Controlling Access to an Oblivious Database using Stateful Anonymous Credentials

Scott Coull Matthew Green Susan Hohenberger

Information Security Institute
The Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218
{coulls,mgreen,susan}@cs.jhu.edu

Abstract

In this work, we consider the task of allowing a content provider to enforce complex access control policies on oblivious protocols conducted with anonymous users. As our primary application, we show how to construct privacy-preserving databases by combining oblivious transfer with an augmented anonymous credential system. This permits a database operator to restrict which items each user may access, without learning anything about users' identities or item choices. This strong privacy guarantee holds even when users are assigned different access control policies and are allowed to adaptively make many queries. Our system is based on standard assumptions in the standard model and, after an initial setup phase, each transaction requires only constant time.

As a main building block of our work, we show how to augment existing anonymous credential systems so that, in addition to certifying a user's attributes, they also store state about a user's access history, which is updated with each use of the credential. In addition to adaptive oblivious transfer, we show that *stateful anonymous credentials* can be efficiently coupled with protocols for blind signatures and oblivious keyword search to privately control access on these requests. Our construction supports a wide range of access control policies, including efficient and private realizations of the Brewer-Nash (Chinese Wall) and Bell-LaPadula (Multilevel Security) policies, which are used for financial and defense applications. Overall, our stateful anonymous credential system provides a balance between the seemingly conflicting goals of access control and user privacy.

1 Introduction

There is an increasing need to provide privacy to users accessing sensitive information, such as medical or financial data. The mere fact that a rare disease specialist accesses a certain patient's medical record exposes information about the private contents of the record. At the same time, newly developed regulations governing such sensitive data (*e.g.,* Sarbanes-Oxley, HIPAA) require content providers to enact strict accounting procedures. These may seem like conflicting goals since the specialist may wish to hide which patient's record she is requesting while the database administrator may wish to ensure that the doctor's collective accesses do not violate regulations. The situation becomes even more precarious when a patient uses such a database to look up information about a potentially sensitive medical condition. In such cases, the patient's identity, as well as her access patterns, must remain hidden from the database administrator. The increasing

trend toward outsourcing and distributing sensitive databases, such as the outsourced medical database provided by Google Health [32], makes these concerns all the more compelling.

Previous works have proposed to construct privacy-friendly databases using Private Information Retrieval [25] or Oblivious Transfer [36, 20]. In a k-out-of-N Oblivious Transfer protocol, a content provider with messages $M_1, \ldots, M_N$ and a user with indices $\sigma_1, \ldots, \sigma_k \in [1, N]$ interact in such a way that at the end the user obtains $M_{\sigma_1}, \ldots, M_{\sigma_k}$ without learning anything about the other messages and the provider does not learn anything about $\sigma_1, \ldots, \sigma_k$. This tool leads to privacy-friendly databases *when the user gets her choice of any files with no restrictions.* Unfortunately that scenario rules out many practical database applications. Worse, the previous work in this area provides no insight as to how access control might ever be incorporated into such a database, since traditional access control mechanisms assume knowledge of the items being requested.

Thus, to realize a practical "oblivious database" for our users, we must couple it with some manner of enforceable access controls. We make two design choices that act as guiding principles for the design of our system. Our first is to maintain all anonymity or privacy guarantees provided by the oblivious transfer protocol. We reject any solutions that use pseudonyms or allow for some form of transaction linking, since it is too difficult to infer what compromise to privacy might result. Secondly, we wish to enforce a strong notion of access control, where the database operator may limit each access based on the user's identity, item requested, and even a *history* of the user's previous requests. Finally, we require our solution to be efficient: each transaction should take constant time, regardless of a user's access history or the complexity of the access policy which she must follow.

Contributions. To achieve the goals above, we show how to efficiently couple an adaptive, oblivious transfer protocol with an *anonymous credential* scheme [23, 16], in order to provide non-trivial, real-world access controls for oblivious databases. Specifically, we present an extension to existing anonymous credential systems to support history-dependent access, by embedding the user's current state into the credential, and *dynamically* updating that state according to well-defined policies governing the user's actions. These *stateful anonymous credentials* are built on top of well-known signatures with efficient protocols [35, 16, 17, 6]. Our constructions are secure in the standard model under basic assumptions, such as Strong RSA. Additionally, we introduce a technique for efficiently proving that a committed value lies in a *hidden* range that is unknown to the verifier, which may be of independent interest.

We then show how our constructions can be used to achieve non-trivial access control policies, including the Brewer-Nash (Chinese Wall) [11] and Bell-LaPadula (Multilevel Security) [3] model, which are used in a number of settings, including financial institutions and classified government systems. In addition, we also show how to combine our anonymous credential system with several other anonymous and oblivious protocols, like blind signing protocols [16, 17, 33] and searches over encrypted data [45]. We provide simulation-based security definitions for our stateful anonymous credentials, as well as an anonymous and oblivious database system with access controls.

Related Work. Several previous works sought to limit *anonymous* user actions, either directly within an existing protocol or through the use of anonymous credentials. Aiello, Ishai, and Reingold [1] proposed *priced* oblivious transfer, in which each user is given a declining balance that is "spent" on each transfer. However, here user anonymity is not protected, and the protocol is also vulnerable to *selective-failure* attacks in which a malicious server induces faults to deduce the user's selections [36, 20]. The more general concept of *conditional* oblivious transfer was proposed by Di Crescenzo, Ostrovsky, and Rajagopolan [28] and subsequently strengthened by Blake and Kolesnikov [4]. In conditional oblivious transfer, the sender and receiver maintain private inputs

(x and y, respectively) to some publicly known predicate $q(\cdot, \cdot)$ (*e.g.,* the greater than equal to relation on integers). The items in the oblivious transfer scheme are encrypted such that the receiver can complete the oblivious transfer and recover her data if and only if $q(x, y) = 1$. In addition, techniques from e-cash and anonymous credentials have been used to place simple limitations on an anonymous user's actions, such as preventing a user from logging in more than once in a given time period [12], authenticating anonymously at most k times [42], or preventing a user from exchanging too much money with a single merchant [14]. Rather than providing a specific type of limitation or restricting the limitation to a particular protocol, our proposed system instead provides a general method by which arbitrary access control policies can be implemented to a wide variety of anonymous and oblivious protocols.

2 Our Model and Definitions

The goal of typical anonymous credential systems is to provide users with a way of proving certain attributes about themselves (*e.g.,* age, or height) without revealing their identity. Users conduct this proof by obtaining a credential from some organization, and subsequently "showing" the credential without revealing their identity. A stateful anonymous credential system adds the additional notion of credential state, which the user may update over the lifetime of the credential. State updates are restricted to some well-defined *policy* dictated by the credential provider. In practice, this may limit the user to a finite number of states, or a particular ordering of states that must be arrived at in succession. The update protocol for a stateful credential must oblivious. In other words, it does not leak information about the credential's current state beyond what the user chooses to reveal. As with typical anonymous credential systems, the user's state and other attributes can be proved without revealing her identity.

At a high level, the stateful anonymous credential system, which is defined by the tuple of algorithms $(\mathsf{Setup}, \mathsf{ObtainCred}, \mathsf{UpdateCred}, \mathsf{ProveCred})$, operates as follows. First, the user and credential provider negotiate the use of a specified policy using the $\mathsf{ObtainCred}$ protocol. The negotiated policy determines the way in which the user will be allowed to update her credential. After the protocol completes, the user receives an anonymous credential that embeds her initial state in the policy, in addition to any other user attributes. Next, the user can prove (in zero-knowledge) that the credential she holds embeds a given state, or attribute, just as she would in other anonymous credential systems by using the $\mathsf{ProveCred}$ protocol. This allows the anonymous access to some service, while the entity checking the credential is assured of the user's attributes, as well as her state in the specified policy – in some cases, as we will show later, these proof can be done in such a way that the verifying entity learns nothing about the user's state or attributes. Finally, when the user wishes to update her credential to reflect a change in her state, she interacts with the credential provider using the $\mathsf{UpdateCred}$ protocol, during which she proves (again, in zero-knowledge) her current state and the existence of a transition in the policy from her current state to her intended next state. As with the $\mathsf{ProveCred}$ protocol, the provider learns nothing about the user other than the fact that her state change is allowed by the policy that was previously negotiated within the $\mathsf{ObtainCred}$ protocol.

Policy Model. To represent the policies for our stateful anonymous credential system, we use directed graphs, which can be thought of as a state machine that describes the user's behavior over time. We describe the *policy graph* Π_{pid} as the set of *tags* of the form $(\mathsf{pid}, S \rightarrow T)$, where pid is the identity of the policy and $S \rightarrow T$ represents a directed edge from state S to state T. Thus, the user's credential embeds the identity of the policy pid and the user's current state in the policy graph. When the user updates her credential, she chooses a tag, then proves that the policy id she

is following is the same as what is provided in the tag and that the tag encodes an edge from her current state to her desired next state.

These policy graphs can be created in such a way that the users may reach a terminal state, and therefore would be unable to continue updating (and consequently using) their credential. In this case, it may be possible for an adversary to perform traffic analysis to infer the policy that the user is following. To prevent this, we consider the use of *null transitions* in the graph. The null transitions occur as self-loops on the terminal states of the policy graph, and allow the user to update her credential as often as she wishes to prevent such traffic analysis attacks. However, the updates performed on these credentials only allow the user access to a predefined null resource. The specifics of this null resource are dependent on the anonymous protocol that the credential system is coupled with, and we describe an implementation for them in oblivious databases in Section 5.

While these policy graphs are rather simplistic, they can represent complicated policies. For instance, a policy graph can encode the user's history with respect to accessing certain resources up to the largest cycle in the graph. Moreover, we can extend the policy graph tags to include auxiliary information about the actions that the user is allowed to perform at each state. By doing so, we allow the graph to dynamically control the user's access to various resources according to her behavior and history, as well as her other attributes. In Section 5, we examine how to extend these policy graphs to provide non-trivial, real-world access control policies for oblivious databases, as well as a variety of other anonymous and oblivious application.

2.1 Protocol Descriptions and Definitions for Stateful Anonymous Credentials

A stateful anonymous credential scheme consists of four protocols: Setup, ObtainCred, UpdateCred, and ProveCred. We will now describe their input/output behavior and intended functionality.

Setup($\mathcal{U}(1^k), \mathcal{P}(1^k, \Pi_1, \ldots, \Pi_n)$)**:** The provider $\mathcal{P}$ generates parameters *params* and a keypair $(pk_{\mathcal{P}}, sk_{\mathcal{P}})$ for the credential scheme. For each graph Π to be enforced, $\mathcal{P}$ also generates a cryptographic representation $\Pi_{\mathcal{C}}$ and publishes this value via an authenticated channel. Each user $\mathcal{U}$ generates a keypair and requests that it be certified by a trusted CA.

ObtainCred($\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{P}(pk_{\mathcal{U}}, sk_{\mathcal{P}}, \Pi_{\mathcal{C}}, S)$)**:** $\mathcal{U}$ identifies herself to $\mathcal{P}$ and then receives her credential Cred which binds her to a policy graph Π and starting state S.

UpdateCred($\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}, \Pi_{\mathcal{C}}, T), \mathcal{P}(sk_{\mathcal{P}}, \Pi_{\mathcal{C}}, D)$)**:** $\mathcal{U}$ and $\mathcal{P}$ interact such that Cred is updated from its current state to state T, but only if this transition is permitted by the policy Π. Simultaneously, $\mathcal{P}$ should not learn $\mathcal{U}$'s identity, attributes, or current state. To prevent replay attacks, $\mathcal{P}$ maintains a database D, which it updates as a result of the protocol.

ProveCred($\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}), \mathcal{P}(pk_{\mathcal{P}}, E)$)**:** $\mathcal{U}$ proves possession of a credential Cred in a particular state. To prevent re-use of credentials, $\mathcal{P}$ maintains a database E, which it updates as a result of the protocol.

A note on our model: In a traditional anonymous credential scheme (*e.g.,* [15]), the user may "show" its credential to many different organizations. We have simplified our protocol descriptions to reflect the assumption that a user need only show its credential to the original credential issuer. This model is sufficient for the applications we consider. We note that our credentials also function in the multi-organization model.

Security Definitions. Security definitions for anonymous credentials have traditionally been game-based. Unfortunately, the existing definitions may be insufficient for the applications considered in this work, as these definitions do not necessarily capture *correctness*. This can lead

to problems when we integrate our credential system with oblivious transfer protocols (see *e.g.,* [36, 20]). To capture the security requirements needed for our applications, we instead use a simulation-based definition, in which security of our protocols is analyzed with respect to an "ideal world" instantiation. We do not require security under concurrent executions, but rather restrict our analysis to atomic, sequential execution of each protocol. We do so because our constructions, which employ standard zero-knowledge techniques, require rewinding in their proof of security and thus are not concurrently secure. An advantage of the simulation paradigm is that our definitions will inherently capture correctness (*i.e.,* if parties honestly follow the protocols then they will each receive their expected outputs). Informally, the security of our system is encompassed by the following two definitions:

Provider Security: A malicious user (or set of colluding users) must not be able to falsely prove possession of a credential without first obtaining that credential, or arriving at it via an admissable sequence of credential updates. For our purposes, we require that the malicious user(s) cannot provide a proof of being in a state if that state is not present in her credential.

User Security: A malicious provider controlling some collection of corrupted users cannot learn any information about a user's identity or her state in the policy graph beyond what is available through auxiliary information from the environment.

Formalizing Definitions. Security for our protocols will be defined using the real-world/ideal-world paradigm, following the approach of [20]. In the real world, a collection of (possibly cheating) users interact directly with a provider according to the protocol, while in the ideal world the parties interact via a trusted party. Informally, a protocol is secure if, for every real-world cheating combination of parties we can describe an ideal-world counterpart ("simulator") who gains as much information from the ideal-world interaction as from the real protocol. We note that our definitions will naturally enforce both *privacy* and *correctness*, but not necessarily *fairness*. It is possible that $\mathcal{P}$ will abort the protocol *before* the user has completed updating her credential or accessing a resource. This is unfortunately unavoidable in a two-party protocol.

Definition 2.1 (Security for a Stateful Anonymous Credential Scheme) Full-simulation security for stateful anonymous credentials is defined according to the following experiments. Note that we do not explicitly specify auxiliary input to the parties, but this information can be provided in order to achieve sequential composition.

Real experiment. The real-world experiment $\mathbf{Real}_{\hat{\mathcal{P}},\hat{\mathcal{U}}_1,\ldots,\hat{\mathcal{U}}_\eta}(\eta, k, \Pi_1, \ldots, \Pi_\eta, \Sigma)$ is modeled as k rounds of communication between a possibly cheating provider $\hat{\mathcal{P}}$ and a collection of η possibly cheating users $\{\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_\eta\}$. In this experiment, $\hat{\mathcal{P}}$ is given the policy graph for each user $\Pi_1, \ldots, \Pi_\eta$, and the users are given an adaptive strategy Σ that, on input of the user's identity and current graph state, outputs the next action to be taken by the user.

At the beginning of the experiment, all users and the provider conduct the Setup procedure. At the end of this step, $\hat{\mathcal{P}}$ outputs an initial state P_1, and each user $\hat{\mathcal{U}}_i$ output state $U_{1,i}$. For each subsequent round $j \in [2, k]$, each user may interact with $\hat{\mathcal{P}}$ to update their credential as required by the strategy Σ. Following each round, $\hat{\mathcal{P}}$ outputs P_j, and the users output $(U_{1,j}, \ldots, U_{\eta,j})$. At the end of the k^{th} round the output of the experiment is $(P_k, U_{1,k}, \ldots, U_{\eta,k})$.

We will define the *honest* provider $\mathcal{P}$ as one that honestly runs its portion of Setup in the first round, honestly runs its side of the ObtainCred and ProveCred protocols when requested by a user at round $j > 1$, and outputs $P_k = params$. Similarly, an honest user $\mathcal{U}_i$ runs the Setup protocol honestly in the first round, and executes the user's side of the Setup, ObtainCred and ProveCred protocols, and eventually outputs the received value Cred along with all messages received.

Ideal experiment. In experiment $\mathbf{Ideal}_{\hat{\mathcal{P}}',\hat{\mathcal{U}}'_1,\ldots,\hat{\mathcal{U}}'_\eta}(\eta, k, \Pi_1, \ldots, \Pi\eta, \Sigma)$ the possibly cheating provider $\hat{\mathcal{P}}'$ sends the policy graphs to the trusted party $\mathcal{T}$. In each round $j \in [1, k]$, every user $\hat{\mathcal{U}}'_i$ (following strategy Σ) may send a message to $\mathcal{T}$ of the form $(\mathsf{update}, i, S_i, T_i)$ to update her credential using the UpdateCred protocol, or (prove, i, S_i) to prove her state using the ProveCred protocol.

- When $\mathcal{T}$ receives an update message, it checks $\hat{\mathcal{U}}'_i$'s current state and policy Π_i to determine whether the requested transition is allowed, setting a bit $b_{\mathcal{T}} = 1$ to so indicate. $\mathcal{T}$ sends $(\mathsf{update}, b_{\mathcal{T}})$ to $\hat{\mathcal{P}}'$, who responds with a bit $b_{\hat{\mathcal{P}}'} \in \{0, 1\}$ to $\mathcal{T}$ that indicates whether the update should succeed or fail. $\mathcal{T}$ returns $(b_{\hat{\mathcal{P}}'} \wedge b_{\mathcal{T}})$ to $\hat{\mathcal{U}}'_i$.
- For a prove message, $\mathcal{T}$ checks that $\hat{\mathcal{U}}'_i$ (setting $b_{\mathcal{T}}$ to so indicate), and relays $(\mathsf{prove}, S, b_{\mathcal{T}})$ to $\hat{\mathcal{P}}'$ who responds with a bit $b_{\hat{\mathcal{P}}'}$, and returns $(b_{\hat{\mathcal{P}}'} \wedge b_{\mathcal{T}})$ to $\hat{\mathcal{U}}'$.[1] Following each round, $\hat{\mathcal{P}}'$ outputs P_j, and the users output $(U_{1,j}, \ldots, U_{\eta,j})$. At the end of the k^{th} round the output of the experiment is $(P_k, V_j, U_{1,k}, \ldots, U_{\eta,k})$.

Let $\ell(\cdot), c(\cdot)$ be polynomially-bounded functions. We now define provider and user security in terms of the experiments above.

Provider Security. A stateful anonymous credential scheme is provider secure if for every collection of possibly cheating real-world p.p.t. receivers $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_\eta$ there exists a collection of p.p.t. ideal-world receivers $\hat{\mathcal{U}}'_1, \ldots, \hat{\mathcal{U}}'_\eta$ such that $\forall \eta = \ell(\kappa)$, $k \in c(\kappa)$, Σ, and every p.p.t. distinguisher:

$$\mathbf{Real}_{\mathcal{P},\hat{\mathcal{U}}_1,\ldots,\hat{\mathcal{U}}_\eta}(\eta, k, \Pi_1, \ldots, \Pi_\eta, \Sigma) \overset{c}{\approx} \mathbf{Ideal}_{\mathcal{P}l,\hat{\mathcal{U}}'_1,\ldots,\hat{\mathcal{U}}'_\eta}(\eta, k, \Pi_1, \ldots, \Pi_\eta, \Sigma)$$

User Security. A stateful anonymous credential scheme provides Receiver security if for every real-world p.p.t. provider $\hat{\mathcal{P}}$ who colludes with some collection of corrupted users, there exists a p.p.t. ideal-world provider $\hat{\mathcal{P}}'$ and users $\hat{\mathcal{U}}'$ such that $\forall \eta = \ell(\kappa)$, $k \in c(\kappa)$, Σ, and every p.p.t. distinguisher:

$$\mathbf{Real}_{\hat{\mathcal{P}},\mathcal{U}_1,\ldots,\mathcal{U}_\eta}(\eta, k, \Pi_1, \ldots, \Pi_\eta, \Sigma) \overset{c}{\approx} \mathbf{Ideal}_{\hat{\mathcal{P}}',\mathcal{U}'_1,\ldots,\mathcal{U}'_\eta}(\eta, k, \Pi_1, \ldots, \Pi_\eta, \Sigma)$$

3 Technical Preliminaries

We recall some basic building blocks, and then introduce a new primitive, *hidden range proofs*, which may be of independent interest. For the remainder of the paper, let 1^κ be the security parameter.

Pedersen and Fujisaki-Okamoto Commitments. In Pedersen commitments [39], the public parameters are a group $\mathbb{G}$ of prime order q, and generators $(g_0, \ldots, g_m)$. In order to commit to the values $(v_1, \ldots, v_m) \in \mathbb{Z}_q^m$, pick a random $r \in \mathbb{Z}_q$ and sets $C = \text{Commit}(v_1, \ldots, v_m; r) = g_0^r \prod_{i=1}^m g_i^{v_i}$. Fujisaki and Okamoto [31] showed how to expand this scheme to composite order groups.

Signatures with Efficient Protocols. Camenisch and Lysyanskaya (CL) [16] designed a signature scheme with two efficient protocols: (1) a protocol for a user to obtain a signature on the value(s) in a Pedersen (or Fujisaki-Okamoto) commitment [39, 31] without the signer learning anything about the message(s), and (2) a proof of knowledge of a signature. CL signatures are based

[1] Note that this reveals the current state S to $\hat{\mathcal{P}}'$. In section 5 we discuss techniques that also hide this information.

on the Strong RSA [2, 31] assumption. We could also use other bilinear signatures with efficient protocols [6, 17], though we do not make use of these in our construction.

Zero-Knowledge Protocols. We use several standard results for proving statements about discrete logarithms, such as (1) a proof of knowledge of a discrete logarithm modulo a prime [40] or a composite [31, 29], (2) a proof of knowledge of equality of representation modulo two (possibly different) prime [24] or composite [19] moduli, (3) a proof that a commitment opens to the product of two other committed values [18, 21, 10], and (4) a proof of the disjunction or conjunction of any two of the previous [27]. These composite-based protocols are secure under Strong RSA and the prime-based ones under the discrete logarithm assumption.

Note that there are several building blocks that are not used in our basic scheme, but which can be used to provide extended functionality or improved performance. These building blocks include:

Bilinear Groups. Let $\mathsf{BMsetup}$ be an algorithm that, on input 1^κ, outputs the parameters for a bilinear mapping as $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g \in \mathbb{G})$, where g generates $\mathbb{G}$, the groups $\mathbb{G}, \mathbb{G}_T$ each have prime order p, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Hidden-Range Proofs. Standard techniques [22, 18, 18, 9] allow us to efficiently prove that a committed value lies in a *public* integer interval (*i.e.,* where the interval is known to both the prover and verifier). In our protocols, we sometimes need to *hide* this interval from the verifier, and instead have the prover show that a committed value lies between the openings of two other commitments.

Fortunately, this can be done efficiently as follows. Suppose we wish to show that $a \leq j \leq b$, for positive numbers a, j, b without revealing them. This is equivalent to showing that $0 \leq (j - a)$ and $0 \leq (b - j)$. We only need to get these two sums reliably into commitments, and can then employ the standard techniques since the range (≥ 0) is now public. Using a group $\mathbf{G} = \langle \mathbf{g} \rangle$, where n is a special RSA modulus, $\mathbf{g}$ is a quadratic residue modulo n and $\mathbf{h} \in \mathbf{G}$. The prover commits to these values as $A = \mathbf{g}^a \mathbf{h}^{r_a}$, $J = \mathbf{g}^j \mathbf{h}^{r_j}$, and $B = \mathbf{g}^b \mathbf{h}^{r_b}$, for random values $r_a, r_j, r_b \in \{0,1\}^\ell$ where ℓ is a security parameter. The verifier next computes a commitment to $(j - a)$ as J/A and to $(b - j)$ as B/J. The prover and verifier then proceed with the standard public interval proofs with respect to these commitments, which for technical reasons require groups where Strong RSA holds.

4 Stateful Anonymous Credentials

We describe how to realize *stateful* credentials. The state records information about the user's *attributes* as well as her prior *access history*. We will consider two separate modes for "showing" a credential. In the first mode, the user exposes her portions of her state during the $\mathsf{ProveCred}$ protocol. This is useful for, say, a DRM application where the user's goal is to prove that her software is in a "licensed" state without revealing her name. In mode two, the user uses her credential to gain access to resources *without* revealing her state. Specifically, we show how to tie this credential system to a number of protocols, such as adaptive oblivious transfer and blind signatures, where the user wants to hide *both* her name and the item she is requesting, while simultaneously proving that she has the credentials to obtain the item.

Camenisch-Lysyanskaya Signatures. Our constructions may be implemented with the Strong RSA signature scheme of Camenisch and Lysyanskaya [16], or with the LRSW-based signatures of [17]. Both schemes consist of the algorithms ($\mathsf{CLKeyGen}, \mathsf{CLSign}, \mathsf{CLVerify}$) as well as two protocols, which we describe below. We first define the algorithms:

$\mathsf{CLKeyGen}(1^\kappa)$. On input a security parameter, outputs a keypair (pk, sk).

$\mathsf{CLSign}(sk, M_1, \ldots, M_n)$. On input one or more messages and a secret signing key, outputs the signature σ.

$\mathsf{CLVerify}(pk, \sigma, M_1, \ldots, M_n)$. On input a signature, message(s) and public verification key, outputs 1 if the signature verifies, 0 otherwise.

Additionally, the scheme consists of two protocols: (1) a protocol for a user to obtain a signature on the value(s) in a Pedersen (or Fujisaki-Okamoto) commitment [39, 31] without the signer learning anything about the message(s), and (2) a proof of knowledge of a signature.

In §5.2 we will use RSA-based CL signatures in conjunction with bilinear groups, *e.g.,* to prove knowledge of a CL signature on a commitment set in a bilinear group. These proofs can be conducted efficiently using techniques described in [13].

4.1 Basic Construction

Our construction begins with the anonymous credentials of Camenisch-Lysyanskaya [35, 16, 17], where the state is embedded as a field in the signature. The core innovation here is a protocol for performing state updates, and a technique for "translating" a history-dependent update policy into a cryptographic representation that can be used as an input to this protocol.

The setup, credential granting, and credential update protocols are presented in Figure 1. We will now briefly describe the intuition behind them.

Setup. First, the credential provider $\mathcal{P}$ generates its keypair and identifies one or more access policies it wishes to enforce. Each policy — encoded as a graph — may be applied to one or more users. The provider next "translates" the graph into a cryptographic representation which consists of the graph description, in addition to a separate CL signature corresponding to each tag in the graph, embedding the graph id, start, and end states. The files are distributed to users via an authenticated broadcast channel (*e.g.,* by signing and publishing them on a website).

A Note on Efficiency. It is important to emphasize that the "translation" of policy graphs may be conducted offline, and thus the cost of the online protocols (executed between user and provider) is *constant* and independent of the size of the policy. Furthermore, if many users share the same policy, this will further amortize the cost. Thus, our scheme is practical even for extremely complex policies containing thousands of distinct states and transition rules.

Obtaining a Credential. When a user $\mathcal{U}$ wishes to obtain a credential, he first generates a keypair that the CA certifies. He then negotiates with the provider to select an update policy to which the credential will be bound, as well the credential's initial state. The user next engages in a protocol to blindly extract a CL signature— under the provider's secret key— binding the user's public key, the initial state, policy id, and two random nonces chosen by the user: an *update nonce* N_u and a *usage nonce* N_s. The update nonce is revealed when the user updates the credential and the usage nonce is revealed when the user show's her credential. This signature, as well as the nonce and state information, form the credential. While the protocol for obtaining a credential, as currently described, reveals the user's identity through the use of her public key, we can apply the techniques found in [15, 16] to provide a randomized pseudonym rather than the public key.

Updating the Credential's State. When the user wishes to update a credential, she first identifies a valid tag within the credential's access policy. She then generates a new pair of nonces and a commitment embedding these values, as well as the new state. Next, the user sends the update nonce along with the commitment. The provider records this nonce and the commitment

into a database — however, if the nonce is already in the database but associated with a *different* commitment, the provider aborts the protocol, which prevents the user from re-using an old version of a credential. By recording the nonce and commitment together, we allow the user to restart the protocol if it has failed as long as she uses the same commitment. Otherwise, the user and provider then interact to conduct zero-knowledge proof that: (1) the remainder of this information is identical to the current credential, (2) the user has knowledge of the secret key corresponding to this credential, and (3) the policy graph contains a signature on a tag from the previous to the new state. If these conditions are met, the user obtains a new credential embedding the new state.

Showing (or Privately Proving Possession of) a Credential. The approach to using a single-show credential (Figure 2) follows [16, 17]. When a user wishes to prove possession of a $\mathcal{P}$ credential to $\mathcal{P}$, he first reveals the credential usage nonce and the current state of the credential. $\mathcal{P}$ must check that this nonce has not been used before. The user then proves knowledge of: (1) a CL signature embedding this state value and nonce formed under $\mathcal{P}$'s public key, and (2) a secret key that is consistent with the CL signature.

Single-show vs. multi-show. This is an example of a "single-show" credential. It can be shown only once, or the verifier will recognize the repeated usage nonce. To restore its anonymity, the user may return to $\mathcal{P}$ and execute the update protocol to replace the usage nonce. This update policy gives users a way to use a single credential multiple times. One can adapt this scheme to support k-times anonymous use by using the Dodis-Yampolskiy [30] pseudorandom function to generate the nonces from a common seed, as shown in [12].

Theorem 4.1 *When instantiated with the RSA (resp., bilinear) variant of CL signatures, the anonymous credential scheme above achieves user, provider, and verifier security (definition 2.1) under the strong RSA (resp., LRSW) assumption.*

Due to space constraints, we omit the proof of Theorem 4.1. However, the proof of Theorem 5.1 naturally includes the security of our credential system.

5 Oblivious Database Access Control

In this section we show how stateful anonymous credentials can be used to control access to *oblivious databases*. Recall that an oblivious database permits users to request data items without revealing their item choices to the database operator (*e.g.,* where the item choices are sensitive as in a medical databases).

Although we possess efficient building blocks such as k-out-of-N Oblivious Transfer (OT), little progress has been made towards the deployment of practical oblivious databases. In part, this is due to a fundamental tension with the requirements of a database operator to provide some form of access control. In this section, we show that it is possible to embed flexible, history dependent access controls into an oblivious database, without compromising the user's privacy. Specifically, we show how to combine our stateful anonymous credential system with an adaptive Oblivious Transfer protocol to construct a multi-user oblivious database that supports complex access control policies. We show how to *efficiently* couple stateful credentials with the recent standard-model adaptive OT construction due to Camenisch, Neven and shelat [20]. Our stateful credentials can also be efficiently coupled with the adaptive OT of Green and Hohenberger [33].

Linking Policies to Database Items. To support oblivious database access, we extend our policy graphs to incorporate *tags* of the form $(\mathsf{pid}, S \rightarrow T, i)$, where pid is the policy, $S \rightarrow T$ is the edge, and i is the message index that is allowed by that tag. Each edge in the graph may be

$\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{P}(1^k, \Pi_1, \ldots, \Pi_n))$**:** The provider $\mathcal{P}$ generates parameters for the CL signature, as well as for the Pedersen commitment scheme.

Party $\mathcal{P}$ runs $\mathsf{CLKeyGen}$ twice, to create the CL signature keypairs $(spk_{\mathcal{P}}, ssk_{\mathcal{P}})$ and $(gpk_{\mathcal{P}}, gsk_{\mathcal{P}})$. It retains $(pk_{\mathcal{P}}, sk_{\mathcal{P}}) = ((spk_{\mathcal{P}}, gpk_{\mathcal{P}}), (ssk_{\mathcal{P}}, gsk_{\mathcal{P}}))$ as its keypair. The provider's public key $pk_{\mathcal{P}}$ must be certified by a trusted CA.

Each party $\mathcal{U}$ selects $u \xleftarrow{\$} \mathbb{Z}_q$ and computes the keypair $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) = (g^u, u)$. The user's public key $pk_{\mathcal{U}}$ must be certified by a trusted CA.

Next, for each policy graph Π, $\mathcal{P}$ generates a cryptographic representation $\Pi_{\mathcal{C}}$.

1. $\mathcal{P}$ parses Π to obtain a unique policy identifier pid.
2. For each tag $t = (\mathsf{pid}, S, T)$ in Π, $\mathcal{P}$ computes a signature $\sigma_{S \to T} \leftarrow \mathsf{CLSign}(gsk_{\mathcal{P}}, (\mathsf{pid}, S, T))$.
3. $\mathcal{P}$ sets $\Pi_{\mathcal{C}} \leftarrow \langle \Pi, \forall t : \sigma_{S \to T} \rangle$ and publishes this value via an authenticated channel.

$\mathsf{ObtainCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{P}(pk_{\mathcal{U}}, sk_{\mathcal{P}}, \Pi_{\mathcal{C}}, S))$**:** On input a graph Π and initial state S, $\mathcal{U}$ first obtains $\Pi_{\mathcal{C}}$. $\mathcal{U}$ and $\mathcal{P}$ then conduct the following protocol:

1. $\mathcal{U}$ picks random show and update nonces $N_s, N_u \in \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N_s, N_u)$.
2. $\mathcal{U}$ conducts an interactive proof to convince $\mathcal{P}$ that A correlates to $pk_{\mathcal{U}}$.
3. $\mathcal{U}$ and $\mathcal{P}$ run the CL signing protocol on committed values so that $\mathcal{U}$ obtains the state signature $\sigma_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{P}}, (sk_{\mathcal{U}}, N_s, N_u, \mathsf{pid}, S))$ with pid, S contributed by $\mathcal{P}$.
4. $\mathcal{U}$ stores the credential $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s, N_u)$.

$\mathsf{UpdateCred}(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}, \Pi_{\mathcal{C}}, T), \mathcal{P}(sk_{\mathcal{P}}, \Pi_{\mathcal{C}}, D))$**:** Given a credential Cred currently in state S, $\mathcal{U}$ and $\mathcal{P}$ interact to update the credential to state T:

1. $\mathcal{U}$ parses $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s, N_u)$ and identifies a signature $\sigma_{S \to T}$ in $\Pi_{\mathcal{C}}$ that corresponds to a transition from state S to T (if none exists, $\mathcal{U}$ aborts).
2. $\mathcal{U}$ selects $N'_s, N'_u \xleftarrow{\$} \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, N'_u, \mathsf{pid}, T)$.
3. $\mathcal{U}$ sends (N_u, A) to $\mathcal{P}$. $\mathcal{P}$ looks in the database D for a pair $(N_u, A' \neq A)$. If no such pair is found, then $\mathcal{P}$ adds (N_u, A) to D. Otherwise $\mathcal{P}$ aborts.
4. $\mathcal{U}$ proves to $\mathcal{P}$ knowledge of values $(sk_{\mathcal{U}}, \mathsf{pid}, S, T, N'_s, N'_u, N_s, \sigma_{\mathsf{state}}, \sigma_{S \to T})$ such that:
   (a) $A = \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, N'_u, \mathsf{pid}, T)$.
   (b) $\mathsf{CLVerify}(spk_{\mathcal{P}}, \sigma_{\mathsf{state}}, (sk_{\mathcal{U}}, N_s, N_u, \mathsf{pid}, S)) = 1$.
   (c) $\mathsf{CLVerify}(gpk_{\mathcal{P}}, \sigma_{S \to T}, (\mathsf{pid}, S, T)) = 1$
5. If these proofs do not verify, $\mathcal{P}$ aborts. Otherwise $\mathcal{U}$ and $\mathcal{P}$ run the CL signing protocol on committed values to provide $\mathcal{U}$ with $\sigma'_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{P}}, A)$.
6. $\mathcal{U}$ stores the updated credential $\mathsf{Cred}' = (\Pi_{\mathcal{C}}, T, \sigma'_{\mathsf{state}}, N'_s, N'_u)$.

Figure 1: Basic algorithms for obtaining and updating a stateful anonymous credential.

ProveCred($\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \mathsf{Cred}), \mathcal{P}(pk_{\mathcal{P}}, E)$): User $\mathcal{U}$ proves knowledge of the Cred as follows:

1. $\mathcal{U}$ parses Cred as $(\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s, N_u)$, and sends its usage nonce N_s to $\mathcal{P}$ (who aborts if $N_s \in E$).
2. Otherwise, $\mathcal{U}$ continues with either:
   - (mode one) Sending her current credential state S to $\mathcal{P}$ in the clear.
   - (mode two) Sending a commitment to S.
3. $\mathcal{U}$ then conducts an interactive proof to convince $\mathcal{P}$ that it possesses a CL signature σ_{state} embedding N_s, S, and that it has knowledge of the secret key $sk_{\mathcal{U}}$.
4. $\mathcal{P}$ adds N_s to E.

Figure 2: Basic algorithm for proving knowledge of a single-show anonymous credential.

associated with one or more tags, which correspond to the items that can be obtained from the database when traversing that edge. As described in Section 2, we place null transitions on each terminal state that allow the user to update her credential and access a predefined null message. The set of all tags, both legitimate and null, are signed by the database and published. Figure 3 shows an example policy for a small database. The interested reader can view a complete discussion of some of the non-trivial access control policies allowed by our credential system in Appendix C.

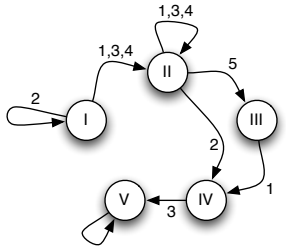

Figure 3: Sample access policy for a small oblivious database. The labels on each transition correspond to the database item indices that can be requested when a user traverses the edge, with null transitions represented by unlabeled edges.

5.1 Protocol Descriptions and Security Definitions for Oblivious Databases

Our oblivious database protocols combine the scheme of Section 4.1 with a multi-receiver oblivious transfer OT protocol. Each transaction is conducted between one of a collection of users and a single database server $\mathcal{D}$. We now describe the protocol specifications.

Setup($\mathcal{U}(1^k), \mathcal{D}(1^k, \Pi_1, \ldots, \Pi_n)$): The database $\mathcal{D}$ generates parameters *params* for the scheme. As in the basic credential scheme, it generates a cryptographic representation $\Pi_{\mathcal{C}}$ for each policy graph, and publishes those values via an authenticated channel. Each user $\mathcal{U}$ generates a keypair and requests that it be certified by a trusted CA.

OTObtainCred($\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{D}(pk_{\mathcal{U}}, sk_{\mathcal{D}}, \Pi_{\mathcal{C}}, S)$): $\mathcal{U}$ registers with the system and receives a credential Cred which binds her to a policy graph Π_{id} and starting state S.

OTAccessAndUpdate($\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \mathsf{Cred}, t), \mathcal{D}(sk_{\mathcal{D}}, E)$): $\mathcal{U}$ requests an item at index i in the database from state S by selecting a tag $t = (\mathsf{pid}, S \rightarrow T, i)$ from the policy graph. The user then up-

dates her credential Cred, in such a way that $\mathcal{D}$ does not learn her identity, her attributes, or her current state. Simultaneously, $\mathcal{U}$ obtains a message from the database at index i. At the end of a successful protocol, $\mathcal{U}$ updates the state information in Cred, and $\mathcal{D}$ updates a local datastore E.

Security. We informally describe the security properties of an oblivious database system, with a formal definition given in Appendix A.

Database Security: No (possibly colluding) subset of corrupted users can obtain any collection of items that is not specifically permitted by the users' policies.

User Security: A malicious database controlling some collection of corrupted users cannot learn any information about a user's identity or her state in the policy graph, beyond what is available through auxiliary information from the environment.

5.2 The Construction

In our model, many users share access to a single database. To construct our protocols, we extend the basic credential scheme of Section 4.1 by linking it to the adaptive OT protocol of Camenisch *et al.* [20]. The database operator commits to a collection of N messages, along with a special *null* message at index $N + 1$. It them distributes these commitments (*e.g.,* via a website). Each user then registers with the database using the OTObtainCred protocol, and agrees to be bound by a policy that will control her ability to access the database.

To obtain items from the database, the user runs the OTAccessAndUpdate protocol, which proves (in zero knowledge) that its request is consistent with its policy. Provided the user does not violate policy, the user is assured that the database operator learns nothing about its identity, or the nature of its request. Figure 4 describes the protocol.

Theorem 5.1 *The scheme described in Figure 4 satisfies definition A.1 under the q-PDDH, q-SDH, and Strong RSA assumptions.*

We provide a proof of Theorem 5.1 in Appendix B.

5.3 Extensions to Compact Access Policies in Practice

Extension #1: Equivalence Classes. In the scheme presented thus far, a tag in the policy graph must be defined on every item index in the database. However, there are cases where many items may have the same access rules applied, and therefore we can reduce the number tags used by referring to the entire group with a single tag. A simple solution is to replace specific item indices with general equivalence classes in the graph tags. The OT database can be easily re-organized to support this concept by renumbering the item indices (previously $[1, N]$) using values of the form $(c||i) \in \mathbb{Z}_q$ where c is the identity of the item class, and $||$ represents concatenation. During the OTAccessAndUpdate protocol, $\mathcal{U}$ can obtain any item $(c||i)$ by performing a zero-knowledge proof on the first half of the selection index, which shows that the user's selected tag contains the class c.

Extension #2: Encoding Contiguous Ranges. An alternative approach requires the database operator to arrange the identities of objects in the same class so that they fall in contiguous ranges. In this case, we will label the graph edges with *ranges* of items rather than single values. The credentials will also replace the value i with an upper and lower bound for the range that the holder of the credential is permitted to access. We make a slight change to the OTAccessAndUpdate protocol so that rather than proving equality between the requested object and the object present

$\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{D}(1^k))$**:** When the database operator $\mathcal{D}$ is initialized with a database of messages $M_1, \ldots, M_{N+1}$, it conducts the following steps:

1. $\mathcal{D}$ selects parameters for the OT scheme as $\gamma = (q, \mathbb{G}, \mathbb{G}_T, e, g) \leftarrow \mathsf{BMsetup}(1^\kappa)$, $h \xleftarrow{\$} \mathbb{G}$, $x \xleftarrow{\$} \mathbb{Z}_q$, and $H \leftarrow e(g, h)$. $\mathcal{D}$ generates two CL signing keypairs $(spk_{\mathcal{D}}, ssk_{\mathcal{D}})$ and $(gpk_{\mathcal{D}}, gsk_{\mathcal{D}})$, and $\mathcal{U}$ generates her keypair $(pk_{\mathcal{U}}, sk_{\mathcal{U}})$ as in the Setup protocol of Figure 1.
2. For $i = 1$ to $(N + 1)$, $\mathcal{D}$ computes a ciphertext $C_i = (A_i, B_i)$ as:
   (a) If $i \leq N$, then $A_i = g^{\frac{1}{x+i}}$ and $B_i = e(h, A_i) \cdot M_i$.
   (b) If $i = (N + 1)$, compute A_i as above and set $B_i = e(h, A_i)$.
3. For every graph Π to be enforced, $\mathcal{D}$ generates a cryptographic representation $\Pi_{\mathcal{C}}$ as follows:
   (a) $\mathcal{D}$ parses Π to obtain a unique policy identifier pid.
   (b) For each tag $t = (\mathsf{pid}, S, T, i)$ with $i \in [1, N + 1]$, $\mathcal{D}$ computes the signature $\sigma_{S \to T,i} \leftarrow \mathsf{CLSign}(gsk_{\mathcal{P}}, (\mathsf{pid}, S, T, i))$. Finally, $\mathcal{D}$ sets $\Pi_{\mathcal{C}} \leftarrow \langle \Pi, \forall t : \sigma_{S \to T,i} \rangle$.
4. $\mathcal{D}$ sets $pk_{\mathcal{D}} = (spk_{\mathcal{D}}, gpk_{\mathcal{D}}, \gamma, H, g^x, C_1, \ldots, C_n)$ and $sk_{\mathcal{D}} = (ssk_{\mathcal{D}}, gsk_{\mathcal{D}}, h)$. $\mathcal{D}$ then publishes each $\Pi_{\mathcal{C}}$ and the OT parameters $pk_{\mathcal{D}}$ via an authenticated channel.

$\mathsf{OTObtainCred}(\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \Pi_{\mathcal{C}}), \mathcal{D}(pk_{\mathcal{U}}, sk_{\mathcal{D}}, \Pi_{\mathcal{C}}, S))$**:** When user $\mathcal{U}$ wishes to join the system, it negotiates with $\mathcal{D}$ to agree on a policy Π and initial state S, then:

1. $\mathcal{U}$ picks a random show nonce $N_s \in \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N_s)$.
2. $\mathcal{U}$ conducts an interactive proof to convince $\mathcal{D}$ that A correlates to $pk_{\mathcal{U}}$, and $\mathcal{D}$ conducts an interactive proof of knowledge to convince $\mathcal{U}$ that $e(g, h) = H$.[a]
3. $\mathcal{U}$ and $\mathcal{P}$ run the CL signing protocol on committed values so that $\mathcal{U}$ obtains the state signature $\sigma_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{P}}, (sk_{\mathcal{U}}, N_s, \mathsf{pid}, S))$ with pid, S contributed by $\mathcal{P}$.
4. $\mathcal{U}$ stores the credential $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s)$.

$\mathsf{OTAccessAndUpdate}(\mathcal{U}(pk_{\mathcal{D}}, sk_{\mathcal{U}}, \mathsf{Cred}, t), \mathcal{D}(pk_{\mathcal{D}}, E))$**:** When $\mathcal{U}$ wishes to obtain the message indexed by $i \in [1, N + 1]$, it first identifies a tag t in Π such that $t = (\mathsf{pid}, S \to T, i)$.

1. $\mathcal{U}$ parses $\mathsf{Cred} = (\Pi_{\mathcal{C}}, S, \sigma_{\mathsf{state}}, N_s)$, and parses $\Pi_{\mathcal{C}}$ to find $\sigma_{S \to T,i}$.
2. $\mathcal{U}$ selects $N'_s \xleftarrow{\$} \mathbb{Z}_q$ and computes $A \leftarrow \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, \mathsf{pid}, T)$.
3. $\mathcal{U}$ then sends N_s to $\mathcal{D}$. $\mathcal{D}$ checks the database E for $(N_s, A' \neq A)$, and if it finds such an entry it aborts. Otherwise it adds (N_s, A) to E.
4. $\mathcal{U}$ parses $C_i = (A_i, B_i)$. It selects a random $v \leftarrow \mathbb{Z}_q$ and sets $V \leftarrow (A_i)^v$. It sends V to $\mathcal{D}$ and proves knowledge of $(i, v, sk_{\mathcal{U}}, \sigma_{S \to T,i}, \sigma_{\mathsf{state}}, \mathsf{pid}, S, T, N'_s)$ such that the following conditions hold:
   (a) $e(V, y) = e(g, g)^v e(V, g)^{-i}$.
   (b) $A = \mathrm{Commit}(sk_{\mathcal{U}}, N'_s, \mathsf{pid}, T)$.
   (c) $\mathsf{CLVerify}(spk_{\mathcal{P}}, \sigma_{\mathsf{state}}, (sk_{\mathcal{U}}, N_s, \mathsf{pid}, S)) = 1$.
   (d) $\mathsf{CLVerify}(\mathcal{P}, \sigma_{S \to T,i}, (\mathsf{pid}, S, T, i)) = 1$.
5. If these proofs verify, $\mathcal{U}$ and $\mathcal{D}$ run the CL signing protocol on committed values such that $\mathcal{U}$ obtains $\sigma'_{\mathsf{state}} \leftarrow \mathsf{CLSign}(ssk_{\mathcal{D}}, A)$. $\mathcal{U}$ stores the updated credential $\mathsf{Cred}' = (\Pi_{\mathcal{C}}, T, \sigma'_{\mathsf{state}}, N'_s)$.
6. Finally, $\mathcal{D}$ returns $U = e(V, h)$ and interactively proves that U is correctly formed (see [20]). $\mathcal{U}$ computes the message $M_i = U^{1/v}$.

[a] This proof can be conducted efficiently in four rounds as in [20].

Figure 4: An access-controlled oblivious database based on the Camenisch, Neven and shelat adaptive oblivious transfer protocol [20]. The database operator and users first run the Setup portion of the protocol. Each user subsequently registers with the database using $\mathsf{OTObtainCred}$, and requests individual database items using $\mathsf{OTAccessAndUpdate}$.

in the tag, the user now proves that the requested object lies in the range described in the user selected tag, as described by the hidden range proof technique in Section 3. Notice that while this approach requires that the database be reorganized such that classes of items remain in contiguous index ranges, it can be used to represent more advanced data structures, such as hierarchical classes.

6 Other Applications of Stateful Anonymous Credentials

Oblivious IBE Key Extraction. Identity-Based Encryption (*e.g.,* [8, 26]) is a form of public-key encryption where users can substitute an arbitrary string— for example, a name or email address— in place of a traditional public key. In an IBE deployment, the corresponding decryption keys are generated by a trusted party known as the Private Key Generator (PKG).

Under normal circumstances, the user cannot hide its identity from the PKG. Indeed, this can be problematic, since the PKG must verify that a user is authorized to obtain a key for a given identity. In some anonymous communication scenarios, however, it can be desirable to anonymously grant temporary decryption keys to users without learning the user's identity.

Green and Hohenberger [33] propose a means by which a user can *blindly* extract a decryption key from a PKG, such that the PKG does not learn the identity extracted. These techniques can also be extended to allow for partially-blind extraction, where a portion of the identity is known to the PKG, which is useful when keys also embed some known, restricted information, such as the time period during which they will be valid. Unfortunately, these techniques deprive the PKG of the ability to control which keys are given out. Using our stateful anonymous credential system, we can realize efficient solutions for blind, yet controlled, access to the IBE keys for the Boneh-Boyen IBE [5] and the Waters IBE [44].

Oblivious (Blind) Signatures. As observed by Moni Naor, there is a connection between decryption keys in IBE schemes and digital signatures.[2] Specifically, the decryption key corresponding to an identity pid in any full-secure IBE scheme is a signature on the message pid where the signature verification key is the master public key of the IBE scheme. Thus, the blind key extraction protocol for the Waters IBE [44] is also a blind signature scheme for the Waters signature. Fortunately, we can put efficient access controls on top of this, as well.

Imagine several scenarios in which this is truly exciting: a signer can now specify a policy under which he is willing to blindly sign messages, and then can enforce this policy without violating any of the user's privacy or even learning her identity. This leads to practical data timestamping services (*e.g.,* [43, 41]) that do not learn anything about what a user is signing, or even who originated a specific request. Alternatively, blind signatures can be useful for forensic purposes: a device can be required to obtain a signature each time it undertakes a controversial action, and use these signatures to convince a later investigator that each action was in fact allowed by policy. Additionally, our access controls can also be placed onto the blind signing protocols of the Strong RSA [16] and bilinear [17] signatures of Camenisch and Lysyanskaya, as well as the short bilinear signatures of Boneh and Boyen [6]. These are all schemes secure in the standard model.

Oblivious Keyword Search. IBE key extraction can also be used to implement public-key searchable encryption [38, 7, 45], which permits users to search a collection of encrypted files for those matching a particular keyword. For example, Waters *et al.* [45] describe a searchable encrypted audit log in which a third party auditor is granted the ability to independently search the encrypted log for specific keywords. In these schemes, the scope of the user's searches is

[2] This observation was credited to Naor by Boneh and Franklin [8].

generally limited by a trusted authority, which generates "search trapdoors" for particular words at the searcher's request. Unfortunately, this trusted party necessarily learns the details of each search term, which may be problematic in circumstances where the pattern of trapdoor requests reveals sensitive information. Using the blind key extraction techniques described above, Green and Hohenberger [33] discuss how an authority can blindly deliver search trapdoors without learning which terms are being monitored. Again, our techniques can help regulate which key word searches are allowed.

References

[1] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, volume 2045, pages 119–135, 2001.

[2] Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97*, volume 1233 of LNCS, pages 480–494, 1997.

[3] D. Elliot Bell and Leonard J. LaPadula. Secure Computer System: Unified Exposition and Multics Interpretation. *Comm. of the ACM*, 1:271–280, 1988.

[4] I. F. Blake and V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals. *ASIACRYPT '04*, 3329 of LNCS:515–529, 2004.

[5] Dan Boneh and Xavier Boyen. Efficient selective-ID secure Identity-Based Encryption without random oracles. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 223–238, 2004.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *EUROCRYPT '04*, volume 3027, pages 382–400, 2004.

[7] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 506–522, 2004.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *CRYPTO '01*, volume 2139 of LNCS, pages 213–229, 2001.

[9] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT '00*, volume 1807 of LNCS, pages 431–444, 2000.

[10] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, volume 1233 of LNCS, pages 318–333, 1997.

[11] David F. C. Brewer and Michael J. Nash. The Chinese Wall Security Policy. In *IEEE Symposium on Security and Privacy*, pages 206–214, 1989.

[12] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In *ACM CCS '06*, pages 201–210, 2006.

[13] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT '05*, volume 3494 of LNCS, pages 302–321, 2005.

[14] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash. In *SCN '06*, volume 4116 of LNCS, pages 141–155, 2006.

[15] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *EUROCRYPT '01*, volume 2045 of LNCS, pages 93–118. Springer, 2001.

[16] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in Communication Networks '02*, volume 2576, pages 268–289, 2002.

[17] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO '04*, volume 3152, pages 56–72, 2004.

[18] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number n is the product of two safe primes. In *EUROCRYPT '99*, volume 1592 of LNCS, pages 107–122, 1999.

[19] Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In *CRYPTO '99*, volume 1666, pages 413–430, 1999.

[20] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, volume 4515, pages 573–590, 2007.

[21] Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998.

[22] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come – easy go divisible cash. In *EUROCRYPT '98*, volume 1403 of LNCS, pages 561–575, 1998.

[23] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[24] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *CRYPTO*, volume 740 of LNCS, pages 89–105, 1992.

[25] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[26] Clifford Cocks. An identity based encryption scheme based on Quadratic Residues. In *Cryptography and Coding, IMA Intl. Conference*, volume 2260 of LNCS, pages 360–363, 2001.

[27] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, volume 839, pages 174–187, 1994.

[28] Giovanni Di Crescenzo, Rafail Ostrovsky, and S. Rajagopolan. Conditional oblivious transfer and time released encryption. In *EUROCRYPT '99*, volume 1592, pages 74–89, 1999.

[29] Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In *ASIACRYPT '02*, volume 2501, pages 125–142, 2002.

[30] Yevgeniy Dodis and Aleksandr Yampolskiy. A Verifiable Random Function with Short Proofs an Keys. In *Public Key Cryptography*, volume 3386 of LNCS, pages 416–431, 2005.

[31] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO*, volume 1294, pages 16–30, 1997.

[32] Google. Google Health. `http://www.google.com/intl/en-US/health/about/index.html`, 2008.

[33] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT '07*, volume 4833, pages 265–282, 2007.

[34] Butler W. Lampson. Dynamic Protection Structures. In *AFIPS Conference*, volume 35, pages 27–38, 1969.

[35] Anna Lysyanskaya. *Signature schemes and applications to cryptographic protocol design.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 2002.

[36] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO '99*, volume 1666 of LNCS, pages 573–590, 1999.

[37] Department of Defense. Trusted Computer System Evaluation Criteria. Technical Report DoD 5200.28-STD, Department of Defense, Dec 1985.

[38] Wakaha Ogata and Kaoru Kurosawa. Oblivious keyword search. *Special issue on coding and cryptography J. of Complexity*, 20(2-3):356–371, 2004.

[39] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, volume 576, pages 129–140, 1992.

[40] Claus-Peter Schnorr. Efficient signature generation for smart cards. *J. of Cryptology*, 4(3):239–252, 1991.

[41] Surety, LLC. Surety LLC. `http://www.surety.com/`.

[42] I. Teranishi, J. Furukawa, and K. Sako. k-Times Anonymous Authentication. In *ASIACRYPT 2004*, volume 3329, pages 308–322, 2004.

[43] Verisign. Verisign Code Signing for Microsoft Authenticode Technology. `http://www.verisign.com/static/030999.pdf`, 2005.

[44] Brent Waters. Efficient Identity-Based Encryption without random oracles. In *EUROCRYPT '05*, volume 3494 of LNCS, pages 114–127, 2005.

[45] Brent R. Waters, Dirk Balfanz, Glenn Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *NDSS'04*, 2004.

A Security Definitions for Oblivious Databases with Access Controls

Our security definition for an oblivious database extends Definition 2.1 by incorporating the concept of a message database $M_1, \ldots, M_N$ held by the database $\mathcal{D}$.

Definition A.1 (Security for Oblivious Databases with Access Controls) Security is defined according to the following experiments. As before, we do not explicitly specify auxiliary input to the parties, but this information can be provided in order to achieve sequential composition.

Real experiment. The real-world experiment $\mathbf{Real}_{\hat{\mathcal{D}},\hat{\mathcal{U}}_1,\ldots,\hat{\mathcal{U}}_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi_\eta, M_1, \ldots, M_N, \Sigma)$ is modeled as k rounds of communication between a possibly cheating database $\hat{\mathcal{D}}$ and a collection of η possibly cheating users $\{\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_\eta\}$. In this experiment, $\hat{\mathcal{D}}$ is given the policy graph for each user $\Pi_1, \ldots, \Pi_\eta$, a message database $M_1, \ldots, M_N$ and the users are given an adaptive strategy Σ that, on input of the user's identity and current graph state, outputs the next action to be taken by the user.

At the beginning of the experiment, the database and users conduct the Setup and OTObtainCred protocols. At the end of this step, $\hat{\mathcal{D}}$ outputs an initial state S_1, and each user $\hat{\mathcal{U}}_i$ output state $U_{1,i}$. For each subsequent round $j \in [2, k]$, each user may interact with $\hat{\mathcal{D}}$ to request an item i as required by the strategy Σ. Following each round, $\hat{\mathcal{D}}$ outputs S_j, and the users output $(U_{1,j}, \ldots, U_{\eta,j})$. At the end of the k^{th} round the output of the experiment is $(S_k, U_{1,k}, \ldots, U_{j,k})$.

We will define the *honest* database $\mathcal{D}$ as one that honestly runs its portion of Setup in the first round, honestly runs its side of the OTObtainCred and OTAccessAndUpdate protocols when requested by a user at round $j > 1$, and outputs $S_k = params$. Similarly, an honest user $\mathcal{U}_i$ runs the Setup protocol honestly in the first round, and executes the user's side of the OTObtainCred, OTAccessAndUpdate protocols, and eventually outputs the received value Cred along with all messages received.

Ideal experiment. In experiment $\mathbf{Ideal}_{\hat{\mathcal{D}}',\hat{\mathcal{U}}'_1,\ldots,\hat{\mathcal{U}}'_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi\eta, M_1, \ldots, M_N, \Sigma)$ the possibly cheating database $\hat{\mathcal{D}}'$ sends the policy graphs to the trusted party $\mathcal{T}$. In each round $j \in [1, k]$, every user $\hat{\mathcal{U}}'$ (following strategy Σ) selects a message index $i \in [1, N+1]$ and sends a message containing the user's identity and (i, S, T) to $\mathcal{T}$. $\mathcal{T}$ then checks the policy graph corresponding to that user to determine if the action is permitted, and sends $\hat{\mathcal{D}}'$ a bit b_1 indicating the outcome of this test. $\hat{\mathcal{D}}'$ then returns a bit b_2 determining whether the transaction should succeed. If $b_1 \wedge b_2$, then $\mathcal{T}$ returns M_i to $\hat{\mathcal{U}}'_i$, otherwise it returns $\perp$. Following each round, $\hat{\mathcal{D}}'$ outputs P_j, and the users output $(U_{1,j}, \ldots, U_{\eta,j})$. At the end of the k^{th} round the output of the experiment is $(P_k, U_{1,k}, \ldots, U_{\eta,k})$.

Let $\ell(\cdot), c(\cdot)$ be polynomially-bounded functions. We now define database and user security in terms of the experiments above.

Database Security. A stateful anonymous credential scheme is database-secure if for every collection of real-world p.p.t. receivers $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_\eta$ there exists a collection of p.p.t. ideal-world receivers $\hat{\mathcal{U}}'_1, \ldots, \hat{\mathcal{U}}'_\eta$ such that $\forall N = \ell(\kappa)$, $N = d(\kappa)$, $k \in c(\kappa)$, PF, Σ, and every p.p.t. distinguisher:

$$\mathbf{Real}_{\mathcal{D},\hat{\mathcal{U}}_1,\ldots,\hat{\mathcal{U}}_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi_\eta, M_1, \ldots, M_N, \Sigma) \overset{c}{\approx} \mathbf{Ideal}_{\mathcal{D},\hat{\mathcal{U}}'_1,\ldots,\hat{\mathcal{U}}'_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi_\eta, M_1, \ldots, M_N, \Sigma)$$

User Security. A stateful anonymous credential scheme provides Receiver security if for every real-world p.p.t. database $\hat{\mathcal{D}}$ and collection of dishonest users, there exists a p.p.t. ideal-world sender $\hat{\mathcal{D}}'$ such that $\forall N = \ell(\kappa)$, $\eta = d(\kappa)$, $k \in c(\kappa)$, PF, Σ, and every p.p.t. distinguisher:

$$\mathbf{Real}_{\hat{\mathcal{D}},\mathcal{U}_1,\ldots,\mathcal{U}_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi_\eta, M_1, \ldots, M_N, \Sigma) \overset{c}{\approx} \mathbf{Ideal}_{\hat{\mathcal{D}}',\mathcal{U}'_1,\ldots,\mathcal{U}'_\eta}(\eta, N, k, \Pi_1, \ldots, \Pi_\eta, M_1, \ldots, M_N, \Sigma)$$

B Proof Sketch of Theorem 5.1

We now sketch a proof of Theorem 5.1. Our sketch will refer substantially to the original proof of Camenisch *et al.* [20]. We note that our proof will consider two components: (1) the security of the

underlying OT scheme (which is based on the proof of [20]), and a separate proof of the anonymous credential scheme.

Proof sketch. Our sketch separately considers User and Database security.

User Security. Let us assume that an adversary has corrupted a database $\mathcal{D}$ and some subset of the users $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_N$. In this model, corruptions will be static. We show that for every such adversary, we can construct a simulator such that the output of the ideal experiment conducted with the simulator will be indistinguishable from the output of the real experiment.

Our simulator operates as follows. First, $\mathcal{D}$ outputs the parameters for the credential system, the cryptographic representation of each graph, and $pk, C_1, \ldots, C_N$. If these parameters are incorrectly formed, the simulator aborts. The simulator next generates a credential key for each uncorrupted user and negotiates with $\mathcal{D}$ to join the system under an appropriate policy. When $\mathcal{D}$ executes the proof of knowledge that $H = e(g, h)$ with some uncorrupted user, our simulator rewinds to extract the value h (this extraction succeeds with all but negligible probability). For $i = 1$ to N, the simulator decrypts C_i using h to obtain M_i. This collection of plaintexts is sent to the trusted party $\mathcal{T}$.

Whenever an uncorrupted user queries $\mathcal{T}$ to obtain message i (according to a state transition defined in their policy), $\mathcal{T}$ verifies that this request is permitted by policy and updates its view of the user's state. Next, it notifies our simulator which runs the OTAccessAndUpdate protocol on an arbitrary (uncorrupted) user's policy under index $N + 1$ (this is the "dummy" transition and is always permitted by the credential system). If this protocol succeeds, the simulator sends a bit 1 to $\mathcal{T}$ which returns M_i to the user.

Claim. The transcript produced by this simulator is indistinguishable from the transcript produced by the real experiment. This is true for following reasons:

1. The probability that the simulator *incorrectly* extracts h (or fails to extract it) is negligible.
2. The probability that the adversary distinguishes a protocol executed on an arbitrary user/dummy index is negligible: this is due to (a) the witness-indistinguishability property of the credential proofs of knowledge, and (b) the element V transmitted to $\mathcal{D}$ during OTAccessAndUpdate is indistinguishable from a random element.

Note that the we need not argue the unforgeability of the anonymous credential scheme here, since we consider only actions taken by the uncorrupted user.

Database Security. Let us assume that an adversary has corrupted some subset of the users $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_N$ (corruptions are static). We show that for every such adversary, we can construct a simulator such that the output of the ideal experiment conducted with the simulator will be indistinguishable from the output of the real experiment.

Our simulator operates as follows. First, it generates the public and privacy parameters for the credential scheme along with the cryptographic representation of the policies provided by $\mathcal{T}$. It generates the parameters for the OT scheme pk, sk as normal, but sets the plaintext for each database element to a dummy value (the identity element) and produces ciphertexts $C_1, \ldots, C_N$ (and generates the dummy message $C_{(N+1)}$ as normal). It sends these parameters to each corrupted user, and to each user proves that $H = e(g, h)$.

Whenever a corrupted user initiates the OTAccessAndUpdate protocol with $\mathcal{D}$, the simulator verifies that the user's request (including ZK proofs) verifies, and that neither N_u or N_s has been seen before. If so, it rewinds and uses the extractors for the ZK proofs to learn the user's identity, the index of the message i being requested, the blinding factor v, and the user's current and previous

credential state S, T. The server transmits the user's identity values (i, S, T) to $\mathcal{T}$ which verifies that they satisfy the policy (updating the policy state in the process). If $\mathcal{T}$ returns $\perp$, then $\mathcal{D}$ aborts the protocol with the user. Otherwise if $\mathcal{T}$ returns M_i, then the simulator parses $C_i = (A_i, B_i)$ and returns $U = (B_i^v)/M_i$. The simulator uses rewinding to simulate the proof and convince the user that U has been correctly formed.

Claim. The transcript produced by this simulator is indistinguishable from the transcript produced by the real experiment. This claim rests on the following points:

1. The false message collection $C_1, \ldots, C_{(N+1)}$ is indistinguishable from the real message by the semantic security of the encryption scheme, which holds under the q-PDDH assumption (see [20] for the full argument).
2. The simulated proof of U's structure is indistinguishable from a correct real proof.
3. The simulator never queries $\mathcal{T}$ on a tuple (i, S, T) that violates the user's policy. This reduces to the unforgeability of the CL signature (which is in turn based on Strong RSA or LRSW). Specifically, to violate policy, a user must satisfy one of the following conditions:
   (a) Prove knowledge of a signature σ_δ that it was not given, or
   (b) Prove knowledge of a signature $\sigma_{S \to T}$ that it was not given. In either case, the simulator can use the extractor for the proof system to obtain the forged signature and win the CL signature forgery game.
   (c) Misuse the CL signing protocol such that it receives a signature that is not equivalent to a signature on the commitment A (or mispresent the structure of A).

□

C Access Control Models

A number of access control models can be used to describe access permissions for resources through the use of our stateful credential system and its extensions. The most widely used form of access controls are discretionary access control systems [37], where access permissions are applied arbitrarily as they are needed. For instance, a systems administrator can describe a list of resources that a given user can access, otherwise known as a capabilities list [34]. Such an access model is trivially achieved in our credential system as a separate graph for each user with a single state and a self loop with tags for each of the contiguous ranges of resources that the user can access.

Mandatory access control models, however, are far more interesting because of their use of the user's access history to enforce an access policy. These history-dependent access controls are difficult to capture with typical capabilities or access list implementations due to their dynamic nature. Here, we describe two non-trivial access control models used in real-world systems, the Brewer-Nash [11] and Bell-LaPadula [3] models, and provide an example policy graph for each in Figures 5 and 6, respectively.

Brewer-Nash Model. The Brewer-Nash model [11], otherwise known as the Chinese Wall, is a mandatory access control model that is widely used in the financial services industry to prevent an employee from working on the finances of two companies that are in competition with one another. Intuitively, the resources in the system are first divided into groups based on the company they are associated with, called *datasets*. These datasets are further grouped into *conflict of interest classes* such that all of the companies that are in competition with one another have their datasets in the same class. The model ensures that once a user chooses an object from a dataset in a given class,

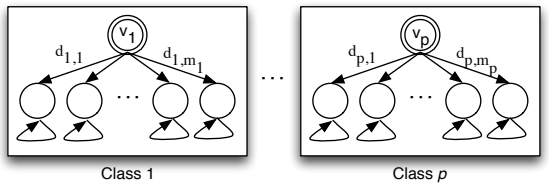

Figure 5: Example access graphs for the Brewer-Nash model. The user receives one access graph per class, where each access graph allows access to at most one of the datasets $d_{i,j}$ for the associated conflict of interest class i.

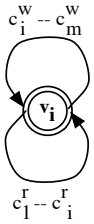

Figure 6: Example access graph for a user with security level i in the Bell-LaPadula model. The graph allows read access to all resources in classes c_1^r through c_i^r and write access to all objects in classes c_i^w to c_m^w.

that user has unrestricted access to all objects in the selected dataset, but no access to objects in any other dataset in that class. In Figure 5, we denote the j^{th} dataset in class i as $d_{i,j}$, which we can succinctly represent in our access graphs using either the class label extension, or hidden range proof extension from Section 5.3.

Bell-LaPadula Model. Another well-known mandatory access control model is the Bell-LaPadula model [3], which is a Multilevel Security model. The Bell-LaPadula model is designed with the intent of maintaining data confidentiality in a classified computer system, and it is typically used in high security environments. In this security model, resources, and users are labeled with a security level (*e.g.,* top secret, secret, etc.). The security level labels are strictly ordered and provide a hierarchy that describes the sensitivity of information. The two basic properties of the Bell-LaPadula model state that a user cannot read a resource with a security level greater than her own, and she cannot write to resources with a security level less than her own. Therefore, the model ensures that information from highly sensitive objects cannot be written to low security objects by using the user as an intermediary. In Figure 6, we denote the security levels as the integers $1, \ldots, m$. Furthermore, we split the access tags into separate read and write access controls through the use of separate indices. Therefore, a user with security level i gets a graph with tags $c_i^w, \ldots, c_m^w$ that allow her to write to any resource with a higher security level, and tags $c_1^r, \ldots, c_i^r$ that allow her to read any resource with a lower security level. Again, these ranges of resources can be succinctly represented by the extensions of Section 5.3.