Controlling Access to an Oblivious Database Using Stateful Anonymous Credentials

Scott Coull, Matthew Green, and Susan Hohenberger

The Johns Hopkins University
Information Security Institute
3400 N. Charles Street; Baltimore, MD 21218, USA
{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. To do so, we show how to augment existing anonymous credential systems so that, in addition to certifying a user's attributes, they also store state about the user's database access history. 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. In addition, our system is based on standard assumptions in the standard model and, after an initial setup phase, each transaction requires only constant time.

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 operator 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 [27], makes these concerns all the more compelling.

Previous works have proposed to construct privacy-friendly databases using Private Information Retrieval [20] or Oblivious Transfer [30,15]. 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 enforceable access controls. We make three design choices that act as guiding principles for our system. Our first is to maintain all anonymity and 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, and thus 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 [18,11], 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 controls 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 [29,11,12,4]. 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.

Our constructions can be used to achieve non-trivial access control policies, including the Brewer-Nash (Chinese Wall) [7] and Bell-LaPadula (Multilevel Security) [2] model, which are used in a number of settings, including financial institutions and classified government systems. We discuss 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 user actions while maintaining privacy, 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 can be 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 [30,15]. The more general concept of conditional oblivious transfer was proposed by Di Crescenzo, Ostrovsky, and Rajagopalan [23] and subsequently strengthened by Blake and Kolesnikov [3]. 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 [8], authenticating anonymously at most k times [34], or preventing a user from exchanging too much money with a single merchant [9]. 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 added to a wide variety of anonymous and oblivious protocols.

2 Stateful Credentials: 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 an organization, and subsequently "showing" the credential without revealing their identity. In addition to the properties of typical credentials, a stateful anonymous credential system adds the additional notion of credential state, which is embedded as an attribute within the credential. The user may update the state in her credential according 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 sequential ordering of states. To maintain the user's anonymity, it is important that the update protocol not leak information about the credential's current state beyond what the user chooses to reveal.

At a high level, the stateful anonymous credential system, which is defined by the tuple of algorithms (Setup, ObtainCred, UpdateCred, ProveCred), operates as follows. First, the user and credential provider negotiate the use of a specified policy using the 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 other 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 ProveCred protocol. This allows the user anonymous access to services, while the entity

checking the credential is assured of the user's attributes, as well as her state in the specified policy. 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 UpdateCred protocol, to prove (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 ProveCred protocol, the provider learns nothing about the user other than the fact that her state change is allowed by the policy previously negotiated within the ObtainCred protocol.

Policy Model. To represent the policies for our stateful 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 and 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 Credentials

A stateful anonymous credential scheme consists of the four protocols: Setup, ObtainCred, UpdateCred, and ProveCred. We will now describe their input/output behavior and intended functionality. For the remainder of the paper, let 1^{κ} be the security parameter.

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($\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \text{Cred}, T), \mathcal{P}(sk_{\mathcal{P}}, D)$): $\mathcal{U}$ 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($\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \text{Cred}), \mathcal{P}(pk_{\mathcal{P}}, E)$): $\mathcal{U}$ 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.

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.,* [30,15]). 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 captured 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.

Due to space considerations, we defer the formal security definitions for stateful anonymous credentials to full version of this paper [21]. In Section 5.1 and Appendix A, we provide definitions for oblivious databases with access control.

3   Technical Preliminaries

In this section, we recall some basic building blocks, and then introduce a new primitive, *hidden range proofs*, which may be of independent interest.

Pedersen and Fujisaki-Okamoto Commitments. In the Pedersen commitment scheme [32], 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 $(v_1, \ldots, v_m) \in \mathbb{Z}_q^m$, the user picks a random $r \in \mathbb{Z}_q$ and sets $C = \mathrm{Commit}(v_1, \ldots, v_m; r) = g_0^r \prod_{i=1}^m g_i^{v_i}$. Fujisaki and Okamoto [26] provided a composite order variant.

Signatures with Efficient Protocols. Camenisch and Lysyanskaya (CL) [11] 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 [32,26] without the signer learning anything about the message(s), and (2) a proof of knowledge of a signature. Our constructions may be implemented with the Strong RSA signature scheme [11] (and with minor modifications, using bilinear signatures based on the LRSW assumption [12]). Both schemes consist of the algorithms (CLKeyGen, CLSign, CLVerify), which we describe below:

CLKeyGen(1^κ). On input a security parameter, outputs a keypair (pk, sk).
CLSign($sk, M_1, \ldots, M_n$). On input one or more messages and a secret signing key, outputs the signature σ.
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.

We could also use other bilinear signatures with efficient protocols (*e.g.,* [4]), 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 [33] or a composite [26,24], (2) a proof of knowledge of equality of representation modulo two (possibly different) prime [19] or composite [14] moduli, (3) a proof that a commitment opens to the product of two other committed values [13,16,6], and (4) a proof of the disjunction or conjunction of any two of the previous [22]. 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 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 [17,13,13,5] 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, it is useful 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

In this section, 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 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 through the use of zero knowledge proofs. Specifically, we show how to tie this credential system to protocols, such as adaptive oblivious transfer, where the user wants to hide both her identity and the item she is requesting while simultaneously proving that she has the credentials to obtain the item.

4.1   Basic Construction

Our construction begins with the anonymous credentials of Camenisch and Lysyanskaya [29,11,12], 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, and a separate CL signature for each tag in the graph. Recall from Section 2 that

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 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 $u \overset{\$}{\leftarrow} \mathbb{Z}_q$ 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 $t = (\text{pid}, S, T)$ in Π, $\mathcal{P}$ computes a signature $\sigma_{S \to T} \leftarrow$ CLSign$(gsk_{\mathcal{P}}, (\text{pid}, S, T))$.
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.

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 usage and update nonces $N_s, N_u \in \mathbb{Z}_q$ and computes $A \leftarrow \text{Commit}(sk_{\mathcal{U}}, N_s, N_u)$.
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 $\sigma_{\text{state}} \leftarrow$ CLSign$(ssk_{\mathcal{P}}, (sk_{\mathcal{U}}, N_s, N_u, \text{pid}, S))$ with pid, S contributed by $\mathcal{P}$.
4. $\mathcal{U}$ stores the credential Cred $= (\Pi_{\mathcal{C}}, S, \sigma_{\text{state}}, N_s, N_u)$.

UpdateCred$(\mathcal{U}(pk_{\mathcal{P}}, sk_{\mathcal{U}}, \text{Cred}, T), \mathcal{P}(sk_{\mathcal{P}}, D))$: 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 Cred $= (\Pi_{\mathcal{C}}, S, \sigma_{\text{state}}, N_s, N_u)$ 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 $N'_s, N'_u \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes $A \leftarrow \text{Commit}(sk_{\mathcal{U}}, N'_s, N'_u, \text{pid}, T)$.
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 $(sk_{\mathcal{U}}, \text{pid}, S, T, N'_s, N'_u, N_s, \sigma_{\text{state}}, \sigma_{S \to T})$ such that:
   (a) $A = \text{Commit}(sk_{\mathcal{U}}, N'_s, N'_u, \text{pid}, T)$.
   (b) CLVerify$(spk_{\mathcal{P}}, \sigma_{\text{state}}, (sk_{\mathcal{U}}, N_s, N_u, \text{pid}, S)) = 1$.
   (c) CLVerify$(gpk_{\mathcal{P}}, \sigma_{S \to T}, (\text{pid}, S, T)) = 1$
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 $\sigma'_{\text{state}} \leftarrow$ CLSign$(ssk_{\mathcal{P}}, A)$.
6. $\mathcal{U}$ stores the updated credential Cred' $= (\Pi_{\mathcal{C}}, T, \sigma'_{\text{state}}, N'_s, N'_u)$.

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

the tags embed the graph id, start, and end states. The cryptographic policy representations are distributed to users via an authenticated broadcast channel (*e.g.,* by signing and publishing them on a website). The user $\mathcal{U}$ generates a keypair that is certified by the CA.

Obtaining a Credential. When a user $\mathcal{U}$ wishes to obtain a credential, she first negotiates with the provider to select an update policy to which the credential will be bound, as well the credential's initial state within the policy graph. The user next engages in a protocol to blindly extract a CL signature under the provider's secret key, which binds the user's public key, her initial state, the policy id, and two random nonces chosen by the user. The *update nonce* N_u is revealed when the user updates the credential and the *usage nonce N_s* 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 readily apply the techniques found in [10,11] 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 associated policy. She then generates a new pair of nonces and a commitment embedding these values, as well as the new state from her chosen tag. Next, the user sends the update nonce from her current credential, along with the commitment, to the provider. 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. If the nonce and commitment are not in the database, the user and provider then interact to conduct a zero-knowledge proof that: (1) the remainder of the information in the commitment is identical to the current credential, (2) the user has knowledge of the secret key corresponding to her current credential, and (3) the policy graph contains a signature on a tag from the current state 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, shown in Figure 2, follows [11,12]. When a user wishes to prove possession of a credential to $\mathcal{P}$, she first reveals the credential usage nonce and the current state of the credential. $\mathcal{P}$ must check that the usage 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. Alternatively, if the user does not want to reveal her state explicitly, the user may generate a commitment to her state and prove (in zero knowledge) that it is the same as that which is found in her credential.

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, assuming it is allowed by the user's policy. This update policy gives users a way to use a single credential multiple times. One can also adapt this scheme to support k-times anonymous use of the credential by using the Dodis-Yampolskiy [25] pseudorandom function to generate the nonces from a common seed, as shown in [8].

A Note on Efficiency. The efficiency of our protocols is of utmost importance in ensuring their practical use in oblivious databases. During the Setup protocol, the provider must "translate" each of the graphs into a cryptographic representation by signing each tag associated with the graphs. This means that the complexity of the Setup protocol is linear in the size of the policy graphs used in controlling access to the database. While this may seem onerous at first, it is important to emphasize that this process may be conducted offline, and only as a one time cost to the provider. Once the setup procedure is completed, the complexity of the remaining protocols is *constant* and independent of the size of the policy in use since they deal with only a single tag at a time. Thus, our scheme is practical even for extremely complex policies containing thousands of distinct states and transition rules.

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.

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

Theorem 1. *When instantiated with the RSA (resp., bilinear) variant of CL signatures, the anonymous credential scheme above achieves user, and provider security under the strong RSA (resp., LRSW) assumption.*

The proof of Theorem 1 is in the full version of this work [21].

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 or their identities to the database operator (*e.g.,* where the item choices are sensitive).

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 scheme due to Camenisch, Neven and shelat [15]. Our stateful credentials can also be efficiently coupled with the adaptive OT of Green and Hohenberger [28].
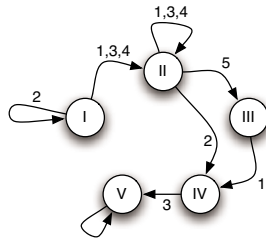

Fig. 3. Sample access policy for a small oblivious database. The labels on each transition correspond to the database item indices that can be requested when a user traverses the edge, with null transitions represented by unlabeled edges.

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 in the database that is allowed by that tag. Each edge in the graph may be 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 fuller discussion of the non-trivial access control policies, including Bell-Lapadula and Brewer-Nash that are allowed by our credential system in the full version of this work [21].

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 OT protocol. Each transaction is conducted between one of a

collection of users and a single database server $\mathcal{D}$. We describe the protocol specifications.

- Setup($\mathcal{U}(1^k), \mathcal{D}(1^k, \Pi_1, \ldots, \Pi_n, M_1, \ldots, M_N)$): The database server $\mathcal{D}$ 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. In addition, $\mathcal{D}$ initializes its database of messages according to the OT protocol in use. 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 updates 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 server. 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.* [15]. 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 her policy, the user is assured that the database operator learns nothing about its identity, or the nature of its request. Figures 4 and 5 describe the protocols in detail.

$\mathsf{Setup}(\mathcal{U}(1^k), \mathcal{D}(1^k, \Pi_1, \dots, \Pi_n, M_1, \dots, M_N))$**:** When the database operator $\mathcal{D}$ is initialized with a database of messages $M_1, \dots, M_N$, 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 credential 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 $pk_\mathcal{D} = (spk_\mathcal{D}, gpk_\mathcal{D}, \gamma, H, y = g^x, C_1, \dots, C_n)$ 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 $A \leftarrow \mathsf{Commit}(sk_\mathcal{U}, N_s)$.
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 $\mathsf{Cred} = (\Pi_\mathcal{C}, S, \sigma_\mathsf{state}, N_s)$.

---

[a] This proof can be conducted efficiently in four rounds as in [15].

Fig. 4. Setup and user registration algorithms for an access controlled oblivious database based on the Camenisch, Neven and shelat oblivious transfer protocol [15]. The database operator and users first run the Setup portion of the protocol. Each user subsequently registers with the database using OTObtainCred.

Theorem 2. *The scheme described above satisfies database and user security (as defined in Definition 1) under the q-PDDH, q-SDH, and Strong RSA assumptions.*

A full proof of Theorem 2 appears in the full version of this work [21]. We sketch the broad outlines of the proof in Appendix B.

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 $N_s' \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes $A \leftarrow \mathsf{Commit}(sk_\mathcal{U}, N_s', \mathsf{pid}, T)$.
3. $\mathcal{U}$ then sends (N_s, A) 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) $A = \mathsf{Commit}(sk_\mathcal{U}, N_s', \mathsf{pid}, T)$.
   (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 [15]). $\mathcal{U}$ computes the message $M_i = B_i / U^{1/v}$.

Fig. 5. Database access protocol for an access-controlled oblivious database based on the Camenisch, Neven and shelat adaptive oblivious transfer protocol [15]

5.3 Extensions to Compact Access Policies in Practice

Extension #1: Equivalence Classes. Thus far, the protocol requires that a tag in the policy graph must be defined on every item index in the database. Yet, 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 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, showing that the 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 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   Conclusion

In this paper, we presented a flexible and efficient system that allows content providers to control access to their data, while simultaneously maintaining the privacy provided by the oblivious and anonymous protocols. Specifically, we described techniques for augmenting traditional anonymous credentials with state, and showed how to combine these credentials with Oblivious Transfer to permit oblivious access to a database enforcing a variety of non-trivial access control policies. The flexibility of our approach makes it relatively straightforward to apply to a diverse set of anonymous and oblivious protocols. For example, our stateful anonymous credentials can be used to control which messages are signed with several blind signature schemes, including those of Waters [35], Boneh and Boyen [4], and Camenisch and Lysyanskaya [11,12], without ever revealing the message to the signer. Other interesting applications include augmenting oblivious versions of Identity-Based key extraction [28] and keyword search protocols [31] with strong access controls.

Acknowledgments

References

1. Aiello, W., Ishai, Y., Reingold, O.: Priced oblivious transfer: How to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Elliot Bell, D., Elliot Bell, D., LaPadula, L.J.: Secure Computer System: Unified Exposition and Multics Interpretation. Comm. of the ACM 1, 271–280 (1988)
3. Blake, I.F., Kolesnikov, V.: Strong Conditional Oblivious Transfer and Computing on Intervals. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 515–529. Springer, Heidelberg (2004)
4. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 382–400. Springer, Heidelberg (2004)
5. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)

6. Brands, S.: Rapid demonstration of linear relations connected by boolean operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
7. Brewer, D.F.C., Nash, M.J.: The Chinese Wall Security Policy. In: IEEE Symposium on Security and Privacy, pp. 206–214 (1989)
8. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: Efficient periodic n-times anonymous authentication. In: ACM CCS 2006, pp. 201–210 (2006)
9. Camenisch, J.L., Hohenberger, S., Lysyanskaya, A.: Balancing Accountability and Privacy Using E-Cash (Extended Abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 141–155. Springer, Heidelberg (2006)
10. Camenisch, J., Lysyanskaya, A.: Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
11. Camenisch, J.L., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
12. Camenisch, J.L., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
13. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number n is the product of two safe primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
14. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 413–430. Springer, Heidelberg (1999)
15. Camenisch, J.L., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007)
16. Camenisch, J.L.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich (1998)
17. Chan, A., Frankel, Y., Tsiounis, Y.: Easy come – easy go divisible cash. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 561–575. Springer, Heidelberg (1998)
18. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
19. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
20. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
21. Coull, S., Green, M., Hohenberger, S.: Controlling access to an oblivious database using stateful anonymous credentials. Cryptology ePrint Archive, Report 2008/474 (2008), http://eprint.iacr.org/
22. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
23. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
24. Damgård, I.B., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)

25. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs an Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
26. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
27. Google. Google Health (2008), http://www.google.com/intl/en-US/health/about/index.html
28. Green, M., Hohenberger, S.: Blind identity-based encryption and simulatable oblivious transfer. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 265–282. Springer, Heidelberg (2007)
29. Lysyanskaya, A.: Signature schemes and applications to cryptographic protocol design. PhD thesis, MIT, Cambridge, Massachusetts (September 2002)
30. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
31. Ogata, W., Kurosawa, K.: Oblivious keyword search. Special issue on coding and cryptography J. of Complexity 20(2-3), 356–371 (2004)
32. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
33. Schnorr, C.-P.: Efficient signature generation for smart cards. J. of Cryptology 4(3), 239–252 (1991)
34. Teranishi, I., Furukawa, J., Sako, K.: k-Times Anonymous Authentication. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 308–322. Springer, Heidelberg (2004)
35. Waters, B.: Efficient Identity-Based Encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

A   Security Definition for an Oblivious Database

Definition 1 (Security for Oblivious Databases with Access Controls).
Security is defined according to the following experiments, which are based on those of Camenisch *et al.* [15]. Although we do not explicitly specify auxiliary input to the parties, we note that 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 D_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 $i \in [1, N + 1]$ as required by the strategy Σ. Following each round, $\hat{\mathcal{D}}$ outputs D_j, and the users output

$(U_{1,j}, \ldots, U_{\eta,j})$. At the end of the kth round the output of the experiment is $(D_k, U_{1,k}, \ldots, U_{j,k})$.

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 $D_k = params$. 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 database items 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 $\hat{\mathcal{D}}'$ generates messages $(M_1^*, \ldots, M_N^*)$ and sends these, along with the policy graphs to the trusted party $\mathcal{T}$ (each policy graph specifies its initial state). 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 $\hat{\mathcal{U}}'$ is in state S and a transition to T is permitted, sending $\hat{\mathcal{D}}'$ 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^* (or $\perp$ if $i = N + 1$) to $\hat{\mathcal{U}}_i'$, otherwise it returns $\perp$. Following each round, $\hat{\mathcal{D}}'$ outputs D_j, and the users output $(U_{1,j}, \ldots, U_{\eta,j})$. At the end of the kth round the output of the experiment is $(D_k, U_{1,k}, \ldots, U_{\eta,k})$.

We will define the *honest* database $\mathcal{D}$ as one that sends $M_1, \ldots, M_N$ in the first round, returns $b_2 = 1$ in all rounds, and outputs $D_k = \epsilon$. Similarly, an honest user $\mathcal{U}_i'$ runs the Setup protocol honestly in the first round, makes queries and transitions according to the selection policy, and eventually outputs all received database items as its output.

Let $\ell(\cdot), c(\cdot), d(\cdot)$ be polynomials. 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 (possibly corrupted) 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 $\forall N = \ell(\kappa), \eta = d(\kappa), k \in c(\kappa), \Sigma$, and every p.p.t. distinguisher:

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

User Security. A stateful anonymous credential scheme provides Receiver security if for every real-world (possibly corrupted) p.p.t. database $\hat{\mathcal{D}}$ and collection of (possibly corrupted) users $\hat{\mathcal{U}}_1, \ldots, \hat{\mathcal{U}}_\eta$, there exists a p.p.t. ideal-world sender $\hat{\mathcal{D}}'$ and ideal (possibly corrupted) users $\hat{\mathcal{U}}_1', \ldots, \hat{\mathcal{U}}_\eta'$ such that $\forall N = \ell(\kappa), \eta = d(\kappa), k \in c(\kappa), \Sigma$, and every p.p.t. distinguisher:

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

B   Proof Sketch of Theorem 2

We now sketch a proof of Theorem 2, arguing security for the oblivious database protocol of §5.2. Our sketch will refer substantially to the original proof of the underlying adaptive oblivious transfer protocol, which is due to Camenisch, Neven and shelat [15]. Our proof will consider two components: (1) the security of the underlying OT scheme (which is based on the proof of [15]), and a separate proof of the anonymous credential scheme.

Proof outline. We separately consider 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 the values $pk, C_1, \ldots, C_N$. 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 $\mathcal{T}$.

When a corrupted user $\hat{\mathcal{U}}$ queries the database, we pass its communications along to $\hat{\mathcal{D}}$ unmodified. Whenever an uncorrupted user $\mathcal{U}$ 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 fact that the element V transmitted to $\mathcal{D}$ during OTAccessAndUpdate is randomly distributed, and (b) the attached proof-of-knowledge is witness indistinguishable and therefore does not reveal the value of i or the user's identity.

We do not need to 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 $\bot$, 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 one 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 [15] for the full argument).
2. The simulated proof of U's structure is indistinguishable from a 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). 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).