shop} *is* 1*. This means that the highest lockout risk is due to the user losing or forgetting the password, and not due to a service provider shutdown. If we added recovery mechanisms to* acc_{SSO}*, then this could change, and service provider shutdown could become the greatest lockout risk.*

Figure 2.9

Example 21. *In the graph from Figure 2.10, there is an account* acc_{A} *with a password* pwd_{A} *that is saved in a password* manager*. For* $V_{\text{init}} := \{\text{manager}, \text{pwd}_{\text{A}}\}$*,*

$$\text{AccessBase}(\text{acc}_{\text{A}}, V_{\text{init}}) = \text{LockoutBase}(\text{acc}_{\text{A}}, V_{\text{init}}) = \{\{\text{manager}\}, \{\text{pwd}_{\text{A}}\}\ \}\ .$$

There is again a different interpretation of the set $\{\text{pwd}_{\text{A}}\}$ *in the access base and the lockout base. In the access base, the set denotes that an attacker who directly compromises* pwd_{A} *can access* acc_{A}*. In the lockout base, however, it reflects the possibility that* pwd_{A} *could be unavailable such that it could not be retrieved* even with *access to*

the password manager. *This models the accidental deletion of* pwd_A *from* manager*'s database. If this scenario is considered unlikely, then* pwd_A *should be assigned a higher initial recoverability score than passwords that are not stored in a password manager, and could thus be forgotten. If this scenario is even considered impossible, then* pwd_A *should not be included in* V_{init} *for computing lockout sets, and thus not be assigned an initial recoverability score at all.*

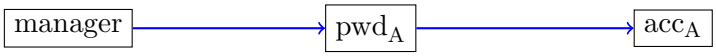

Figure 2.10

2.3.4 Vertex evaluation functions and centrality

The presented definitions for security and recoverability scoring schemes are useful when evaluating a user's account setups with respect to these properties. However, despite their flexibility, there may be properties of a user's accounts and credentials that do not naturally fit these definitions. In particular, a vertex's score only depends on its access or lockout base, but not on the rest of the graph. This restriction may be undesirable in certain cases where we require additional context. As an example, consider the notion of vertex *centrality* in a graph, which we use for evaluating the graphs of our study participants in Chapter 3. We wish to measure a vertex's centrality with respect to the entire account access graph it appears in. Thus, it is not sufficient to consider only its access or lockout base, and this notion cannot be expressed as a scoring scheme.

To express such notions, we give the following general definition of *vertex evaluation functions*. The value a vertex is assigned with respect to such a function can depend also on the other vertices in the graph, enabling a more contextual analysis of the vertex within the user's setup.

Definition 32. *A* vertex evaluation function *with domain* D *is a function that maps a graph* G*, a vertex* $v \in G$*, and a set of vertices* V_{init} *to a value in* D*.*

V_{init} defines the set of initial vertices. Its interpretation may depend on the given function; including this set as a parameter allows for evaluation functions that depend on vertices' access or lockout bases, since these are defined with respect to a given V_{init}.

We next define the centrality evaluation function we use in Chapter 3. This function aims to evaluate how central a given vertex is within its account access graph. The

idea is that a vertex that occurs in the access sets of many other vertices' access bases has a high centrality: Access to the vertex helps accessing many other vertices in the graph. For example, a device with many open sessions or a password manager with many saved passwords would be a central vertex in a graph.

This notion of centrality is transitive in the following sense: A vertex does not necessarily need many outgoing edges to be considered central; it could also, directly or indirectly, provide access to other vertices with many outgoing edges. For example, a master password for a password manager could also receive a high centrality value.

The formal function definition makes use of the multiset of *all access sets* in a graph, across all vertices' access bases. This multiset contains the same access set once for each time it occurs in the access base of a vertex.

Definition 33.

$$\text{AllAccessSets}(G, V_{\text{init}}) := \biguplus_{v \in V_G} \text{AccessBase}(v, V_{\text{init}}) \ .$$

Furthermore, we are interested in those access sets that contain a specific vertex v. For this, we provide the following auxiliary definition.

Definition 34.

$$\text{AccessSetsWithVertex}(G, V_{\text{init}}, v) := \{ S \mid S \in \text{AllAccessSets}(G, V_{\text{init}}) \ \wedge \ v \in S \ \} \ .$$

We can now formally define the *centrality* evaluation function. This function counts how often a vertex appears in access sets, weighted inversely proportionally to the access sets' sizes. Note that this function will be 0 for vertices that are not part of V_{init}, and at least 1 otherwise, since $\{v\}$ is always an access set for a vertex $v \in V_{\text{init}}$.

Definition 35. *The* vertex centrality evaluation function *is a vertex evaluation function with domain $\mathbb{Q}$, given as follows:*

$$\text{vertexCentrality}(G, V_{\text{init}}, v) := \sum_{S \in \text{AccessSetsWithVertex}(G, V_{\text{init}}, v)} \frac{1}{|S|} \ .$$

Example 22. *Consider the account access graph in Figure 2.11.*

Let $V_{\text{init}} := \{\text{manager}, \text{device}, \text{pwd}_\text{A}, \text{pwd}_\text{B}\}$. *The two passwords for accounts* acc_A *and* acc_B *are stored in the password* manager*, and* acc_B *additionally requires a* code *generated on the* device. *The password* manager *occurs in a singleton access set* {manager}

for itself as well as for pwd_A*,* pwd_B*, and* acc_A*, which each contribute* 1 *to its centrality value. It also occurs in* $\{\text{manager}, \text{device}\}$ *for* acc_B*, which contributes another* 0.5*. Therefore, the centrality value for the password* manager *is* 4.5*. The* device *occurs in two singleton access sets* $\{\text{device}\}$*, for itself and for the* code*, which each contribute* 1 *to its centrality value, and in the access sets* $\{\text{manager}, \text{device}\}$ *and* $\{\text{pwd}_\text{B}, \text{device}\}$ *for* acc_B*, which each contribute* 0.5*. Thus, the* device *has a centrality value of* 3*.*

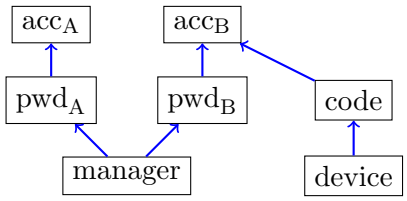

Figure 2.11

As this example shows, a single account with a password contributes 2 to the password manager's centrality value, which may be unexpected. However, the purpose of this definition is to compare the centrality across different vertices in a graph and obtain an order. Thus, the actual centrality value is less meaningful than how it compares to that of other vertices. In particular, centrality values are a good way to identify critical parts of a user's setup, as we will show in our user study results in Section 3.3.

2.4 Identifying Account Setup Weaknesses

In this section, we show how to leverage scoring schemes to answer concrete questions about an account setup, such as which accounts contain *backdoors*, or whether more important accounts are always better secured than less important accounts. That is, we discover critical weaknesses without manually analyzing each score.

Formally, we define *predicates*, which are Boolean-valued functions on vertices that evaluate to true when there is a potential weakness. We will give example predicates for concrete weaknesses, but note that this is a general definition that can be instantiated to answer many relevant questions about account setups.

2.4.1 General Definitions

We give two kinds of predicates: *scoring scheme* predicates and *direct* predicates. The former require a scoring scheme and identify weaknesses based on accounts' scores. The

latter identify weaknesses directly with respect to accounts' access and lockout bases.

Definition 36. *A* scoring scheme predicate *on n vertices is a function $P_{S,\mathrm{Add}} : V_G^n \to \{\mathrm{true}, \mathrm{false}\}$ that is defined with respect to a scoring scheme S and additional information* Add*, and takes as input an n-tuple of vertices (with $n \geq 1$, where a 1-tuple is a single vertex).*

Definition 37. *A* direct predicate *on n vertices is a function $P_{V_{\mathrm{init}},\mathrm{Add}} : V_G^n \to \{\mathrm{true}, \mathrm{false}\}$ that is defined with respect to a set of initial vertices V_{init} and additional information* Add*, and takes as input an n-tuple of vertices (with $n \geq 1$, where a 1-tuple is a single vertex).*

2.4.2 Concrete Instances

We next give examples of both kinds of predicates and discuss why both definitions are useful in different situations.

2.4.2.1 Recovery paths and backdoors

One kind of weakness in an account access graph is a *backdoor*. An account has a backdoor when it can be accessed more easily using recovery access methods than using its primary authentication method.

We formalize this notion with a scoring scheme predicate. For this, we first explicitly define the subset of edges that are associated with recovery methods. We then consider for an account access graph G a reduced version G' of that graph that does not contain the recovery edges. An account has a backdoor with respect to a security scoring scheme if its score in G is lower than its score in G', i.e., it is easier to access this account by using at least one recovery method.

Definition 38. *Let S be a security scoring scheme with scoring function* Score *and $E_{\mathrm{rec}} \subsetneq E_G$ a set of edges used in recovery methods. Let G' be the graph obtained from G by removing the edges in E_{rec}, $G' := (V_G, E_G \setminus E_{\mathrm{rec}}, C_G)$. Then, we define the following scoring scheme predicate.* $\mathrm{HasBackdoor}_{S,E_{\mathrm{rec}}}(v) := \mathrm{Score}_G(v) \prec \mathrm{Score}_{G'}(v)$.

Example 23. *In the account access graph G given in Figure 2.12, we evaluate the predicate* $\mathrm{HasBackdoor}(\mathrm{acc}_{\mathrm{bank}})$*. We apply the sum-then-min security scoring scheme S with V_{init} containing all leaves. Let $E_{\mathrm{rec}} := \{(\mathrm{acc}_{\mathrm{mailA}}, \mathrm{acc}_{\mathrm{bank}}, \mathrm{red}), (\mathrm{acc}_{\mathrm{mailB}}, \mathrm{acc}_{\mathrm{mailA}}, \mathrm{red})\}$.*

Then, the backdoor predicate is evaluated as follows.

$\text{AccessBase}_G(\text{acc}_{\text{bank}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{bank}}, \text{device}\}, \{\text{pwd}_{\text{mailA}}, \text{device}\}, \{\text{pwd}_{\text{mailB}}\}\ \}$.
$\text{AccessBase}_{G'}(\text{acc}_{\text{bank}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{bank}}, \text{device}\}\}$.
Thus, $\text{HasBackdoor}_{S,E_{\text{rec}}}(\text{acc}_{\text{bank}})$ *since* $\text{Score}_G(\text{acc}_{\text{bank}}) = 1 < 3 = \text{Score}_{G'}(\text{acc}_{\text{bank}})$.

The backdoor of acc_{bank} is not directly due to its recovery method, using $\text{acc}_{\text{mailA}}$, but due to the recovery method of $\text{acc}_{\text{mailA}}$, using $\text{acc}_{\text{mailB}}$. Thus, the predicate also identifies indirect backdoors.

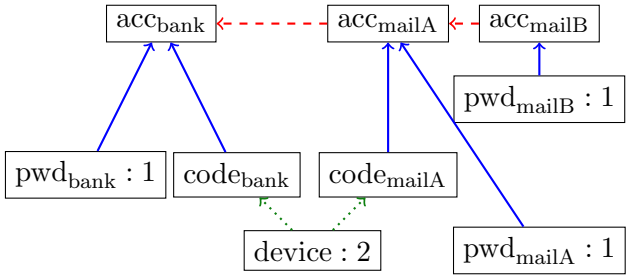

Figure 2.12

From a recoverability point of view, we are interested in the effectiveness of recovery methods. The analogous weakness to backdoors are *ineffective* recovery methods, namely those that do not actually improve an account's recoverability. We define this as another scoring scheme predicate.

Definition 39. *Let R be a recoverability scoring scheme with scoring function* Score *and $E_{\text{rec}} \subsetneq E_G$ a set of edges used in recovery methods. Let G' be given as in Definition 38. Then, we define the scoring scheme predicate* IneffectiveRec *as follows:*

$$\text{IneffectiveRec}_{R,E_{\text{rec}}}(v) := \text{Score}_G(v) \preceq \text{Score}_{G'}(v) \ .$$

This predicate identifies accounts whose recoverability scores in G, the graph that contains recovery methods, are not better than in G', the graph without any recovery methods. Thus, the account's recovery methods are ineffective.

Example 24. *In the graph in Figure 2.13, the* device *provides access to two-factor codes as well as to a password* manager. *Let* $E_{\text{rec}} := \{(\text{acc}_{\text{mail}}, \text{acc}_{\text{shop}}, \text{red})\}$, $V_{\text{init}} := \{\text{pwd}_{\text{mail}}, \text{pwd}_{\text{shop}}, \text{device}\}$. *We apply the sum-then-min recoverability scoring scheme. We assign an initial score of* 1 *to* pwd_{mail} *and an initial score of* 2 *to the* device.

However, we assign an initial score of 3 *to* pwd_{shop}*, as its initial score reflects the unlikely scenario of the password being deleted from the password* manager*'s database, as explained in Example 21.*

We obtain the following lockout bases.

$$\text{LockoutBase}_G(\text{acc}_{\text{shop}}, V_{\text{init}}) = \{\{\text{device}\}, \{\text{pwd}_{\text{mail}}, \text{pwd}_{\text{shop}}\}\} \ .$$
$$\text{LockoutBase}_{G'}(\text{acc}_{\text{shop}}, V_{\text{init}}) = \{\{\text{device}\}, \{\text{pwd}_{\text{shop}}\}\} \ .$$

This results in $\text{Score}_G(\text{acc}_{\text{shop}}) = \text{Score}_{G'}(\text{acc}_{\text{shop}}) = 2$ *and thus*

$$\text{IneffectiveRec}_{R,E_{\text{rec}}}(\text{acc}_{\text{shop}}) \text{ is true.}$$

The reason is that device loss is the highest lockout risk with or without the recovery method. The recovery method only helps in case pwd_{shop} would be deleted from the password manager's database, a scenario we deemed unlikely.

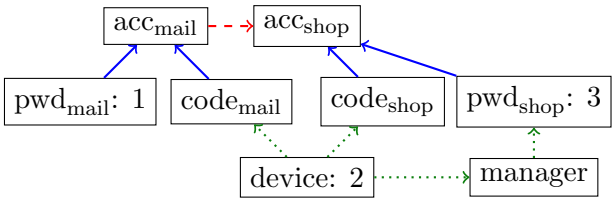

Figure 2.13

We note that there is an important difference in these predicates. The *backdoor* predicate only identifies a weakness when two scores *differ*, while the *ineffective recovery* also identifies a weakness when two scores are the *same*. This informs which kinds of scoring schemes we wish to use for these predicates to avoid false positives. The backdoor predicate should be based on a *coarse-grained* scoring scheme such that a change in score displays a real difference in security. In contrast, the ineffective recovery predicate should be based on a *fine-grained* scoring scheme such that *no* change in score means that there is no meaningful improvement.

Another possibility is to define the ineffective recovery predicate as a *direct predicate* based on the lockout bases directly instead, as follows.

Definition 40. *Let* $E_{\text{rec}} \subsetneq E_G$ *be a set of edges used in recovery methods and* V_{init} *a set of vertices that the user could get locked out of directly. Let* G' *be given as in*

Definition 38. Then, we define the scoring scheme predicate IneffectiveRecDirect *as follows:*

$$\text{IneffectiveRecDirect}_{V_{\text{init}},E_{\text{rec}}}(v) :=$$
$$\text{LockoutBase}(v, V_{\text{init}})_G = \text{LockoutBase}(v, V_{\text{init}})_{G'} \ .$$

That is, we only consider a recovery method ineffective if removing it does not change the accounts' lockout base at all.

Example 25. *In Figure 2.14,* $\text{pwd}_{\text{reused}}$ *is reused for both* acc_{shop} *and* acc_{mail}*, and* acc_{mail} *can be used to recover* acc_{shop}*. Let* $E_{\text{rec}} := \{(\text{acc}_{\text{mail}}, \text{acc}_{\text{shop}}, \text{red})\}$ *,* $V_{\text{init}} := \{\text{pwd}_{\text{reused}}, \text{acc}_{\text{mail}}\}$ *.*

Then

$$\text{LockoutBase}_G(\text{acc}_{\text{shop}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{reused}}\}\} \ .$$
$$\text{LockoutBase}_{G'}(\text{acc}_{\text{shop}}, V_{\text{init}}) = \{\{\text{pwd}_{\text{reused}}\}\} \ .$$

That is, the lockout bases are equal, and $\text{IneffectiveRecDirect}_{V_{\text{init}},E_{\text{rec}}}(\text{acc}_{\text{shop}})$ *is true. Including* acc_{mail} *in* V_{init} *opens up the possibility that* acc_{mail} *could be unavailable inherently, but this is irrelevant: if the user still knows* $\text{pwd}_{\text{reused}}$*, she can access* acc_{shop} *directly, and if she has forgotten the password, she cannot access* acc_{mail} *anyway.*

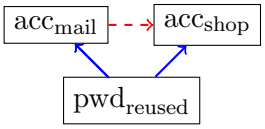

Figure 2.14

2.4.2.2 Account importance

Different accounts are of varying value and importance for the user. For example, an online banking account may be more valuable than other accounts. A risk analysis may indicate that a lower score is acceptable for a less important account. When importance values are assigned to each account, we can then combine this information with a scoring scheme to discover inconsistencies.

Definition 41. *Let S be a security or recoverability scoring scheme with a scoring function* Score *and let $I : V \to D_I$ be a function that assigns to a vertex an importance value from a partially ordered domain D_I. Then, the scoring scheme predicate* Inconsistent *is given as follows.*

$$\text{Inconsistent}_{S,I}(v_1, v_2) := I(v_1) \succ I(v_2) \wedge \text{Score}(v_1) \preceq \text{Score}(v_2) \ .$$

That is, v_1 represents a more important account than v_2, but receives the same or a lower score value.

The definition is sensible for both security and recoverability scoring schemes, since it only depends on more important accounts receiving higher scores.

Example 26. *In Figure 2.15, we use the attacker security scoring scheme S given in Definition 30 and $D_I = \{\text{low}, \text{med}, \text{high}\}$ with the expected ordering. There are two accounts, which both require two-factor authentication. However,* acc_{bank}*, the more important account, can also be accessed with the* answers *to security questions.*

$$I(\text{acc}_{\text{bank}}) = \text{high} \succ \text{med} = I(\text{acc}_{\text{shop}}), \textit{ but}$$
$$\text{Score}(\text{acc}_{\text{bank}}) = \{(\text{rem}, \text{some})\} \prec \{(\text{loc}, \text{some})\} = \text{Score}(\text{acc}_{\text{shop}}) \ .$$
$$\textit{Thus, } \text{Inconsistent}_{S,I}(\text{acc}_{\text{bank}}, \text{acc}_{\text{shop}}) \textit{ is true.}$$

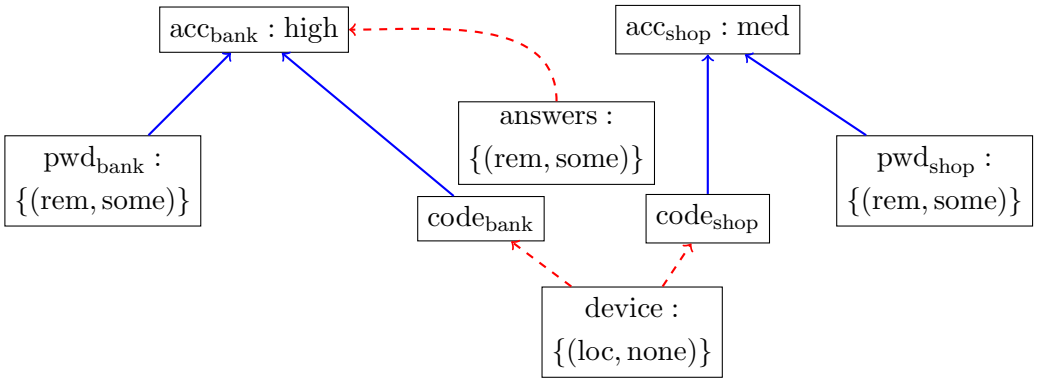

Figure 2.15

That is, even though acc_{bank} is the more important account compared with acc_{shop}, it is less secure.

2.5 Tooling

We have implemented all described algorithms and functions in the functional programming language Haskell [12]. The function declarations in the Haskell code are very close to the definitions in the thesis, making the correspondence between the code and the definitions apparent. Therefore, a functional programming language such as Haskell is a good choice for implementing these functions. The Haskell code is available at [13].

Furthermore, we have implemented an accompanying front-end tool in Java, using JSP (JavaServer Pages) technology [14]. The front-end source code is available at [15]. This front-end provides a GUI (graphical user interface) that enables entering users' account data to generate account access graphs. It automatically generates Haskell files that are executed by the Haskell back-end to perform an automated analysis. The analysis results are then reported back to the front-end, where they are displayed to the user.

The front-end tool consists of a *client interface* and an *admin interface*. In the client interface, users' account setup data can be entered. The admin interface allows the definition of credential, device, and account types, which simplify entering information. For example, default scores for each type can be defined, such that each vertex of that type will have this score. Note that the notion of types is limited to the front-end and not part of our formalism; we intentionally did not introduce vertex types into our formalism to keep it simple and flexible. The client interface allows to enter vertices by picking one of the vertex types previously defined in the admin interface. It also allows for the manual entry of vertices that do not fit one of these types. We show an overview of the front-end tool in Figure 2.16. Account graphs are visualized using the mxGraph Java library [16] and persisted or exported as simple JSON structures. These structures could also be interpreted by other tools. For example, the graphs shown in Chapter 3 have been obtained by exporting the front-end tool data into another tool with a different drawing style. An interesting direction for future work would be to standardize an account access graph JSON structure to enable interoperability across different tools operating on them.

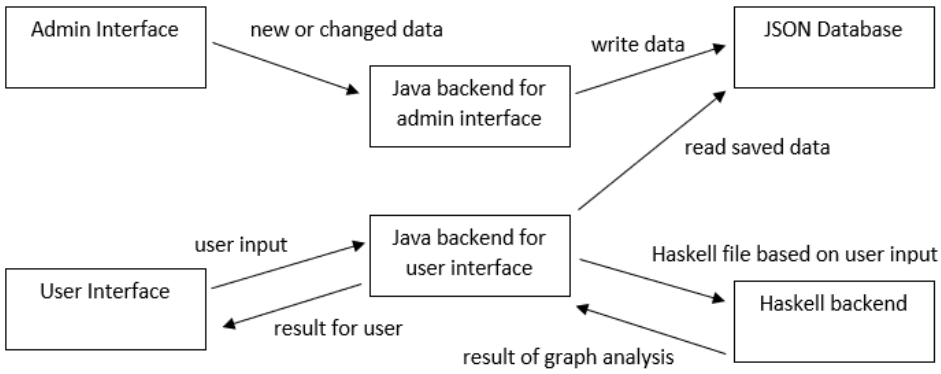

Figure 2.16: Overview of the front-end tool components and their interfaces. Note that we refer to "Java backend" to denote server-side components, but refer to the overall Java code as front-end as opposed to the Haskell back-end.

2.6 Related Work

We consider three kinds of related work: (i) graph-based threat modeling, (ii) logic programming, and (iii) end-user authentication and account recovery in general.

There are many graph-based threat modeling formalisms that are related to our model, originating from safety engineering techniques such as fault trees [17]. The most popular model used for threat modeling today is attack trees [18, 19]. Different formalizations of attack trees have been given, such as that by Mauw et al. [11]. The survey by Kordy et al. [20] provides an extensive overview of attack and defense models based on directed acyclic graphs (DAGs). Attack trees are the formalism that most closely resembles ours, so we compare the formalisms in more detail.

2.6.1 Comparison to attack trees

First, we note that our formalism is specific to the domain of user accounts, whereas attack trees more generally refine a high-level attacker goal into sub-goals and attacks. Nevertheless, there are similarities; accessing an account can be an attacker's goal, and accessing other accounts or credentials can be interpreted as the attacker's sub-goals.

We next investigate the similarities and differences between the two formalisms in more detail.

We start with outlining the *differences* on a high level to establish the strengths of our formalism with respect to the domain of user accounts, showing that attack trees would not be sufficient for the kind of analyses we wish to perform.

- Account access graphs *can contain cycles*, whereas attack trees, as their name suggests, must be acyclic. Our formalism naturally handles these cycles; the general definition of V_{init} even allows analysis of graphs that do not have any leaves.
- Account access graphs *do not have a hierarchical structure*, i.e., there is no vertex corresponding to the top-level goal of attack trees. While one might be particularly interested in a specific account, our analysis considers all accounts in the graph simultaneously.
- Account access graphs admit the analysis of *both security and recoverability* aspects, where only the security aspect could easily be represented by an attack tree. Attack trees are not a natural fit for recoverability analysis, which is not defined with respect to an attacker.

We next perform a more in-depth comparison on aspects where *similarities* are present between our formalism and attack tree formalizations, initially given by Mauw et al. [11] and expanded upon since. We also point out subtle differences between the formalisms, showing how even similar definitions are nevertheless not exact mappings of one another.

2.6.1.1 Only leaves as initial vertices

In what follows, we consider a subset of account access graphs and accompanying definitions that allow us to focus on the similarities between the formalisms. Concretely, we consider account access graphs that are *acyclic*, i.e., trees, and consider *security scoring schemes* where V_{init} consists of the *leaves*. For this case, some of our definitions have close similarities to the attack tree formalism introduced in [11].

Basic definitions. An *access set* related to an account corresponds to an *attack*, realizing the goal of compromising the account, but with the notable difference that

an attack is a *multiset*. A set of access sets corresponding to an *attack suite*, showing different ways how an attacker could compromise the account.

An access set's *elements*, vertices in V_{init}, correspond to *attack components*, which *"[...] are at the lowest level of abstraction that we consider and thus have no internal structure."* [11] Attack components are also identified with leaves in an attack tree.

Scoring schemes and attributes. The Init function of a security scoring scheme corresponds to the un-extended *attribute* function α in attack trees: Init assigns initial scores to vertices in V_{init}, whereas the attribute function assigns values to attack components.

The scoring function Score corresponds to the extension of the attribute function to attack suites, where the Combine function corresponds to a *disjunctive combinator* and the Eval function to a *conjunctive combinator*. Thus, a security scoring scheme corresponds to the combination of an *attribute domain* and an *attribute* as follows. The domain D corresponds to the attribute values V, Init and Eval correspond to its combinators as just outlined, and Init corresponds to the attribute.

A notable difference is that the soundness conditions we impose on security scoring schemes enforce that a more secure account must receive a higher score. Attack tree attributes are more *flexible*; while its combinators must also follow certain algebraic properties, they allow for a wider range of different semantics. Example 3 in [11] gives both the attribute domains $(\mathbb{N}, \min, +)$ and $(\mathbb{N}, \max, +)$. The first one, to be interpreted as *"cost of the cheapest attack"* [11] corresponds to our sum-then-min security scoring scheme. The second one, however, to be interpreted as *"maximal damage"* [11] would not correspond to a sound security scoring scheme. The reason is that the disjunctive combinator, corresponding to our Eval function, is instantiated with max, a function that does not meet our soundness requirements; it might lead to less secure accounts receiving higher scores.

This difference is expected when considering the use cases of each formalism. Security scoring schemes aim to specifically measure the security of accounts, so more secure accounts should never receive lower scores than less secure ones. Attribute domains can express a wider variety of semantics, e.g., the "maximal damage" interpretation just discussed. Therefore, the soundness requirements for security scoring schemes would be overly restrictive for attribute domains. They may be meaningful for attribute do-

mains that aim to capture the difficulty or cost of performing an attack, metrics that are expected to be higher for more secure systems.

2.6.1.2 General initial vertices

If we allow for V_{init} to not only contain leaves, the correspondences just outlined do not hold directly anymore. Consider a vertex v that represents an account. When $v \in V_{\text{init}}$, then $\{v\} \in \text{AccessBase}(v, V_{\text{init}})$. In this case, a security scoring scheme assigns *two* scores to v: Init assigns an initial score that represents the score, e.g., the difficulty, of compromising the account directly, without presenting credentials, as described in Section 2.3.3. The scoring function Score assigns a final score according to the scoring scheme. Since the attribute function for attack trees α is defined only as a single function that extends to attacks, it cannot assign two different values to v.

In fact, according to the rules given in the previous section, v would correspond to an *attack component*, and the access base of v would correspond to an *attack*. Thus, we can still find a correspondence by having v correspond to two different nodes in the attack tree: One goal node, *Compromise account* v, and one leaf node, i.e., attack component, *Compromise* v *directly (without presenting credentials)*. The initial score then corresponds to the attribute of the attack component, and the final score to the attribute of the goal.

Going in a similar direction, Buldas et al. [21] have shown how attributes can be assigned to intermediate nodes in attack trees. However, their method assumes that some attributes of intermediate nodes are known, which would correspond to final scores being known before evaluating a scoring scheme, whereas some initial values on leaves are unknown. This does not directly correspond to account access graphs with a general V_{init}, which should be translated to attack trees in the way we just described.

2.6.2 Other related work

Logic programming [22] is also well-suited for problems of the kind we consider. In particular, its stable model semantics [23, 24], which is the basis of answer set programming [25, 26], is commonly used for problems of a similar form as our access and lockout set computation. For example, a query of the form $v \in^{?} accessFrom(V)$ can be answered using Horn clause resolution. Nevertheless, our domain-specific formalism offers many advantages. It facilitates visualization, and is well suited to directly analyze

both access and lockout. The translation to Horn clauses given in Section 2.2.2 requires different sets for access and lockout, respectively, and also requires the introduction of auxiliary variables that do not directly model a particular entity. In contrast, in our formalism, each vertex directly corresponds to an entity, such as an account, credential, or device.

There is extensive previous work on end-user authentication with services; the survey by Bonneau et al. [27] provides a good overview. However, there is substantially less work on account recovery, and existing work mostly belongs to one of two categories: empirical studies of existing systems and proposals for new systems. Since these are not directly related to our work, we only give a few examples. Bonneau et al. [28] and Rabkin [29] have conducted studies on security questions. Social recovery systems based on trustees have been proposed by Brainard et al. [30] and Schechter et al. [31]. Jakobsson et al. [32] suggest the use of personal preference questions as a replacement for traditional security questions. To our knowledge, there is no previous work analyzing a user's entire account connection setup.

3

User Account Access Graphs: Practical User Study

In this chapter, we describe the methodology and results of our qualitative interview study with twenty participants. In this study, we leveraged the formalism and tooling we developed in the previous chapter to model our participants' account setups as account access graphs. This allowed us to both analyze structural features of their setups in terms of graph properties, and served as context for semi-structured interviews about our participants' account security.

We have performed both manual and automated analysis to discover security weaknesses in our participants' account access graphs. The manual analysis focused on standard notions, such as password reuse [33, 34, 35, 36] and lack of two-factor authentication [37, 38, 39]. The automated analysis focused on the discovery of *backdoors*, the predicate given in Definition 38, using the *single attacker vulnerablity* scoring scheme given in Definition 31. That is, we also show how our formalism can be leveraged for automated analysis in practice.

The results presented in this chapter were obtained with the help of the following researchers, who are co-authors of the article this chapter is based on, *"Making Security Personal: A Study on Users' Individual Account Connections"*: Michael Crabb, Saša Radomirović, and Ralf Sasse. These article co-authors and the author of this thesis are referred to as researchers in the methodology section of this chapter. The interviews were performed and transcribed by the author of this thesis.

The sections in this chapter are organized as follows. In Section 3.1, we describe

our methodology, our research questions, and our user participant group. We give an overview of our results from analyzing our participants' account graph structure and our interview transcripts in Section 3.2. We provide more detailed results from our structural analysis in Section 3.3 and more detailed interview results in Section 3.4. In Section 3.5, we leverage our results to give concrete recommendations for improving users' account security. We discuss related work in Section 3.6.

3.1 Methodology

We have formulated two main aims for our study. First, we aimed to determine how users' account setups look with respect to the connections between each user's accounts, credentials, and devices. That is, we aimed to model the users' account setups as account access graphs and determine, based on this model, what structural features and security weaknesses they exhibit.

Second, we aimed to obtain individual insights into users' awareness of their setups' security weaknesses, what kinds of threats they consider, and their willingness to improve the security of their setup.

To this end, we conducted twenty 60-90 minute semi-structured in-person interviews in February and March 2020 after obtaining approval of the ETH Zurich institutional review board (IRB). Each participant received a supermarket voucher valued 20 CHF for their participation. Each interview was conducted by the same interviewer, the author of this thesis, to ensure consistency in the process.

In preparation for the interviews, we have performed multiple pilot interviews with colleagues to refine our process and tool. These pilot interviews were not recorded, and no data of them is included in our results.

According to the guidelines of the ETHZ IRB, the participants were informed of their rights and responsibilities, including that they may withdraw from the study at any time. No participants made use of this right. The interviewer then asked the participants for their gender and age, and additionally asked them to mark one of the following options to assess their background knowledge and interest.

- I am an information security professional or my study is directly related to information security.

- I am an IT professional or study Computer Science or a closely related field, but not directly related to information security.
- I am interested in the topic of information security, but not professionally or in my studies.
- I am not interested specifically in information security, but am interested in IT technology topics.
- I am not particularly interested in IT technology topics.

The interviewer also asked them whether they agree to an audio recording of the second interview part. They were informed that the recording stays on an off-line device, and that the transcript is stored off-line on the interviewer's computer as well as on the secured server of the Information Security Group at ETH Zurich. 18 out of 20 participants agreed to the recording.

The interviews consisted of two parts. In the first part, the interviewer asked the participant about their accounts and connections, and entered the obtained information into our front-end tool described in Section 2.5. The tool was deployed locally on the interviewer's computer. We now describe the approach in more detail.

3.1.1 Part I: Obtaining the participant's account access graph

In the first part, we obtained *structural* information about the participant's account setup. The aim of the first part was to answer the following research questions:

RQ 1. Structure: What does a typical user account setup look like?

RQ 2. Weaknesses: What are potential weaknesses in users' account setups?

To answer the first question, we used the aforementioned tool to model the participants' account setups as account access graphs. The structure of a participant's account setup is comprised of the connections between their accounts, credentials, and devices. Modeling these connections as directed edges in a graph enables an analysis of various structural features of the participant's setup.

The interviewer first asked the participant about which devices they use to connect to the Internet, such as computers, smartphones, or tablets. For each device the

participant uses, he asked how that device can be unlocked, i.e., whether it requires a password, a PIN, or biometrics such as a fingerprint or FaceID.

Second, the interviewer asked whether the participant uses a password manager, or otherwise stores their passwords, e.g., in a browser. Afterwards, he asked about the participant's email accounts, since these are often connected to many other accounts. Participants were not required to disclose which service providers they use for email, but the interviewer did ask them to give each email account a temporary identifier to refer to during the interview.

For each account, the interviewer asked whether login requires a password and/or a second factor. He also asked how the participant could reset the password if they forgot it. If the participant was not sure how the recovery process for one of their accounts works, the interviewer asked them if they can obtain this information, using their own devices. The interviewer never told participants that they had to actually attempt a password reset, but some participants did so voluntarily. In cases where the information about the recovery process was not guaranteed to be accurate, the interviewer still asked the participants how they believed the process works. In some of these cases, the participant's account access graph had to be revised slightly in the second part of the interview.

The interviewer then asked the participants whether they have accounts they use to log in to other services, explicitly asking if they use their Google or Facebook accounts for this purpose. For each participant who answered that they do, he next asked about the accounts they use for single sign-on. The Google account was already covered in the email section if the participant used GMail. If they mentioned GMail in this part but did not explicitly mention that one of their used email accounts is the GMail account, the interviewer added this information. This ensured that the Google account was modeled properly as both a GMail account and an account used for single sign-on.

For all accounts with at least one outgoing connection, the interviewer also asked whether the participant is usually logged in, i.e., has an open session on any of their devices, and on which. In some cases, the participants noted that this varies over time. In this case, the device was modeled to be connected to the account even if the participant was not always logged in.

At this stage, the interviewer has finished gathering the information about the participant's devices and their accounts that have outgoing connections, i.e., provide

access to other accounts. The interviewer's goal was to gather this information as completely and accurately as possible.

The interviewer then asked some questions about additional accounts that are more likely to be leaf vertices in the graph, i.e., that are not used to recover or log in to another account. For these kinds of accounts, the interviewer's goal was *not* to gather information about every single account the participant owns. This would not have been feasible within a reasonable time frame. We aimed to estimate which of the participant's accounts and devices provide access to many other accounts, but the exact numbers were not important. To this end, the interviewer asked whether the participant has accounts in some categories, such as social media, web shops, and online banking. He also asked about default behavior as follows.

- When the participant has multiple email addresses, the interviewer asked which one of them they are most likely to register as a recovery method when they set up a new account at a service.
- When the participant mentioned that they use single sign-on, the interviewer asked them for example services for which they use this option. He also asked them which option they are most likely to use, e.g., if a service provider offered single sign-on with different providers or offered both single sign-on and the option of registering a separate account.

The interviewer then included generic vertices in the account access graph, such as a Default vertex to capture the participant's behavior they described as most likely when registering at a new service.

Before finishing the first part, the interviewer asked the participant whether there was an account they considered important and they have not talked about, or whether they can think of more accounts with an unusual setup for authentication or recovery. If the participant mentioned any accounts at this stage, the interviewer also asked about how these accounts can be accessed and recovered, as for all the previous ones.

Between the first and second part, the interviewer ran an automated analysis on the obtained account access graph to automatically detect backdoors with respect to specific attacker models. The backdoors were defined with respect to *single attacker vulnerability* scoring schemes according to Definition 31 in Section 2.3. In particular, we used the following instantiations of the scheme.

- A *text message attacker* can compromise text messages directly (without access to the phone). That is, vertices representing text messages receive a Vulnerable score in the corresponding scoring scheme. A text message attacker models, for example, an attacker who can perform SIM swap attacks [2, 3]. In a SIM swap attack, the attacker obtains access to the victim's phone number by impersonating the victim to their mobile phone provider, obtaining a new SIM card for their phone number.

- A *password attacker* can compromise passwords. That is, vertices representing passwords receive a Vulnerable score in the corresponding scoring scheme. A backdoor with respect to a password attacker could for example mean that two-factor authentication can be circumvented using only a password.

- A *weak secrets attacker* can compromise credentials labeled as *weak secrets*. In our model, weak secrets include in particular security questions as well as patterns that must be drawn to unlock a device (e.g., a smartphone). Vertices representing such weak secrets receive a Vulnerable score in the corresponding scoring scheme.

- A *device-theft attacker* can compromise (steal) physical devices, but cannot compromise any secrets. That is, vertices representing physical locked devices receive a Vulnerable score in the corresponding scoring scheme. A backdoor with respect to a device-theft attacker means that an attacker who has obtained one of the user's devices can access the account without knowing its password, e.g., through password reset using an open email account session on a phone without a locking mechanism.

3.1.2 Part II: Discussion based on the account access graph

The aim of the second part was to answer the following research questions by means of a personal discussion about the participant's account access graph.

RQ 3. Awareness: How *aware* are users of the weaknesses in their setup?

RQ 4. Willingness to improve security:

- *RQ 4-A.* For weaknesses users are aware of, why have they not made improvements so far? Have there been obstacles preventing them from improving their security?
- *RQ 4-B.* For weaknesses that users were not aware of, are they *willing* to make improvements once we have made them aware of these weaknesses?

RQ 5. Attacker models: What do users *perceive* as threats to their setup? Does this align with the threats typically considered in the security community?

The interviewer created an audio recording of the second part for each participant who explicitly agreed to this. At the start of the second part, he showed the participant the visualization of their account access graph produced by the tool. Any accounts with *backdoors* that were discovered by the automated analysis were marked red. While these provided an entry point into the discussion, the interviewer also discussed potential weaknesses he noticed by manually inspecting the graph. Note that the interviewer had the required expertise on account security to perform such a manual analysis. The combined automated and manual analysis aimed to discover a wide range of potential security weaknesses, but was not designed to exhaustively discover every single weakness.

The second part was intentionally less structured than the first. The goal was to discuss individual weaknesses in each participant's graph. Even though no rules were formalized in our process for the second part, the interviewer informally followed the guidelines described next.

The interviewer pointed out when an account with many outgoing edges, such as an email account, was only protected with a password. In these cases, the interviewer asked whether this password was used for other accounts. If the participant did reuse the password for such an account, the interviewer steered the conversation to the risk arising from password reuse. In particular, the interviewer asked whether the participant has seen *haveibeenpwned.com*, or a similar website or tool that can be used to determine whether a password has been exposed in a password database breach. He also asked the participant whether they were aware that password reuse had an associated risk. If they were aware, he asked them why they still do it, and advised them to use unique passwords at least for their accounts with many outgoing edges. If they were not aware,

he explained the risk of credential stuffing attacks using password database breaches to the participant, and asked them if they were willing to change their behavior.

When such a connected account belonged to a service that offers two-factor authentication, the interviewer asked the participant whether they were aware of this option. If they were aware, he asked them why they do not use it. That is, the interviewer tried to understand the barriers to using two-factor authentication given the additional context of an important account. If the participant was not aware of two-factor authentication options, the interviewer explained them, in particular text messages and authenticator apps such as Google Authenticator. If the participant was aware of the option using text messages, but not authenticator apps, the interviewer explained the differences. He afterwards asked the participant if they were willing to adopt one of these options.

When an account could be recovered by using a text message only, the interviewer mentioned risks associated with text messages. This included SIM swap attacks and potential risk incurred by text messages visible on the locked phone screen, which he asked for in the first part.

There were other topics discussed and we intentionally did not constrain the topics covered in the second part. This allowed the interviewer to touch on unique circumstances encountered in each participant's account access graph. In particular, some participants made interesting comments in the first part with respect to their thought process on how they have set up their accounts. The interviewer followed up on these comments in the second part to discuss them in the recorded part of the interview.

3.1.3 Ensuring the anonymity of participants

The following steps were taken to ensure the participants' anonymity. The participants' names were never included in their data, neither in the account access graph data nor in the interview transcripts. All email correspondence and calendar entries for scheduling the interview appointments have been deleted. Consent forms containing the participant's names are stored only physically, separated from any other data, and their order has been randomized to prevent correlation with participant numbers.

3.1.4 Data evaluation

The 18 recordings of the second interviews were transcribed by the interviewer. 11 of the sessions were conducted in English and 7 sessions in German. Transcripts and analysis were created in the language the interviews were conducted in. Final quotes in Sections 3.2 and 3.4 are all shown in English with translations made by the interviewer. Original language transcripts are provided within the supplementary material [40] in line with existing style guidance [41].

An initial code book identifying topics in the transcripts was generated to assist in analyzing transcripts. A subset of 5 transcripts were coded by the interviewer and 3 different transcripts were coded by another researcher. Both researchers used an inductive coding technique [42] with themes then generated as emergent categories [43]. That is, the researchers marked participants' quotes with codes representing the topics that best matched the quotes and categorized the resulting codes into different themes. The two researchers discussed their different approaches to coding, noting similarities in generated codes/themes and discussed differences. Agreement between researchers was achieved and a preliminary code book was created consisting of 34 codes and 9 themes.

The entire corpus was then analyzed, including transcripts that had been analyzed in the previous stage. All researchers that were involved in the study took part in this activity and first became familiar with the initial code book. Each transcript was analyzed, separately, by two researchers. Individual transcript coding was compared between researchers and agreement sought for individual codes. The researchers then discussed the analysis with the aim of condensing the overall number of codes in areas where similarity was present [44].

A privacy-enhanced version of the coded transcripts and account access graphs, where the names of service providers have been replaced with generic identifiers, is available at [40].

3.1.5 Participants

From our 20 participants, 12 were female and 8 were male. The ages ranged from 19 to 40, with a mean of 25.5. 4 reported to study computer science. 12 reported to be interested in information security topics, but not professionally or in their studies, and

2 reported interest in IT topics, but not security specifically. One participant reported no particular interest in IT topics, and another participant reported to be between the last two options (no particular interest, and interest in IT, but not security specifically). We refer to our participants as P1, ..., P20. Note that P3 and P19 did not agree to the recording and thus were not quoted.

We next note some limitations of our participant selection methods. The study was advertised as a *User Study on Security Risks in Accounts for Internet Services*. As with every study with voluntary participation, there is a self-selection bias that may lead to participants that have an above average interest in the discussed topic. This is likely the reason for the high number of participants who reported an interest in information security. Participants were required to actively use internet services and own at least one portable device with an Internet connection. The study was also advertised in places, both online and physical, that are mostly, but not exclusively, frequented by university students. Thus, even though we did not specifically ask about this, we assume that a large fraction of our participants are university students.

3.2 Result Overview

In this section, we give an overview of our results in terms of answers to our research questions. We mainly leverage the structural results from our participants' account access graphs to answer RQ1 and RQ2 and the interview results to answer RQ3-RQ5.

We present here only a selection of quotes that directly relate to our research questions. Quotes that were translated from German are marked *[GER]*. We show the final code book we have obtained from the interview transcripts and give an overview of our themes, including concrete participant numbers for each code, in Section 3.4.

In what follows, "most participants" refers to three quarters of participants or more, "many participants" refers to half or more, and "several participants" refers to a third or more. All ratios refer to all participants unless a subset is mentioned explicitly. Other words, such as "some" or "multiple", are not assigned a special meaning; these either refer to a lower ratio or are used where such a measure would not be meaningful.

RQ1. Structure: What does a typical user account setup look like?

We modeled our participants' account setups as account access graphs as a basis for answering this research question. The 20 account access graphs elicited by interviewing our participants have between 30 and 59 vertices. However, as we described in Section 3.1.1, we did not require participants to mention all of their accounts. That is, we were more interested in structural features than exact numbers.

Figures 3.1 and 3.2 are simplified versions of P6's and P18's account access graphs, whose structural features we describe next. These graphs combine most of the structural features we have observed in the elicited graphs. The semantics of different edge styles in these figures differs slightly from those in Chapter 2 to improve readability. Multiple *black* edges should be interpreted as edges of distinct colors, that is, any *one* vertex with such an edge to a vertex v is sufficient to access v. This is a simplification for presentation purposes only and does not affect the underlying formalism.

Access patterns The first structural features we examined were how our participants' accounts are generally accessed and recovered. In particular, we examined what common *patterns* exist, where many different accounts are accessed in a similar manner.

We consider two kinds of patterns: Those reported by our participants directly, and those we identified in the graphs ourselves. An example of the first kind of pattern is the grey-shaded vertex Default in Figure 3.2. The participant did not name specific accounts for this vertex, but indicated the use of accounts that only had a password but no recovery method. Similarly, using the Mail account for single sign-on was identified by the participant as a pattern for Serious accounts.

We identified patterns ourselves by considering different vertex types: Passwords, email accounts, other accounts, and devices. We consider three or more credentials or accounts to have the same access pattern if they are of equal type and are accessed by equal (non-empty) combinations of types of credentials. A typical such pattern is the use of some kind of password manager to store passwords. This pattern can be seen in the grey-shaded pwds vertices in Figures 3.1 and 3.2. We bundle accounts, devices, and any other credentials that are accessible with the same pattern as follows. We contract the vertices that have the same access pattern and indicate in the resulting grey-shaded vertex the number of contracted vertices. Figure 3.1 shows 3 different access patterns that we identified for P6. P6 reported on 16 passwords, contracted into the vertex

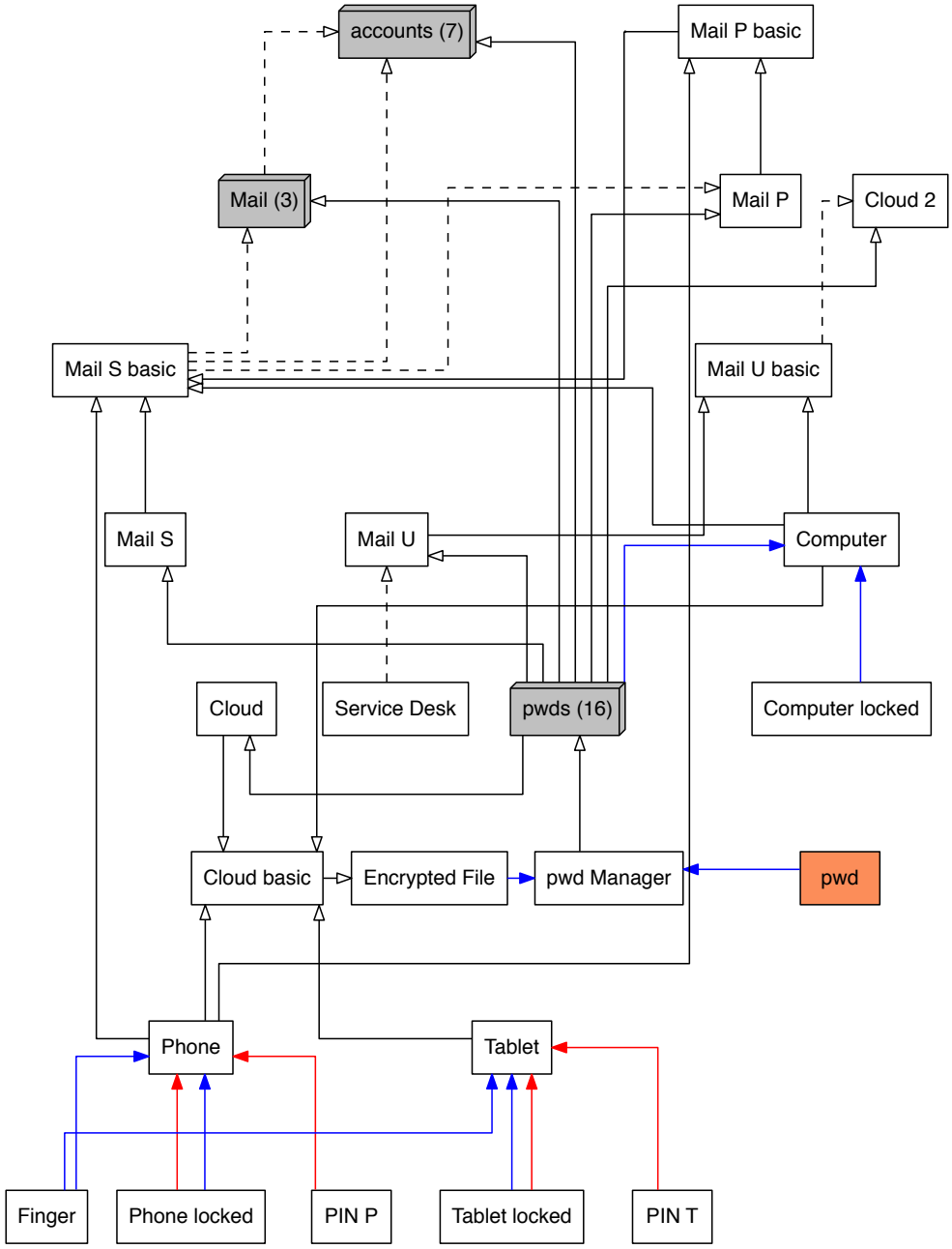

Figure 3.1: Simplified Access Graph of participant P6. Some accounts have been removed. Grey shaded vertices are contractions of vertices with the same access pattern. The orange shaded vertex is the most critical for access to other accounts.

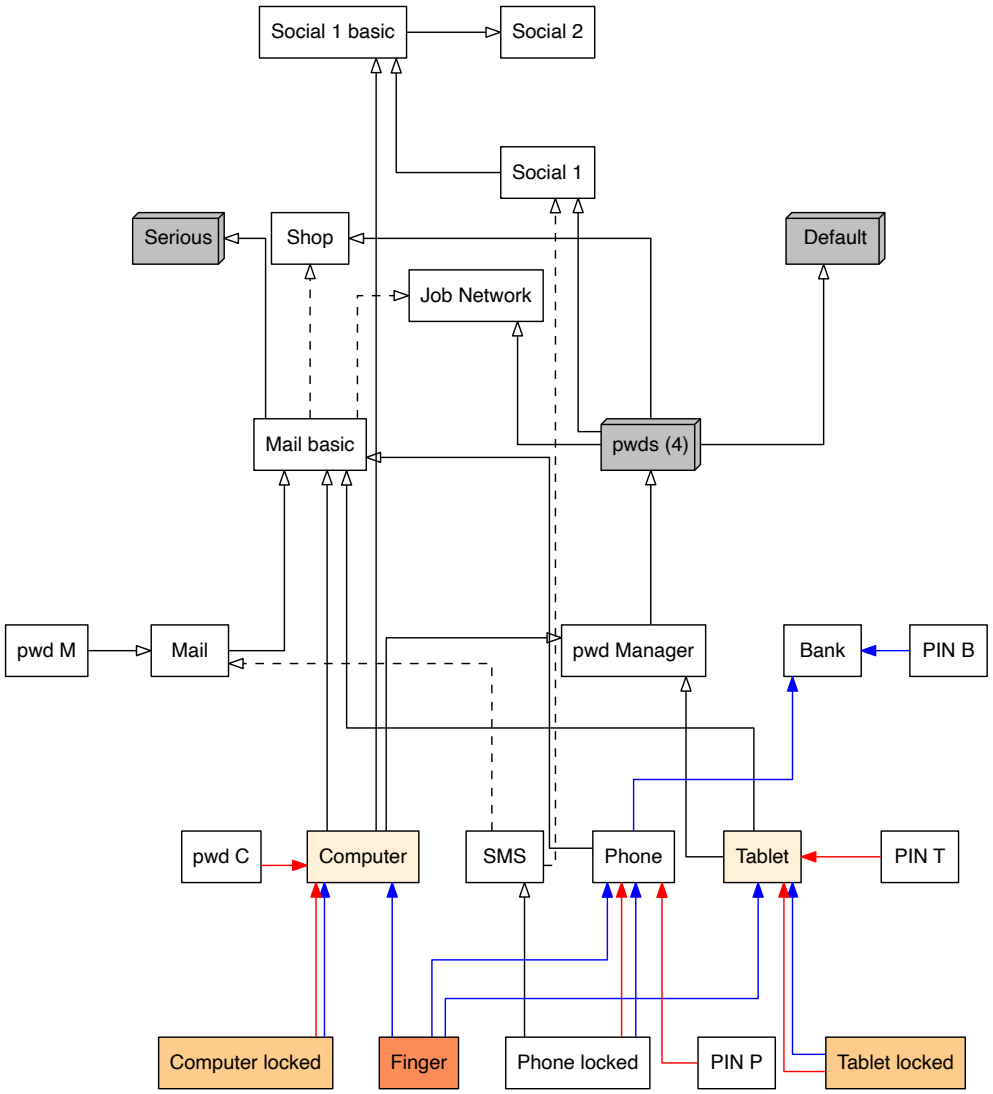

Figure 3.2: Simplified Access Graph of participant P18. Some accounts have been removed. Grey shaded vertices are contractions of vertices with the same access pattern. The orange shaded vertices are the most critical for access to other accounts. More intensely shaded is more critical.

pwds, all of which are accessed using a password manager. P6 mentioned 7 different accounts, each of which is accessed with a password and recoverable by means of an email account, and 3 email accounts that are accessed and recoverable in the same way.

Central vertices The next structural feature we analyzed was which vertices are the most central in our participants' graphs in terms of providing access to other vertices. We used the centrality evaluation function given in Definition 35, which evaluates a vertex's centrality by counting the number of ways it contributes to the access to other vertices. Vertices that receive high centrality values indicate critical parts of the user's setup. If these vertices were protected poorly, this would constitute a weakness.

We next give a high-level overview of our insights with respect to central vertices. We provide more details on our analysis of centrality values in Section 3.3.

Passwords are used as an authentication factor for most accounts, so **password managers or vertices providing access to them are central vertices.** Our definition of password managers is broad and includes dedicated applications as well as passwords stored locally on a file system or in a browser. For example, in Figure 3.1, the vertex pwd has the highest centrality value. This is a decryption password for an Encrypted File stored in the Cloud storage. We will discuss the reason for pwd's high centrality value in the context of cycles below.

While we found that **most participants stored some of their passwords in a password manager, several participants did not store all of them.** For example, in P18's setup (Figure 3.2), the password pwd M for the Mail account is not stored in the pwdManager.

In many participants' setups, a device is the most central vertex according to our centrality values. Our participants' devices commonly provide access to many accounts by means of open sessions or through passwords saved on the device. Most participants' devices were protected with a locking mechanism. The most common locking mechanisms were PINs and fingerprints. The graph in Figure 3.2 highlights the Finger vertex because it is an option to unlock any of the three devices. Note that in comparison with the Computer and Tablet, the Phone is not as central in this graph. The reason for this is that it does not provide access to the password manager.

Cycles A structural feature closely related to central vertices are cycles. Cycles occur, for example, when multiple accounts can be used directly or indirectly to recover each other. Vertices that are contained in a cycle or provide access into a cycle can have comparatively high centrality values, particularly when access to any one of its elements provides access to the whole cycle. The reason is that such a cycle with several incoming and outgoing links acts like a hub connecting all incoming paths to all outgoing paths. If the cycle includes or provides access to a critical vertex such as a password manager, then all vertices that are part of the cycle or provide access into the cycle obtain high centrality values. These accounts must therefore be well-protected.

Several participants' account access graphs contained cycles, for example due to email accounts recovering each other or passwords saved in cloud storage. P6's graph in Figure 3.1 contains two cycles: Mail S basic → Mail P → Mail P basic → Mail S basic, and pwd Manager → pwds → Cloud → Cloud Basic → Encrypted File → pwd Manager. All vertices in these cycles are sufficient to provide access to the next vertex in the cycle, except for the access provided to the password manager, which, in addition to Encrypted File, requires pwd. This explains why pwd, the password for the password manager, has by far the highest centrality value: Most of the access paths to passwords and accounts in this graph are contingent on pwd. In contrast, Encrypted File, unlike the password, is accessible through multiple devices that have open sessions to the Cloud storage, denoted by Cloud basic.

We give more detailed information about cycles, central vertices, and other structural graph features in Section 3.3.

Compartmentalization Several participants' account graphs consist of multiple connected components. In the second part of the interview, we learned that **some participants consciously separated their accounts with the goal of reducing spam and notification load**, e.g., into work-related and not work-related. In particular, they were logged in to different accounts on different devices, and connected accounts to different email accounts. For example, P8 stated that *"if something is important, I would connect it to [one email address], something that just needs an email to log in, I would connect to [another email address]."* P6 followed a similar process, describing their use of the three email accounts shown in Figure 3.1 as follows: One email account is for things that were described as *"important,"* a second email account

for things that were *"related to university"* and a final account for things that were *"more creative, I would say, social media and stuff."* P10 explained that they used their tablet only rarely and thus consciously was not logged in on many accounts there, concluding *"my configuration follows my routine, basically."*

RQ2. Weaknesses: What are potential weaknesses in users' account setups?

We briefly outline the most common weaknesses that we discovered by automatically and manually analyzing our participants' account access graphs, as described in Section 3.1.2. We provide a more detailed discussion and give related quotes when discussing later research questions.

Most participants fully or partially reused passwords, exposing themselves to risk from password database breaches [45]. Some participants even reused their password for accounts that provide access to many others, such as their email accounts. Note that we elicited password reuse in the second part of the interview only, so the graphs do not differentiate between unique and reused passwords.

Several participants did not use two-factor authentication unless it was mandatory, as is the case for many online banking accounts. While using only a single authentication factor might not directly constitute a critical weakness, it does become critical when this factor is a reused or weak password. Therefore, using two-factor authentication is security advice commonly given by experts [46, 47, 48]. In particular, some participants had email accounts that were central to their setup, and with email providers that offered two-factor authentication, but they did not have it enabled.

Many participants used text-message-based account recovery, leaving them vulnerable to SIM Swap attacks [2, 3]. Furthermore, some participants who used such a recovery method either did not have a locking mechanism on their phone, or enabled text message preview on the locked phone. That is, anyone with physical access to the phone would have access to recovery codes sent to the phone.

Security questions were used as recovery methods in a few participants' setups, but often served only as a single step in a multi-step recovery process. Thus, they did not constitute a critical weakness in most setups.

RQ3. Awareness: How *aware* are users of the weaknesses in their setup?

We next discuss our participants' awareness of the weaknesses described previously, their relevance for their personal setup, and potential solutions for them.

Multiple participants reused passwords only for accounts they considered relatively unimportant. P8 described reusing passwords for *"the stuff that I don't care about,"* while *"the things that are important to me have different passwords."* P18 similarly said *"I do reuse passwords, I just make sure that the [accounts] that I know are connected to a lot of different things are unique."* **Participants generally did not have an understanding of the concrete password reuse risk arising from data breaches.** While participants who reused passwords often expressed some awareness that this was not a secure practice, they did not mention concrete risks or concerns. For example, P4 was aware that reused passwords were *"not so secure,"* and P7 used one password *"for almost everything else,"* wanting *"something different"* for their most important email account. Most participants who were asked about it were unaware of *haveibeenpwned.com*, a website that allows one to check if a password has been compromised in a password database breach.

Participants were more aware of text-message-based two-factor authentication than of dedicated authenticator apps. Their experience with authenticator apps was generally restricted to online banking services that authenticate users using their own specific app. Participants were largely unaware about authenticator apps used for multiple services, such as *Google Authenticator*. Participants that did not use two-factor authentication did not understand its details, e.g., they overestimated how often they would be asked for the second factor. P4 said: *"I thought you would have to use [two-factor authentication] every time [you log in], or something, and then it's annoying."* This affected participants' willingness to use two-factor authentication; we will discuss this in more detail when we discuss RQ4, related to willingness.

With respect to text-message-based account recovery or two-factor authentication, **most participants asked about SIM Swap attacks were unaware of them.** However, we believe that increasing end user awareness for these kinds of attacks does not necessarily have to be prioritized; they can be prevented directly by mobile phone providers more diligently authenticating users before issuing new SIM cards.

Security risks arising from security questions were discussed by some participants. For example, P2 said: *"I was easily able to go into [my sister's] account because of the security questions,"* leading them to believe they *"shouldn't be too honest"* when answering security questions, or else it would be *"very easy for somebody to get in."* P2 then explained their strategy as follows: *"I didn't write the name of my best friend, I just wrote the name of the first friend I ever had, which also not a lot of people know."* P11 thought that it was *"so easy to find such information"* related to security questions.

Participants had an incomplete view of their account setup. Thus, they underestimated the importance of their central accounts, such as their email accounts, and devices. When asked what they found surprising about the study, P15 replied *"I'm surprised that so much is connected, especially to my [email account] [GER],"* and P8 said *"Yeah, like, everything is almost accessible if they have your phone physically."*

RQ4. Willingness to improve security

RQ4-A. For weaknesses users are aware of, why have they not made improvements so far? Have there been obstacles preventing them from improving their security?

Some participants consciously favored convenience over security, considering their accounts not sufficiently important to warrant much effort. For example, P14 reused a single password for all their accounts despite having heard of the risk, explaining *"one just doesn't get around to doing it [setting up unique passwords], because convenience wins out [...] I am aware that this might not be optimal, but [...] even if I resolve to do it, I don't, until it is too late [GER]."* P7 said *"I of course don't want anyone to access my email, but I feel like, worse comes to worst, somebody does actually [access it], there's nothing that I'm afraid they might read, I don't have anything like that."*

Other participants mentioned concrete obstacles, in particular with respect to adopting two-factor authentication. Multiple participants had usability or privacy concerns about using their phone number for text-message-based two-factor authentication. P8, who had moved between countries, described resulting usability problems as follows: *"I lost my [...] country's phone number, and when I entered my email [...] I need to have the SMS, and I couldn't receive the SMS, so I was locked out for 30 days on both my emails."* P2 said *"if you change your number, maybe you have to go back and change your number in the account,"* further explaining *"it's just now*

that I'm looking for a job and then I might move countries." P4 similarly said *"I've been moving a lot, so I got different phone numbers from different countries [...] if I'm in a different country I won't receive the message."*

With respect to privacy, participants were concerned about their phone number becoming public. P13 explained that they intentionally did not enter their current phone number for account recovery: *"then not everyone has my current number [GER]."* Similarly, P2 asked about two-factor authentication *"if I put my phone number it can become public, is that so?"*

Participants also expressed a higher willingness to keep a default configuration than to actively switch to another. P7 explained *"I feel like all of my accounts are protected [...] there's nothing that's gone wrong with any of my accounts, so it's just an extra hassle if I have to [set up two-factor authentication],"* but later also mentioned an account for which two-factor authentication was *"by default set up,"* for which they did not disable it. Similarly, P20 explained *"if an app or platform asks me upon installation if I want to use two factors, I usually say yes, but if it does not ask explicitly, I don't [GER]."*

Participants were influenced by personal security advice given by family members or friends. P2 talked about their sister, *"she's a computer scientist, she also helped me with how to [set up two-factor authentication]."* P15 said about one of their passwords *"[...] it's randomly generated, because my brother told me to [...] it was something because it was good to do, but also he insisted on this thing."*

RQ 4-B. For weaknesses that users were not aware of, are they willing *to make improvements once we have made them aware of these weaknesses?*

Participants were willing to better protect accounts that the interviewer pointed out as central to their setup, such as email accounts used to recover many other accounts. When the interviewer discussed using unique passwords and/or two-factor authentication for such accounts, participants were receptive to the advice, saying that they would implement it. When the interviewer asked participants about actions they would take after the interview, they usually mentioned at least one. For example, P2 responded *"I will [change my security questions] when I go home today,"* and P4 said: *"I'll probably change my passwords and write them down with pen at home."* P7, who the interviewer had shown *haveibeenpwned.com*, said *"I'm definitely*

going to check if my password has been visible and then I'm going to ask a bunch of people to do that as well."

Participants were also motivated to think about security more generally after the study, for example P10, *"this study makes me reflect a lot on how some things you hear, or that you see online, could affect the way that you configure your security stuff [...],"* or P18, *"I haven't really thought about security in a while, but this is a good reminder."*

RQ5. Attacker models: What do users *perceive* as threats to their setup? Does this align with the threats typically considered in the security community?

Privacy with respect to service providers was a concern for multiple participants, impacting their willingness to use security mechanisms. In particular, they did not trust service providers with their personal information. They were hesitant to enter their phone number for authentication purposes, or did not want to synchronize their saved passwords across devices using a cloud service. P6 said *"I don't really like to have all my data in the cloud,"* and switched from one browser to another because *"you can control more"* with respect to data sharing. P18 said *"I actually make sure that a lot of things are not in the cloud."* Not all our participants shared this concern; for example, P17 had *"no concerns with respect to this"* when using social login, explaining *"it's good that I can pay with my data [...] then I get ads that actually interest me [GER]."*

Authentication based on biometrics, such as fingerprints or facial recognition, was also mentioned as a potential privacy concern, for example by P10: *"you're giving more data and more information as you need to unlock a device."*

Participants did not mention whether they were aware of service providers' efforts to preserve their privacy, such as Apple's FaceID storing data encrypted and locally [49]. However, trust in the service provider is still required in such cases, since users cannot easily verify the service providers' claims about privacy. An interesting question that we did not address is whether users who expressed such privacy concerns would be more willing to use more user-centric solutions, such as password managers that are synchronized across devices manually rather than using a cloud service.

Participants considered attempts to hack accounts to be targeted at specific individuals, and considered themselves an unlikely target for attacks.

P5 wondered if they *"might be too paranoid,"* because they were *"not some politician who has important things [GER],"* and P9 said that they *"don't have that much wealth to make people want to hack my account."*

Finally, some participants believed that an attacker with physical access to their device, e.g., having stolen their laptop, would find a way to circumvent any digital locking mechanism on that device, such as a password or PIN. Such a belief was expressed by P2: *"I just increased the degree of difficulty [...] for the password, but of course if there is a hacker he could find other ways to go into that,"* and P11: *"if someone steals [my computer], this person probably knows how to access things, I don't know how safe this PIN really is."*

While this belief may slightly overestimate an attacker's capabilities, it likely increases users' diligence in protecting their physical devices. The belief might be problematic if it caused users to consider digital locking mechanisms useless and not use them at all. Our results do not suggest this, however; most of our participants did digitally lock their devices.

3.3 Detailed Structural Results

In this section, we give an overview over all participants' account graph features. All interviewed participants use smartphones in addition to a laptop or desktop computer. Table 3.3 summarizes the data we discuss and elaborate on in the following.

The *Vertices* column indicates the number of vertices in each participant's account access graph that were elicited during the interview. As discussed in Section 3.2, the actual number of vertices is likely to be higher, since the interview process did not aim for completeness.

Several participants had cycles in their account access graph. The *El. in Cycles* column shows the size of each cycle. 0 indicates that there is no cycle, while more than one number indicates that there is more than one cycle. As seen in Table 3.3, five participants have one cycle in their account setup and two participants have two cycles. Five of these cycles provide sufficient access to each vertex in the cycle, that is, each of the vertices in the cycle is sufficient to access the next without any other authentication factors. In three of these cases, this is the consequence of two email providers being used as recovery email addresses for each other. In the other two cases,

Participant	Vertices	El. in Cycles	Comp.	PwdManager	Open Sessions	Lockscreen prevents SMS Preview	SMS for 2FA	SMS for recovery	Central Vertices
P1	50	4	1	yes	2	yes	no	yes	Computer
P2	45	0	2	no	4	no	no	yes	Phone
P3	32	0	4	no	1	no	yes	yes	Password
P4	44	0	1	partial	4	n/a	n/a	n/a	Computer
P5	33	0	3	no	1	no	yes	no	Password, Old Password
P6	58	3;5	1	yes	5	n/a	n/a	n/a	Password for PwdManager
P7	59	0	1	yes	4	yes	yes	yes	Computer
P8	37	4	1	yes	3	no	no	yes	Phone
P9	43	6	1	partial	4	yes	yes	yes	Password
P10	53	0	3	yes	4	yes	yes	yes	Computer
P11	41	0	1	partial	4	yes	no	yes	Computer
P12	49	0	2	partial	6	yes	yes	yes	Phone
P13	30	0	2	yes	4	no	yes	no	Phone, PwdManager
P14	42	0	3	yes	3	yes	yes	yes	Phone
P15	44	0	1	yes	3	yes	yes	yes	Phone
P16	35	2	2	partial	2	no	yes	yes	Computer
P17	33	0	2	partial	2	no	yes	no	Phone
P18	35	0	1	partial	2	no	no	yes	Finger, Tablet, Computer
P19	52	4; 6*	1	yes	6	yes	no	yes	All Devices, Account
P20	51	3	1	partial	2	yes	yes	yes	Old Password

Figure 3.3: Summary of participants' account graph features. *: the cycles are overlapping.

a password manager provides access to a cloud service, which in turn provides access to the password manager.

The *Comp.* column indicates the number of connected components in the graph. Many participants' graphs consist of a single connected component. The reasons for several components may be lack of detail provided or an intentional separation.

Few participants make no use of any kind of password manager, as indicated by *no* in the *PwdManager* column. However, many make only *partial* use of it, as they still access some accounts with passwords that are not stored in the password manager. Recall that we also consider passwords saved locally by browsers or in the file system to be password managers for our purpose.

The *Open Sessions* column indicates the number of accounts that one of the participant's devices has an open session with. However, we only elicited this information for accounts that could potentially be used to access other accounts, such as email accounts or accounts used for single sign-on.

As can be seen from the next three columns in Table 3.3, all but two participants use text messages (SMS) as a second authentication factor or a recovery factor for at least one account. Almost half of the participants allow SMS previews to be displayed on the lock screens. For participants whose account access graphs did not contain SMS at all, SMS-related entries are indicated by "n/a".

The *Central Vertices* column indicates the types of vertices that are most central for providing access to other vertices in a participant's graph. We evaluated the centrality of vertices using the function given in Definition 35. See Section 2.3.4 for more details on how we define such vertex evaluation functions. The column shows the types of vertices with a value that is within 5% of the graph's highest centrality value. Note that we do not distinguish between locked and unlocked devices in this column.

For fourteen out of the twenty setups, a device (phone, laptop, tablet, or computer) is the most central vertex, the phone being the most central in eight of these. In three of the remaining six setups the central password vertex is a source vertex that provides access to an email or service provider vertex with many outgoing edges. In P18's setup, shown in Figure 3.2 on page 69, the central vertex is a fingerprint that provides access to all of the participant's devices (phone, tablet, and laptop). P6, shown in Figure 3.1 on page 68, P9, and P19 each have a central vertex that provides access to a large cycle (of 5 or 6 vertices). Moreover, P19's setup is special in that there are two overlapping

cycles. In P19's setup all highly ranked vertices directly or indirectly provide access to at least one of the two cycles. All vertices in the cycles also received comparatively high centrality values.

3.4 Detailed Interview Results

Our final code book contains 22 codes across 5 themes. We next present our themes in separate subsections. For each code, we provide quotes that best illustrate it. Note that we focus on demonstrating the breadth of responses that our participants have given. That is, the quotes shown are not exhaustive, but provide a broad overview of our results. We repeat some quotes from Section 3.2, but in the context of the assigned code and theme rather than the related research question.

3.4.1 Account graph structure

During the semi-structured interviews participants discussed items that relate to the structure of their online security setup.

3.4.1.1 Separation and sharing

5 participants mentioned that they consciously separate their accounts, e.g., into work-related and not work-related. This affected on which devices they were logged in to those accounts, commonly with the goal of reducing spam and notification load. P8 justified this approach, stating that *"if something is important, I would connect it to [one email address], something that just needs an email to log in, I would connect to [another email address]."* P6 followed a similar process, having an email account for things that were described as *"important,"* a second email account for things that were *"related to university"* and a final account for things that were *"more creative, I would say, social media and stuff."*

P7 mentioned sharing an account *"with my brother for some work that we do, and also with some people in the office who would still need, not my [email] password."*

3.4.1.2 Routines and preferences

5 participants discussed their routine relating to their accounts and which devices they are logged in on. For example, P11 described that *"usually I just use my laptop and*

my phone [...] using the university's computer, which is not that common, [...] I always make sure that I'm logged out," and P15 mentioned being *"always logged in"* to their email account. P10 explained that they used their tablet only rarely and thus consciously was not logged in on many accounts there, concluding *"my configuration follows my routine, basically."*

3.4.1.3 Digital management

5 participants discussed digital security management techniques and tools they currently employ, which we illustrate with some examples. P7 used two-factor authentication *"if I'm logging in to a new device, not on my devices, but if I'm logging in on a separate computer."* P13 usually used a *"fake name [GER]"* as username for accounts. P15 applied a mixed password generation strategy; their password for their email account *"was suggested by the Phone, by the App [...] it's really long, and diverse, so I will not remember it,"* while their password for a social media account was *"not randomly generated, but different from the other ones."*

3.4.1.4 Non-digital management

12 participants discussed non-digital security management techniques. P4 described that they were selectively *"writing [passwords] down with pen and pencil, usually I know where I need to log in to, like if I go traveling,"* while P1 *"didn't want to write them down [GER]."*

With respect to choosing passwords, P20 explained that they have *"a very complex basis, and then a specific one [for each website] [GER]."* P10 similarly described that they had a *"structure [...] numerical and alphabetical, and in the end, maybe just a slight change of the symbol or something [...] I wouldn't say like, algorithm."* P5 described a similar strategy, concluding *"I have always done it this way, and never forgotten a single [password], ever [GER]."*

3.4.2 Awareness and understanding of security problems and solutions

Participants discussed different aspects relating to their awareness of security problems, such as password reuse risk, and solutions, such as two-factor authentication and password managers.

3.4.2.1 Lockout risks

6 participants made statements related to the risk of locking themselves out of their devices or accounts. With respect to usernames and passwords, P5 mentioned *"that one could forget the email address, that could be more likely [GER]*, while P11 described that they do not use randomly generated passwords because they *"wouldn't feel comfortable in not having this power over my accounts."*

Two-factor authentication leading to lockout over a longer time were discussed by P8, who had moved between countries, *"I lost my [...] country's phone number, and when I entered my email [...] I need to have the SMS, and I couldn't receive the SMS, so I was locked out for 30 days on both my emails,"* and P20 noted: *"I had an authenticator for a while, for example for [a gaming platform], and it took two months until I could enter again because I changed my mobile phone [...] [GER]."*

3.4.2.2 Security risks

10 participants made statements related to security risks arising from password reuse, weak passwords, or security questions. P4 was aware that reused passwords were *"not so secure,"* and P7 used one password *"for almost everything else,"* wanting *"something different"* for their most important email account. P12 suspected that the reason why one of their accounts had been hacked was that it had *"a very simple password [GER]."*

P2 said *"I was easily able to go into [my sister's] account because of the security questions,"* leading them to believe they *"shouldn't be too honest"* when answering security questions, or else it would be *"very easy for somebody to get in."* P2 also understood that some questions could be brute-forced: *"second is the color of the first car that we owned [...] but of course it's another thing that there's not a lot of colors."* P11 thought that it was *"so easy to find such informations"* related to security questions.

3.4.2.3 Security mechanisms

9 participants mentioned whether they are aware of some security mechanism or tool. The interviewer asked participants whether they are aware of an authenticator app, to which P9 replied *"after you describe it, I feel like I used it, not the same application, but..."* and P10 replied *"I've already like seen the icon, so I know what you're talking*

about." P12, without being asked, mentioned the *"option to use, for example, TOTP,"* referring to the time-based one-time passwords used by many two-factor authentication apps. P4 mentioned physical security keys, *"I know of these devices that can just like plug in to your computer."*

P6 described their search for a good password manager, highlighting several different aspects that also connect to other themes. *"I searched [...] reviews about password managers [...] I don't want to go with [large service provider], because they already know too much [...] I wanted to have access to it if I'm on vacation, and I don't have data [...] maybe use it on another computer, I just look and type it on another one, and then, this one, it was open source, and I found it interesting because the code is already online, you know what's up with your data [...]."*

3.4.2.4 Influence of others

5 participants mentioned that other people, most commonly family members, made them aware of a particular security problem or solution. P2 talked about their sister, *"she's a computer scientist, she also helped me with how to [set up two-factor authentication]."* P15 said about one of their passwords *"[...] it's randomly generated, because my brother told me to [...] it was something because it was good to do, but also he insisted on this thing."*

3.4.2.5 Security mental models

10 participants expressed security beliefs or ways they think about their account security. Participants had different opinions about the security of certain practices. In the example of writing down passwords physically, P11 said *"I also thought that [writing down a password] wouldn't be that safe, anyway, so it's always better to just trust your mind and the security system each platform has for recovery,"* while P12 said *"I think that the physical copy is actually the safest of my password managers."*

P10 thought that *"it's better to have something that provides a double step,"* and *"that it was better to put a phone number, to be reachable by another device, than to put the information of another email address,"* then asked the interviewer *"if more steps are always equal to more security."*

3.4.2.6 Risk implications of connections

13 participants discussed how they believe the connections between their accounts and/or devices are associated with risks. P8, when asked about if they found anything about the study surprising, replied *"Yeah, like, everything is almost accessible if they have your phone physically."* P20 said that they use two-factor authentication for their GMail account *"because I know it is like the root of my accounts."* P15 knew *"that there are so many connections"* but was *"not aware of the risks."* P10 asked *"about this frequent question you receive, from webmail services, to connect one mail service to another - is that a positive one, or is that in general maybe better to keep them separate."* P16 thought of one of their email accounts *"as more of a risk, since there one has the app on the mobile phone where one does not have to log in [GER]."* P17 identified that *"when the mobile phone is turned on, and the person knows how the pattern works, then this is a big security risk for me [GER]."*

3.4.2.7 Misconceptions and inefficient use of tools

8 participants made comments that showed a misconception they had about a security concept, or described using a tool in an inefficient or ineffective manner. P4 *"thought you would have to use [two-factor authentication] every time [you log in], or something, and then it's annoying."* P6 had trouble shifting their own viewpoint to an attacker's, asking about two-factor authentication, *"if I'm logged in all the time, what's the really difference?"*

For an example of inefficient tool usage, P16 described losing track of the passwords saved in their browser: *"I'm not even sure what's saved, it asks me, update password, and I'm thinking, this is the current password, what is update supposed to mean [GER]."*

3.4.2.8 Noticing or recovering from an account breach

3 participants described how they would notice or recover from an account breach. P7 noted that *"these days"* if someone else were to login to their bank account, *"the bank will instantly notify you, and your money is kind of protected."* P16 described a similar belief: *"I've never had the problem that someone debited or transferred money off my account, and I would notice it within a day, so that's not such a great risk [GER]."*

3.4.3 Willingness to improve security

Participants discussed various aspects relating to their willingness to improve their account setups' security.

3.4.3.1 Account importance

8 participants discussed account importance, such as P7, saying *"I wouldn't really worry about somebody logging in to"* their account at a hotel booking service, but *"what I would worry about is my bank."* P8 described reusing passwords for *"the stuff that I don't care about,"* while *"the things that are important to me have different passwords."* P18 similarly said *"I do reuse passwords, I just make sure that the [accounts] that I know are connected to a lot of different things are unique."* Password strength was mentioned by P15: *"I just differentiate where there are my personal informations, then I put a long password."*

Furthermore, participants explained how much they would care about account breaches, such as P7, *"I of course don't want anyone to access my email, but I feel like, worse comes to worst, somebody does actually [access it], there's nothing that I'm afraid they might read, I don't have anything like that,"* or P14, who *"wouldn't really care"* if their accounts at *"web shops or [a social media platform] [GER]"* got hacked.

3.4.3.2 Usability and convenience

13 participants mentioned either concrete usability concerns or more generic convenience concerns with respect to security mechanisms. In particular, participants discussed usability of two-factor authentication. P2 said *"if you change your number, maybe you have to go back and change your number in the account,"* further explaining *"it's just now that I'm looking for a job and then I might move countries."* P4 similarly said *"I've been moving a lot, so I got different phone numbers from different countries [...] if I'm in a different country I won't receive the message."* P6 was reluctant to install a two-factor authentication app *"because my phone is getting full."*

P14 reused a single password for all their accounts despite having heard of the risks, explaining *"one just doesn't get around to doing it, because convenience wins out [...] I am aware that this might not be optimal, but [...] even if I resolve to do it, I don't, until it is too late [GER]."*

3.4.3.3 Accepting defaults

5 participants expressed that they prefer default configurations, whether they are secure or not. P7 explained *"I feel like all of my accounts are protected [...] there's nothing that's gone wrong with any of my accounts, so it's just an extra hassle if I have to do [set up 2FA],"* but later also mentioned an account for which two-factor authentication was *"by default set up,"* for which they did not disable it. Similarly, P20 explained *"if an app or platform asks me upon installation if I want to use two factors, I usually say yes, but if it does not ask explicitly, I don't [GER]."*

3.4.4 Attacker models

Participants mentioned both concrete *attack personas*, which we labeled with the first two codes that follow, and more general *attack vectors*, labeled with the latter three codes.

3.4.4.1 Service and application providers

9 participants discussed whether they had concerns sharing their personal data with a cloud service provider or web applications. P6 said *"I don't really like to have all my data in the cloud,"* and switched from one browser to another because *"you can control more"* with respect to data sharing. P10 was uncertain about using biometrics because *"you're giving more data and more information as you need to unlock a device."* P18 said *"I actually make sure that a lot of things are not in the cloud."* In contrast, P17 had *"no concerns with respect to this"* when using social login, explaining *"it's good that I can pay with my data [...] then I get ads that actually interest me [GER]."*

P13 explained that they purposefully did not enter their current phone number for account recovery: *"then not everyone has my current number [GER]."* Similarly, P2 asked about two-factor authentication *"if I put my phone number it can become public, is that so?"*

3.4.4.2 Friends and family

4 participants mentioned how they feel about friends or family members potentially accessing their accounts. P1 mentioned they don't like to write down recovery codes: *"If they're just lying around, then family members can access them, too [...] I don't*

have the feeling that they're trying to hack me, but the thought that the codes are just lying around makes me uncomfortable. [GER]." The interviewer then asked P4 if they had similar concerns, who replied *"My partner, I hope not, then I'd have other issues."* P7 shared some accounts with colleagues, but made sure to use a different password for their personal account: *"I'm definite they wouldn't misuse it, but I don't want to take any chance."*

3.4.4.3 Physical device security

6 participants discussed whether they were concerned about others physically accessing their devices. P11 mentioned not saving too many passwords on their laptop because they *"don't want to let things too easy for someone else,"* explaining that, in their home country, *"people steal many computers."* P1 had an additional security PIN required for accessing apps on their phone because *"it is unlocked on occasion, and then one cannot simply enter [the apps] [GER]."* P18 said *"I make sure that my phone is on me at most times."* P15 noted that if they use a public computer they *"always choose not to remember the password."*

3.4.4.4 Targeted attacks

5 participants mentioned reasons why they would consider a targeted attack on their accounts likely or unlikely. P5 wondered if they *"might be too paranoid,"* because they were *"not some politician who has important things [GER]."* P9 said that they *"don't have that much wealth to make people want to hack my account."* P20 had been targeted before: *"the reason why my [social media account] got hacked, I won tickets [for an event] and that was announced on the radio [GER]."*

3.4.4.5 Attacker capabilities

5 participants expressed beliefs on what an attacker might be able to do. P2 doubted the effectiveness of even a strong password against a dedicated attacker, *"I just increased the degree of difficulty [...] for the password, but of course if there is a hacker he could find other ways to go into that."* P2 also thought about what an attacker might know when answering security questions: *"I didn't write the name of my best friend, I just wrote the name of the first friend I ever had, which also not a lot of people know."*

P11 expressed doubts about the effectiveness of their computer's PIN: *"if someone steals [my computer], this person probably knows how to access things, I don't know how safe this PIN really is."* With respect to biometrics for unlocking devices, P1 said *"if someone wants to access my phone so badly that they copy my fingerprint then I have other problems than the few apps protected by it [GER]."*

3.4.5 Effects of study

13 participants mentioned how the study affected them. Concrete improvement plans were for example mentioned by P2: *"I will [change my security questions] when I go home today,"* and by P4: *"I'll probably change my passwords and write them down with pen at home."* P7, who the interviewer had shown *haveibeenpwned.com*, said *"I'm definitely going to check if my password has been visible and then I'm going to ask a bunch of people to do that as well."*

When asked what they found surprising about the study, P6 replied *"I wasn't expecting something as complicated,"* and P15 was *"surprised that so much is connected, especially to my [email account] [GER]."*

Finally, participants seemed motivated to think about security after the study, for example P10, *"this study makes me reflect a lot on how some things you hear, or that you see online could affect the way that you configure your security stuff [...],"* or P18, *"I haven't really thought about security in a while, but this is a good reminder."*

3.5 Security Recommendations

We leverage our results to give concrete recommendations that can help service providers and security experts in improving users' account security.

Two-factor authentication Our participants noted usability and privacy concerns specific to phone-number-based two-factor authentication, but were often unaware of alternatives. Furthermore, some participants thought they would necessarily have to provide the second factor with every login. Participants also generally overestimated the number of times they log in to their accounts. Finally, participants were happy to use two-factor authentication as a secure default that is set up at registration, while they would not go through the steps to set it up afterwards.

These points suggest that the following recommendations for *service providers* may increase adoption of two-factor authentication:

- They should support and advertise *authenticator apps linked to devices* rather than only phone-number-based options.
- They should inform users clearly that they can *configure trusted devices* for which the second factor is not required.
- They should offer two-factor authentication as the *default option* during registration.

Password reuse and attacker models Participants that *reused passwords* were not too concerned with attacks because they believed that they had to be *targeted*, thinking of an attacker as someone who had decided in advance to hack them specifically. They were also largely unaware of password database breaches [45]. The interviewer showed *haveibeenpwned.com* to participants who reused passwords, and the vast majority were unaware of the site or any similar tools. This suggests that *security experts* should focus on the following points when communicating risks to users:

- They should increase *awareness of password database breaches* and resulting password reuse risk.
- They should *explain the untargeted nature of credential stuffing attacks*, emphasizing that an attacker does not have to consciously pick a user as their target.

Personalized security advice Security advice that we gave to our participants was tailored to their specific setup and experience, unlike more generic advice that is commonly given. In particular, we could single out accounts that were central to each participant's setup, providing access to many other accounts. When these accounts were protected weakly, such as with a reused or weak password, we gave advice on how to better protect this account specifically. Most participants were highly receptive to this kind of advice, with many saying that they would implement it after the interview.

Thus, we believe that *security experts* should seek out opportunities for *personalized security counseling* rather than only giving generic advice. We also believe that it would be promising to automate our process, providing personal security advice at scale.

Privacy concerns Privacy with respect to service providers was an obstacle for adopting security mechanisms. Participants not only had concerns about phone-number-based authentication, but also about biometric authentication methods. Another obstacle was privacy concerns with respect to tool providers; some participants were hesitant to use cloud-based password synchronization.

This suggests that *service and application providers* should offer *privacy-preserving* authentication methods and tools, as these are more likely to be adopted by users. They should also *explain the privacy properties clearly*, since it is the users' perception that guides their decision.

3.6 Related Work

Studies related to ours generally fall into one of three categories. The first kind are studies directly related to the connections between users' accounts and devices arising from recovery methods. The second kind more broadly aims to understand users' motivations to follow security best practices, their security-related mental models, and their risk awareness. The third kind more narrowly focuses on specific aspects of account security, such as passwords and two-factor authentication. Our study bridges a gap between the latter two kinds of studies. While we combine different aspects, all are directly related to users' account security.

3.6.1 Account recovery

Despite its importance to user account security, there is only little previous work on the topic of account recovery specifically. Li et al. have analyzed account recovery methods of websites, finding that most websites use e-mail based account recovery [50]. They also propose a protocol for selectively protecting account recovery e-mails with a second factor, called SEAR (Secure Email Account Recovery). They have more recently expanded on their work, providing a detailed security analysis of their proposed SEAR protocol [51].

Ulqinaku et al. [52] have recently performed a user study on FIDO (FastIdentity Online) [53] downgrade attacks, which allow attackers to downgrade a non-phishable FIDO authentication to a phishable alternative factor. This is closely related to our notion of backdoors, and is an interesting example of a security weakness incurred

by alternative account access methods. Their work focuses on this kind of backdoor specifically, and can thus be seen as investigating a special case of the more general problems analyzed in this thesis.

3.6.2 Following security advice

We first discuss research that aims to understand why users follow or do not follow security advice. This relates to our research question on willingness to improve security.

Fagan and Khan [46] conducted a survey on users' motivations to follow security advice. In particular, they studied perceived benefits, costs, and risks of following, or not following, four pieces of security advice: Keep your software up to date, use a password manager, use two-factor authentication, and change passwords frequently. They compared the perceptions of users who follow the advice with users who do not. Most people who did not follow the advice valued convenience highly, and this effect was particularly pronounced for two-factor authentication. This is consistent with our observation that our participants overestimated the required effort for using two-factor authentication.

Ion et al. [47] compared the security practices recommended and followed by security experts with those of non-experts. With respect to account security, they found that using two-factor authentication and password managers was mentioned more frequently by experts, while using strong passwords and changing passwords frequently was mentioned more frequently by non-experts. A replication study was performed by Busse et al. [48], with largely similar results, additionally asking the experts how realistic they consider each piece of advice to be. In particular, they considered the advice of using two-factor authentication as not very realistic to be followed. This was partly due to lack of support by service providers. Our results suggest that, if supported by service providers, it may be realistic for users to use two-factor authentication for their most important accounts.

3.6.3 Security mental models and risk awareness

We next discuss work on users' mental models and security risk awareness. This relates to our research questions on awareness and attacker models, and is also closely linked to users' willingness to improve security.

Wash [54] conducted an interview study about users' mental models of viruses and hackers. He showed that a user's mental model affects how they perceive threats and thus how likely they are to follow different security advice. This relates to our insights that our participants' attacker models influence their tendency for password reuse.

Harbach et al. [55] conducted a survey on which security risks are considered by Internet users. This includes risks related to their accounts with websites. They noted that *"what it really means to have an account hijacked or how credentials can or cannot get hacked is likely unclear to participants, as their often abstract and unspecific responses suggest." [55]* Our study provides a more differentiated view on the topic, with users having different levels of awareness with respect to different kinds of threats.

Kang et al. [56] performed a qualitative study to understand people's knowledge and mental models about the Internet. Their participants commonly did not take action because they did not consider their accounts important. They had participant groups from different backgrounds, and even one of the participants with a technical background said *"I don't care who sees and reads my email." [56]* This further suggests that many users are unaware that their email accounts often provide access to most of their other accounts, as can be seen in our participants' account access graphs.

3.6.4 Passwords

The topic of user passwords and related risks and solutions has been studied extensively. For a literature review on the topic up to 2014, we refer the reader to Taneski et al. [57]. We next discuss some work that closely relates to the aspects we focused on in our study, in particular to password reuse.

Gaw and Felten [33] studied users' password management strategies, with a focus on password reuse. They found that their participants use unique passwords for their more important accounts, in line with our findings. They also discussed users' perceived threat models, but with respect to attackers trying to compromise passwords specifically.

Das et al. [34] investigated partial password reuse and how an attacker can leverage a known password for guessing different passwords of the same user. They describe in detail the transformation rules users employ to generate new passwords from existing ones, a topic that is also mentioned by our participants.

Pearman et al. [35] conducted a large-scale *in situ* study of users' password behaviors, obtaining detailed data about password reuse in particular. They did not find a correlation between using password managers and password reuse, suggesting that their participants did not use randomly generated passwords. They found correlations between website categories and password reuse, with decreased reuse on government websites and increased reuse on shopping and job search websites. They conjectured that the difference was due to perceived account importance, and noted that it was *somewhat surprising considering that shopping website passwords may protect sensitive credit card data and that job- and work-related sites may contain [...] payroll and employment information. [35]* This is another indicator that users may not accurately assess the importance of their accounts.

Lyastani et al. [36] explicitly studied the relationship between using password managers and password reuse in a study with both qualitative and quantitative elements. Their work poses the question why people that use password managers still reuse passwords. Our study provides some insight into this: some participants felt uncomfortable only using randomized passwords, and others used password managers as a digital notebook on a separate device rather than having their passwords synchronized across all their devices.

Pearman et al. [58] conducted an interview study on the use of password managers. Our study corroborates their findings that people reuse passwords, despite using a browser-based password save feature.

3.6.5 Two-factor authentication

We next discuss work that is related to the usability of two-factor authentication and how its adoption could be increased, a topic on which we also provide recommendations.

Redmiles et al. [37] conducted a study on messages asking users to enable two-factor authentication. Their participants both critiqued existing messages and designed new ones. They preferred simple and clear messages, including information about the required time investment for setup. Participants also mentioned they were more likely to use two-factor authentication only for their more important accounts.

Albayram et al. [38] investigated how video tutorials on two-factor authentication can increase users' willingness to adopt the technology. Interestingly, one of their participants gave the following reason for not enabling two-factor authentication after

watching the video: *"I do not want [...] sites I use to have my cell phone number, as I don't feel like I can trust them with it." [38]* That is, their participant was unaware of solutions that do not require the phone number, similar to our participants.

De Cristofaro et al. [39] performed a comparative usability study of three two-factor authentication solutions: codes generated by security tokens, received via email or SMS, or generated by smartphone apps. Their participants chosen were already two-factor authentication users. They found that two-factor authentication was overall perceived as usable by their participants. Their participants mentioned troubles with SMS-based authentication when abroad, again similar to what we observed.

4

Privacy-preserving OpenID Connect

In our user study results, summarized in Section 3.2, we have observed that many participants reused passwords for their less important accounts, but were willing to put effort into protecting the accounts central to their setup. Therefore, *single sign-on* solutions, where users use their account with an identity provider to log in to other services, called relying parties, are a promising way to improve users' overall account security. Our user study results suggest that users may realistically use a strong and unique password as well as two-factor authentication for their central account with an identity provider. They can then use this account to log in to various different services, for which they might otherwise have reused passwords or used weak passwords.

However, the current de-facto standard for single sign-on, the OpenID Connect [1] protocol, comes with a significant privacy trade-off. The identity provider learns to which relying parties the user logs in, and the exact time of these logins. This is in contrast to one of our security recommendations we give in Section 3.5, namely to focus on *privacy-preserving* technology to increase user adoption of security best practices and tools. In this chapter, we provide such a technology. We present privacy-preserving extensions of OpenID Connect, which prevent the identity provider from learning to which relying parties the user logs in. Thus, users no longer have to choose between security and privacy, but provably obtain strong guarantees for both aspects.

This chapter is organized as follows. In Section 4.1, we give background on OpenID Connect, and in Section 4.2, we explain the two different proof systems we used for

security and privacy proofs, respectively. In Section 4.3, we explain the system model and attacker model that we consider for this chapter. In Section 4.4, we explain our design of the Privacy-preserving OpenID Connect (POIDC) protocol in general first, and then give multiple variants of the protocol that achieve different functional properties in addition to the desired privacy property. We also give cryptographic privacy proofs for all variants. In Section 4.5, we show how POIDC can be further extended to pairwise POIDC, achieving stronger privacy also with respect to colluding relying parties that aim to track the user. In Section 4.6, we define our desired authentication security property, and explain how we proved it using the protocol verification tool TAMARIN. We compare the work presented in this chapter with related work in Section 4.7.

4.1 Background on OpenID Connect

OpenID Connect is a *delegated authentication protocol* built on top of the OAuth 2.0 standard [59]. It involves three parties: identity providers (IdPs), relying parties (RPs), and users. A user can use their account at an IdP to log in to an RP. In the process, the IdP sends a short-lived JSON Web Token (JWT) [60], called *id_token*, to the RP, either directly or through the user agent, which is typically a web browser.

The IdP is called the OpenId Connect Provider (OP) or authorization server in the official specification [1]; we use the term IdP because it is more generally used for delegated authentication. The RP is sometimes also called the client, its OAuth 2.0 terminology [59]. In particular, the RP's identifier assigned to it by the IdP is called its *client_id*. We avoid using the term client directly to prevent confusion with the user and her user agent.

Example 27. *A popular OpenID Connect implementation is* Google Sign-In*. A user can click on a* Sign In With Google *button, for example on* stackoverflow.com*. She is then redirected to Google, where she is asked to log in to her Google account, if she is not already logged in. The Google page then displays a message that asks the user to confirm that she wants to log in to* Stack Overflow*. She must also confirm that* Stack Overflow *may access her e-mail address. Afterwards, she is redirected to* stackoverflow.com*, where she is logged in and her user profile already contains her verified e-mail address.*

More generally, when a user wishes to log in to an RP using OpenID Connect, she first selects her IdP from a given list of alternatives. This is usually done by selecting

the appropriate button on the RP's page. The user is then redirected to the IdP, where she authenticates herself. Note that the IdP's authentication mechanism is not part of the OpenID Connect protocol definition. The IdP then asks the user for consent to log in to the RP and potentially to release some of her profile information to the RP, such as the e-mail address with which she is registered at the IdP. The next steps depend on which OpenID Connect flow is used.

In *implicit flow*, when the user gives consent, the IdP sends a signed *id_token* to the RP via the user's browser. The *id_token* includes an *issuer identifier* identifying the IdP, a *subject identifier* identifying the user, and an *audience identifier* denoting the identity (called *client_id*) of the intended recipient RP. This ensures that the *id_token* will only be accepted by that RP. The *id_token* may contain additional information, such as the user's e-mail address, if requested by the RP.

In *authorization code* flow, a *code* is first forwarded via the user's browser to the RP. The *code* is a random value generated by the IdP, which is then exchanged for an *id_token* via a direct channel between the RP and the IdP. In this flow, the RP might additionally have to authenticate to the IdP by providing a secret it shares with the IdP, called the *client_secret*.

Our first privacy goal is to prevent the IdP from learning to which RP the user logs in. A direct channel between the RP and IdP is incompatible with this goal. We thus use implicit flow as the base for our extensions.

Note that we focus on the *id_token* functionality offered by OIDC only and we do not consider OAuth 2.0 access tokens. An access token could be used by an OAuth 2.0 client, corresponding to the RP in OpenID Connect, to access the user's resources at a resource server, which is commonly part of the IdP. This functionality again requires direct communication between the client (RP) and resource server (part of the IdP). Thus, our privacy-preserving protocols focus on pure OpenID Connect functionality, sending an *id_token* only. For this reason, using the implicit flow does not expose our protocols to security risks [61] that are present when using this flow to exchange an access token. In particular, the *id_token* is usually shorter lived than an access token and contains a mandatory *nonce* in implicit flow for replay protection. Throughout this chapter, we denote standard OpenID Connect by OIDC, referring to the implicit flow, unless stated otherwise.

Example 28. *In Example 27, the id_token contains:* www.google.com *as the issuer identifier; the* identifier for stackoverflow *(called client_id) chosen by Google as the audience identifier; the identifier Google assigned to the* user *as the subject identifier; the user's e-mail address; and a nonce that was generated by the RP.*

There are two kinds of subject identifiers in OIDC, called *public* and *pairwise* subject identifiers. A public subject identifier is a unique identifier for a user at an IdP. Pairwise subject identifiers are specific to the RP, so the IdP looks up the user's pairwise identifier for a specific RP when creating an *id_token* for that RP.

OIDC has the following setup: The RP must register with the IdP and provide a set of URIs called *redirect_uris* that belong to the RP and to which *id_token*s may be sent. The RP provides additional metadata, such as its human-readable name. The IdP displays this metadata to the user when asking for consent to log in to this RP.

OIDC (implicit flow) consists of the following nine steps, illustrated in Figure 4.1. Note that we omitted the *response_type* parameter; since we do not consider the combination of OpenID Connect *id_token*s with OAuth 2.0 access tokens, we assume that the *response_type* is set to *id_token*. We do not explain every detail of the protocol, but focus on the parts that will be relevant for our privacy-preserving extensions.

1. The user, using her user agent (typically a web browser), initiates the protocol by requesting to log in to the RP. She specifies which IdP she wants to use.

2. The RP redirects the user agent to the IdP, sending its *client_id* and a freshly generated *nonce* as query parameters.

3. The user agent is redirected to the IdP, forwarding these query parameters to the IdP (the user also logs in at the IdP, if necessary, which is not part of OIDC).

4. The IdP opens a dialogue to be displayed in the user agent, asking the user to confirm that she wishes to log in to the RP belonging to the *client_id*. For this, the IdP looks up a human-readable *client_name* that belongs to the *client_id*.

5. The user clicks the confirm button in the dialogue.

6. The IdP returns an *id_token* as described earlier. In particular, the *id_token* contains the *client_id* in its *aud* (audience) field, and the *nonce* that was chosen by the RP.

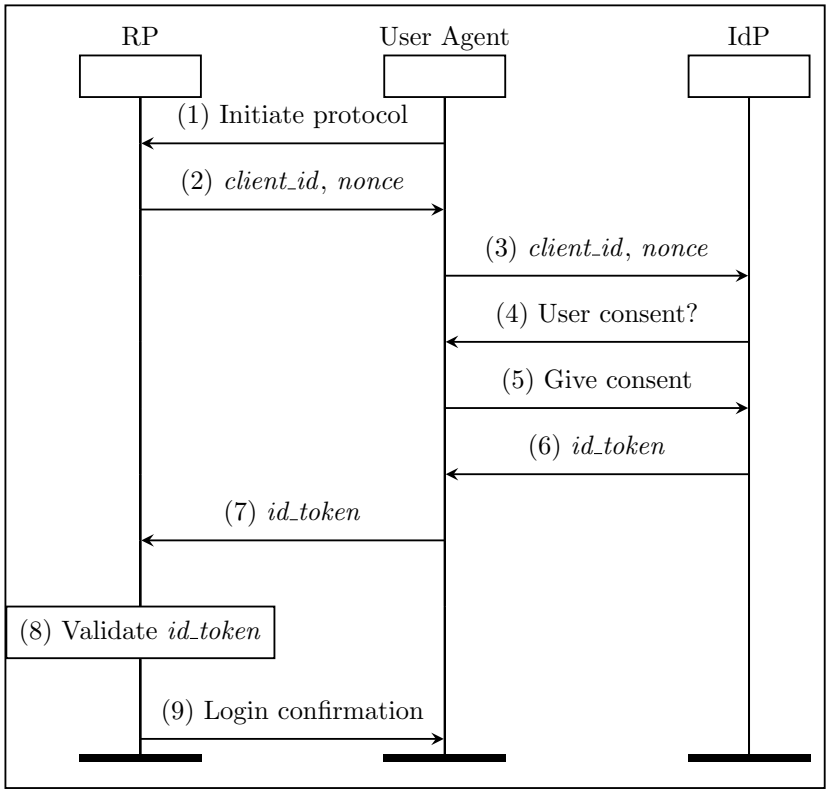

Figure 4.1: OpenID Connect implicit flow.

7. The user agent forwards the *id_token* to the RP (namely, to an endpoint called the *redirect_uri*, which must be one of the *redirect_uris* that are registered at the IdP for the RP).

8. The RP validates the IdP's signature on the *id_token* and compares the *aud* field with its own *client_id*. The RP accepts the *id_token* only if these values are equal (and signature validation succeeds).

9. The RP notifies the user agent whether the login was successful.

4.2 Background on proof systems

We use cryptographic game-based pen-and-paper proofs to prove our *privacy* properties and symbolic machine-checked proofs using the Tamarin prover to prove our security property related to user authentication. We next explain in more detail why we use these two different proof systems.

Our primary *privacy* property is formulated with respect to an honest-but-curious IdP who tries to correlate messages received in different protocol runs. We explain the motivation for modeling the IdP in this way in Section 4.3.2. The proper way to prove privacy is thus to analyze in detail the information that the IdP can obtain from these messages. Concretely, the IdP tries to distinguish scenarios where a user logs in to different RPs. Thus, we introduce cryptographic distinguishability games for these kinds of proofs.

Our *security* property is formulated with respect to a standard Dolev-Yao network attacker. Standard OpenID Connect has previously been proven secure with respect to such an attacker [5], and our goal is to prove that our privacy-preserving extensions do not weaken the protocols' security. To this end, we use the protocol verification tool Tamarin that works in the symbolic model. We extend its built-in Dolev-Yao attacker model with a model for abstract TLS [62] channels.

4.2.1 Cryptographic definitions for privacy proofs

We will formalize our privacy properties as cryptographic games. We use the following standard definitions, see, e.g., [63, 64].

An *efficient attacker* is an attacker that is bounded by probabilistic polynomial-time in the security parameter λ. A *negligible* function is a function that vanishes faster than any inverse polynomial. A *cryptographic game* $G^{\mathcal{A}}(\lambda)$ is a randomized algorithm that depends on the security parameter λ and the attacker $\mathcal{A}$. The *advantage* of an attacker $\mathcal{A}$ in a cryptographic game is given as $|\Pr[G^{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$. We say that an adversary *wins* a cryptographic game if and only if the game outputs 1. Note that we often leave the security parameter λ implicit in our definitions.

We write $x \leftarrow\!\!\$\, D$ to denote that x is sampled from probability distribution D. We write $x \leftarrow\!\!\$\, S$ to denote that x is sampled uniformly at random from a set S. We write $\mathrm{rnd} \leftarrow\!\!\$\, \mathcal{R}$ to denote that rnd is a cryptographically secure random number.

Definition 42. *A* public-key encryption scheme *[63] is a tuple of polynomial-time algorithms* $\Pi := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$. $\mathsf{KGen}(1^\lambda)$ *is a probabilistic key-generation algorithm that outputs a key-pair* $(\mathsf{pk}, \mathsf{sk})$. $\mathsf{Enc}_{\mathsf{pk}}(\mathrm{m})$ *is a probabilistic encryption algorithm taking as argument a public key* pk *and a plaintext* m *and outputs a ciphertext. For definitions that follow later in this chapter, it will be necessary to make the randomness used in* $\mathsf{Enc}_{\mathsf{pk}}$ *explicit. We thus introduce the non-standard notation* $\mathsf{Enc}_{\mathsf{pk}}(\mathrm{m}, \mathrm{rnd})$*, where* rnd *is the randomness used. That is, calling* $\mathsf{Enc}_{\mathsf{pk}}$ *with the same* m *and* the same value for rnd *always produces the same result.* $\mathsf{Dec}_{\mathsf{sk}}(c)$ *is a deterministic decryption algorithm that takes as input a secret key* sk *and a ciphertext* c*, and outputs a plaintext or a special symbol* $\perp$ *denoting failure.*

Definition 43. *A public-key encryption scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *provides* indistinguishability under chosen plaintext attacks *[63], or is* IND-CPA secure *for short, if any efficient attacker* $\mathcal{A}$ *only has negligible advantage in winning the following distinguishability game. Note that* $\mathcal{A}$ *keeps state between the different stages of the game.*

1. *The challenger invokes* $\mathsf{KGen}(1^\lambda)$ *and obtains* $(\mathsf{pk}, \mathsf{sk})$*. It sends* pk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *is given oracle access to* $\mathsf{Enc}_{\mathsf{pk}}(.)$*, i.e., it can call* $\mathsf{Enc}_{\mathsf{pk}}(\mathrm{m}_i)$ *for arbitrary messages* m_i*.*
3. $\mathcal{A}$ *outputs a pair of plaintexts* $(\mathrm{m}_0, \mathrm{m}_1)$ *of equal length.*
4. *The challenger picks* $b \leftarrow\!\!\$\, \{0, 1\}$*, chooses* rnd *at random with respect to the distribution given in the definition of* Enc [1]*, and sends the ciphertext* $c := \mathsf{Enc}_{\mathsf{pk}}(\mathrm{m}_b, \mathrm{rnd})$ *to* $\mathcal{A}$*.* $\mathcal{A}$ *continues to have oracle access to* $\mathsf{Enc}_{\mathsf{pk}}(.)$ *at this stage.*
5. $\mathcal{A}$ *outputs a bit* b'*, and wins if* $b = b'$*.*

4.2.2 Symbolic model for security proofs

We next explain the Tamarin prover and the symbolic model we use for our security proofs in more detail.

4.2.2.1 The Tamarin prover

Tamarin [6, 7] is a state-of-the-art automated tool for security protocol verification in the symbolic model of cryptography. In Tamarin, protocols are described by multiset

[1] The usual definition of Enc does not contain the randomness as an argument. Instead, the choice of random bits is part of the encryption algorithm's definition itself.

rewriting rules, and trace properties, like secrecy and authentication, are specified in a first-order logic fragment. TAMARIN provides a counterexample, representing an attack, if a verification attempt terminates without proof. In general, termination is not guaranteed due to the undecidability of the underlying protocol verification problem.

TAMARIN works in the symbolic model, also called Dolev-Yao (DY) model, where messages are represented as terms. The DY attacker represents an attacker who controls the network. He sees all messages and can block, reorder, and manipulate them, but he cannot break cryptography, e.g., he cannot forge signatures.

Security protocol execution is represented by a labeled state transition system. Here, a *state* is a finite multiset of so-called facts, including terms as their arguments, and models the current snapshot of the protocol's execution. This includes the local state of all participants, as well as all messages sent over the network and the messages an attacker can construct from those. The steps performed by a protocol participant or the attacker are specified as *rules*, rewriting one state into another. Rules are of the form `name: l -[ a ]--> r` with `l`, `a`, and `r` all finite sequences of facts. `l` are called the premises, `a` the actions, and `r` the conclusions of the rule. A step of multiset rewriting starts in a given state s, uses a rule `name: l -[ a ]--> r`, and can be executed if there is a substitution σ such that $\sigma(l) \in s$. Triggering the rule results in a new state $s' = (s \setminus \sigma(l)) \cup \sigma(r)$. Note that we simply write `name: l --> r` if there are no actions, or we present a rule where actions are omitted for brevity.

An execution starts with the empty multiset and alternates states and rewriting rule instances. The rule instance between two states is a step between those states. The *trace* of the execution is the sequence of the used rules' actions. A *trace property* is defined over traces, and holds for a protocol if and only if it holds on all possible traces for that protocol, resulting from all possible interleavings of executions of multiset rewriting rules modeling the protocol. These rewriting rules model protocol execution steps from an unbounded number of concurrent protocol sessions and attacker actions.

4.2.2.2 Signature model

Symbolic models of cryptography have traditionally formalized signature verification by the equation

$$\text{verify}(\text{sign}(\text{msg}, \text{sk}), \text{msg}, \text{pk}(\text{sk})) = \text{true} \ .$$

Any input to verify that is not exactly of this form does not match the pattern and thus implicitly returns false. Our models instead use an improved, more precise, way to model signatures recently introduced by Jackson et al. [8]. They show that using the equation above does not accurately capture subtle behaviors that are present even in signature schemes that are considered secure with respect to standard cryptographic definitions. We next give the cryptographic definition for signature schemes, and then explain how the symbolic model introduced in [8] relates to these definitions.

Definition 44. *A digital signature scheme [63] is a tuple of polynomial-time algorithms* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$. $\mathsf{KGen}(1^\lambda)$ *is a probabilistic key-generation algorithm that outputs a key-pair* $(\mathsf{pk}, \mathsf{sk})$. $\mathsf{Sig}_{\mathsf{sk}}(\mathrm{m})$ *is a probabilistic signing algorithm that takes as input a secret key* sk *and a message* m *and produces a signature* σ *on* m. $\mathsf{Vf}_{\mathsf{pk}}(\mathrm{m}, \sigma)$ *is a deterministic verification algorithm that takes as input a public key* pk*, a message* m*, and a signature* σ*. It returns either* true *or* false*.*

We require that a digital signature scheme satisfies correctness*, defined as follows: For any* $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{KGen}(1^\lambda)$, $\mathsf{Vf}_{\mathsf{pk}}(\mathrm{m}, \mathsf{Sig}_{\mathsf{sk}}(\mathrm{m})) = \text{true}$.

This standard correctness definition explicitly only prescribes whether Vf must return true when it is called with a public key pk that was generated by the key-generation algorithm $\mathsf{KGen}(1^\lambda)$. We call such a public key *honestly generated* for short.

We next explain the standard security definition for digital signature schemes.

Definition 45. *A digital signature scheme* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *provides* existential unforgeability under chosen message attacks*, or is* EUF-CMA secure *for short, if an efficient attacker* $\mathcal{A}$ *only has negligible probability of winning the following game.*

1. *The challenger invokes* $\mathsf{KGen}(1^\lambda)$ *and obtains* $(\mathsf{pk}, \mathsf{sk})$*. It sends* pk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *is given oracle access to* $\mathsf{Sig}_{\mathsf{sk}}(.)$*. Let* $\mathrm{M}_{\text{queried}}$ *denote the set of messages* m_i *for which* $\mathcal{A}$ *has invoked the oracle to obtain* $\mathsf{Sig}_{\mathsf{sk}}(\mathrm{m}_i)$*.*
3. $\mathcal{A}$ *outputs* (σ^*, m^*) *for* $\mathrm{m}^* \notin \mathrm{M}_{\text{queried}}$*. He wins if* $\mathsf{Vf}_{\mathsf{pk}}(\mathrm{m}^*, \sigma^*) = \text{true}$*.*

Note that the probability *of* $\mathcal{A}$ *winning this game must be negligible, and not the* advantage*, since this is not a distinguishability game.*

Similar to the correctness definition, EUF-CMA security is defined with respect to an honestly generated public key. That is, neither the correctness nor the security

definition impose any restrictions on signatures that are verified with respect to a public key that is *not* generated honestly, i.e., that was not generated by calling $\mathsf{KGen}(1^\lambda)$.

For example, it may be possible for an attacker to generate a public key pk' and a signature σ' such that $\mathsf{Vf}_{\mathsf{pk}'}(\sigma', \mathrm{m}) = \text{true}$ for *any* message m. These kinds of behaviors are not modeled by the equation traditionally used in symbolic verification, and thus verification would not find any attacks that exploit such behavior. We use the updated symbolic *verification model* introduced in [8] that works as follows.

- The verify check *must* return true if not doing so would violate the *correctness* of the signature scheme.
- The verify check must *not* return true if doing so would violate *EUF-CMA security* of the signature scheme.
- In all other scenarios, e.g., when verify is used with respect to a public key that was *not* generated using $\mathsf{KGen}(1^\lambda)$, *the attacker chooses the result* returned by verify, while maintaining consistency between multiple checks with the same arguments.

This symbolic model provides a sound over-approximation of the subtle behaviors that may be possible even for an EUF-CMA secure signature scheme. In particular, a proof in this model holds for any EUF-CMA secure signature scheme, even if the attacker could exploit unexpected behavior that may arise when signatures are verified using non-honestly generated public keys.

4.3 Setup, assumptions, and properties

In this section, we explain our setup, including the attacker models that we consider when analyzing privacy and security.

4.3.1 System model

The following parties are involved in our protocols.

The RP exposes an interface, such as a button, which allows the user to send login requests from its user agent. It also contains endpoints for requests sent to a *redirect_uri* registered for the RP.

The user agent is a standard browser that can perform actions, such as cryptographic operations, by loading JavaScript code from web pages. This code is executed locally in the browser and is therefore part of the user agent. Messages that are sent to the user agent but are not forwarded are sent as URI fragment identifiers [65]. These identifiers are accessible to the JavaScript code and can thus be processed by the user agent. However, unlike query parameters, they are not sent to the IdP back-end.

The IdP corresponds to the *IdP back-end only*. Messages sent to the IdP in our protocol descriptions are sent as query parameters.

4.3.2 Attacker models

We employ different attacker models for the properties we analyze. We consider *security* with respect to a standard *Dolev-Yao network attacker*, which we formalize in Section 4.6. For *privacy*, we consider two different attacker models: An *honest-but-curious IdP*, and a set of *colluding RPs*.

Definition 46. *An* honest-but-curious IdP *follows the protocol, but retains detailed transcripts of each protocol run that contain every message the IdP received during the run. The IdP analyzes these transcripts and correlates information from transcripts for different protocol runs.*

We argue that this is the right adversary model for analyzing OpenID Connect or variants thereof. First, a malicious IdP can trivially violate its users' privacy and security. The IdP can log in under one of its users' account at any RP by forging *id_token*s. Thus, it can learn which RPs the user visits, namely those where the user has an account. It can even obtain additional data from the user's account at the RP.

Our assumption that the user's IdP is not malicious is clearly a basic assumption underlying standard OpenID Connect. Moreover, this assumption is realistic for IdPs associated with major companies such as Google or Microsoft, which are concerned about their reputation. If an IdP's misbehavior were detected, then news of this would spread quickly, with disastrous effects for the IdP. While corporate IdPs are extremely sensitive about their reputation, they are also businesses and their business model often includes monetizing user data, for example through targeted ads. Thus, while the IdP has strong incentives to adhere to its protocol definition, it still wishes to gather as much user data as possible, provided this cannot be detected.

The other attacker model that we consider with respect to privacy is that of colluding RPs.

Definition 47. *A set of* colluding RPs *consists of (potentially malicious) RPs that share information obtained during protocol runs with each other, such as information contained in id_tokens.*

Colluding RPs try to link their users' accounts to the same person. When an *id_token* includes the user's e-mail address, then they can trivially link this information. However, when no such information is explicitly given, linking accounts should not be possible. In standard OIDC, there are two different kinds of subject identifiers: public and pairwise. If public identifiers are used, then the same identifier is used for the user at each RP, and thus colluding RPs can link two accounts to the same person. Pairwise identifiers prevent this by assigning a different identifier for the user for each RP.

Finally, note that when analyzing privacy, in contrast to security, we do not consider a network attacker. A network attacker can observe the traffic between users and RPs to see where a user logs in. The problem of protecting user privacy against a network attacker is independent of protecting user privacy with respect to the IdP or RPs, and it cannot be solved by the delegated authentication protocol alone. To ensure privacy against a network attacker, a user must employ other methods such as mix networks [66] or Tor [67] to realize an anonymous communication channel.

4.3.3 Privacy properties

We now state the privacy properties to be achieved by our protocols, first informally and then formalized as cryptographic games.

Definition 48. Login unlinkability. *A delegated authentication protocol P provides* login unlinkability *if, for any given honest user* u*, and any two honest RPs* rp_0 *and* rp_1 *chosen by the attacker, an honest-but-curious IdP attacker $\mathcal{A}$ cannot distinguish between a transcript of a protocol run where user* u *logs in to* rp_0*, and one where* u *logs in to* rp_1*, even if it has access to additional transcripts from previous protocol runs by any users. An honest user is a user that is not controlled by the attacker, i.e., none of the user's credentials are known by the attacker. An honest RP follows its protocol description and does not collude with the IdP.*

Formally, an attacker $\mathcal{A}$ can distinguish these two scenarios if he has non-negligible advantage in the IdP-login-linking *game, defined as follows for protocol P.*

1. *The challenger sets up the environment that is required to run protocol P according to a set of parameters* setupParams. *We do not define this set exhaustively, but only describe the relevant elements. In particular,* setupParams *contains a list of registered RPs and their associated client_ids. Note that we assume that all client_ids have the same length. The challenger also chooses an honest user* u, *and sends* setupParams *and* u *to* $\mathcal{A}$.

2. $\mathcal{A}$ *is given oracle access to* $\mathrm{RunProtocolIdP}_P(\mathrm{u}', \mathrm{rp}')$. *This function returns a transcript* tr' *of messages sent* to the IdP *during the protocol run where user* u' *logs in to RP* rp'.

3. $\mathcal{A}$ *outputs a pair of RPs* $(\mathrm{rp}_0, \mathrm{rp}_1)$.

4. *The challenger picks* $b \leftarrow\!\$\, \{0,1\}$, *obtains* $\mathrm{tr} := \mathrm{RunProtocolIdP}_P(\mathrm{u}, \mathrm{rp}_b)$, *and sends* tr *to* $\mathcal{A}$. $\mathcal{A}$ *continues to have oracle access to* $\mathrm{RunProtocolIdP}_P(\mathrm{u}', \mathrm{rp}')$ *at this stage.*

5. $\mathcal{A}$ *outputs a bit* b', *and wins if* $b = b'$.

Note that this defines a stronger requirement than secrecy of the RP's identity. In particular, it also prevents the IdP from linking repeated logins of the same user to the same RP. The oracle access to $\mathrm{RunProtocolIdP}_P$ overapproximates the information that the IdP has from previous protocol runs by any users.

We later present a setting where the IdP must be allowed to learn the total number of logins to an RP during a given time period. Any protocol that allows the IdP to obtain this information cannot satisfy login unlinkability: The number of total logins to RPs for the two scenarios where the user logs in to rp_0 or to rp_1 will be different. Thus, if we are in a setting with this requirement, we aim for a slightly weaker privacy property, which we call login exchangeability.

Definition 49. Login exchangeability. *A delegated authentication protocol provides* login exchangeability *if, for any two given honest users* u_0 *and* u_1, *and any two honest RPs* rp_0 *and* rp_1 *chosen by the attacker, an honest-but-curious IdP attacker* $\mathcal{A}$ *cannot distinguish between transcripts of protocol runs for the following two scenarios, even if it has access to additional transcripts from previous protocol runs (by any users).*

1. u_0 *logs in to* rp_0, *and* u_1 *logs in to* rp_1.

2. u_0 *logs in to* rp_1, *and* u_1 *logs in to* rp_0.

Formally, an attacker can distinguish these two scenarios if he can win the IdP-login-matching *game with non-negligible advantage, defined as follows for protocol P.*

1. *The challenger sets up the environment that is required to run protocol P according to a set of parameters* setupParams. *In particular,* setupParams *contains a list of registered RPs and their associated client_ids. Note that we assume that all client_ids have the same length. The challenger also chooses* two honest users, u_0 *and* u_1, *and sends* setupParams *and* $(\mathrm{u}_0, \mathrm{u}_1)$ *to* $\mathcal{A}$.

2. $\mathcal{A}$ *is given oracle access to* $\mathrm{RunProtocolIdP}_P(\mathrm{u}', \mathrm{rp}')$. *This function returns a transcript* tr' *of messages sent* to the IdP *during the protocol run where user* u' *logs in to RP* rp'.

3. $\mathcal{A}$ *outputs a pair of RPs* $(\mathrm{rp}_0, \mathrm{rp}_1)$.

4. *The challenger picks* $b \leftarrow\!\!\$\, \{0,1\}$, *obtains* two transcripts, $\mathrm{tr}_0 := \mathrm{RunProtocolIdP}_P(\mathrm{u}, \mathrm{rp}_b)$ *and* $\mathrm{tr}_1 := \mathrm{RunProtocolIdP}_P(\mathrm{u}, \mathrm{rp}_{1-b})$, *and sends* $(\mathrm{tr}_0, \mathrm{tr}_1)$ *to* $\mathcal{A}$. $\mathcal{A}$ *continues to have oracle access to* $\mathrm{RunProtocolIdP}_P(\mathrm{u}', \mathrm{rp}')$ *at this stage.*

5. $\mathcal{A}$ *outputs a bit* b', *and wins if* $b = b'$.

Next, we define login unlinkability with respect to *colluding RPs* that aim to track users across RP boundaries.

Definition 50. Login unlinkability with respect to colluding RPs. *A delegated authentication protocol P provides* login unlinkability with respect to a set of colluding RPs *if a set S of (potentially malicious) RPs cannot distinguish, given only information learned from honest users' logins, whether any two logins to two different RPs* rp_0 *and* rp_1 *in S were performed by the same honest user* u_0 *or by two different honest users* u_0 *and* u_1. *An* honest user *is a user for which none of the user's credentials with the IdP are known by any of the colluding RPs. That is, they cannot distinguish the following two scenarios:*

1. u_0 *logs in to both* rp_0 *and* rp_1.

2. u_0 *logs in to* rp_0 *and* u_1 *logs in to* rp_1.

Formally, an attacker can distinguish these two scenarios if he can win the RP-login-linking *game with non-negligible advantage, defined as follows for protocol P.*

1. *The challenger sets up the environment that is required to run protocol P according to a set of parameters* setupParams*. In particular,* setupParams *contains a list of registered users and their associated* subject identifiers*, as well as a set of registered RPs S. The challenger also chooses* two honest users*,* u_0 *and* u_1*, and sends* setupParams *and* $(\mathrm{u}_0, \mathrm{u}_1)$ *to* $\mathcal{A}$*.*

2. *The challenger picks* $b \leftarrow\!\!\$\, \{0,1\}$ *and then makes calls to* $\mathrm{RunProtocolRP}_P$ *depending on* b*. This function returns a transcript* tr' *of messages sent* to the RP rp' *during the protocol run where user* u' *logs in to this RP. If* $b = 0$*, the challenger runs* $\mathrm{RunProtocolRP}_P(\mathrm{u}_0, \mathrm{rp}_0)$ *and* $\mathrm{RunProtocolRP}_P(\mathrm{u}_0, \mathrm{rp}_1)$*. If* $b = 1$*, the challeger runs* $\mathrm{RunProtocolRP}_P(\mathrm{u}_0, \mathrm{rp}_0)$ *and* $\mathrm{RunProtocolRP}_P(\mathrm{u}_1, \mathrm{rp}_1)$*. It then sends the results of these function calls to* $\mathcal{A}$*.*

3. $\mathcal{A}$ *outputs a bit* b'*, and wins if* $b = b'$*.*

Note that the adversary in this game does *not* obtain oracle access to the function $\mathrm{RunProtocolRP}_P$. The reason is that this function takes as input a user identifier u, which is not known to the set of colluding RPs. This user identifier abstractly models some global identifier for the user, which we use in our proofs. If such an identifier was known to the colluding RPs, then they could trivially link logins of the same user.

4.4 POIDC: Protecting against the IdP

In this section, we show how OpenID Connect implicit mode can be modified to obtain the privacy properties presented above while preserving the provided authentication security property.

4.4.1 The general POIDC protocol

The IdP learns to which RP the user logs in using OIDC because the *client_id* identifying the RP is sent to the IdP. The *client_id* serves two purposes. First, it is included in the *aud* (audience) field of the *id_token*, so that the token is only valid for that one RP. Second, the IdP looks up the following data for the RP based on its *client_id*:

1. a human-readable name for the RP (called the *client_name*), which it displays when asking the user for consent to log in to that RP; and

2. a set of *redirect_uris* that belong to the RP and can be used for sending an *id_token* to that RP.

Note that the trivial solution for privacy where the *client_id* is not sent to the IdP, and thus not included in the *id_token* either, would not be secure. Such an *id_token* would be valid for any RP, and a malicious RP could use any *id_token* it was sent to log in under the user's account to any other RP.

We now show how POIDC achieves privacy by hiding the RP's identity from the IdP while preserving security. That is, our goal is to provide the following properties:

1. Login-unlinkability with respect to an honest-but-curious IdP; and

2. security with respect to a Dolev-Yao network attacker.

POIDC relies on two main features. First, POIDC replaces the *client_id* with a one-time pseudonym for the RP, used as the audience of the *id_token*. Second, POIDC enables the user agent to ask the user for consent to log in to an RP and perform a redirect to the RP, since these steps can no longer be performed by the IdP.

4.4.1.1 Masking the *client_id*

We replace the *client_id* that is sent to the IdP with a hashed or encrypted value that gives the IdP no information about the actual *client_id*.

We use a different function to mask the *client_id* in each variant of POIDC to achieve different functional requirements. We first describe the general pattern. We abstractly define a masking function $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$ that takes as arguments the *client_id* and *nonce* sent by the RP, as well a cryptographically secure random value rnd supplied by the user agent. The function may also depend on the user u, as we will see later. The output $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$ is used as the one-time pseudonym for the RP, and should be instantiated such that it provides no information about the *client_id*. Note that we do not require that the *nonce* sent by the RP is a cryptographically secure random value with respect to our privacy properties; the randomization should depend only on rnd. We simply include the *nonce* in the masked value since it could leak information about the RP if not all RPs choose the *nonce* in the same way.

We include $F(client\_id, nonce, \mathrm{rnd}, \mathrm{u})$ in the *id_token*, in a field called *private_aud*. This field replaces OIDC's *aud* (audience) field, and we call a token that has a *private_aud* field a *private_id_token*. Such a token is different from a usual *id_token* (the otherwise mandatory *aud* field is missing). An RP that runs OIDC would therefore not accept a *private_id_token*. Thus, POIDC can be run in parallel with standard OIDC with no risk of interference.

4.4.1.2 User consent in user agent

In OIDC, the IdP presents a user consent dialogue asking her if she wishes to log in to a particular RP. In POIDC, the IdP cannot look up the required information since it does not know the RP's identity. Thus, a user consent dialogue is instead shown locally by the user agent without looking up information in the IdP back-end. For this, we must enable the user agent to map a *client_id* to a user-readable *client_name*. This mapping must be authentic, since an attacker could otherwise impersonate any RP to the user. This would allow the attacker to obtain an *id_token* of the user for that RP, and log in to the user's account at the RP.

We therefore introduce an additional step when the RP originally registers at the IdP (or when the IdP enables support for POIDC), where the IdP sends a signed token called a *client_id_binding* to the RP. The *client_id_binding* contains:

- the RP's *client_id*;
- the RP's *client_name*, the name of the RP as it can be understood and checked by the user; and
- the RP's set of registered *redirect_uris* that are valid endpoints to send the *private_id_token* to. The IdP must verify that these endpoints belong to the RP.

During a POIDC protocol run, the RP includes its *client_id_binding* token in a URI fragment so that it is sent to the user agent, but not to the IdP back-end. With this information, the user agent can construct a user consent dialogue displaying the *client_name*. The user agent will further only forward the *private_id_token* to one of the *redirect_uris* listed in the *client_id_binding*.

4.4.1.3 Protocol steps

The protocol steps of POIDC are illustrated in Figure 4.2. Variants of POIDC depend on the choice of the function F.

1. The user u initiates the protocol by requesting to log in to the RP. She specifies which IdP she wants to use, e.g., by clicking on the corresponding button on the RP's web page. This triggers a request sent from the user agent to the RP.

2. The RP redirects the user agent to the IdP, sending the *client_id_binding* (containing its *client_id*) and its *nonce* to the user agent as URI fragments. These URI fragments are accessible to JavaScript, but are *not* sent to the IdP back-end. This redirect must *not* contain an HTTP referer header.

3. The user agent is redirected to the IdP to load its website, but this request does not contain any query parameters.

4. The IdP sends its website to the user agent. The website contains JavaScript code that allows the user agent to perform the actions in Steps 5–9, as well as the IdP's public verification key $\mathrm{pk}_{\mathrm{IdP}}$ that is used in the next step.

5. The user agent executes the JavaScript code to verify the IdP's signature on the *client_id_binding* using $\mathrm{pk}_{\mathrm{IdP}}$. If this verification succeeds, the user agent then opens a dialogue for the user to confirm that she wishes to log in to the RP with the *client_name* in the *client_id_binding*.

6. The user agent generates a cryptographically secure random value rnd [1] and computes $\mathit{masked\_aud} := F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$. Note that if the instantiation of F depends on the user u, then it must specify which user-specific input it expects at this point.

7. It sends *masked_aud* to the IdP back-end, using, for example, an XMLHttpRequest [68].

[1] Note that the exact way rnd is generated may depend on the choice of F. For example, if F is a randomized public-key encryption function, then rnd would be generated properly as randomness for this function, according to its definition.

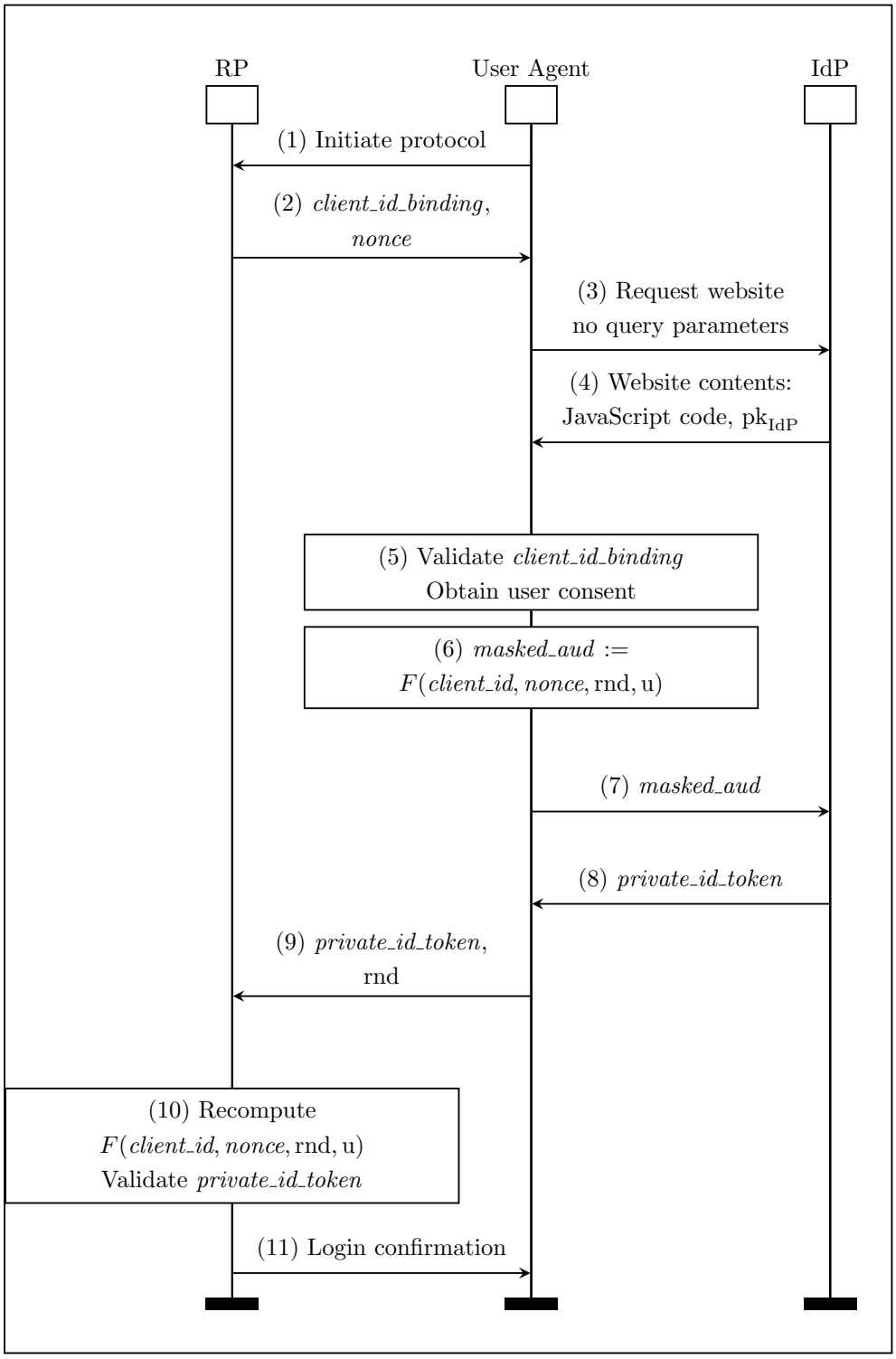

Figure 4.2: Privacy-preserving OpenID Connect.

8. The IdP back-end returns a *private_id_token* to the user agent, which contains $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$ in its *private_aud* field.

9. The user agent verifies that the *private_aud* field of the received *private_id_token* contains the value for $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$ it computed earlier. If this verification succeeds, the user agent sends this token and its rnd value to the *redirect_uri* given in the *client_id_binding*, using, for example, *window.open(redirect_uri)*.

10. The RP validates the IdP's signature on the *private_id_token* and recomputes $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$ from its own *client_id*, the *nonce* that it generated earlier, and the rnd value that it received from the user agent. It accepts the *private_id_token* only if its *private_aud* field matches this recomputed value (and signature validation succeeds).

11. The RP notifies the user agent whether the login was successful.

Recall that the IdP is honest-but-curious and thus, in Step 4, provides JavaScript code that correctly performs Steps 5–9.

Note that the user may have to log in to her account at the IdP if she does not have an existing session. This authentication step is, however, not explicitly part of the OIDC protocol, and thus is not part of POIDC either. A user authenticating to the IdP is also independent of the RP, so there is no risk of leaking information about the RP's identity during that process.

Parameter lengths. POIDC prescribes a fixed length L_c for *client_id*s assigned by the same IdP and L_n for *nonce*s chosen by all RPs using the same IdP. The concrete values of L_c and L_n do not matter as long as the parameters are of sufficient length to not be efficiently guessable. However, we assume in our privacy proofs that inputs to F have the same length, which requires this standardization.

4.4.1.4 Masking function privacy

We next define privacy properties with respect to $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$. The privacy properties of a masking function F directly determine the privacy properties provided by POIDC instantiated with F.

Definition 51. *A masking function* $F(\textit{client\_id}, \textit{nonce}, \mathrm{rnd}, \mathrm{u})$ *that belongs to a POIDC variant* P *is* *client_id*-unlinkable *if, for client_ids* cid_0 *and* cid_1 *and values* $\textit{nonce}_0$ *and* $\textit{nonce}_1$ *chosen by the attacker, and given an honest user* u*, any efficient attacker* $\mathcal{A}$ *cannot distinguish whether a value is of the form* $F(\mathrm{cid}_0, \textit{nonce}_0, \mathrm{rnd}, \mathrm{u})$ *or* $F(\mathrm{cid}_1, \textit{nonce}_1, \mathrm{rnd}, \mathrm{u})$.

Formally, F *is client_id-unlinkable for* P *if* $\mathcal{A}$ *has only negligible advantage to win the client_id-*linking game *for function* F*, defined as follows.*

1. *The challenger sets up the environment that is required to run protocol* P *according to a set of parameters* setupParams*. In particular,* setupParams *contains a list of registered RPs and their associated client_ids. Note that we assume that all client_ids have the same length* L_c *defined in* setupParams*. The challenger also chooses an honest user* u*, and sends* setupParams *and* u *to* $\mathcal{A}$*. Note that initializing the protocol environment is necessary for client_ids to exist, even though the protocol is not run directly during the game.*

2. $\mathcal{A}$ *is given oracle access to* F*, i.e., it can call* F *with any parameters.*

3. $\mathcal{A}$ *outputs a pair of client_ids* $(\mathrm{cid}_0, \mathrm{cid}_1)$ *and two values* $(\textit{nonce}_0, \textit{nonce}_1)$ *of length* L_n *as defined in* setupParams*.*

4. *The challenger picks* $b \leftarrow\!\!\$\, \{0,1\}$ *and* $\mathrm{rnd} \leftarrow\!\!\$\, \mathcal{R}$ *and sends* $F(\mathrm{cid}_b, \textit{nonce}_b, \mathrm{rnd}, \mathrm{u})$ *to* $\mathcal{A}$*.* $\mathcal{A}$ *continues to have oracle access to* F *at this stage.*

5. $\mathcal{A}$ *outputs a bit* b'*, and wins if* $b = b'$*.*

Theorem 5. *Let* F *be a client_id-unlinkable function. Then POIDC instantiated with* F *provides login unlinkability with respect to the IdP.*

Proof. We prove this by contradiction. Assume that an attacker can break login unlinkability for POIDC instantiated with a *client_id*-unlinkable F. That is, assume that there is an efficient attacker $\mathcal{A}$ that, with non-negligible advantage, can win the IdP-login-linking game.

Then, there exists an efficient attacker $\mathcal{A}'$ that makes use of $\mathcal{A}$ to win the *client_id*-linking game with non-negligible advantage, who works as follows. Note that $\mathcal{A}'$ acts as the challenger for $\mathcal{A}$ in the IdP-login-linking game.

1. When $\mathcal{A}'$ receives setupParams and u from its challenger, it uses these parameters to set up the same protocol environment for P to act as the challenger for $\mathcal{A}$. It then forwards setupParams and u to $\mathcal{A}$.

2. When $\mathcal{A}$ issues an oracle request of the form $\text{RunProtocolIdP}_P(\text{u}', \text{rp}')$, $\mathcal{A}'$ looks up the *client_id* cid' assigned to rp' based on setupParams, chooses a value $nonce'$ of length L_n as defined in setupParams and a cryptographically secure random value rnd', then sends $F(\text{cid}', nonce', \text{rnd}', \text{u}')$ to $\mathcal{A}$. The transcript from the IdP's point of view only contains a single value of this form, so this constitutes a valid transcript from a protocol run.

3. When $\mathcal{A}$ sends $(\text{rp}_0, \text{rp}_1)$, $\mathcal{A}'$ looks up their *client_id*s cid_0 and cid_1 from the provided setupParams, then outputs cid_0 and cid_1 and two values $nonce_0$ and $nonce_1$ of length L_n as defined in setupParams.

4. When $\mathcal{A}'$ receives its challenge $c = F(\text{cid}_b, nonce_b, \text{rnd}, \text{u})$, it passes $tr := c$ to $\mathcal{A}$. As explained previously, this value is a valid transcript from a protocol run.

5. $\mathcal{A}$ sends its bit b' to $\mathcal{A}'$, who outputs the same bit b'.

$\mathcal{A}'$ answers correctly for the *client_id*-linking game if and only if $\mathcal{A}$ answered correctly for the login-linking game. The input for $\mathcal{A}$, tr, is a valid transcript for a login to rp_b, with *client_id* cid_b. Thus, $\mathcal{A}'$ has the same advantage for the *client_id*-linking game as $\mathcal{A}$ has for the login-linking game. In particular, if the advantage of $\mathcal{A}$ is non-negligible, so is that of $\mathcal{A}'$. This contradicts our assumption that F is *client_id*-unlinkable, and thus POIDC instantiated with F provides login unlinkability with respect to the IdP. □

We next give the analogous definition for login exchangeability.

Definition 52. *A masking function* $F(client\_id, nonce, \text{rnd}, \text{u})$ *that belongs to a POIDC variant* P *is* *client_id*-exchangeable *if an attacker cannot distinguish, for client_ids* cid_0 *and* cid_1 *and values* $nonce_0$ *and* $nonce_1$ *chosen by the attacker, and given two users* u_0 *and* u_1*, whether a value is of the form* $F(\text{cid}_0, nonce_0, \text{rnd}_0, \text{u}_0)||F(\text{cid}_1, nonce_1, \text{rnd}_1, \text{u}_1)$ *or* $F(\text{cid}_1, nonce_1, \text{rnd}_0, \text{u}_0)||F(\text{cid}_0, nonce_0, \text{rnd}_1, \text{u}_1)$.

Formally, F *is client_id-exchangeable if* $\mathcal{A}$ *has only negligible advantage to win the* *client_id*-matching game *for function* F*, defined as follows.*

1. *The challenger sets up the environment that is required to run protocol* P *according to a set of parameters* setupParams. *In particular,* setupParams *contains a list of registered RPs and their associated client_ids. Note that we assume that all client_ids have the same length* L_c *defined in* setupParams. *The challenger also chooses* two honest users u_0 *and* u_1*, and sends* setupParams *and* (u_0, u_1) *to* $\mathcal{A}$.

2. $\mathcal{A}$ *is given oracle access to* F*, i.e., it can call* F *with any parameters.*

3. *$\mathcal{A}$ outputs a pair of client_ids* $(\mathrm{cid}_0, \mathrm{cid}_1)$ *and two values* $(nonce_0, nonce_1)$ *of length* L_n *as defined in* setupParams*.*

4. *The challenger picks* $b \leftarrow\!\!\$\, \{0,1\}$ *as well as* $\mathrm{rnd}_0 \leftarrow\!\!\$\, \mathcal{R}$ *and* $\mathrm{rnd}_1 \leftarrow\!\!\$\, \mathcal{R}$*, then computes* $c_0 := F(\mathrm{cid}_b, nonce_b, \mathrm{rnd}_0, \mathrm{u}_0)$ *and* $c_1 := F(\mathrm{cid}_{1-b}, nonce_{1-b}, \mathrm{rnd}_1, \mathrm{u}_1)$*. It sends* (c_0, c_1) *to* $\mathcal{A}$*.* $\mathcal{A}$ *continues to have oracle access to* F *at this stage.*

5. *$\mathcal{A}$ outputs a bit* b'*, and wins if* $b = b'$*.*

Theorem 6. *Let F be a client_id-exchangeable function. Then, POIDC instantiated with F provides login exchangeability with respect to the IdP.*

Proof. We prove this by contradiction. Assume that an attacker can break login exchangeability for POIDC instantiated with a *client_id*-exchangeable F. That is, there is an efficient attacker $\mathcal{A}$ that, with non-negligible advantage, can win the login-matching game.

Then, there exists an efficient attacker $\mathcal{A}'$ that can win the *client_id*-matching game with non-negligible advantage, who works as follows. Note that $\mathcal{A}'$ acts as the challenger for $\mathcal{A}$ in the IdP-login-matching game.

1. When $\mathcal{A}'$ receives setupParams and $(\mathrm{u}_0, \mathrm{u}_1)$ from its challenger, it uses these parameters to set up the same protocol environment for P to act as the challenger for $\mathcal{A}$. It then forwards setupParams and $(\mathrm{u}_0, \mathrm{u}_1)$ to $\mathcal{A}$.

2. When $\mathcal{A}$ issues an oracle request of the form $\mathrm{RunProtocolIdP}_P(\mathrm{u}', \mathrm{rp}')$, $\mathcal{A}'$ looks up the *client_id* cid' assigned to rp' based on setupParams, chooses a value $nonce'$ of length L_n as defined in setupParams and a cryptographically secure random value rnd', then sends $F(\mathrm{cid}', nonce', \mathrm{rnd}', \mathrm{u}')$ to $\mathcal{A}$. The transcript from the IdP's point of view only contains a single value of this form, so this constitutes a valid transcript from a protocol run.

3. When $\mathcal{A}$ sends $(\mathrm{rp}_0, \mathrm{rp}_1)$, $\mathcal{A}'$ looks up their *client_id*s cid_0 and cid_1 from the provided setupParams, then outputs cid_0 and cid_1 and any values $nonce_0$ and $nonce_1$ of length L_n as defined in setupParams..

4. When $\mathcal{A}'$ receives its challenge

$$(c_0, c_1) = (F(\mathrm{cid}_b, nonce_b, \mathrm{rnd}_0, \mathrm{u}_0), F(\mathrm{cid}_{1-b}, nonce_{1-b}, \mathrm{rnd}_1, \mathrm{u}_1)) \ ,$$

   it passes $(tr_0, tr_1) := (c_0, c_1)$ to $\mathcal{A}$. As explained previously, these values are valid transcripts from two protocol runs.

5. $\mathcal{A}$ sends its bit b' to $\mathcal{A}'$ who outputs the same bit b'.

$\mathcal{A}'$ answers correctly for the *client_id*-matching game if and only if $\mathcal{A}$ answered correctly for the login-matching game: The input tr_0 is a transcript for a login to rp_b, with *client_id* cid_b, and tr_1 is a transcript for a login to rp_{1-b}, with *client_id* cid_{1-b}. Thus, different values for $c_0 = tr_0$ and $c_1 = tr_1$ directly result in transcripts where the RPs to which u_0 and u_1 log in are swapped. This means that $\mathcal{A}'$ has the same advantage for the *client_id*-matching game as $\mathcal{A}$ has for the login-matching game. In particular, if the advantage of $\mathcal{A}$ is non-negligible, so is that of $\mathcal{A}'$. This contradicts our assumption that F is *client_id*-exchangeable, and thus POIDC instantiated with F provides login exchangeability. □

4.4.2 POIDC variants

We present four different variants of POIDC, three of which satisfy additional functional requirements. The first two variants provide login unlinkability and the second two provide login exchangeability.

4.4.3 Baseline privacy

Our first POIDC variant provides privacy without considering additional functional requirements. It is based on the following instantiation of F.

Definition 53. *Let H be a cryptographic hash function, and $F_H(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u}) := H(\mathit{client\_id}||\mathit{nonce}||\mathrm{rnd})$, where $||$ denotes concatenation. We call POIDC instantiated with this function POIDC-F_H.*

Theorem 7. *POIDC-F_H provides login unlinkability with respect to the IdP in the random oracle model.*

Proof. We first show that F_H is *client_id*-unlinkable in the random oracle model. We want to bound the advantage of an attacker in winning the *client_id*-linking game for F_H, i.e., distinguishing between $H(\mathrm{cid}_0||\mathit{nonce}_0||\mathrm{rnd})$ and $H(\mathrm{cid}_1||\mathit{nonce}_1||\mathrm{rnd})$, where $\mathrm{rnd} \leftarrow\!\!\$\, \mathcal{R}$ and rnd is not known by the attacker. Let D_H be the domain of H. Then, in the random oracle model, the distributions of $H(\mathrm{cid}_0||\mathit{nonce}_0||\mathrm{rnd})$ and $H(\mathrm{cid}_1||\mathit{nonce}_1||\mathrm{rnd})$ are both uniform at random in $D_H \times D_H$. That is, both values are drawn from the same distribution. Thus, the only way how an attacker could have any advantage in distinguishing them would be to guess a valid pre-image for one of the

hashes, which requires guessing rnd. However, rnd is a cryptographically secure random value; in particular, it is chosen from a sufficiently large domain that an efficient attacker only has negligible probability in guessing it.

Thus, in the random oracle model, F_H is *client_id*-unlinkable, and due to Theorem 5, POIDC-F_H is login unlinkable with respect to the IdP. □

A cryptographic hash function is the most straightforward instantiation of F, and POIDC-F_H requires little additional setup. Support for hash functions such as SHA-256 is widespread: numerous JavaScript implementations are available thereof. Thus, POIDC-F_H is a good candidate for adoption when the additional functionality provided by our other variants is not required.

4.4.4 Private user logs

We now consider the addition of logging.

Definition 54. *The* user-accessible logs *requirement states that a user can, at any time, request a log from her IdP. This log contains a* timestamp *of each login that she performed using that IdP, and to which RP that login was performed.*

Note that it is easy to fulfill this requirement in standard OIDC, as the IdP can simply keep a log containing the *client_id* for every login of the user. However, in POIDC-F_H, the hashed values are insufficient for this purpose since they would be meaningless to the users. Thus, we propose the following POIDC variant.

Additional setup. The user generates a key-pair $(\mathsf{pk}_\mathrm{u}, \mathsf{sk}_\mathrm{u})$ upon registration at the IdP, stores sk_u offline, and sends pk_u to the IdP.

Definition 55. *Let* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an IND-CPA secure public-key encryption scheme. Then, we define* $F_{\mathrm{Enc_u}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u}) := \mathsf{Enc}_{\mathsf{pk}_\mathrm{u}}(\mathit{client\_id}||\mathit{nonce}, \mathrm{rnd})$*, where* pk_u *is the public key of user* u*, and the function* $\mathsf{Enc}_{\mathsf{pk}_\mathrm{u}}$ *is randomized with* rnd*. We call POIDC instantiated with this function POIDC-*$F_{\mathrm{Enc_u}}$*.*

Theorem 8. *POIDC-*$F_{\mathrm{Enc_u}}$ *provides login unlinkability with respect to the IdP.*

Proof. We first show that $F_{\mathrm{Enc_u}}$ is *client_id*-unlinkable. We prove this by contradiction. Assume that there is an efficient attacker $\mathcal{A}$ that can win the *client_id*-linking game of $F_{\mathrm{Enc_u}}$with non-negligible advantage.

Then, there is an efficient attacker $\mathcal{A}'$ that can use $\mathcal{A}$ to win the IND-CPA game for the public-key encryption scheme $\Pi := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ used to instantiate $F_{\mathrm{Enc_u}}$ with

non-negligible advantage, who works as follows. Note that $\mathcal{A}'$ acts as the challenger for $\mathcal{A}$ in the *client_id*-linking game.

1. When $\mathcal{A}'$ receives pk from its challenger, it chooses a user id u and setupParams for $F_{\mathrm{Enc_u}}$ such that these parameters assign $\mathsf{pk}_\mathrm{u} := \mathsf{pk}$, that is, the public key for user u is chosen to be pk. It forwards setupParams and u to $\mathcal{A}$.

2. When $\mathcal{A}$ issues an oracle request of the form $F_{\mathrm{Enc_u}}(\mathrm{cid}', \mathit{nonce}', \mathrm{rnd}', \mathrm{u}')$, $\mathcal{A}'$ looks up the public key pk' assigned to user u' and sends $\mathsf{Enc}_{\mathsf{pk}'}(\mathrm{cid}'||\mathit{nonce}', \mathrm{rnd}')$ to $\mathcal{A}$. This is a correct function application of $F_{\mathrm{Enc_u}}$ according to its definition.

3. When $\mathcal{A}$ sends $(\mathrm{cid}_0, \mathrm{cid}_1)$ and $(\mathit{nonce}_0, \mathit{nonce}_1)$, $\mathcal{A}'$ sets $\mathrm{m}_0 := \mathrm{cid}_0||\mathit{nonce}_0$ and $\mathrm{m}_1 := \mathrm{cid}_1||\mathit{nonce}_1$, then outputs $(\mathrm{m}_0, \mathrm{m}_1)$.

4. $\mathcal{A}'$ receives $c = \mathsf{Enc}_{\mathsf{pk}}(\mathrm{cid}_b||\mathit{nonce}_b, \mathrm{rnd}) = F_{\mathrm{Enc_u}}(\mathrm{cid}_b, \mathit{nonce}_b, \mathrm{rnd}, \mathrm{u})$ and forwards this value to $\mathcal{A}$. Note that this equality holds because $\mathsf{pk} = \mathsf{pk}_\mathrm{u}$ according to the chosen setupParams.

5. $\mathcal{A}'$ then gives the same answer as $\mathcal{A}$.

$\mathcal{A}'$ answers correctly for the IND-CPA game if and only if $\mathcal{A}$ answered correctly for the *client_id*-linking game of $F_{\mathrm{Enc_u}}$. Thus, $\mathcal{A}'$ has the same advantage for the IND-CPA game as $\mathcal{A}$ has for the *client_id*-linking game. In particular, if the advantage of $\mathcal{A}$ is non-negligible, so is that of $\mathcal{A}'$. This contradicts our assumption that Enc provides indistinguishability under chosen plaintext attacks, and thus $F_{\mathrm{Enc_u}}$ is *client_id*-unlinkable.

Thus, due to Theorem 5, POIDC-$F_{\mathrm{Enc_u}}$ is login unlinkable with respect to the IdP. □

In each run of POIDC-$F_{\mathrm{Enc_u}}$, the IdP additionally stores a log entry of the form $(\mathrm{timestamp}, \mathsf{Enc}_{\mathsf{pk}_\mathrm{u}}(\mathit{client\_id}||\mathit{nonce}, \mathrm{rnd}))$. A user may, at any time, request to receive these log entries, and can decrypt them using her secret key sk_u.

The user agent and the RP both need access to the user's public key pk_u. Thus, the IdP provides pk_u to the user agent upon loading the JavaScript, and includes pk_u in the *private_id_token* as a new field called *user_pk*.

POIDC-$F_{\mathrm{Enc_u}}$ and POIDC-F_H both provide login unlinkability with respect to the IdP, while POIDC-$F_{\mathrm{Enc_u}}$ additionally implements user-accessible logs. The main trade-off is that the user is required to generate a key-pair and keep the secret key safe.

However, since the user's secret key is only required for decrypting logs, it can and should be kept offline. For example, the key could be printed as a QR code or stored on a USB key. Alternatively, a randomness seed for generating the key could be written down, such as a BIP-39 [69] mnemonic phrase. Furthermore, if the user loses access to her secret key, she loses access to the logs, but not to her account.

4.4.5 Private aggregation

We now consider a functional requirement relevant when IdPs wish to bill RPs for services rendered.

Definition 56. *An IdP has access to* aggregated login data *if, given a series of* aggregation time periods $[t_1, t_2), [t_2, t_3), \ldots, [t_i, t_{i+1}), \ldots, [t_{n-1}, t_n]$*, the IdP can learn how many logins to any RP have been made in total by its users during each time period.*

When an IdP provides valuable user data to its RPs, the IdP may require the RP to pay a fee based on the amount of data provided. For this, the IdP must know at least the aggregated number of logins to each RP during a billing time period. Note that this requirement is again trivially satisfied in standard OIDC, and the question is how to aggregate login data while preserving the privacy of individual logins. For this, we propose the following POIDC variant.

Additional setup. We introduce a new party, called an *aggregator*. An aggregator can serve multiple IdPs, but each IdP is served by exactly one aggregator. Aggregators are also honest-but-curious, and do not collude with the IdPs they serve. An aggregator serving an IdP may learn the aggregated login counts to RPs using that IdP, but no information about the individual logins. Each aggregator A has a key-pair $(\mathsf{sk}_\mathrm{A}, \mathsf{pk}_\mathrm{A})$, and publishes pk_A.

We usually consider settings focusing on a single IdP, and we denote the aggregator serving this IdP simply as *the aggregator*.

Definition 57. *Let* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be an IND-CPA secure public-key encryption scheme. Then, we define* $F_{\mathrm{Enc_A}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u}) := \mathsf{Enc}_{\mathsf{pk}_\mathrm{A}}(\mathit{client\_id}||\mathit{nonce}, \mathrm{rnd})$*, where* pk_A *is the aggregator's public key, and the function* $\mathsf{Enc}_{\mathsf{pk}_\mathrm{A}}$ *is randomized with* rnd*. We call POIDC instantiated with this function POIDC-*$F_{\mathrm{Enc_A}}$.

To show that POIDC-$F_{\mathrm{Enc_A}}$ provides login exchangeability with respect to the IdP, we must make our attacker's additional capabilities precise.

Definition 58. *An attacker $\mathcal{A}$ in the client_id-linking or client_id-matching games has access to an* aggregation oracle *if it can access the following information: A list of pairs (cid_i, n_i), one pair for each registered client_id cid_i, where n_i is the number of times that a value of the form $F(\mathrm{cid}_i, \_, \_, \_)$ was sent to the attacker during the game, where $(\_)$ represents arbitrary data. We call n_i the aggregate for cid_i.*

No instantiation of F can provide *client_id*-unlinkability when the attacker has access to an aggregation oracle. In particular, it can determine b given a value of the form $F(\mathrm{cid}_b, \_, \_, \_)$, since the output of the oracle is always $[(\mathrm{cid}_b, 1), (\mathrm{cid}_{1-b}, 0)]$.

However, $F_{\mathrm{Enc_A}}$ is *client_id-exchangeable*, even when the attacker has access to an aggregation oracle. We can thus prove the following theorem.

Theorem 9. *POIDC-$F_{\mathrm{Enc_A}}$ is login exchangeable with respect to the IdP, even when the attacker has access to an aggregation oracle.*

Proof. We first show that $F_{\mathrm{Enc_A}}$ is *client_id*-exchangeable, even when the attacker has access to an aggregation oracle.

We will bound the advantage that any efficient attacker $\mathcal{A}$ can have for winning the *client_id*-matching game for $F_{\mathrm{Enc_A}}$.

First, we note that the output of the aggregation oracle is independent of the bit b. For its challenge, $\mathcal{A}$ always receives one value of the form $F_{\mathrm{Enc_A}}(\mathrm{cid}_0, \_, \_, \_)$ and one value of the form $F_{\mathrm{Enc_A}}(\mathrm{cid}_1, \_, \_, \_)$. Thus, the challenge will always increase both the aggregate for cid_0 and the aggregate for cid_1 by one, and so the aggregates cannot help the attacker in determining b.

We call the *client_id*-matching game for $F_{\mathrm{Enc_A}}$ Game 0. We cannot directly perform a reduction from IND-CPA to this game, so we introduce a slight variation of the *client_id*-matching game for $F_{\mathrm{Enc_A}}$, called Game 1. In Game 1, the input c_1 is of the form $F_{\mathrm{Enc_A}}(0^{L_c}, 0^{L_n}, \mathrm{rnd}, \mathrm{u})$ rather than $F_{\mathrm{Enc_A}}(\mathrm{cid}_{1-b}, \mathit{nonce}_{1-b}, \mathrm{rnd}, \mathrm{u})$, where L_c is the length of *client_id*s defined in the setupParams for Game 0 and L_n is the defined length for *nonce*s.

Game 1 is computationally indistinguishable from Game 0, which can be seen as follows[1]. An efficient distinguisher $\mathcal{D}$ between the games could be used to distinguish between $F_{\mathrm{Enc_A}}(0^{L_c}, 0^{L_n}, \mathrm{rnd}, \mathrm{u}) = \mathsf{Enc}_{\mathsf{pk_A}}(0^{L_c+L_n}, \mathrm{rnd})$ and $F_{\mathrm{Enc_A}}(\mathrm{cid}_1, \mathit{nonce}_1, \mathrm{rnd}, \mathrm{u}) = \mathsf{Enc}_{\mathsf{pk_A}}(\mathrm{cid}_1 || \mathit{nonce}_1, \mathrm{rnd})$ for some $\mathrm{cid}_1 || \mathit{nonce}_1 \neq 0^{L_c+L_n}$. That is, $\mathcal{D}$ could be used to

[1] Note that we do not consider the aggregation oracle for Game 1. We have shown that it does not help at all in the original game, so we can now safely assume an attacker who does not have access to it.

win the IND-CPA game for the public-key encryption scheme $\Pi := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ used to instantiate $F_{\mathrm{Enc_A}}$.

Next, assume that an efficient attacker $\mathcal{A}$ can win Game 1 with non-negligible advantage. Then, there exists an efficient attacker $\mathcal{A}'$ that can win the IND-CPA game for Π with non-negligible advantage. $\mathcal{A}'$ works as follows. Note that $\mathcal{A}'$ acts as the challenger for $\mathcal{A}$ in the Game 1 variant of the *client_id*-matching game.

1. When $\mathcal{A}'$ receives pk from its challenger, it chooses a user id u and setupParams for $F_{\mathrm{Enc_A}}$ such that these parameters assign $\mathsf{pk_A} := \mathsf{pk}$, that is, the public key for the aggregator is chosen to be pk. Note that the choice of u is not relevant as $F_{\mathrm{Enc_A}}$ does not depend on u. It forwards setupParams and u to $\mathcal{A}$.

2. When $\mathcal{A}$ issues an oracle request of the form $F_{\mathrm{Enc_A}}(\mathrm{cid}', \mathit{nonce}', \mathrm{rnd}', \mathrm{u}')$, $\mathcal{A}'$ sends $\mathsf{Enc}_{\mathsf{pk_A}}(\mathrm{cid}'||\mathit{nonce}', \mathrm{rnd}')$ to $\mathcal{A}$. This is a correct function application of $F_{\mathrm{Enc_A}}$ according to its definition.

3. When $\mathcal{A}$ sends $(\mathrm{cid}_0, \mathrm{cid}_1)$ and $(\mathit{nonce}_0, \mathit{nonce}_1)$, $\mathcal{A}'$ sets $\mathrm{m}_0 := \mathrm{cid}_0||\mathit{nonce}_0$ and $\mathrm{m}_1 := \mathrm{cid}_1||\mathit{nonce}_1$, then outputs $(\mathrm{m}_0, \mathrm{m}_1)$.

4. When $\mathcal{A}'$ receives $c = \mathsf{Enc}_{\mathsf{pk}}(\mathrm{cid}_b||\mathit{nonce}_b, \mathrm{rnd}_0) = F_{\mathrm{Enc_A}}(\mathrm{cid}_b, \mathit{nonce}_b, \mathrm{rnd}_0, \mathrm{u}_0)$, it sets $c_0 := c$, picks rnd_1 randomly according to the definition of Enc, constructs $c_1 := \mathsf{Enc}_{\mathsf{pk}}(0^{L_c+L_n}, \mathrm{rnd}_1)$, and sends (c_0, c_1) to $\mathcal{A}$. Note that these are valid inputs for Game 1. In particular, $\mathsf{pk} = \mathsf{pk_A}$ according to the choice of setupParams.

5. $\mathcal{A}'$ then gives the same answer as $\mathcal{A}$.

$\mathcal{A}'$ answers correctly for the IND-CPA game if and only if $\mathcal{A}$ answered correctly for Game 1. Thus, $\mathcal{A}'$ has the same advantage for the IND-CPA game as $\mathcal{A}$ has for Game 1. In particular, if the advantage of $\mathcal{A}$ is non-negligible, so is that of $\mathcal{A}'$.

Since Game 1 is computationally indistinguishable from Game 0, the *client_id*-matching game for $F_{\mathrm{Enc_A}}$, no efficient attacker can win Game 0 with non-negligible advantage for an IND-CPA secure Π.

Thus, $F_{\mathrm{Enc_A}}$ is *client_id*-exchangeable and, due to Theorem 6, POIDC-$F_{\mathrm{Enc_A}}$ is login exchangeable with respect to the IdP. □

$F_{\mathrm{Enc_A}}$ is similar to $F_{\mathrm{Enc_u}}$, except that we use the *aggregator's* public key rather than the user's. Analogous to POIDC-$F_{\mathrm{Enc_u}}$, the IdP provides the aggregator's public key $\mathsf{pk_A}$ to the user, and also includes $\mathsf{pk_A}$ in each *private_id_token* in an additional field called *aggregator_pk*. We now explain how the IdP can obtain the number of logins to each RP during a time period.

Aggregation protocol. We define a series of time periods $[t_i, t_{i+1})$. The IdP stores every value of the form $\mathsf{Enc}_{\mathsf{pk}_\mathrm{A}}(\mathit{client\_id}||\mathit{nonce}, \mathrm{rnd})$ that it receives during a time period. After the end of a time period, the IdP and the aggregator perform the following steps. First, the IdP shuffles (puts into a uniformly random order) all $\mathsf{Enc}_{\mathsf{pk}_\mathrm{A}}(\mathit{client\_id}||\mathit{nonce}, \mathrm{rnd})$ values it received during the time period and sends them to the aggregator. Then, the aggregator decrypts the values and counts how often each *client_id* occurs. It sends the IdP a list with entries of the form $(\mathit{client\_id}, n)$, denoting n occurrences of *client_id*.

Privacy with respect to the aggregator. We also consider privacy with respect to an honest-but-curious *aggregator*. The aggregator receives a *shuffled* list of encrypted *client_id*s from the IdP, where each ordering of the entries is equally likely (uniform distribution). The information content of this shuffled list is equivalent to the list of aggregated login counts produced by the aggregator. In fact, one could generate a shuffled list with the same probability distribution from the list of aggregated login counts: For each (cid, n) in the list of aggregated login counts, denoting that there were n total logins to RP with *client_id* cid, generate n entries of the form $\mathsf{Enc}_{\mathsf{pk}_\mathrm{A}}(\mathrm{cid}||\mathit{nonce}, \mathrm{rnd})$ with random *nonce* values. Gather these entries in a list, then shuffle the resulting list. Thus, the shuffled list is just another representation of the same information (except for the *nonce* values, which do not contain additional information), and all the aggregator learns are the aggregated login counts.

POIDC-$F_{\mathrm{Enc_A}}$ gives the IdP access to aggregated login data while preserving the privacy of individual logins. The reason for the slightly weaker privacy property is inherent in the *requirement* of providing the IdP with aggregated login data, and not due to our solution. In practice, the aggregation time periods should be chosen such that there are many logins by different users to each RP during each time period. The problem of choosing appropriate time periods is orthogonal to the design of POIDC-$F_{\mathrm{Enc_A}}$ itself, since any other solution that provides aggregated login data would also need to solve this problem.

The main trade-off of POIDC-$F_{\mathrm{Enc_A}}$ with respect to POIDC-$F_{\mathrm{Enc_u}}$ or POIDC-F_H is the introduction of the aggregator as an additional party. Thus, we do not propose POIDC-$F_{\mathrm{Enc_A}}$ as the POIDC variant that should primarily be used in practice. Rather, we use it to illustrate that, even with a requirement that seems to directly contradict

our privacy goals, much can still be achieved. While introducing aggregators does come at the cost of additional complexity and trust assumptions, note that the aggregators do not have to be online during POIDC-F_{Enc_A}'s execution. The aggregation protocol is run only once at the end of each aggregation time period. This may make it acceptable to introduce this additional party.

There are several options for how the aggregator could be instantiated in practice: It could be run as part of the OpenID Foundation's infrastructure, or as a commercial service by some provider. The essential requirement for privacy is that it must not collude with the IdP. In particular, it must not share its secret key, and it must only perform decryption operations as specified in the aggregation protocol. Ideally, the aggregation service would be run by a third party that is not affiliated with any IdP, and whose business model depends on providing the service in the specified way.

Trust assumptions between the IdP and the aggregator must then also be considered, since the IdP might not want a third party to learn the aggregated number of logins by the IdP's users to RPs. An IdP that is unwilling to use an aggregator service provided by a third party could potentially run the aggregator itself, isolating the aggregator from the rest of its infrastructure except for the aggregation protocol interface. Furthermore, no employee should have access to both the aggregator and the encrypted *client_id* entries. While this may still require the users to trust the IdP, external audits could certify that a proper separation of the systems has been put into place.

An open question is how the aggregated data can be hidden from the aggregator itself, so that the protocol could be used by IdPs who do not wish to share their aggregated login data with a third party. For this, a private aggregation protocol, such as that by Leontiadis et al. [70], is an option. However, the protocol they present cannot be used directly for our purposes, since it assumes that all users only submit correct data, and a single incorrect data point causes aggregation to fail completely. Instead, a protocol that is resilient against malicious and erroneous user input is required for our setting.

4.4.6 Combination of user logs and aggregation

We now present our last POIDC variant, which combines POIDC-F_{Enc_u} and POIDC-F_{Enc_A}, covering both the requirement of user-accessible logs and aggregated login data. We define F_{cmb} as the combination of F_{Enc_u} and F_{Enc_A} as follows.

Definition 59.

$$F_{\mathrm{cmb}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd_u}||\mathrm{rnd_A}, \mathrm{u}) :=$$
$$F_{\mathrm{Enc_u}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd_u}, \mathrm{u})||F_{\mathrm{Enc_A}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd_A}, \mathrm{u}) \ ,$$

where the randomness string $\mathrm{rnd} = \mathrm{rnd_u}||\mathrm{rnd_A}$ *is twice the length of that for* $F_{\mathrm{Enc_u}}$ *or* $F_{\mathrm{Enc_A}}$ *individually. The first half is used as input for* $F_{\mathrm{Enc_u}}$ *and the second half is used as input for* $F_{\mathrm{Enc_A}}$*. We call POIDC instantiated with this function POIDC-*F_{cmb}*.*

Theorem 10. *POIDC-*F_{cmb} *is login exchangeable with respect to the IdP, even when the attacker has access to an aggregation oracle.*

Proof. We first show that F_{cmb} is *client_id*-exchangeable, even when the attacker has access to an aggregation oracle.

We will bound the advantage that any efficient attacker $\mathcal{A}$ can have for winning the *client_id*-matching game for F_{cmb}.

First, we again note that the output of the aggregation oracle is independent of the bit b. For its challenge, $\mathcal{A}$ always receives one value of the form $F_{\mathrm{cmb}}(\mathrm{cid}_0, \_, \_, \_)$ and one value of the form $F_{\mathrm{cmb}}(\mathrm{cid}_1, \_, \_, \_)$. Thus, the challenge will always increase both the aggregate for cid_0 and the aggregate for cid_1 by one, and so the aggregates cannot help the attacker in determining b.

We now consider two games: Game 0 is the *client_id*-matching game for F_{cmb}. Game 1 is the *client_id*-matching game for a variant of F_{cmb} that we call F'_{cmb} and define as follows:

$$F'_{\mathrm{cmb}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd_u}||\mathrm{rnd_A}, \mathrm{u}) :=$$
$$F_{\mathrm{Enc_u}}(0^{L_c}, 0^{L_n}, \mathrm{rnd_u}, \mathrm{u})||F_{\mathrm{Enc_A}}(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd_A}, \mathrm{u}) \ ,$$

i.e., we replace the encryption of $\mathit{client\_id}||\mathit{nonce}$ in $F_{\mathrm{Enc_u}}$ by an encryption of $0^{L_c+L_n}$.

Game 1 is computationally indistinguishable from Game 0, which can be seen as follows. An efficient distinguisher $\mathcal{D}$ between the games could be used to distinguishing between $F_{\mathrm{Enc_u}}(0^{L_c}, 0^{L_n}, \mathrm{rnd_u}, \mathrm{u}) = \mathsf{Enc}_{\mathsf{pk_u}}(0^{L_c+L_n}, \mathrm{rnd_u})$ and $F_{\mathrm{Enc_u}}(\mathrm{cid}, \mathit{nonce}, \mathrm{rnd_u}, \mathrm{u}) = \mathsf{Enc}_{\mathsf{pk_u}}(\mathrm{cid}||\mathit{nonce}, \mathrm{rnd_u})$ for some $\mathrm{cid}||\mathit{nonce} \neq 0^{L_c+L_n}$. That is, $\mathcal{D}$ could be used to win the IND-CPA game for the public-key encryption scheme $\Pi := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ used to instantiate F_{cmb}.

Next, assume that there exists an efficient attacker $\mathcal{A}$ that can win Game 1 with non-negligible advantage. Then, there exists an efficient attacker $\mathcal{A}'$ that can win the *client_id*-matching game for $F_{\mathrm{Enc_A}}$(instantiated with Π) with non-negligible advantage,

contradicting Theorem 9. $\mathcal{A}'$ works as follows. Note that $\mathcal{A}'$ acts as the challenger for $\mathcal{A}$ in the Game 1 variant of the *client_id*-matching game for F_{cmb}.

1. When $\mathcal{A}'$ receives setupParams for $F_{\text{Enc}_\text{A}}$and (u_0, u_1) from its challenger, it extends setupParams with the additional parameters required to initialize F_{cmb}, in particular assigning public keys to users. In particular, $\mathsf{pk}_{\text{u}0}$ is assigned to u_0 and $\mathsf{pk}_{\text{u}1}$ is assigned to u_1. Let setupParams$'$ be the set of extended parameters. $\mathcal{A}'$ then sends setupParams$'$ and u to $\mathcal{A}$.

2. When $\mathcal{A}$ issues an oracle request of the form $F'_{\text{cmb}}(\text{cid}', \mathit{nonce}', \text{rnd}_\text{u}'||\text{rnd}_\text{A}', \text{u}')$, $\mathcal{A}'$ looks up the public key pk' for u$'$ and sends

$$\mathsf{Enc}_{\mathsf{pk}'}(0^{L_c+L_n}, \text{rnd}_\text{A}')||\mathsf{Enc}_{\mathsf{pk}_\text{A}}(\text{cid}'||\mathit{nonce}', \text{rnd}_\text{u}')$$

   to $\mathcal{A}$. This is a correct function application of F'_{cmb} as defined for Game 1.

3. When $\mathcal{A}$ sends $(\text{cid}_0, \text{cid}_1)$ and $(\mathit{nonce}_0, \mathit{nonce}_1)$, $\mathcal{A}'$ outputs these values directly.

4. When $\mathcal{A}'$ receives $c_0 = F_{\text{Enc}_\text{A}}(\text{cid}_b, \mathit{nonce}_b, \text{rnd}_{\text{A}0}, \text{u}_0)$ and $c_1 = F_{\text{Enc}_\text{A}}(\text{cid}_{1-b}, \mathit{nonce}_{1-b}, \text{rnd}_{\text{A}1}, \text{u}_1)$, it picks $\text{rnd}_{\text{u}0} \leftarrow\!\!\$\, \mathcal{R}$ and $\text{rnd}_{\text{u}1} \leftarrow\!\!\$\, \mathcal{R}$. It constructs $c'_0 := \mathsf{Enc}_{\mathsf{pk}_{\text{u}0}}(0^{L_c+L_n}, \text{rnd}_{\text{u}0})||c_0$ and $c'_1 := \mathsf{Enc}_{\mathsf{pk}_{\text{u}1}}(0^{L_c+L_n}, \text{rnd}_{\text{u}1})||c_1$, and sends (c'_0, c'_1) to $\mathcal{A}$. Note that these are valid inputs for Game 1 according to the definition of F'_{cmb}.

5. $\mathcal{A}'$ then gives the same answer as $\mathcal{A}$.

$\mathcal{A}'$ answers correctly for the F_{Enc_A} *client_id*-matching game if and only if $\mathcal{A}$ answered correctly for Game 1. Thus, $\mathcal{A}'$ has the same advantage for the F_{Enc_A} *client_id*-matching game as $\mathcal{A}$ has for Game 1. In particular, if the advantage of $\mathcal{A}$ is non-negligible, so is that of $\mathcal{A}'$.

Since Game 1 is computationally indistinguishable from Game 0, the *client_id*-matching game for F_{Enc_A}, no efficient attacker can win Game 0 with non-negligible advantage due to Theorem 9.

Thus, F_{cmb} is *client_id*-exchangeable and, due to Theorem 6, POIDC-F_{cmb} is login exchangeable with respect to the IdP. □

The setup for this variant combines the setup of the user log and aggregation variants. The user log can be recovered from the first half of F_{cmb} with the user's secret key. The aggregated login data can be extracted from the second half of F_{cmb} by running the aggregation protocol.

In case only one of the two functional requirements is desired, POIDC-$F_{\mathrm{Enc_u}}$ or POIDC-$F_{\mathrm{Enc_A}}$ should be used directly, since less setup is necessary. For POIDC-$F_{\mathrm{Enc_u}}$, we also have the stronger login unlinkability property rather than login exchangeability. However, POIDC-F_{cmb} allows IdPs that require both features to still preserve their users' privacy, and illustrates the flexibility of our definitions.

4.5 Pairwise POIDC: Protecting also against colluding RPs

All presented variants of POIDC so far require public subject identifiers. Namely, there is a single subject identifier for each user at the IdP, and this identifier is included in each *id_token* for the user, regardless of the intended audience RP. Thus, it does not provide privacy with respect to colluding RPs, who can link accounts to the same user.

Standard OIDC provides a solution to this problem called pairwise identifiers, where a different subject identifier is assigned to the user for each RP. The IdP can simply look up the correct pairwise identifier for a user at an RP when the user wishes to log in to that RP. When the IdP should no longer learn to which RP the user logs in, designing pairwise identifiers becomes more challenging. In particular, this cannot be done solely by instantiating the POIDC protocol we have presented differently; the masking function F is unrelated to *subject* identifiers.

We outline next the requirements for implementing pairwise identifiers in POIDC. Our goal is to provide the following properties:

1. Login-unlinkability with respect to an honest-but-curious IdP;
2. login-unlinkability with respect to colluding RPs; and
3. security with respect to a Dolev-Yao network attacker.

Each requirement follows from one of these goals, which we list next to the requirement.

User Agent Computable Identifiers *(Login-unlinkability with respect to the IdP)*
The *user agent* must be able to compute the pairwise identifier for the user at a specific RP, since the IdP does not know to which RP the user wishes to log in.

No Identifier Registration *(Login-unlinkability with respect to the IdP)*
There must be no difference between the first login (registration) of a user at an RP and to an RP where the user has logged in before. Thus, the IdP cannot generate new pairwise identifiers or even keep any sort of pairwise identifier list for the user.

Pairwise Identifier Unlinkability *(Login-unlinkability with respect to colluding RPs)*
Comparing pairwise identifiers must not provide the RPs with any information that enables linking them to the same user. This is the main purpose of pairwise identifiers, and is easy to solve in isolation, but is challenging when combined with the other requirements.

IdP-hidden Identifiers *(Login-unlinkability with respect to the IdP)*
While the pairwise identifier for a specific user at an RP must be persistent for functionality reasons, the IdP must not be able to link protocol runs with the same pairwise identifier.

User-bound Identifiers *(Security with respect to a Dolev-Yao attacker)*
An IdP must only sign an *id_token* that contains a subject identifier that belongs to the user who is currently logged in at the IdP. While this requirement is again simple in isolation, it rules out many potential solutions for the other requirements.

We first explain our design for *User Agent Computable Identifiers* that have *No Identifier Registration*. First, we introduce a global user identifier *user_id*, which is unique to each user at the IdP, and is not shared with RPs. Then, we define the following mapping from a *user_id* to pairwise identifiers for RPs. The pairwise identifier for an RP with *client_id* cid is given as $pairwise\_sub := H(user\_id||\text{cid})$, where H is a cryptographic hash function. Thus, the user agent can compute the pairwise identifier, and it is not necessary to keep a list of identifiers for each user.

We ensure *Pairwise Identifier Unlinkability* by requiring that the *user_id* is an unpredictable random value from a large domain so that an RP with *client_id* cid cannot feasibly obtain a user's *user_id* from a pairwise identifier $H(user\_id||\text{cid})$ by guessing partial pre-images. That is, the RP should not be able to efficiently compute

an x such that $H(x||\text{cid})$ matches the pairwise identifier. In particular, the *user_id* must not be a username chosen by the user.

For *IdP-hidden Identifiers*, the user agent must mask the identifier similarly to how it masks the audience field. For this, it generates a cryptographically secure, unpredictable, random value $\overline{\text{rnd}}$ and computes $H(\textit{pairwise\_sub}||\overline{\text{rnd}}) = H(H(\textit{user\_id}||\text{cid})||\overline{\text{rnd}})$. This masked value is then sent to the IdP to be used as a masked subject identifier in a run of pairwise POIDC.

Finally, to ensure *User-bound Identifiers*, the IdP must be certain that the masked subject identifier provided by the user agent has been computed from the currently authenticated user's *user_id*, i.e., that it is of the form $H(H(\textit{user\_id}||x)||y)$ for some x and y. To solve this, the user agent proves this statement in zero-knowledge, e.g., using ZKBoo [71]. ZKBoo provides an efficient way to prove this kind of statement. In particular, ZKBoo is well suited for proving statements of the form $y = H(x)$ and has been evaluated for SHA-256 to require 55ms of time and 836 KB of space [71]. While our statement uses a nested hash and would thus result in a larger circuit, the scale-up is linear in the circuit size. Thus, ZKBoo supports a practical implementation of pairwise POIDC. ZKBoo's reference implementation uses 136 rounds. To avoid a large latency overhead caused by performing an interactive zero-knowledge proof with many rounds, the user agent instead computes a non-interactive zero-knowledge proof (NIZKP) using the Fiat-Shamir heuristic [72] and sends this NIZKP to the IdP.

4.5.1 The general pairwise POIDC protocol

We next give the protocol description for pairwise POIDC, as illustrated in Figure 4.3. The protocol description uses a generic masking function F to compute *masked_aud*, just as regular POIDC, but uses a fixed hashing function for computing *masked_sub*. This is sufficient to obtain our additional privacy property with respect to colluding RPs, and we do not require the flexibility here that a generic function would give us.

In general, pairwise POIDC variants can be obtained by instantiating F just as for POIDC. That is, each protocol variant described in Section 4.4.2 can be defined similarly with pairwise POIDC. However, not all of these variants actually provide the desired privacy property with respect to colluding RPs. The problem is that $F(\textit{client\_id}, \textit{nonce}, \text{rnd}, \text{u})$ may depend on the user u, which is the case for POIDC-F_{Enc_u} and POIDC-F_{cmb}. These variants also include the user's public key, pk_u, in the

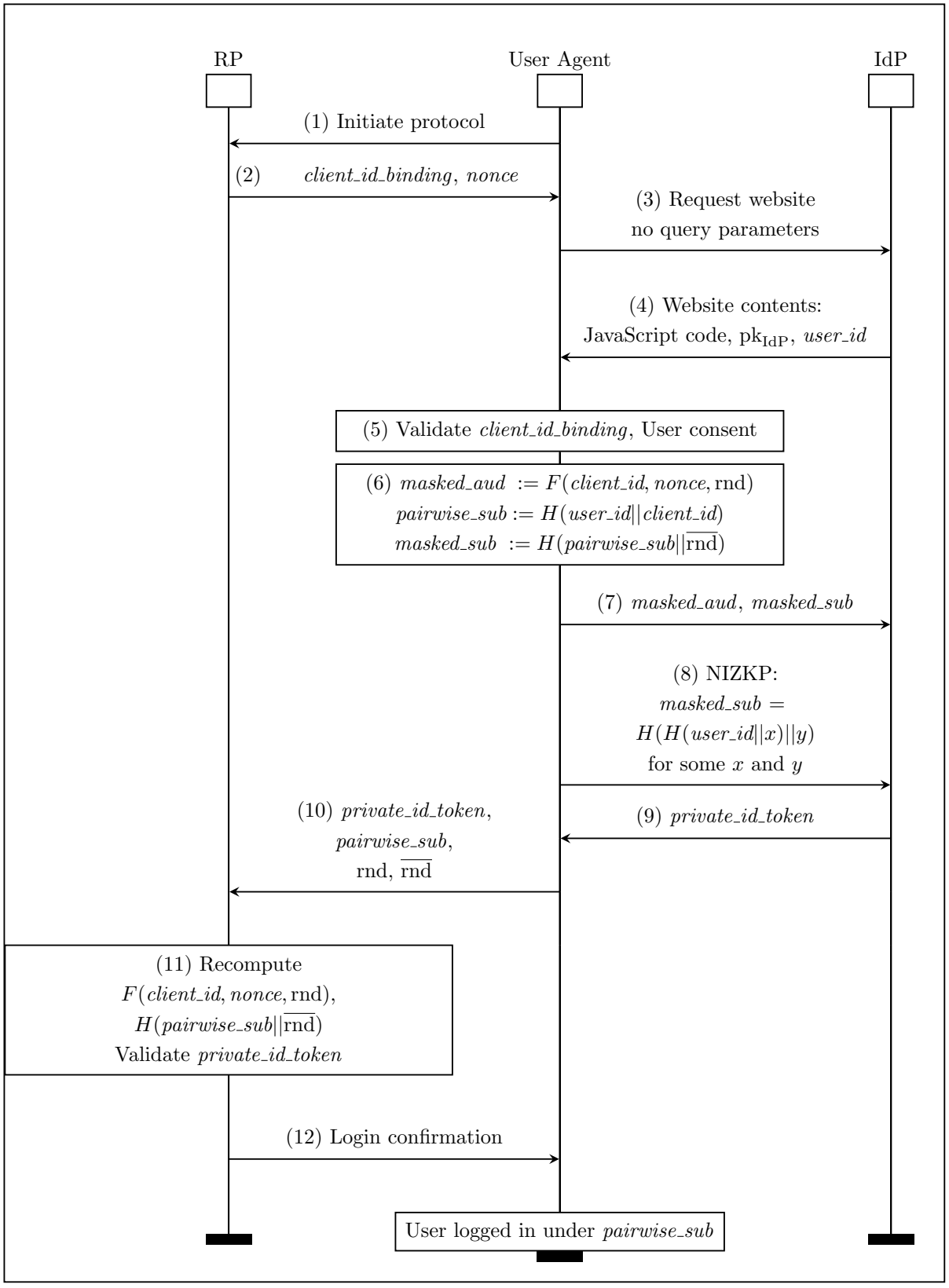

Figure 4.3: Privacy-preserving OpenID Connect with pairwise subject identifiers.

private_id_token. For these reasons, the POIDC variants implementing user-accessible logs do not have associated pairwise POIDC variants.

The variants of pairwise POIDC for which we will show the desired privacy property are pairwise POIDC-F_H and pairwise POIDC-$F_{\mathrm{Enc_A}}$. The masking functions used for these two variants do not depend on the user, which is an assumption that we will need for the privacy proof of pairwise POIDC. For this reason, we write $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd})$ rather than $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd}, \mathrm{u})$ in the definition for pairwise POIDC. We will also make the assumption that F must not depend on u explicit when proving privacy with respect to colluding RPs.

1. The user u initiates the protocol by requesting to log in to the RP. She specifies which IdP she wants to use, e.g., by clicking on the corresponding button on the RP's web page. This triggers a request sent from the user agent to the RP.

2. The RP redirects the user agent to the IdP, sending the *client_id_binding* (containing its *client_id*) and its *nonce* to the user agent as URI fragments. These URI fragments are accessible to JavaScript, but are *not* sent to the IdP back-end. This redirect must *not* contain an HTTP referer header.

3. The user agent is redirected to the IdP to load its website, but this request does not contain any query parameters.

4. The IdP sends its website to the user agent. The website contains JavaScript code that allows the user agent to perform the actions in Steps 5–10, as well as the IdP's public verification key $\mathrm{pk_{IdP}}$ and the user's unique identifier *user_id*.

5. The user agent executes the JavaScript code loaded from the IdP's side to verify the IdP's signature using $\mathrm{pk_{IdP}}$ on the *client_id_binding*. If this verification succeeds, the user agent then opens a dialogue for the user to confirm that she wishes to log in to the RP with the *client_name* in the *client_id_binding*.

6. The user agent generates *two* cryptographically secure random values, rnd and $\overline{\mathrm{rnd}}$, then computes the following values:

   - The masked audience identifier for the RP,
     $\mathit{masked\_aud} := F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd})$.

- The pairwise subject identifier for the RP to which the user wishes to log in, $pairwise\_sub := H(user\_id||client\_id)$.
- The masked version thereof, $masked\_sub := H(pairwise\_sub||\overline{\text{rnd}})$.

7. The user agent sends *masked_aud* and *masked_sub* to the IdP back-end, using, for example, an XMLHttpRequest.

8. The user agent provides a non-interactive zero-knowledge proof (NIZKP) to the IdP back-end showing that *masked_sub* is a value of the form $H(H(user\_id||x)||y)$ for some x and y, where *user_id* is the user's unique identifier. The NIZKP is constructed using the Fiat-Shamir heuristic [72].

9. If the IdP back-end accepts the zero-knowledge proof, it then returns a *private_id_token* to the user agent, which contains $F(client\_id, nonce, \text{rnd})$ in its *private_aud* field and *masked_sub* in its subject field.

10. The user agent verifies that the received *private_id_token* contains the expected values in the *private_aud* and subject fields. If this verification succeeds, the user agent sends this token, as well as *pairwise_sub*, rnd, and $\overline{\text{rnd}}$ to the *redirect_uri* given in the *client_id_binding*, using, for example, *window.open(redirect_uri)*.

11. The RP validates the IdP's signature on the *private_id_token*. It then recomputes $F(client\_id, nonce, \text{rnd})$ from its own *client_id*, the *nonce* that it generated earlier, and the rnd that it received from the user agent. It also recomputes the *masked_sub* from *pairwise_sub* and $\overline{\text{rnd}}$ that the user provided. It accepts the *private_id_token* and considers the user logged in to the account under the identifier *pairwise_sub* only if the *id_token*'s *private_aud* and subject field match the expected, recomputed, values (and signature validation succeeds).

12. The RP notifies the user agent whether the login was successful.

4.5.1.1 Transitioning to pairwise POIDC

To transition to pairwise POIDC from standard OIDC using pairwise subject identifiers, it is necessary to switch to the new kind of subject identifiers of the form $H(user\_id||client\_id)$.

If standard OIDC pairwise subject identifiers were used before, then these identifiers can be converted to the subject identifiers introduced by pairwise POIDC as follows. The user requests an identifier switch for a specific RP, and the IdP creates a signed token called a *transition_token*. A *transition_token* contains the user's old pairwise identifier for that RP and the new one of the form $H(user\_id||client\_id)$. Upon the next login to the RP, the user presents a *transition_token* in addition to an *id_token*, and the RP will transfer the user's account to the new subject identifier.

Note that the login where the switch happens can still be observed by the IdP. Login unlinkability holds with respect to the set of logins to all RPs for which the switch has already been made and thus pairwise POIDC login is used.

4.5.2 Privacy proofs

We show that pairwise POIDC instantiated with a *client_id*-unlinkable F provides login-unlinkability with respect to the IdP, that is, the additional messages sent to the IdP in pairwise POIDC preserve the privacy property established by regular POIDC.

Theorem 11. *If F is client_id-unlinkable, then POIDC instantiated with F is login-unlinkable with respect to an honest-but-curious IdP in the random oracle model, even if that IdP knows the value of any nonce or pairwise_sub of the form $H(user\_id||client\_id)$ that were used during protocol runs.*

Proof. The proof of this theorem is a reduction similar to that of Theorem 5, but must consider the additional messages sent to the IdP during a protocol run of pairwise POIDC as compared to regular POIDC. A valid transcript of messages received by the IdP in pairwise POIDC contains a *masked_sub* value and the non-interactive zero-knowledge proof (NIZKP) in addition to the *masked_aud*.

The messages that the IdP receives during a protocol run contain one value of the form $F(client\_id, nonce, \mathrm{rnd}, \mathrm{u})$, one value of the form $H(H(user\_id||client\_id)||\overline{\mathrm{rnd}})$, and the NIZKP. Note that the fact that F must not depend on the user is irrelevant for privacy with respect to the IdP, so we use the more generic form of F in this proof to be compatible with our existing definitions.

We cannot perform the reduction directly as in Theorem 5 since the adversary in the *client_id*-linking game only receives a *masked_aud* value but no corresponding *masked_sub* value. Since it does not know the *client_id*, it might not generate a valid *masked_sub* value. It would then also not be able to generate a valid NIZKP.

We therefore introduce two variations of the IdP-login-linking game for pairwise POIDC with associated protocol variations. Note that these protocols variations are used for the purpose of this proof only, and would not be functional or secure in practice.

- In Game 1, the IdP does not receive the NIZKP, but receives valid *masked_aud* and *masked_sub* values.
- In Game 2, the IdP does not receive the NIZKP, and receives a random value R from the hash function H's domain rather than a valid *masked_sub* value. It still receives a valid *masked_aud* value.

We first show that Game 1 and Game 2 are computationally indistinguishable. In the random oracle model, both *masked_sub* and R are random values chosen from the same distribution. The only way how an attacker could gain an advantage in distinguishing them is by guessing the pre-image for *masked_sub*. An efficient attacker has only negligible probability of doing so, and thus an efficient attacker has only negligible probability in distinguishing Game 1 from Game 2.

We now show by contradiction that, if F is *client_id*-unlinkable, the protocol variation defined by Game 2 provides login unlinkability. Afterwards, we will show that this implies login unlinkability also for the original pairwise POIDC protocol with a *client_id*-unlinkable F.

Assume that there is an efficient attacker $\mathcal{A}$ that, with non-negligible advantage, can win Game 2. Then, there exists an attacker $\mathcal{A}'$ that makes use of $\mathcal{A}$ to win the *client_id*-linking game with non-negligible advantage, who works as follows. Note that $\mathcal{A}'$ acts as the challenger for $\mathcal{A}$ in the IdP-login-linking game.

1. When $\mathcal{A}'$ receives setupParams and u from its challenger, it uses these parameters to set up the same protocol environment for P to act as the challenger for $\mathcal{A}$. It then forwards setupParams and u to $\mathcal{A}$. Note that setupParams contains the *user_id* that is assigned to u.

2. When $\mathcal{A}$ issues an oracle request of the form $\text{RunProtocolIdP}_P(\text{u}', \text{rp}')$, $\mathcal{A}'$ looks up the *client_id* cid' assigned to rp' based on setupParams, chooses a value $nonce'$ of length L_n as defined in setupParams and a cryptographically secure random value rnd', then computes the following *masked_aud* value: $F(\text{cid}', nonce', \text{rnd}', \text{u}')$. However, in pairwise POIDC, this value alone does *not* yet constitute a valid transcript from a protocol run for the IdP. Therefore, $\mathcal{A}'$ additionally generates another cryptographically secure random value $\overline{\text{rnd}}'$, and then picks a random

value R' from H's domain, according to the definition of Game 2. It then sends *masked_aud* and R to the IdP, which corresponds to the transcript of the messages the IdP receives in Game 2.

3. When $\mathcal{A}$ sends $(\mathrm{rp}_0, \mathrm{rp}_1)$, $\mathcal{A}'$ looks up their *client_id*s cid_0 and cid_1 from the provided setupParams, then outputs cid_0 and cid_1 and two values $nonce_0$ and $nonce_1$ of length L_n as defined in setupParams.

4. When $\mathcal{A}'$ receives its challenge $c = F(\mathrm{cid}_b, nonce_b, \mathrm{rnd}, \mathrm{u})$, it expands this to a valid transcript for Game 2 by picking a random value R from H's domain. It then sends c and R to $\mathcal{A}$.

5. $\mathcal{A}$ sends its bit b' to $\mathcal{A}'$ who outputs the same bit b'.

$\mathcal{A}'$ answers correctly for the *client_id*-linking game if and only if $\mathcal{A}$ answered correctly for Game 2. Thus, $\mathcal{A}'$ has the same advantage for the *client_id*-linking game as $\mathcal{A}$ for Game 2. In particular, if the advantage of $\mathcal{A}$ is non-negligible, so is that of $\mathcal{A}'$.

This contradicts our assumption that F is *client_id*-unlinkable, and thus the protocol variant used in Game 2 provides login unlinkability with respect to the IdP. Since Game 2 is computationally indistinguishable from Game 1, this also shows that Game 1 instantiated with a *client_id*-unlinkable F provides login unlinkability with respect to the IdP.

Game 1 differs from the original IdP-login-linking game only by omitting the NIZKP. However, these two games are not indistinguishable. Therefore, we must argue differently to show login unlinkability of the original game. First, note that in the random oracle model, the Fiat-Shamir heuristic [72] preserves the zero-knowledge property of the interactive zero-knowledge proof, since the challenges computed in the NIZKP are assumed to be truly random. We now prove login unlinkability of the original IdP-login-linking game by contradiction.

Assume that an adversary could win the original game with non-negligible probability. Due to the zero-knowledge property of the NIZKP, the adversary could generate a valid NIZKP itself by running a simulator. Therefore, the adversary for the original game could be used to construct an adversary to win Game 1 with non-negligible probability as follows: The first time the adversary for the original game would access any information about the NIZKP, the adversary for Game 1 first uses the simulator to generate a valid NIZKP itself, and then accesses the information of the simulated proof instead. Therefore, it can win Game 1 with the same probability as the original game even though it is not provided with the NIZKP. □

We next also show that pairwise POIDC instantiated with an F that does not depend on the user provides login unlinkability with respect to colluding RPs. Note that we assume that *id_token*s contain no optional fields to ensure that it contains no user-identifying information in addition to the subject identifiers.

Theorem 12. *Let the private_id_token contain only the masked pairwise subject identifier* sub*, the issuer identifier* iss*, the private_aud, and fields containing user-independent date or time values. Furthermore, let S be a set of colluding RPs that do not know the user_id of any honest user. Then, pairwise POIDC instantiated with an F that does not depend on the user is login-unlinkable with respect to these colluding RPs in the random oracle model.*

Proof. We want to bound the advantage of an attacker $\mathcal{A}$ in winning the RP-login-linking game for pairwise POIDC. In this game, $\mathcal{A}$ obtains a value $\mathrm{RunProtocolRP}_P(\mathrm{u}_0, \mathrm{rp}_0)$ and then either a value $\mathrm{RunProtocolRP}_P(\mathrm{u}_0, \mathrm{rp}_1)$ or a value $\mathrm{RunProtocolRP}_P(\mathrm{u}_1, \mathrm{rp}_1)$, and its goal is to distinguish these two scenarios.

These values each contain a *private_id_token*, a pairwise subject identifier *pairwise_sub*, and a random value $\overline{\mathrm{rnd}}$. The fields in the *private_id_token* for pairwise POIDC are the masked pairwise subject identifier *sub*, the issuer identifier *iss* and the *private_aud*, and potentially user-independent date or time values. These values, which by definition do not depend on the user, cannot help $\mathcal{A}$ in distinguishing the scenarios. We next consider the other values consecutively.

The issuer identifier *iss* is the same for each *private_id_token* issued by the IdP, which we assume to be the same across these protocol runs, so this is also user-independent and thus cannot help $\mathcal{A}$ in distinguishing these scenarios, either. The audience identifier *masked_aud* is of the form $F(\mathit{client\_id}, \mathit{nonce}, \mathrm{rnd})$. Here, it is critical that F does not depend on the user, as we have assumed for this proof. When this assumption is met, then this value will also be the same in both scenarios.

The last field to consider is *sub*, in conjunction with the values sent outside of the *private_id_token*. Note that *sub* is of the form $H(\mathit{pairwise\_sub}||\overline{\mathrm{rnd}})$ and can be recomputed by the RP from *pairwise_sub* and $\overline{\mathrm{rnd}}$. Thus, it depends fully on *pairwise_sub* and $\overline{\mathrm{rnd}}$, and it is sufficient to consider these two values. Since $\overline{\mathrm{rnd}}$ is a random value, it also does not depend on the user and cannot help $\mathcal{A}$ in distinguishing the scenarios. This leaves *pairwise_sub*, which is of the form $H(\mathit{user\_id}||\mathit{client\_id})$. Let $\mathit{user\_id}_0$ belong to u_0 and $\mathit{user\_id}_1$ to u_1, and let cid_1 be the *client_id* of rp_1. Furthermore, let D_H be the domain of H. Then, in the random oracle model, the distributions of the values that are different between the two scenarios, $H(\mathit{user\_id}_0||\mathrm{cid}_1)$ and $H(\mathit{user\_id}_1||\mathrm{cid}_1)$,

are both uniform at random in $D_H \times D_H$. That is, both values are drawn from the same distribution. Thus, the only way how an attacker could have any advantage in distinguishing them would be to guess a valid pre-image for one of the hashes, which requires guessing $user\_id_0$ or $user\_id_1$. However, these values are chosen randomly from a sufficiently large domain, such that an efficient attacker only has negligible probability in guessing them. □

Theorem 13. *Let the private_id_token contain only the masked pairwise subject identifier* sub*, the issuer identifier* iss*, the private_aud, and fields containing user-independent date or time values. Then pairwise POIDC-F_H and pairwise POIDC-$F_{\mathrm{Enc_A}}$ are login-unlinkable with respect to colluding RPs in the random oracle model.*

Proof. This follows directly from Theorem 12. F_H and $F_{\mathrm{Enc_A}}$ do not depend on the user u, and thus satisfy the required assumption. □

It is worth pointing out that obtaining privacy with respect to colluding RPs requires more assumptions than obtaining privacy with respect to the IdP. In particular, we must guarantee that the *private_id_token* does not contain additional user-dependent fields, and that the user's *user_id* is not leaked to any RPs. The second point must be ensured by IdPs implementing pairwise POIDC, which must hold these values secret. In particular, its front-end must not persist the *user_id* in the user agent's state in a way that could be accessible by malicious RPs.

In general, these stronger assumptions combined with the additional complexity incurred by the zero-knowledge proof mean that pairwise POIDC should be employed only in scenarios where the user's privacy with respect to both the IdP *and* colluding RPs is critical. Furthermore, if the user provides additional correlatable information to the RP after the login, such as her e-mail address, this would undermine the purpose of pairwise POIDC. If providing such information to the RPs is a functional requirement, then regular POIDC should be used instead.

4.6 Security proofs with Tamarin

We used the Tamarin prover [6, 7] to prove security properties for standard OIDC, both implicit flow and code flow, as well as POIDC and pairwise POIDC.

We next explain our Tamarin models for OIDC and POIDC. These protocols differ from traditional cryptographic protocols, and the differences affect modeling.

The first important difference is that these protocols operate on the application layer. As such, they rely on an underlying secure channel, usually provided by TLS. Thus, it is insufficient to model an insecure network and cryptography: TLS channels with abstract security properties must also be modeled.

Another important difference is that the user directly participates in the protocol via her user agent (browser), while usually security protocols describe only the machines at both ends. The user's intent and consent to log in to an RP are critical for defining our main security property. Thus, the user must be modeled as a separate protocol role who interacts with her user agent.

We next describe our abstract TLS channel model and our user-browser communication model. These models provide the baseline for our protocol models and are included in each of them.

Abstract TLS model and agent compromise. We assume that TLS and the underlying public-key infrastructure are secure. A formal analysis of TLS is out of scope for this work, but can be found in [73]. We model a TLS connection abstractly as a channel between a browser endpoint, which is not authenticated, but is invariant during a session, and an authenticated server endpoint. This models the common scenario where server certificates are used, but no client certificates. The server endpoint is modeled as a public name `$Server` and the browser endpoint as a session identifier `~brID_TLS`, which is freshly generated for each session.

We next show in more detail how we model abstract TLS channels using rules in our Tamarin models. The rule for initiating a TLS session in our Tamarin models looks as follows.

```
rule Init_TLS_Session:
[ !St_Browser_Init(~brID, $User), Fr(~brID_TLS) ] -->
[ !St_Browser_Session(~brID, $Server, ~brID_TLS)
, !St_Server_Session($Server, ~brID_TLS) ]
```

`Fr(~brID_TLS)` models `~brID_TLS` as a freshly generated, unique, and unpredictable value. The rule generates two facts that are used in protocol rules to exchange messages over the TLS channel. Note that facts starting with an exclamation mark, such as `!St_Browser_Init(...)` can be used multiple times, i.e., they are not consumed when used as the input of a rule. We show an example protocol rule to illustrate how these

facts model received and sent messages. The rule models an authentication request to the IdP in OIDC implicit flow. We omit some details, since the purpose is just to illustrate the TLS channel model.

```
rule AuthRequest_IdP:
[ !Client_to_Server_TLS(~brID_TLS, $IdP, <'authRequest', ...>)
, !St_Server_Session($IdP, ~brID_TLS)
, !St_RP_Registered($RP, $IdP, $client_id, ...) ] -->
[ St_IdP_1(...)
, !Server_to_Client_TLS($IdP,~brID_TLS,<'show', <'authDialog', $IdP>>) ]
```

We show the corresponding rules executed by the browser later. Note that the facts denoting messages on a channel can be used as input multiple times, modeling a channel that is not inherently replay-protected.

We add rules to TAMARIN's built-in Dolev-Yao attacker that allow him also to create TLS channels. We also provide rules that allow the attacker to compromise RPs and IdPs. A compromised RP or IdP leaks its secrets to the attacker. Furthermore, an attacker can send arbitrary messages over TLS channels controlled by a compromised RP or IdP as well as learn any message received on such channels.

Modeling the user and user agent. We differentiate between the user (person) and the user agent (browser). We model communication between the user and her user agent as a secure, replay-protected channel. Note that we assume that the platform running the user agent is not compromised.

We assume that a single user agent is only used by one user at a time, but that a user can have multiple user agents. The user performs checks with respect to human-readable names and URIs. The user checks whether the RP's name in a consent dialogue is the one she intended to log in to. That is, the user remembers the RP's name to which she wants to log in, and only gives consent if the displayed name is the one she remembers.

The user performs actions, such as initiating the protocol (e.g., by clicking on a button on the RP page) or giving consent (by clicking on a button in a consent dialogue). The browser is reactive, and performs actions only if prompted to do so. The following three kinds of prompts can be sent to the browser.

1. User actions, such as clicking on a link or entering a URI.

2. A server sending content, triggering a web page to be displayed to the user, and/or the execution of JavaScript code.

3. A server redirecting the browser to another URI.

We next show how we formalize each of these actions in our TAMARIN models. User actions are modeled as follows.

```
rule Browser_Performs_User_Action:
[ User_InputTo_Browser($User, ~brID, <'userAction', $Server, message>)
, !St_Browser_Init(~brID, $User)
, !St_Browser_Session(~brID, $Server, ~brID_TLS) ] -->
[ !Client_to_Server_TLS(~brID_TLS, $Server, message) ]
```

The `User_InputTo_Browser` fact models a secure message from the user to the browser. For example, when the user clicks on a *Login with IdP* idp button on RP rp, this is modeled as a secure message `<'userAction', $rp, <'loginWith', $idp>>` from the user to the browser. The browser then reacts to this input by sending a message to the server named. In the example, the browser sends a TLS message `<'loginWith', $idp>` to the RP where the user clicked on the button.

We next show the rule for displaying web pages to the user.

```
rule Browser_Shows:
[ !Server_to_Client_TLS($Server, ~brID_TLS, <'show', message>)
, !St_Browser_Session(~brID,  $Server, ~brID_TLS)
, !St_Browser_Init(~brID, $User) ] -->
[ Browser_Shows_User(~brID, $User, $Server, message) ]
```

When the browser receives a TLS message `<'show', message>` from a server, the browser then displays `message` to the user, modeled as a `Browser_Shows_User` fact.

For example, when the IdP requests the user to authenticate, the IdP sends the message `<'show', 'authDialog'>` to the browser. The browser then displays a dialogue asking the user to enter her credentials for IdP idp. This is modeled with the fact `Browser_Shows_User(~brID, $User, $idp, 'authDialog')`.

We next show the rule for redirection.

```
rule Browser_Redirects_To_URI:
[ !Server_to_Client_TLS($Server1, ~brID_TLS1,
 <'redirectToURI', $uri, message>)
, !St_Browser_Session(brID, $Server1, ~brID_TLS1)
, !St_Browser_Session(brID, $Server2, ~brID_TLS2)
, !Uri_belongs_to($uri, $Server2) ] -->
[ !Client_to_Server_TLS(~brID_TLS2, $Server2, message) ]
```

The browser receives `<'redirectToURI', $uri, message>` from `$Server1`, requesting to redirect `message` to `$uri`. The fact `!Uri_belongs_to($uri, $Server2)` denotes that `$uri` belongs to `$Server2`, so the browser redirects `message` to `$Server2`.

Note that all three rules require that the browser has active TLS sessions with one or more servers, which are modeled by the `!St_Browser_Session(...)` facts. If the browser does not yet have an active session with one of these servers, then the `Init_TLS_Session` rule must first be executed.

In general, the browser is stateless and unaware of any protocol logic. The one notable exception is the JavaScript code that the browser loads from the IdP website in POIDC and pairwise POIDC. In this case, the browser keeps state that models its progression through the code execution. In particular, the browser is in one of two possible states after loading the code and displaying the consent dialogue: (1) waiting for the user to give consent, or (2) waiting for a *private_id_token* to be returned from the IdP back-end. After the browser sends the *private_id_token* to the *redirect_uri* given in the *client_id_binding*, the JavaScript code execution is finished.

4.6.1 Security property and proofs

We formalize *user authentication* similarly to the authentication property given by Fett, Küsters, and Schmitz [5]: An attacker should not be able to log in under an honest user's account at an honest RP using an honest IdP. For our security property definition, an honest user is one where the user's account at the IdP has not been compromised, and the user does not use compromised browsers or platforms. An honest RP or IdP is an entity that is not controlled by the attacker.

We explicitly model the user's actions separate from the browser's, and our security property also refers to the user's actions. An honest user using an uncompromised user agent should only be logged in at an honest RP using an honest IdP if the user has

both expressed her *intent* (by starting the protocol) and her *consent* (by confirming a consent dialogue). We formalize this as follows.

Definition 60. *A delegated authentication protocol is* secure with respect to user authentication *if the following holds for any honest RP* rp*, honest IdP* idp*, honest user* u*, and browser* b*. Whenever* u *is logged in at RP* rp *with IdP* idp *on* b *as a result of running the protocol, then* b *belongs to* u*,* u *has initiated the protocol, and* u *has given consent to log in to RP* rp *with IdP* idp*.*

Our formalization of this definition in Tamarin differs slightly for the different flows. For OIDC implicit flow and POIDC, we use the following formalization.

```
lemma User_Authentication:
"All rp uid idp #finish.
RP_gets_IDToken(rp, uid, idp)@finish
& not (Ex #j. Is_Compromised(idp)@j)
& not (Ex #j. Is_Compromised(rp)@j)
& not (Ex #j. AdversaryRegisters(uid)@j) ==>
(Ex usr browserSession browser
#k #start #consent #m #n.
UserID_belongs_To(uid, usr, idp)@k // (1)
& UserGivesConsent(usr, rp, idp)@consent // (2)
& UserStartsSession(usr, rp, idp)@start // (3)
& BrowserUser(browser, usr)@n // (4)
& BrowserServerSession(browser, rp, session)@m // (4)
& RPgetsIDToken_FromBr(rp, session, idp)@finish // (4)
& consent < finish & start < consent)" // (5)
```

This lemma can be understood as follows. Whenever an honest RP `rp` receives an *id_token* that contains a subject identifier `uid` that belongs to an honest user and an issuer identifier `idp` that belongs to an honest IdP, then the following hold.

1. The subject identifier `uid` belongs to the user `usr`.
2. The user `usr` has given consent to log in to `rp` using `idp`.
3. The user `usr` has started the protocol session intending to log in to `rp` using `idp`.
4. The RP obtained the *id_token* in a session with a browser that belongs to `usr`.

5. The protocol start happened before giving consent, and giving consent happened before the RP received the *id_token*.

We next explain the adjustments that must be made for the OIDC code flow models. When the code flow is used, the RP receives the *id_token* from the IdP directly rather than from the browser. Thus, in condition (4), we require that the RP receives the *code* instead of the *id_token* from the user's browser, denoted by `RPgetsCode(rp, browserSession, idp)@getCode` in that model.

For pairwise POIDC, the *id_token* does not directly contain the user's global user identifier `uid`, but rather a pairwise subject identifier of the form `h(<uid, cid>)`, and we write `(RP_gets_IDToken(rp, h(<uid, cid>), idp)@finish` in the left-hand side of the implication. Thus, the lemma ensures that this identifier was computed from the *user_id* `uid` that belongs to the user `usr` who gave consent and started the session.

We provide nine TAMARIN models, each with between 300 and 400 lines of code. The models are available at [40] and at [74]. The proofs for the following theorems were computed automatically on a server with 2 CPUs of type Intel(R) Xeon(R) E5-2650 v4 @ 2.2 GHz, 256 GB of RAM, running Ubuntu 16.04.3 LTS, using TAMARIN release version 1.4.1. Each CPU has 12 cores, but we limited our computation to use 10 cores only. The proof runtimes are given in Figure 4.4. The proofs are also available at [40] and at [74].

Theorem 14. *OIDC implicit flow and authorization code flow, both with and without client secret, are secure with respect to user authentication.*

Theorem 15. *POIDC-F_H, POIDC-$F_{\mathrm{Enc_u}}$, POIDC-$F_{\mathrm{Enc_A}}$, and POIDC-F_{cmb} are secure with respect to user authentication.*

Recall from Section 4.5 that pairwise POIDC does not admit variants of POIDC-$F_{\mathrm{Enc_u}}$ and POIDC-F_{cmb}, as these would not provide the desired privacy property with respect to colluding RPs. Therefore, we only prove security of the two meaningful pairwise POIDC variants.

Theorem 16. *Pairwise POIDC-F_H and pairwise POIDC-$F_{\mathrm{Enc_A}}$ are secure with respect to user authentication.*

	Proof Runtime
OIDC implicit flow	14s
OIDC code flow without client secret	34min 41s
OIDC code flow with client secret	3min 52s
POIDC-F_H	31min 44s
POIDC-F_{Enc_u}	31min 59s
POIDC-F_{Enc_A}	34min 08s
POIDC-F_{cmb}	60min 42s
Pairwise POIDC-F_H	28min 32s
Pairwise POIDC-F_{Enc_A}	29min 59s

Figure 4.4: Proof runtime for the automatic proof of user authentication in nine different Tamarin models.

4.7 Related Work

We first compare POIDC and pairwise POIDC to other privacy-preserving single sign-on solutions. Afterwards, we compare our formal security analysis of OIDC, POIDC, and pairwise POIDC to work related to the formal analysis of OIDC and the underlying OAuth 2.0 protocol.

4.7.1 Privacy-preserving single sign-on

Fett, Küsters, and Schmitz have proposed the privacy-preserving single sign-on system SPRESSO [75], for which they prove that the IdP does not learn the RP where the user logs in. Their approach is different from ours: They develop an entirely new protocol whereas we propose changes to the existing OIDC standard. Furthermore, they do not protect privacy with respect to colluding RPs. Their proposal (and ours) are the only ones that we are aware of that actually prove the claimed security and privacy properties.

Other single sign-on systems with enhanced user privacy have been proposed. For example, Maheswaran et al. propose Crypto-Book [76], which builds a privacy-preserving cryptographic layer on top of existing protocols such as OAuth 2.0. Their system requires a user to obtain credentials from multiple credential producers (IdPs), and while it makes use of OAuth 2.0, it is an entirely different credential system rather than

an extension of an existing standard. Dey and Weis propose PseudoID [77], a privacy-preserving extension for OpenID, OIDC's predecessor, based on blind signatures. Their solution introduces a blind signing service where the user must first blind a token that she then presents to the IdP. The use of this service adds steps to the user login process. The Mozilla Persona (or BrowserID) single sign-on system [78], which is no longer supported, was also designed to provide privacy-preserving single sign-on based on a user's e-mail address. However, several attacks on its privacy properties were given by Fett, Küsters, and Schmitz [79].

An alternative to systems based on an identity provider are user-centric systems that require the user to use a secret key when logging in. The most prominent line of work in this area is on anonymous credentials, introduced by Camenisch and Lysyanskaya [80] and implemented in idemix [81]. Camenisch and Pfitzmann explain the privacy issues of federated identity management and how idemix can solve them [82]. There are also other user-centric identity systems based on similar ideas, such as U-Prove [83]. However, these systems have so far not challenged the prominence of standards such as OIDC. One reason for this lack of adoption is the difficulty of key management. This problem also applies to proposals that combine delegated authentication protocols with secret keys held by users, such as the proposal by Camenisch, Gross, and Sommer to improve privacy in the WS-Security standard [84], or UnlimitID by Isaakidis et al. [85].

More recently, Zhang et al. have presented EL PASSO [86], a privacy-preserving single sign-on protocol based on anonymous credentials. Similarly to our proposal, they also rely on the IdP to provide the code executed in the user agent, in their case as a WebAssembly (WASM) [87] module. However, they also require the user agent to hold a secret, for which they leverage browser-based password stores. They do describe how multi-device support is achieved even with such a user secret.

4.7.2 Formal analysis of OIDC or OAuth 2.0

Fett, Küsters, and Schmitz have used their web infrastructure model (WIM) [79] for the manual analysis of OIDC [5]. They also analyzed the OpenID Financial-grade API using the same model [88], as well as OAuth 2.0 [89]. In our OIDC analysis using Tamarin, we model the web infrastructure more abstractly to allow for proof automation and machine-checked proofs.

In the realm of automated analysis, Bansal et al. have used the WebSpi [90] framework for analyzing web applications based on ProVerif [91] to automatically discover attacks on OAuth 2.0. However, neither WebSpi nor WIM model the user as an entity separate from the browser, which makes it difficult to reason about explicit user consent. An interesting direction for future work in this area is to formally model a more detailed web infrastructure, closer to WIM, while still allowing for automation, and additionally model the user as a separate entity. Such a model could be used to obtain even stronger guarantees about the security of web-based protocols such as OIDC.

More recently, Li et al. have presented a modular analysis of OAuth 2.0 in the computational model [92]. They complement existing analyses which have been performed in the symbolic model by providing a detailed analysis of attacker advantages with respect to different security games. However, they only provide manual security proofs.

5

Conclusion

In the introduction, we listed the following relevant aspects of users' digital identity: *security risk, lockout risk, usability, convenience, and privacy.* We next examine how this thesis advances digital identity with respect to all of these points, and then illustrate problems that remain open, showing promising directions for future research.

5.1 Advancements to Digital Identity

In Chapter 2, we have provided the first formalism and methodology that systematically models users' account setups and automatically detects *security risks* and *lockout risks*. These risks arise from subtle connections across users' accounts, credentials, devices, and other aspects of the physical world. They are individual for each user and are often hard to detect without such a systematic approach.

The systematic modeling approach enabled by our *account access graphs* is a game changer in how we think of users' digital identity. Before this work, we at best had an incomplete picture of a user's account setup. In fact, the question of *how a user's account setup looks* was not even well defined. Our work not only allows us to answer this question in a clearly defined way, but we can answer the even more important question of *what security risks arise from the connections in a user's account setup.*

In Chapter 3, we answer these and other questions with respect to real users in a qualitative user study. While our formalism allows for a comprehensive analysis with respect to security and recoverability, users may also optimize their setup for *usability*

and *convenience*. Our user study shows which aspects users prioritize in practice and illustrates their decision-making process with respect to their real account setups.

Our results showed a range of different opinions and beliefs among our participants. It is easy to get stuck in the mindset of a security expert, who thinks about security every day, and then expect that non-expert users should think about security similarly. It is just as easy to disregard non-expert users as not caring about security at all. Neither view reflects what we have observed in our qualitative user study. Our participants commonly considered the importance of their accounts, and applied security best practices only for the more important ones. This is unsurprising given how we all prioritize different aspects of life in our daily routine. While security plays a lesser role for non-expert users than for security experts, there certainly are accounts and digital assets that they value highly, and do worry about protecting. However, usability and convenience must not be disregarded. If a certain technology would improve security but is simply too inconvenient to use, or cannot even be used in certain contexts, it is unlikely to achieve high adoption. We observed this, for example, with text-message-based two factor authentication, where users could not receive their authentication codes while abroad. This was not only inconvenient but actually increased users' lockout risk.

Privacy concerns were also mentioned by our participants. Many convenient tools, such as cloud-synced password managers, improve a user's security with respect to potential account compromise by an attacker. However, they also require the user to trust the cloud service provider not to act as such an attacker itself. Participants also mentioned privacy concerns with respect to biometric authentication methods, e.g., to unlock a phone with fingerprint or face recognition. Our interviews showed that having to make privacy trade-offs may prevent some users from adopting security best practices and tools.

Single sign-on using OpenID Connect exhibits such a trade-off between privacy and security. With respect to security, it offers the benefit of reducing the number of passwords a user has to manage. This eases the load on the user's memory, which may reduce their tendency to reuse passwords or use weak passwords. We also found in our study that users are often willing to use two-factor authentication for a few important accounts. Thus, using single sign-on would generally result in a more secure account setup for the user, especially if they use a strong, unique password and two-factor authentication for their account at the identity provider. However, OpenID Connect

allows the identity provider to learn valuable metadata about the user, namely which services they use, when they use them, and how often. That is, a user who wishes to reap the mentioned security benefits must trade them off against privacy.

In Chapter 4, we offer a way out of this dilemma. *Privacy-preserving OpenID Connect* (POIDC) comes with all the security benefits that the user could obtain from standard OpenID Connect while eliminating the mentioned privacy issue. Usability, convenience, and privacy concerns may all present barriers that prevent users from improving their account setup security. POIDC shows how to tear down one such barrier: The identity provider no longer learns the sensitive metadata about the users' logins. This thesis presents a proof-of-concept, that is, we show how OpenID Connect could in theory be modified to remove this privacy issue. Unfortunately, when designing protocols and solutions in practice, such privacy concerns are usually not the priority. We hope that this thesis contributes to changing this fact. Our interview results from Chapter 3 show that users care about privacy, while the POIDC proposal shows how it could be implemented.

Overall, this thesis improves our *systematic understanding* of the security of users' digital identity, which is comprised of their account setups. It highlights that secure digital identity is *personal* and that *trade-offs* between security and lockout risk, usability, convenience, and privacy must be considered. Our user study shows that these trade-offs are a common cause of security weaknesses in setups. It is therefore critical that we work on making such trade-offs unnecessary. That is, the security community must continue to provide tools, services, and protocols that are not only secure but also *usable, convenient, and privacy-preserving*. POIDC, a single sign-on protocol that is provably both privacy-preserving and secure, is an important step in this direction.

5.2 Directions for Future Work

Our thesis has made significant progress in how we think about users' digital identity, model it systematically as account access graphs, and leverage these results to improve its security. Nevertheless, there is still much work to be done. A single thesis could not realistically aim to solve all problems in the realm of secure digital user identity. Indeed, there are numerous aspects that we considered as promising directions for future work, but were ultimately beyond the scope of this thesis. We next outline these directions

to the reader, and hope that some will feel inspired in continuing our work. After all, digital identity is of critical importance to all Internet users.

The first two directions for future work are directly enabled by the work of this thesis, in particular by the account access graphs framework. In particular, they concern automation of the two major steps in analyzing a user's overall account security: First, we must obtain an accurate representation of the user's account setup in the form of an account access graph. Second, we must analyze this graph in a way that yields actionable advice to the user. We believe that both steps could be automated. The third direction concerns future digital identity standards. It is not directly enabled by this work, but has a strong thematical connection to it.

Account access graph mining. The approach we took in our user study for eliciting our participants' account access graphs does not scale to larger participant sample sizes. The process was manual, with the interviewer entering the data into a tool himself. A first step for improving the process would be to develop a tool that is geared more towards *end-users*, which is mostly a challenge in terms of user interface and experience design. An open research question, however, is how account access graphs can be elicited *automatically*, or semi-automatically, with the user only providing access to select documents or interfaces. We call this process *account access graph mining* in reference to data mining [93].

There are various data sources that could be leveraged by such a mining process. One such source is users' *password manager* data, which already contains a lot of information about accounts. Other valuable data sources are *GDPR data requests* according to the *right of access by the data subject* [94]. Data about a user of a large service provider contains a lot of valuable information on the user's account setup. For example, if the user has an iPhone and an Apple account, the request would also contain information about the user's iPhone; similarly for Google and Android devices.

It is not a priori clear what level of *completeness* and *accuracy* such an account access graph mining process should aim for. If the goal is to discover the ground truth about the user's accounts, this implies different requirements than if the goal is to discover simply what information about the users would be available to colluding service providers. Furthermore, it is not clear what levels of completeness or accuracy are necessary with respect to the user's real setup. For example, are false attacks

arising from incorrectly modeled account connections acceptable? Is it acceptable to miss attacks that exist in the real setup? These questions look related to commonly considered completeness and soundness notions, e.g., with respect to formal protocol or program verification. However, it is unclear whether similar definitions make sense for this scenario. Therefore, finding meaningful definitions is an open theoretical research question in addition to the practical problem of building an account access graph miner.

Automating security advice. While account access graph mining would automate the first part of our interview process, it would also be promising to automate the second part. As we have mentioned in Section 3.5, we believe that personalized security advice shows much potential. However, if every user would have to sit down with a security expert for an hour or longer, as in our interviews, this process would not scale. Most advice we gave to our participants was based on the security weaknesses we discovered in the interviews. Due to the flexible definitions given in Section 2.4, *predicates* could be designed to automatically capture weaknesses we have discovered manually, such as password reuse. We instead chose to perform a manual analysis as part of a discussion for our interviews since this better aligned with our aim of answering our research questions.

When the aim is primarily to provide security advice, additional predicates could help to automate a larger part of the security analysis. A conceivable end goal is a tool where users can enter their account access graphs, or have them mined automatically, and then obtain automated security advice. Such a tool would have to present this advice in a way that is understandable for non-experts and differentiated with respect to criticality. The second point is especially important since our user study showed that users prioritize their more important accounts, but do not necessarily care about every single security weakness in their setup. How to design an effective such tool is an open question, and tool candidates would have to be evaluated in further user studies.

Analyzing and improving future identity standards. While the current de-facto standard for single sign-on is OpenID Connect, this has not always been the case, and likely will not always remain the case. Before OpenID Connect and OAuth 2.0 was introduced, SAML [95] was used for similar use cases, and it is still in use today. Unsurprisingly, numerous standardization bodies and communities are actively working on new identity standards. A promising concept is that of *self-sovereign identity* [96]

and the associated W3C Recommendations, Decentralized Identifiers (DIDs) [97] and Verifiable Credentials (VCs) [98]. The goal of these standards and related community efforts is to bring digital identity closer to identity in the physical world, where people hold credentials, such as passports and driver's licenses, who are issued by different issuers. They can present these credentials to verifiers upon request. In particular, the physical world does not have the equivalent of an *identity provider*, a central party who handles a person's identity. Instead, there are numerous *credential issuers*, but the process of presenting a credential does not involve a central party, and the verifier can verify the credential without contacting its issuer. Due to this lack of a central identity provider, this concept by design has the potential to be more respectful of the user's privacy than even our POIDC proposal.

There is promising community progress around development of open-source software [99] and open standards [100]. However, there is a lack of representation of the academic formal methods and security communities in the involved groups. Recent experience with TLS 1.3 [73, 101] has shown that formal protocol analysis can be used effectively to detect security problems in early standard drafts preemptively rather than when it is already widely deployed.

The author of this thesis was involved in a community effort related to self-sovereign identity topics, called Rebooting the Web of Trust (RWOT) [102], and helped produce a whitepaper [103]. However, academic work on the topic was outside of the scope of this thesis. The main reason for this is that the entire scope of the topics surrounding self-sovereign identity proved too large to address as what would have been only a minor part of this thesis. Furthermore, the rapidly changing landscape of community efforts, e.g., with respect to protocol definitions, made it difficult to understand the relevance of different efforts. However, the work is now converging towards standardization, with the DIDComm working group aiming to standardize protocols [100] and the Trust over IP (ToIP) foundation [104] focused on governance in addition to technology. We strongly believe that formal methods and verification could play an important role in guaranteeing the security of these emerging next-generation digital identity standards.

Bibliography

[1] Nat Sakimura, John Bradley, Mike Jones, Breno de Medeiros, and Chuck Mortimore. **OpenID Connect Core 1.0 incorporating errata set 1**. 2014. 1, 95, 96

[2] Nathanael Andrews. **Can I Get Your Digits: Illegal Acquisition of Wireless Phone Numbers for Sim-Swap Attacks and Wireless Provider Liability**. *Nw. J. Tech. & Intell. Prop.*, **16**:79, 2018. 2, 62, 72

[3] Kevin Lee, Ben Kaiser, Jonathan Mayer, and Arvind Narayanan. **An Empirical Study of Wireless Carrier Authentication for SIM Swaps**, 2020. 2, 62, 72

[4] **verimi.de**. https://verimi.de/. Accessed: 2018-04-20. 5

[5] Daniel Fett, Ralf Küsters, and Guido Schmitz. **The Web SSO Standard OpenID Connect: In-depth Formal Security Analysis and Security Guidelines**. *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 189–202, 2017. 5, 9, 100, 142, 146

[6] Simon Meier, Benedikt Schmidt, Cas J. F. Cremers, and David Basin. **The TAMARIN Prover for the Symbolic Analysis of Security Protocols**. In *CAV*, **8044** of *LNCS*, pages 696–701. Springer, 2013. 8, 101, 138

[7] Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David Basin. **Automated analysis of Diffie-Hellman protocols and advanced security properties**. In *Computer Security Foundations Symposium (CSF)*, pages 78–94. IEEE, 2012. 8, 101, 138

[8] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. **Seems Legit: Automated Analysis of Subtle Attacks on Protocols That Use Signatures**. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, pages 2165–2180, New York, NY, USA, 2019. ACM. 9, 103, 104

[9] William F Dowling and Jean H Gallier. **Linear-time algorithms for testing the satisfiability of propositional Horn formulae**. *The Journal of Logic Programming*, **1**(3):267–284, 1984. 27

[10] Nachum Dershowitz and Zohar Manna. **Proving termination with multiset orderings**. *Communications of the ACM*, **22**(8):465–476, 1979. 34

[11] Sjouke Mauw and Martijn Oostdijk. **Foundations of attack trees**. In *International Conference on Information Security and Cryptology*, pages 186–198. Springer, 2005. 34, 51, 52, 53

[12] **Haskell Language**. https://www.haskell.org/. Accessed: 2020-09-08. 50

[13] Sven Hammann, Saša Radomirović, Ralf Sasse, and David Basin. **Haskell code for Account Access Graphs**. https://infsec.ethz.ch/research/software/account_access_graphs.html, 2019. 50

[14] **JavaServer Pages Technology**. https://www.oracle.com/java/technologies/jspt.html. Accessed: 2020-09-08. 50

[15] **Account Access Graphs Frontend**. http://doi.org/10.5905/ethz-1007-306. Accessed: 2020-10-22. 50

[16] **mxGraph library**. https://jgraph.github.io/mxgraph/. Accessed: 2020-10-02. 50

[17] William E Vesely, Francine F Goldberg, Norman H Roberts, and David F Haasl. **Fault tree handbook**. Technical report, Nuclear Regulatory Commission Washington DC, 1981. 51

[18] Chris Salter, O Sami Saydjari, Bruce Schneier, and Jim Wallner. **Toward a secure system engineering methodolgy**. In *Proceedings of the 1998 workshop on New security paradigms*, pages 2–10. ACM, 1998. 51

[19] Bruce Schneier. **Attack trees**. *Dr. Dobb's journal*, **24**(12):21–29, 1999. 51

[20] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. **DAG-based attack and defense modeling: Don't miss the forest for the attack trees**. *Computer Science Review*, **13**:1–38, 2014. 51

[21] Ahto Buldas, Olga Gadyatskaya, Aleksandr Lenin, Sjouke Mauw, and Rolando Trujillo-Rasua. **Attribute evaluation on attack trees with incomplete information**. *Computers & Security*, **88**:101630, 2020. 54

[22] John W Lloyd. *Foundations of logic programming*. Springer Science & Business Media, 2012. 54

[23] Michael Gelfond and Vladimir Lifschitz. **The Stable Model Semantics for Logic Programming**. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, **88**, pages 1070–1080. MIT Press, 1988. 54

[24] Michael Gelfond and Vladimir Lifschitz. **Classical Negation in Logic Programs and Disjunctive Databases**. *New Generation Computing*, **9**(3-4):365–386, 1991. 54

[25] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003. 54

[26] Victor W Marek and Miroslaw Truszczyński. **Stable models and an alternative logic programming paradigm**. In *The Logic Programming Paradigm*, pages 375–398. Springer, 1999. 54

BIBLIOGRAPHY

[27] Joseph Bonneau, Cormac Herley, Paul C van Oorschot, and Frank Stajano. **The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes**. In *IEEE Symposium on Security and Privacy, SP 2012*, pages 553–567. IEEE, 2012. 55

[28] Joseph Bonneau, Elie Bursztein, Ilan Caron, Rob Jackson, and Mike Williamson. **Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google**. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015*, pages 141–150. International World Wide Web Conferences Steering Committee, 2015. 55

[29] Ariel Rabkin. **Personal knowledge questions for fallback authentication: Security questions in the era of Facebook**. In *Proceedings of the 4th symposium on Usable privacy and security*, pages 13–23. ACM, 2008. 55

[30] John G Brainard, Ari Juels, Ronald L Rivest, Michael Szydlo, and Moti Yung. **Fourth-factor authentication: somebody you know**. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pages 168–178. ACM, 2006. 55

[31] Stuart E Schechter, Serge Egelman, and Robert W Reeder. **It's not what you know, but who you know: a social approach to last-resort authentication**. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009*, pages 1983–1992. ACM, 2009. 55

[32] Markus Jakobsson, Erik Stolterman, Susanne Wetzel, and Liu Yang. **Love and authentication**. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 197–200. ACM, 2008. 55

[33] Shirley Gaw and Edward W Felten. **Password management strategies for online accounts**. In *Proceedings of the second symposium on Usable privacy and security*, pages 44–55, 2006. 57, 92

[34] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. **The tangled web of password reuse.** In *NDSS*, **14**, pages 23–26, 2014. 57, 92

[35] Sarah Pearman, Jeremy Thomas, Pardis Emami Naeini, Hana Habib, Lujo Bauer, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, and Alain Forget. **Let's go in for a closer look: Observing passwords in their natural habitat**. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 295–310, 2017. 57, 93

[36] Sanam Ghorbani Lyastani, Michael Schilling, Sascha Fahl, Michael Backes, and Sven Bugiel. **Better managed than memorized? Studying the Impact of Managers on Password Strength and Reuse**. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 203–220, 2018. 57, 93

[37] Elissa M Redmiles, Everest Liu, and Michelle L Mazurek. **You Want Me To Do What? A Design Study of Two-Factor Authentication Messages.** In *SOUPS*, 2017. 57, 93

[38] Yusuf Albayram, Mohammad Maifi Hasan Khan, and Michael Fagan. **A study on designing video tutorials for promoting security features: A case study in the context of two-factor authentication (2FA)**. *International Journal of Human–Computer Interaction*, **33**(11):927–942, 2017. 57, 93, 94

[39] Emiliano De Cristofaro, Honglu Du, Julien Freudiger, and Greg Norcie. **A Comparative Usability Study of Two-Factor Authentication**. *NDSS Workshop on Usable Security (USEC 2014)*, 2014. 57, 94

[40] **Supplementary material: Account graphs and coded transcripts for Chapter 3; Tamarin models for Chapter 4**. `https://www.research-collection.ethz.ch/handle/20.500.11850/443942`, 2020. Accessed: 2020-10-22. 65, 144

[41] Joseph Gibaldi, Walter S Achtert, and Modern Language Association of America. *MLA handbook for writers of research papers*. Modern Language Association of America New York, 2003. 65

[42] David R Thomas. **A general inductive approach for qualitative data analysis**. 2003. 65

[43] Graham R Gibbs. *Analyzing qualitative data*, **6**. Sage, 2018. 65

[44] Heather L Stuckey. **The second step in data analysis: Coding qualitative research data**. *Journal of Social Health and Diabetes*, **3**(01):007–010, 2015. 65

[45] Dennis Mirante and Justin Cappos. **Understanding password database compromises**. *Dept. of Computer Science and Engineering Polytechnic Inst. of NYU, Tech. Rep. TR-CSE-2013-02*, 2013. 72, 89

[46] Michael Fagan and Mohammad Maifi Hasan Khan. **Why do they do what they do?: A study of what motivates users to (not) follow computer security advice**. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, pages 59–75, 2016. 72, 91

[47] Iulia Ion, Rob Reeder, and Sunny Consolvo. **"... no one can hack my mind": Comparing Expert and Non-Expert Security Practices**. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015) 2015*, pages 327–346, 2015. 72, 91

[48] Karoline Busse, Julia Schäfer, and Matthew Smith. **Replication: no one can hack my mind revisiting a study on expert and non-expert security practices and advice**. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019) 2019*, 2019. 72, 91

[49] **About Face ID advanced technology**. `https://support.apple.com/en-us/HT208108`, 2020. Accessed: 2020-07-08. 76

[50] Yue Li, Haining Wang, and Kun Sun. **Email as a master key: Analyzing account recovery in the wild**. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1646–1654. IEEE, 2018. 90

[51] Yue Li, Zeyu Chen, Haining Wang, Kun Sun, and Sushil Jajodia. **Understanding Account Recovery in the Wild and Its Security Implications**. *IEEE Transactions on Dependable and Secure Computing*, 2020. 90

[52] Enis Ulqinaku, Hala Assal, AbdelRahman Abdou, Sonia Chiasson, and Srdjan Čapkun. **Is Real-time Phishing Eliminated with FIDO? Social Engineering Downgrade Attacks against FIDO Protocols**. Cryptology ePrint Archive, Report 2020/1298, 2020. https://eprint.iacr.org/2020/1298. 90

[53] **FIDO Alliance**. https://fidoalliance.org/. Accessed: 2020-10-22. 90

[54] Rick Wash. **Folk models of home computer security**. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, pages 1–16, 2010. 92

[55] Marian Harbach, Sascha Fahl, and Matthew Smith. **Who's afraid of which bad wolf? A survey of IT security risk awareness**. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 97–110. IEEE, 2014. 92

[56] Ruogu Kang, Laura Dabbish, Nathaniel Fruchter, and Sara Kiesler. **"My Data Just Goes Everywhere:" User mental models of the internet and implications for privacy and security**. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015) 2015)*, pages 39–52, 2015. 92

[57] Viktor Taneski, Marjan Heričko, and Boštjan Brumen. **Password security—No change in 35 years?** In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1360–1365. IEEE, 2014. 92

[58] Sarah Pearman, Shikun Aerin Zhang, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. **Why people (don't) use password managers effectively**. In *Fifteenth Symposium On Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA*, pages 319–338, 2019. 93

[59] D. Hardt. **The OAuth 2.0 Authorization Framework**. RFC 6749, RFC Editor, October 2012. http://www.rfc-editor.org/rfc/rfc6749.txt. 96

[60] M. Jones, J. Bradley, and N. Sakimura. **JSON Web Token (JWT)**. RFC 7519, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7519.txt. 96

[61] **OAuth 2.0 Security Best Current Practice**. https://tools.ietf.org/html/draft-ietf-oauth-security-topics-09, 2018. Accessed: 2020-07-17. 97

[62] E. Rescorla. **The Transport Layer Security (TLS) Protocol Version 1.3**. RFC 8446, RFC Editor, August 2018. 100

[63] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014. 100, 101, 103

[64] Victor Shoup. **Sequences of games: a tool for taming complexity in security proofs.** *IACR Cryptology ePrint Archive*, **2004**:332, 2004. 100

[65] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. **Uniform Resource Identifier (URI): Generic Syntax, Section 3.5**. STD 66, RFC Editor, January 2005. http://www.rfc-editor.org/rfc/rfc3986.txt. 105

[66] David L Chaum. **Untraceable electronic mail, return addresses, and digital pseudonyms**. *Communications of the ACM*, **24**(2):84–90, 1981. 106

[67] Roger Dingledine, Nick Mathewson, and Paul Syverson. **Tor: The second-generation onion router**. Technical report, Naval Research Lab Washington DC, 2004. 106

[68] **XMLHttpRequest** . https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest. Accessed: 2020-07-17. 112

[69] **Bitcoin Improvement Proposal 39: Mnemonic code for generating deterministic keys**. https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki. Accessed: 2020-07-17. 121

[70] Iraklis Leontiadis, Kaoutar Elkhiyaoui, and Refik Molva. **Private and Dynamic Time-Series Data Aggregation with Trust Relaxation**. In *Proceedings of the 13th International Conference on Cryptology and Network Security - Volume 8813*, pages 305–320, 2014. 125

[71] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. **ZkBoo: Faster zero-knowledge for boolean circuits**. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1069–1083, 2016. 130

[72] Amos Fiat and Adi Shamir. **How to prove yourself: Practical solutions to identification and signature problems**. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986. 130, 133, 136

[73] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. **A Comprehensive Symbolic Analysis of TLS 1.3**. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, pages 1773–1788, 2017. 139, 154

[74] **Tamarin models for Chapter 4 on github**. https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/thesis-SvenHammann-POIDC, 2020. Accessed: 2020-10-22. 144

[75] Daniel Fett, Ralf Küsters, and Guido Schmitz. **SPRESSO: A Secure, Privacy-Respecting Single Sign-On System for the Web**. In *Proceedings of the ACM CCS, 2015*, pages 1358–1369. 145

[76] John Maheswaran, Daniel Jackowitz, Ennan Zhai, David Isaac Wolinsky, and Bryan Ford. **Building privacy-preserving cryptographic credentials from federated online identities**. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 3–13. ACM, 2016. 145

[77] Arkajit Dey and Stephen Weis. **PseudoID: Enhancing Privacy in Federated Login**. In *Hot Topics in Privacy Enhancing Technologies*, pages 95–107, 2010. 146

[78] **Mozilla Persona Website**. https://developer.mozilla.org/en-US/docs/Archive/Mozilla/Persona. Accessed: 2018-04-26. 146

BIBLIOGRAPHY

[79] Daniel Fett, Ralf Küsters, and Guido Schmitz. **An Expressive Model for the Web Infrastructure: Definition and Application to the Browser ID SSO System**. In *2014 IEEE Symposium on Security and Privacy*, pages 673–688, 2014. 146

[80] Jan Camenisch and Anna Lysyanskaya. **An efficient system for non-transferable anonymous credentials with optional anonymity revocation**. *Advances in Cryptology - EUROCRYPT 2001*, pages 93–118, 2001. 146

[81] Jan Camenisch and Els Van Herreweghen. **Design and implementation of the idemix anonymous credential system**. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 21–30. ACM, 2002. 146

[82] Jan Camenisch and Birgit Pfitzmann. **Federated Identity Management**. In *Security, Privacy, and Trust in Modern Data Management*, pages 213–238. Springer, 2007. 146

[83] **U-Prove**. https://www.microsoft.com/en-us/research/project/u-prove/. Accessed: 2020-03-02. 146

[84] Jan Camenisch, Thomas Gross, and Dieter Sommer. **Enhancing privacy of federated identity management protocols: anonymous credentials in ws-security**. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, pages 67–72. ACM, 2006. 146

[85] Marios Isaakidis, Harry Halpin, and George Danezis. **UnlimitID: Privacy-preserving federated identity management using algebraic MACs**. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 139–142. ACM, 2016. 146

[86] Zhiyi Zhang, Michał Król, Alberto Sonnino, Lixia Zhang, and Etienne Rivière. **EL PASSO: Privacy-preserving, Asynchronous Single Sign-On**. *arXiv preprint arXiv:2002.10289*, 2020. 146

[87] 146

[88] Daniel Fett, Pedram Hosseyni, and Ralf Küsters. **An Extensive Formal Security Analysis of the OpenID Financial-grade API**. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 453–471. IEEE, 2019. 146

[89] Daniel Fett, Ralf Küsters, and Guido Schmitz. **A comprehensive formal security analysis of OAuth 2.0**. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1204–1215. ACM, 2016. 146

[90] Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Sergio Maffeis. **Discovering concrete attacks on website authorization by formal analysis 1**. *Journal of Computer Security*, **22**(4):601–657, 2014. 147

[91] Bruno Blanchet. **An Efficient Cryptographic Protocol Verifier Based on Prolog Rules**. In *Proceedings of the 14th IEEE workshop on Computer Security Foundations*, page 82, 2001. 147

[92] Xinyu Li, Jing Xu, Zhenfeng Zhang, Xiao Lan, and Yuchen Wang. **Modular Security Analysis of OAuth 2.0 in the Three-Party Setting**. 2020. 147

[93] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011. 152

[94] **Art. 15 GDPR: Right of access by the data subject**. https://gdpr-info.eu/art-15-gdpr/. Accessed: 2020-09-11. 152

[95] **Security Assertion Markup Language (SAML) V2.0 Technical Overview**. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html. Accessed: 2020-09-11. 153

[96] Allen, Christopher. **The Path to Self-Sovereign Identity**. http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html. Accessed: 2020-09-11. 153

[97] **Decentralized Identifiers (DIDs) v1.0**. https://www.w3.org/TR/did-core/. Accessed: 2020-09-11. 154

[98] Sporny, Manu and Longley, Dave and Chadwick, David. **Verifiable Credentials Data Model 1.0**. https://www.w3.org/TR/vc-data-model/. Accessed: 2020-09-11. 154

[99] **Hyperledger Aries**. https://www.hyperledger.org/use/aries. Accessed: 2020-09-11. 154

[100] **DID Communication Working Group**. https://identity.foundation/working-groups/did-comm.html. Accessed: 2020-09-11. 154

[101] Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. **Automated analysis and verification of TLS 1.3: 0-RTT, resumption and delayed authentication**. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 470–485. IEEE, 2016. 154

[102] **Rebooting Web-of-Trust**. https://www.weboftrust.info/. Accessed: 2020-09-11. 154

[103] **Alice Attempts to Abuse a Verifiable Credential**. https://github.com/WebOfTrustInfo/rwot9-prague/blob/master/final-documents/alice-attempts-abuse-verifiable-credential.pdf. Accessed: 2020-09-11. 154

[104] **Trust over IP Foundation**. https://trustoverip.org/. Accessed: 2020-09-11. 154