# Attribute-Based Access Control with Hidden Policies and Hidden Credentials

Keith Frikken, *Member*, *IEEE*, Mikhail Atallah, *Fellow*, *IEEE*, and Jiangtao Li, *Student Member*, *IEEE*

**Abstract**—In an open environment such as the Internet, the decision to collaborate with a stranger (e.g., by granting access to a resource) is often based on the characteristics (rather than the identity) of the requester, via digital credentials: Access is granted if Alice's credentials satisfy Bob's access policy. The literature contains many scenarios in which it is desirable to carry out such trust negotiations in a privacy-preserving manner, i.e., so as minimize the disclosure of credentials and/or of access policies. Elegant solutions were proposed for achieving various degrees of privacy-preservation through minimal disclosure. In this paper, we present protocols that protect both sensitive credentials and sensitive policies. That is, Alice gets the resource only if she satisfies the policy, Bob does not learn anything about Alice's credentials (not even whether Alice got access), and Alice learns neither Bob's policy structure nor which credentials caused her to gain access. Our protocols are efficient in terms of communication and in rounds of interaction.

**Index Terms**—Electronic commerce-security, management of computing and information systems, security and protection, authentication, access control, trust negotiation, hidden credentials, privacy.

## 1 Introduction

Whereas, in the past, access decisions were based on the identity of the entity requesting a resource, in open systems such as the Internet, this approach is ineffective when the resource owner and the requester belong to different security domains controlled by different authorities that are unknown to each other. One alternative is to use *digital credentials* for satisfying access policies. Digital credentials, the digital equivalent of paper credentials, are digitally signed assertions about the credential owner by a credential issuer. Each digital credential contains an attribute (or set of attributes) about the owner. The decision to allow or deny access to a resource is based on the attributes in the requester's credentials, such as age, citizenship, employment, group membership, or credit status. This approach is called attribute-based access control [3], [27], [12], [22], [21].

A typical scenario for accessing a resource using digital credentials is for the requester Alice to send her request to Bob, who responds with the policy that governs access to that resource. If Alice's credentials satisfy Bob's policy, she sends the appropriate credentials to Bob. After Bob receives the credentials and verifies them, he grants Alice access to the resource. Observe that, in this scenario, Alice learns Bob's policy and Bob learns Alice's credentials. Such a strategy is straightforward and efficient; however, if the credentials or the access control policies are sensitive, then this strategy is unacceptable. We now give a simple example where both the credentials and the policy are sensitive.

**Example 1.** Consider an online business that grants access to media records by sending access keys to its client's special media-reader software—keys that the reader uses to "unlock" encrypted media records that are freely downloaded in encrypted form (or are given away, in encrypted form, on CDs that are widely distributed for free). Certain records are treated differently from the rest: The online business grants access to these records only if the requester has a disability or is a senior citizen or is terminally ill and has an income of under 30 K a year. This requirement involves four attributes (denote them by $attr_1$, $attr_2$, $attr_3$, $attr_4$) and the policy is $(attr_1 \vee attr_2 \vee attr_3) \wedge attr_4$. In order to gain access to the sensitive records in Bob's database, Alice needs to prove to Bob that she satisfies the policy. However, neither Alice nor Bob is willing to disclose her/his private information. Alice does not want to reveal her credentials as her credentials contain sensitive information about her (e.g., health, age, income, etc.). Bob does not want to reveal the policy, even to those who satisfy the policy, so as to make it harder for an adversary to know which credentials he needs to forge or otherwise illicitly gain access.

In other examples, the motivation for hiding the policy is not security from an evil adversary, but simply the desire to prevent legitimate users from "gaming" the system—e.g., changing their behavior based on their knowledge of the policy (which can render economically motivated policies less effective). This is particularly important for policies that are not incentive-compatible in economic terms (i.e., they suffer from perverse incentives in that they reward the wrong kinds of behavior, such as free-loading). In yet other

• *The authors are with the CERIAS and Department of Computer Sciences, Purdue University, 250 N. University Street, West Lafayette, IN 47907. E-mail: {kbf, mja, jtli}@cerias.purdue.edu.*

*Manuscript received 4 Oct. 2004; revised 5 Apr. 2006; accepted 3 May 2006; published online 22 Aug. 2006.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0313-1004.*

examples, the policy is simply a commercial secret—e.g., Bob has pioneered a novel way of doing business and knowledge of the policy would compromise Bob's strategy and invite unwelcome imitators.

Finally, it is important to point out that a process that protects Alice's credentials from Bob is ultimately not only to Alice's advantage but also to Bob's: Bob no longer needs to worry about rogue insiders in his organization illicitly leaking or selling Alice's private information and may even lower his liability insurance rates as a result of this. Privacy-preservation is a win-win proposition, one that is appealing even if Alice and Bob are honest and trustworthy entities.

## 1.1 Related Work

Our work is related to the area of automated trust negotiation [4], [28], [31], [32], [33], [34], [37], [38]. The goal of trust negotiation is to enable strangers to access sensitive data in open environments without violating the data's access policy. In trust negotiation, two parties establish trust through iterative disclosure of credentials and requests for credentials. Most of the research in this area focuses on protecting resources and credentials and assumes that policies can be freely disclosed. Some work [4], [28], [37] considers access policies to be sensitive information. Bonatti and Samarati [4] proposed a framework for regulating service access and information release on the Web. Their scheme protects the revelation of policies by dividing policies into two parts: service prerequisite rules and service requisite rules. A requisite rule is disclosed only after prerequisite rules are satisfied. Seamons et al. proposed the concept of policy graphs in [28]. Instead of using a single policy, their scheme uses policy graphs to represent complex policies and it protects polices by gradual disclosure of the graph nodes. Furthermore, Yu and Winslett proposed a unified scheme (UniPro) for resource protection [37]. The basic idea of the UniPro scheme is to model policies as protected resources and to protect them in the same way as other resources. Our work offers better protection of policies; in their schemes, Alice learns (part of) the policy if her credentials satisfy the policy, whereas, in our protocols, Alice does not learn the policy even if her access request is granted.

Li et al. introduced the notion of Oblivious Signature-Based Envelope (OSBE) [20] to protect sensitive credentials. They assume the content of a credential is nonsensitive (as anyone can come up with it) and only the signature of the credential needs to be protected. In OSBE, Alice sends Bob the content of her credential or a credential that she does not have and Bob runs an OSBE protocol with Alice, sending an encrypted message to Alice such that Alice can decrypt it if and only if she has the signature on the content. The difference between their work and ours is that the policies in OSBE have to be revealed, whereas the policies in our protocols are protected.

Balfanz et al. proposed a construct called secret handshakes [1]. In their scheme, each party receives, from a central authority, a certificate that consists of a pseudonym and a corresponding secret. When Alice and Bob meet, they exchange their pseudonyms and then each of them computes a key based on their own secret and the other party's pseudonym in order to achieve mutual authentication. Compared to their scheme, our work supports more complex policies and protects the content of the policies.

Recently, Holt et al. proposed hidden credentials [18], a system that protects sensitive credentials and policies. Furthermore, their system reduces the network overhead as it needs fewer rounds of interaction compared to traditional trust negotiation. Their system also solves the "going first" problem in PKI-authentication systems, where one of the two parties must be the first to reveal a certificate to a potentially malicious stranger. Our protocols directly build on their work. However, we believe that the protection of policies in their system is not sufficient, for three reasons:

1. The policy structures are revealed in their system. For instance, if Bob's policy is $(attr_1 \wedge attr_2) \vee attr_3$, where $attr_1$, $attr_2$, and $attr_3$ are three attributes, Alice learns that the structure of the policy is of the form $(x \wedge y) \vee z$ even if her credentials do not contain any one of the attributes $attr_1$, $attr_2$, and $attr_3$.
2. If an access request to a resource is granted, Alice learns which attributes gave her access. For instance, if Bob's policy is $(attr_1 \wedge attr_2) \vee attr_3$ and Alice's credentials contain $attr_1$ and $attr_2$, Alice gets the resource and she knows that $attr_1 \wedge attr_2$ is part of the access policy.
3. Even if Alice cannot access the resource, she might learn some partial information about the policy. For instance, if Bob's policy is $(attr_1 \wedge attr_2) \vee (attr_3 \wedge attr_4)$ and Alice's credentials contain $attr_1$ and $attr_3$, Alice learns that $attr_1$ and $attr_3$ are part of Bob's policy.

Bradshaw et al. [6] extended the hidden credentials system to support complex access policies expressed as monotonic Boolean functions. They applied a secret splitting system to conceal the structure of such policies. The extended hidden credentials system protects the structure of Bob's polices; however, it is still unable to solve the problems described above in items 2 and 3, whereas, in our protocols, if Alice's credentials match an attribute in Bob's policy, she will not necessarily learn that the attribute is part of the policy (of course, Alice will learn whether she obtained access and some inferences can be made about the attributes in Bob's policy; we will discuss this in more detail in the next section). In summary, the hidden credentials system [18], [6] and the trust negotiation systems ([33], [31], [38], [37], to list a few) do not achieve the privacy goal of our work. Of course, we do so at a cost in protocol complexity; therefore, our work should be viewed as providing another point on the privacy-performance curve, rather than as an unqualified improvement over the previous work.

## 1.2 Problem Definition

Before we formally define the hidden policies with hidden credentials problem, we first define attributes and policies for our problem.

An attribute is a statement about a credential holder. For example, an attribute could be gender (male or female), job type (student, faculty, FBI agent, etc.), state of residence (California, Indiana, Ohio, etc.), status (secret clearance, disabled, homeless), age (between [0-17], between [18-20], between [21-59], or age $\geq$ 60), or annual income (between $[0-15K]$, between $[15K-30K]$, between $[30K-60K]$, or income $> 60K$). Let $\mathcal{S}$ be the set of all possible attributes in the hidden credentials system. The Certificate Authority (CA) publishes $\mathcal{S}$ to every user.

Each credential binds a username $nym$ and an attribute $attr$. Each user in the system has a unique username $nym$. $nym$ can be either a real name or a pseudonym. We use $cred.attr$ to denote the attribute associated with the credential $cred$. If a user has $m$ attributes, she can get $m$ credentials from the CA, one for each attribute. For instance, if Alice is a professor, she can have a professor credential $cred_1$, where $cred_1.attr = professor$; if her age is 30, she has an age credential $cred_2$, where $cred_2.attr = age \in [21 \dots 59]$.[1]

We now define the policies for our problem. Let $\mathcal{P}$ be a family of policy functions. Each $P \in \mathcal{P}$ is a policy function: Given any set of credentials $C$, $P(C) \in \{0,1\}$. More specifically, a policy function $P$ is defined by a Boolean expression $p$ that is relevant to $n$ attributes $\{attr_1, attr_2, \dots, attr_n\} \subset \mathcal{S}$. That is, $P(C) = p(x_1, x_2, \dots, x_n)$, where $p : \{0,1\}^n \rightarrow \{0,1\}$ and $x_i = 1$ if $\exists\, cred_j \in C$ such that $attr_i = cred_j.attr$; otherwise, $x_i = 0$.

We now give an example to clarify the above definitions. Let $C_A$ be Alice's credential set; to check whether Alice satisfies a policy $P$ is equivalent to verifying whether $P(C_A) = 1$. For example, suppose Bob's policy is $female \wedge student$, i.e., $p(x_1, x_2) = x_1 \wedge x_2$, and that Alice has a female credential ($cred_1.attr = female$), a professor credential ($cred_2.attr = professor$), and an Indiana resident credential ($cred_3.attr = Indiana$). Alice does not satisfy Bob's policy as $P(cred_1, cred_2, cred_3) = p(x_1, x_2) = 1 \wedge 0 = 0$. Note that $x_1 = 1$ because Alice has a female credential and $x_2 = 0$ because Alice does not have a student credential.

The goal of our work is to provide privacy-protection for *both* access control policies and credentials. The framework for our *hidden policies with hidden credentials* problem is informally described as follows: Alice has $m$ credentials issued by the CA. Bob has a resource $M$ and a policy $P$ for controlling access to $M$. When Alice wants to access $M$ from Bob, she engages in a protocol with Bob. Alice provides the protocol with a subset of her credentials (she may choose to omit certain credentials), whereas Bob provides $M$ and the policy $P$. If the attributes in the credentials that Alice inputs into the protocol satisfy $P$, she gets $M$; otherwise, she gets nothing. We want Alice to learn as little information as possible about Bob's policy and Bob to learn as little information as possible about Alice's credentials.

We now formally state the problem. Let $M$ be a private message. Alice has private input $C_A$, a subset of her credentials. Bob has a private message $M$ and a private access control policy $P$. Alice and Bob want to compute a function $F$ defined as follows:

$$F_{Bob}(M, P) = \perp$$

$$F_{Alice}(C_A) = \begin{cases} M & \text{if } P(C_A) = 1; \\ \perp & \text{otherwise,} \end{cases}$$

where $F_{Alice}$ represents Alice's output, $F_{Bob}$ represents Bob's output, and $\perp$ denotes an empty message. In other words, our goal is for Bob to learn nothing from the function evaluation, and for Alice to learn $F_{Alice}(C_A)$ without learning anything else, i.e., she learns $M$ iff $P(C_A)$ is equal to 1 (i.e., her credentials satisfy Bob's policy) and she learns nothing otherwise. Observe that Alice can infer the result of $P(C_A)$ from her output.

The preceding problem can be solved using general solutions from two-party Secure Function Evaluation (SFE) [36], [17], [16] as follows: Alice and Bob first build a universal circuit [30] in which policy evaluation and credential verification are performed and then Alice and Bob securely evaluate the circuit using the standard two-party SFE techniques [36], [17]. However, this approach is inefficient, as the size of the circuit is very large.

It is possible in our framework to set the policy to be an arbitrary computable function (even nonmonotonic functions); however, it is common in the literature to assume that the policy does not require the absence of a credential. The reason for this is the practical difficulty of verifying an absence, e.g., if the policy is $\neg attr$, then a party, Alice, who has $attr$ can choose not to input the corresponding credential (if she suspects it can cause her to be denied access). Furthermore, if Alice knows that her credentials will not be revealed by the protocol and that the policy is monotonic, then she has an incentive to input all of her credentials. It is therefore a practical consideration, rather than an inherent limitation of our scheme (in fact, our scheme can support nonmonotonic policies), that motivates this assumption that the policy does not require the absence of a credential. Sometimes the absence of a credential ("under 21 years of age") can be replaced by a requirement for the presence of the opposite ("at least 21 years of age"), but this is not always possible (e.g., consider requiring the absence of a credential for millionaire).

Another issue that needs to be discussed is probing attacks by either party. Our protocols prevent Alice from probing a policy offline (i.e., requesting a resource once and then trying several subsets of her credentials). However, Alice can engage in the protocol with Bob multiple times using different credential sets (all subsets of her credentials) to gain information about Bob's policy. Restricting adversaries to online attacks is a significant step forward, but does not entirely solve the problem. Other means for preventing online probing attacks can be taken, for example, making sure that Alice can request a specific resource from Bob no more than three times a week. As Bob does not know whether Alice gained access, he cannot carry out probing attacks.

## 1.3 Our Contributions

In this paper, we give efficient protocols that solve the hidden policies with hidden credential problem. Our protocols are built on the hidden credentials system [18] and have two phases: In the first phase, we use homomorphic encryption, oblivious transfer, scrambled circuit

1. For simplicity, if an attribute takes multiple values such as age and income, we bucket these values into several categories such that it is easier to construct a policy. Consider the policy $age \geq 21$: If we allow age-attribute to be any integer value, then the policy would be $age = 21 \vee age = 22 \vee \dots$; however, if we bucket age-attribute into four groups: $[0-18], [18-20], [21-60], [60+]$, the new policy is simply $[21 \dots 59] \vee [60+]$.

TABLE 1
Summary of Communication and Number of Rounds of Interaction of Our Protocols

| | Simple Policies | Arbitrary Policies |
|---|---|---|
| Protocol 1 | $O(\rho n), 3$ | $O(\rho n + 2^n), 3$ |
| Protocol 2 | $O(\rho mn), 5$ | $O(\rho mn + 2^n), 5$ |
| Protocol 3 | $O(\rho^2 mn), 5$ | $O(\rho^2 mn + 2^n), 5$ |

*In this table, $\rho$ is a security parameter (e.g., $\rho$ could be the ciphertext size of IBE encryptions [5], [9] or homomorphic encryptions [11], [26], [25]), $m$ is the number of credentials Alice possesses, and $n$ is the number of attributes in Bob's policy. The simple policies include monotonic Boolean expressions as well as threshold-based functions.*

evaluation, and shuffling to prevent an attacker from offline probing; in the second phase, we use scrambled circuit evaluation to protect the policies.

Our scheme is *credential indistinguishable* in that the communication between two clients with two credential sets of equal size is indistinguishable (at least to a computationally bounded server) from each other. Thus, the server will not learn which credentials a client has from our protocols. Furthermore, our scheme is *policy indistinguishable* in that two policies that evaluate to the same value for the client's credential set have indistinguishable communication transcripts and, thus, the client does not learn anything about the policy other than what can be deduced from the outcome (i.e., whether access was granted). For efficiency reasons, we sometimes weaken this latter requirement by requiring only indistinguishability for policies of a certain class (this will become clear in Section 3.2.2).

We provide three protocols, each with different performance and levels of protection for the policy. All of these protocols leak an upper bound on the number of credentials that Alice is using to obtain the resource and an upper bound on the number of attributes in the policy.[2] Protocol 1 reveals to Alice a superset of the credentials in the policy. Protocol 2 reveals the number of credentials in the policy that Alice satisfies. Protocol 3 reveals nothing beyond the above-mentioned upper bounds. Our protocols can support arbitrary policies at exponential cost. However, for certain classes of policies (e.g., monotonic Boolean expression, threshold-based functions), our protocols require only polynomial cost. A summary of our protocols' performance is listed in Table 1.

In the above table, $\rho$ is a security parameter and is typically about 1,024 or 2,048. Hence, protocols 1 and 2 are efficient and protocol 3 is quite expensive in practice. Compared to our scheme, the hidden credentials schemes in [18], [6] require at least $O(\rho n)$ communication (depending on the policy structure). Again, we stress that our protocols achieve better privacy at a cost in protocol complexity.

2. Note that, when Alice and Bob engage in our protocols, Alice learns the number of attributes in Bob's policy and Bob learns the number of credentials that Alice inputs. Since Alice can insert some dummy credentials and Bob can add dummy attributes into his policy before engaging the protocols, the *actual* number of Alice's credentials and *actual* size of Bob's policy are not necessarily leaked from our protocols.

### 1.4 Organization of Document

The rest of this paper is organized as follows: In Section 2, we review the identity-based encryption schemes, the hidden credentials system, and some building blocks that are used in our protocols. We present our protocols in detail in Section 3 and give the security proofs in Section 4. We conclude in Section 5.

## 2 Review of Cryptographic Tools

### 2.1 Review of Identity-Based Encryption

The concept of Identity-Base Encryption (IBE) was first proposed by Shamir [29] in 1984; however, the first usable IBE systems were discovered only recently [5], [9]. An IBE scheme is specified by the following four algorithms:

1. *Setup:* A Private Key Generator (PKG) takes a security parameter $k$ and generates system parameters $\mathsf{params}$ and a master secret $s$. $\mathsf{params}$ is public, whereas $s$ is private to PKG.
2. *Extract:* Given any arbitrary ID $\in \{0,1\}^*$, PKG uses $\mathsf{params}$, $s$, and ID to compute the corresponding private key $d_{\mathrm{ID}}$.
3. *Encrypt:* It takes $\mathsf{params}$, ID, and plaintext $M$ as input and returns ciphertext $C$.
4. *Decrypt:* It takes $\mathsf{params}$, $d_{\mathrm{ID}}$, and ciphertext $C$ as input and returns the corresponding plaintext $M$.

An IBE scheme enables Bob to encrypt a message using Alice's ID as the public key and, thus, he avoids obtaining the public key from Alice or a directory. Boneh and Franklin proposed an IBE scheme from the Weil pairing [5] that is secure against adaptive chosen ciphertext attacks.

### 2.2 Review of Hidden Credentials System

In the hidden credentials system proposed by Holt et al. [18], there is one or more Credential Authority (CA) who issue credentials for users in the system. We assume CAs are trusted entities. Each user in the system is assigned a unique $nym$, where $nym$ could be either a real name or a pseudonym. In hidden credential systems [18], [6], a credential $cred$ is a tuple ($nym$, $attr$, $sig$). The CA issues a credential asserting that $nym$ has attribute $attr$ by providing $sig$ to the owner of $nym$. Note that $sig$ is known only to the credential holder (and the CA) and must be kept secret. Given an IBE scheme as described in Section 2.1, the signature $sig$ of a hidden credential $cred$, for username $nym$ and attribute $attr$, is the private key corresponding to the identity $nym \| attr$. More specifically, assuming public value $\mathsf{params}$ is conventionally agreed upon for use by all CAs, the hidden credentials system has the following four programs:

1. $CA\_Create()$: A CA chooses a private key $sk$ and computes the corresponding public key $pk$. The CA keeps $sk$ private and publishes $pk$. The CA may also publish the whole list of attribute names for which she can issue credentials.
2. $CA\_Issue(nym, attr)$: A CA issues a hidden credential $cred$ for the user with username $nym$ and attribute $attr$. The signature of the credential $cred$ is the IBE private key corresponding to the public

identity $nym \parallel attr$. Given a hidden credential $cred$, we use $cred.nym$ to denote the corresponding username, $cred.attr$ to denote the corresponding attribute in the credential, and $cred.sig$ to denote the signature of the credential.

3. $I(M, nym \parallel attr)$: This program corresponds to the encrypt algorithm of the IBE system with system parameters params, $\mathrm{ID} = nym \parallel attr$, and plaintext $M$. The output of this program is a ciphertext $C$.
4. $I^{-1}(C, cred)$: This function corresponds to the decrypt program of the IBE system with system parameters params, credential $cred$, and ciphertext $C$. The output of this function is plaintext $M$. These programs must satisfy the standard consistency constraint, namely, for any credential $cred$ and any message $M$, $I^{-1}(I(M, cred.nym \parallel cred.attr), cred) = M$. When a user computes $I^{-1}(I(M, cred.nym \parallel cred.attr), cred')$ for some $cred' \neq cred$, the value obtained is computationally indistinguishable from a random value.

Hidden credentials are secure against an adaptive chosen ciphertext attack where an attacker can obtain an unlimited number of other arbitrary credentials [18]. Hidden credentials are also unforgeable [18]. We now give a simple example (based on [18]) of how Alice can access Bob's resource using a hidden credential. Suppose Bob's resource $M$ can be accessed only by a student. Alice has a student credential $cred$, i.e., $cred.nym = Alice$ and $cred.attr = student$. To access $M$, Alice sends her username $Alice$ to Bob. Bob responds with $I(M, Alice \parallel student)$. Alice uses her credential $cred$ to decrypt $I(M, Alice \parallel student)$ and obtains $M$. Bob does not learn whether Alice possesses a student credential from the above interaction.

## 2.3 Review of Other Building Blocks

**Scrambled Circuit Evaluation.** There are two parties in scrambled circuit evaluation, one is a *generator* of a scrambled circuit and the other is an *evaluator*. Let $x$ be the evaluator's input, $y$ be the generator's input, and $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a function known to both parties. The goal of the scrambled circuit evaluation is for the two parties to securely compute $f(x, y)$ without the generator learning $x$ or the evaluator learning $y$. The generator creates a scrambled circuit where each wire of the circuit has two encodings (one for each possible value of the wire) and the evaluator learns only one encoding per wire and, thus, the intermediate results are oblivious to the evaluator. The generator creates gate information that allows the evaluator to obtain the encoding of a gate's output wire when given the encoding for each of the gate's input wires. Yao's original scheme [35] was secure only in the passive model (where the participants follow the protocol) and, recently, there has been an implementation of this protocol [23]. Also, Katz and Ostrovsky introduced a scheme that is secure in the malicious model that requires only five rounds of interaction [19]. We discuss this in more detail in Appendix A.

**Homomorphic Encryption.** A cryptographic scheme is said to be homomorphic if, for its encryption function $E$, the following holds: $E(x) * E(y) = E(x + y)$. Some homomorphic schemes are described in [11], [26], [25]. A homomorphic encryption scheme is *semantically secure* if $E(x)$ reveals no information about $x$; hence, from a pair $(E(x), E(y))$, it is computationally infeasible to distinguish between the case $x \neq y$ and $x = y$.

**Symmetric Key Encryption.** We use $Enc$ to denote the encryption algorithm of a symmetric key encryption scheme (such as AES [10]) and $Dec$ to denote the corresponding decryption algorithm.

**Oblivious Transfer.** We use chosen 1-out-of-2 Oblivious Transfer (OT) [13], [16] as a basic building block in our protocols. A 1-out-of-2 OT protocol involves a sender and a chooser. A sender has two private messages, $\langle m_0, m_1 \rangle$, and a chooser has a bit, $b \in \{0, 1\}$. At the end of the protocol, the chooser obtains $m_b$ without learning anything else, while the sender learns nothing about $b$. More formally, 1-out-of-2 OT is defined as $\mathrm{OT}_1^2((m_0, m_1), b) = (\perp, m_b)$. Efficient 1-out-of-2 OT protocols have been proposed by Bellare and Micali [2] and Naor and Pinkas [24].

**Set Intersection.** Freedman et al. [14] introduced an elegant scheme for computing the intersection of two $k$ element sets with only $O(k)$ communication and $O(k \ln \ln k)$ modular exponentiations. In this paper, we use a simple modification of their result. We define a primitive $SetInt(x, S, E_A)$, where Bob provides an element $x$ and Alice provides a set $S$ with $n$ elements and a semantically secure homomorphic encryption system $E_A$. The output of the protocol is that Bob learns $E_A(y)$ where $y$ is 0 if and only if $x \in S$ and is otherwise a random value. This primitive can be easily defined from the protocols in [14], requires a single message from Alice to Bob with $O(\rho \parallel S \parallel)$ communication (where $\rho$ is the size of the security parameter), and is secure against a malicious Alice.

## 3 Our Protocols

There are two primary phases in our protocols: 1) a credential hiding phase and 2) a blinded policy evaluation phase. During the credential hiding phase, Alice and Bob engage in a protocol that in some way (to be specified later) hides which credentials Bob's policy requires. During the blinded policy evaluation, if Alice satisfies Bob's policy, then she learns the requested message and, otherwise, she learns nothing. We now describe each phase in more detail:

- *Credential Hiding Phase:* Suppose Bob's policy contains $n$ attributes $attr_1, \ldots, attr_n$. At the end of this phase, Alice has a set of values $\ell_1, \ldots, \ell_n$ (i.e., one for each attribute), where $\ell_i \in \{r_i[0], r_i[1]\}$ and $r_i[0], r_i[1]$ are random values generated by Bob. These values will either be encryption keys or seeds for a pseudorandom generator that can produce such keys. Furthermore, the value of $\ell_i$ is subject to the following constraints:

    1. $\ell_i = r_i[1]$ only if Alice has a credential $cred$ such that $cred.attr = attr_i$ (i.e., if Alice does not possess attribute $attr_i$, she cannot learn the value $r_i[1]$). Otherwise, Alice gets $r_i[0]$.

2. A computationally bounded Alice learns nothing about the value $\{r_i[0], r_i[1]\} - \{\ell_i\}$. In other words, Alice learns one and *only* one value from the set $\{r_i[0], r_i[1]\}$.

- *Blinded Policy Evaluation Phase:* Given the $\ell$ values from the previous phase, Alice and Bob engage in a protocol that allows Alice to learn message $M$ if she satisfies Bob's policy. His policy is represented by a Boolean function $p : \{0,1\}^n \rightarrow \{0,1\}$ (i.e., it maps $n$ values, which correspond to which attributes Alice has, to a binary value that corresponds to whether Alice satisfies Bob's policy). To formalize the definition of this phase, suppose that, after the previous phase, Alice's values are $r_1[x_1], r_2[x_2], \ldots, r_n[x_n]$, where $x_i \in \{0,1\}$, then Alice will receive $M$ if and only if $p(x_1, x_2, \ldots, x_n) = 1$.

## 3.1 Credential Hiding Phase

In this section, we introduce three protocols for the credential hiding phase; we describe the input and output for this phase below:

**Input.** Bob has a policy $P$ relative to set of attributes $attr_1, \ldots, attr_n$ and, for each attribute $attr_i$, Bob has two random values, $r_i[0]$ and $r_i[1]$. Alice has a set of credentials $cred_1, \ldots, cred_m$.

**Output.** Alice learns a value $\ell_i$ for each attribute $attr_i$, where $\ell_i = r_i[1]$ if and only if there exists a $cred_j$, $1 \leq j \leq m$, in her credential set such that $cred_j.attr = attr_i$, and $\ell_i$ is $r_i[0]$ otherwise. Recall that a crucial element of this protocol is that Alice learns exactly one value for each attribute.

There is an inherent security/communication complexity trade-off for these protocols. We denote the number of attributes in Bob's policy by $n$, the number of credentials Alice possesses by $m$, and a security parameter by $\rho$. Note that Alice can input some dummy credentials to disguise her actual number of credentials and, analogously, Bob can add dummy attributes to hide the size of the actual policy. The protocols can be summarized as follows:

1. *Protocol 1* (Section 3.1.1): In this protocol, it is assumed that Bob is willing to reveal to Alice a superset of the attributes in his policy (he chooses this superset and can make it large enough to achieve a "hiding in a crowd" effect that suits him). While this is not acceptable for all applications, there are many cases where Alice could guess with high probability the set of attributes in Bob's policy before the protocol and, in such cases, this protocol may be acceptable to Bob. Example 2 gives a scenario where Bob is willing to reveal a superset of attributes in his policy. The communication complexity of this protocol is $O(\rho n)$ and it requires three rounds of interaction.
2. *Protocol 2* (Section 3.1.2): Unlike Protocol 1, this protocol does not assume that Bob is willing to reveal a superset of the attributes in his policy. In this protocol, Bob learns the value $m$ and Alice learns: a) the value $n$ and b) the number of attributes in Bob's policy that she satisfies (she does not know which of her credentials are responsible for this). This protocol requires $O(\rho m n)$ communication and five rounds of interaction.
3. *Protocol 3* (Section 3.1.3): This protocol is similar to Protocol 2, but Alice does not even learn how many attributes she satisfies in Bob's policy. This protocol requires $O(\rho^2 m n)$ communication and five rounds of interaction.

**Example 2.** An online auto insurance company gives a special promotion to those who are married and have good credit history. The policy for this special promotion is $married \wedge good\_credit$. The insurance company may publish a superset of attributes that the policies may contain such as age, gender, marital status, credit history, number of accidents in the last three years, state of residence, etc. The actual promotion policy contains only a small subset of the published attributes. The auto insurance company may treat the superset of the attributes as nonsensitive information, but treat the attributes used in the policy as well as the policy structure as private information. The insurance company does not want its competitors to know the content of the policy as it is a commercial secret.

### 3.1.1 *Protocol CHP 1*

This protocol assumes that Bob is willing to reveal to Alice a superset of the attributes in his policy. This is the most efficient of the credential hiding phase protocols.

**Protocol.**

1. For each attribute $attr_i$: Bob generates a random key $k_i$ as well as information $C_i$ that reveals to Alice what credential she needs to satisfy $attr_i$. Bob then generates an encryption $\alpha_i = I(k_i, nym \parallel attr_i)$. He sends the following information to Alice: $(\alpha_1, C_1), \ldots, (\alpha_n, C_n)$.
2. For each ordered pair $(\alpha_j, C_j)$, Alice generates a value $b_j$ which is 1 if she possesses a credential $cred_p$ that satisfies $C_j$ and is 0 otherwise. If $b_j = 1$, she computes $k_j$, which is $I^{-1}(\alpha_j, cred_p)$.
3. For each attribute $attr_i$, Alice and Bob engage in a chosen 1-out-of-2 OT protocol where Bob's input is the list $\{r_i[0], Enc(r_i[1], k_i)\}$ and Alice's input is $b_i$.
4. If $b_i$ is 1, then Alice decrypts $Enc(r_i[1], k_i)$ with $k_i$ (she computed this in Step 2) and sets the result as her output and, otherwise, she sets her output to be $r_i[0]$.

**Complexity Analysis.** For each attribute, the protocol requires a single encryption to be sent between Alice and Bob as well as the descriptions $C_i$ (which we assume are of size $O(1)$) and a single chosen 1-out-of-2 OT. Thus, the communication complexity is $O(\rho n)$. The OT can be bootstrapped with the first step and, thus, this protocol requires only three rounds of interaction.

**Intuition.** The intuition of this protocol is that Alice will learn $k_i$ iff she has $attr_i$. This is because this value is encrypted with IBE using $attr_i$ as the key. Furthermore, without $k_i$, the value $Enc(r_i[1], k_i)$ reveals nothing (in a computational sense). Thus, Alice will learn $r_i[1]$ iff she has $attr_i$. Now, due to the nature of the OT, Alice will learn at most one of the values $r_i[0]$ or $Enc(r_i[1], k_i)$.

### 3.1.2 Protocol CHP 2

In this protocol, Bob does not reveal to Alice a superset of the attributes in his policy, but Alice learns how many, but not which, attributes she satisfies in Bob's policy. Clearly, if Bob pads his list with superfluous credentials, as in Protocol 1, this reveals less information to Alice.

**Protocol.**

1. For each attribute $attr_i$, Bob generates two random keys, $k_i[0]$ and $k_i[1]$, and a public marker $q$, where $q$ could be specified beforehand and is of length $\rho$, the length of the security parameter. He also computes $\alpha_i = I(k_i[0], nym \parallel attr_i)$. He then sends to Alice $\alpha_1, \ldots, \alpha_n$ along with $q$.
2. Alice generates a semantically secure homomorphic encryption system $E_A$. Then, for each value $\alpha_i$ and for each of her credentials $cred_j$, she creates a value $\beta_{i,j} = I^{-1}(\alpha_i, cred_j)$. Alice and Bob then engage in protocol $SetInt(k_i[0], \{\beta_{i,1}, \ldots, \beta_{i,m}\}, E_A)$, where Bob learns the value $E_A(\gamma_i)$ (note that $\gamma_i$ is 0 if Alice has a credential for $attr_i$ and is random if she does not). Bob computes $\delta_i = E_A(\gamma_i) * E_A(k_i[1]) = E_A(\gamma_i + k_i[1])$; he then forms ordered pairs $(\delta_i, Enc(q, k_i[1]))$. Bob randomly permutes these pairs and sends them to Alice.
3. For each value $(\delta_j, Enc(q, k_j[1]))$, Alice computes $\eta_j = D_A(\delta_j)$ and then computes $Dec(Enc((q, k_j[1])), \eta_j)$. If this value is $q$, then she stores $\eta_j$ (which is $k_j[1]$) and sets $b_j$ to 1 and, otherwise, she sets $b_j$ to 0.
4. For each attribute $attr_i$ (note that these are permuted), Alice and Bob engage in a chosen 1-out-of-2 OT protocol where Bob's input is the list $\{r_i[0], Enc(r_i[1], k_i[1])\}$ and Alice's input is $b_i$. Note: These values are permuted by Bob above, but Alice does not need to know which value corresponds to which initial attribute.
5. If $b_i$ is 1, then Alice decrypts $Enc(r_i[1], k_i[1])$ with $k_i[1]$ (she computed this in Step 3) and sets the result as her output; otherwise, she sets her output to be $r_i[0]$.

**Complexity Analysis.** It is clear that, for each attribute, $O(m)$ encryption/decryption operations are required (in the form of homomorphic/identity-based encryptions and decryptions). Thus, there are $O(mn)$ such operations and the system requires $O(\rho mn)$ communication. Furthermore, the OT in Step 5 can be started in Step 3 and, thus, this protocol requires five rounds of interaction.

**Intuition.** The set of values $\{\beta_{i,1}, \ldots, \beta_{i,m}\}$ will contain $k_i[0]$ iff Alice has $attr_i$. This is due to the IBE encryption of $k_i[0]$. In the set intersection protocol, Bob will learn $E_A(0)$ if Alice has $attr_i$ and he will learn $E_A(r_1)$ for some random value $r_1$ otherwise. Since $E_A$ is semantically secure, Bob will not learn which value he has received. However, the value $\delta_i$ will be $E_A(k_i[1])$ iff Alice has $attr_i$; otherwise, it will be $E_A(r_2)$ for some random value $r_2$. Thus, Alice will learn $k_i[1]$ iff she has $attr_i$. In Step 3 of the protocol, Alice will learn if a specific value is a correct key, but, since the values were permuted by Bob earlier, she does not learn which credential this corresponds to. The rest of the intuition is identical to the previous protocol except that $k_i[1]$ now plays the role of $k_i$.

### 3.1.3 Protocol CHP 3

This protocol protects which attributes are in Bob's policy more than the previous protocols in that Alice does not learn how many credentials she has that match with the attributes in Bob's policy. As stated earlier, this protocol requires more communication and computation than the previous protocols. It also uses Scrambled Circuit Evaluation extensively (see Appendix A).

**Protocol.**

1. For each attribute $attr_i$, Bob generates two keys, $k_i[0]$ and $k_i[1]$. He also computes $\alpha_i = I(k_i[0], nym \parallel attr_i)$. He then sends to Alice $\alpha_1, \ldots, \alpha_n$.
2. Alice generates a semantically secure homomorphic encryption system $E_A$. Then, for each value $\alpha_i$ and for each of her credentials $cred_j$, she creates a value $\beta_{i,j} = I^{-1}(\alpha_i, cred_j)$. Alice and Bob then engage in protocol $SetInt(k_i[0], \{\beta_{i,1}, \ldots, \beta_{i,m}\}, E_A)$, where Bob learns the value $E_A(\gamma_i)$ (note that $\gamma_i$ is 0 if Alice has a credential for $attr_i$ and is random if she does not). Bob computes $\delta_i = E_A(\gamma_i) * E_A(k_i[1]) = E_A(\gamma_i + k_i[1])$. Bob sends $\delta_1, \ldots, \delta_n$ values to Alice.
3. For each value $\delta_j$, Alice computes $\eta_j = D_A(\delta_j)$. Alice and Bob engage in a Scrambled Circuit Evaluation, with Bob as the generator and Alice as the evaluator. The circuit that Bob creates is a circuit for an equality test (which requires $O(k)$ gates for $k$-bit values). Bob's input into the circuit is $k_j[1]$ and Alice's input is $\eta_j$. If the values match, then Alice learns $r_j[1]$ and, otherwise, she learns $r_j[0]$.

**Complexity Analysis.** There are $n$ circuits in the above protocol, each of which compares $m$ values with $\rho$ bits each. Thus, there will be $O(\rho mn)$ encryption operations and $O(\rho^2 mn)$ communication. This can be reduced slightly by using a one-way function on the values in Step 4 before engaging in the circuit. The protocol requires five rounds of interaction.

**Intuition.** The $\delta$ values are identical to the previous protocol. However, in this protocol, Alice and Bob will use SCE (where Alice is the evaluator) to reveal the outcome. Now, a property of SCE is that the evaluator will learn at most one encoding per wire and the evaluator does not learn which encoding (i.e., 0 or 1) that it received. Thus, Alice will learn either $r_i[0]$ or $r_1[1]$ but not both and she will not know which of them she has received. Furthermore, since Bob constructs the circuit, she will learn $r_1[1]$ iff her value $\eta_i$ is equal to Bob's value $k_i[1]$. Therefore, she will learn $r_i[1]$ iff she has $attr_i$.

## 3.2 Blinded Policy Evaluation Phase

In this section, we outline two protocols for blinded policy evaluation. The first protocol requires $O(2^n)$ communication for a policy with $n$ attributes and the second requires a communication polynomial in $n$. However, the first protocol is for arbitrary functions and the second protocol is for a special class of functions that are useful in expressing policies. We formally define the input and output for this phase below. In the input and output description, we use $p(x_1, \ldots, x_n)$ to denote a policy. Recall that, given a policy $P$ on a set of credentials $C$, $P(C) = p(x_1, \ldots, x_n)$, where $x_i = 1$

if there is a credential in $C$ that matches with $cred_i$ in policy $P$ and $x_i = 0$ otherwise.

**Input.** Bob has a policy $p : \{0,1\}^n \rightarrow \{0,1\}$, several pairs of values $\{r_1[0], r_1[1]\}, \ldots, \{r_n[0], r_n[1]\}$, and a message $M$. Alice has $n$ values $\ell_1, \ldots, \ell_n$, where $\ell_i \in \{r_i[0], r_i[1]\}$.

**Output.** Alice learns $M$ if and only if $p(\ell_1 \stackrel{?}{=} r_1[1], \ldots, \ell_n \stackrel{?}{=} r_n[1]) = 1$ and learns nothing otherwise.

### 3.2.1 Arbitrary Policies

Recall that, after the previous phase, Alice has a set of values $\ell_1, \ldots, \ell_n$, where each of these is one of two keys[3] that Bob generated, i.e., $\ell_i$ is either $r_i[0]$ if Alice does not possess attribute $attr_i$ and is $r_i[1]$ if Alice does possess attribute $attr_i$.

**Protocol.**

1. For each binary string $\delta = x_1 x_2 \ldots x_n \in \{0,1\}^n$, Bob generates a value $\beta_{x_1 x_2 \ldots x_n}$ along with a public marker, $q$, where $q$ is of length $\rho$, the length of the security parameter. There are two cases for such a value:
   - $p(x_1, \ldots, x_n) = 1$:
   $$\beta_{x_1 x_2 \ldots x_n} = Enc\left(q \parallel k, \bigoplus_{i=1}^{n} (r_i[x_i])\right).$$
   - $p(x_1, \ldots, x_n) = 0$: $\beta_{x_1 x_2 \ldots x_n}$ is chosen to be a random value indistinguishable from an encryption with $Enc$.

   Bob randomly permutes the $\beta$ values and sends them to Alice, along with $q$, and $Enc(M, k)$.
2. Alice computes $k' = \bigoplus_{i=1}^{n} \ell_i$ and then computes $Dec(\beta, k')$ for each $\beta$ value. If she finds $q$, then she uses this value to obtain $M$.

**Complexity Analysis.** Note that the communication complexity of the above scheme is $O(2^n \rho)$ and, thus, is intractable for large policies. However, any computable policy function can be supported with such a scheme and this is impossible with polynomial computation.[4] In the following section, we look at a class of policies that can be computed with polynomial communication.

**Intuition.** The above is secure since Alice only knows one value for each of the attributes. Assuming a strong encryption system, Alice cannot deduce information about other keys given the encrypted messages.

### 3.2.2 Oblivious Circuit Policies

Typically, with Scrambled Circuit Evaluation (see Appendix A) during the construction of a scrambled circuit, the gates are constructed for some publicly defined function $f$; when the computation must be verified (i.e., in the presence of malicious adversaries), it is required that the generator of the circuit prove to the evaluator that the circuit is well-formed (i.e., that it computes $f$). However, there are cases where it is useful not to have the function publicly defined, especially when the function is private (as is the case with the application in this exposition). In this case, the construction can easily be modified to use *oblivious gates*, where the evaluator does not know the function that each gate computes.

**Protocol.**

1. Bob constructs a circuit $C$ that computes his policy (several "useful circuits" are described below) that uses the $r_i$ values as the input encodings and that has an output wire with two encodings: $k$ and some other random value. He sends the encodings of the circuit's gates to Alice (note that she already has input encodings) along with $Enc(M, k)$.
2. Alice evaluates the circuit and then tries to decrypt the message with the value she computes for the output wire of the circuit.

We now give several examples of such oblivious circuits:

1. A binary tree of oblivious gates could be used to compute several common types of policies, including: conjunction (does Alice have all of the attributes?), disjunction (does Alice have at least one of the attributes?), conjunction/disjunction of a subset (does Alice have all (one of) of a subset of the attributes), and other policies. Note that the size of this circuit is $O(\rho n)$.
2. An addition circuit followed by a comparison circuit would allow for computation of a threshold based function (i.e., "does Alice have at least four of the credentials"). This could easily be modified to support policies that require a threshold number of attributes for a subset of the attributes. Note that the size of the circuit is $O(\rho n)$.
3. By having one instance of each of the above, combined with a single oblivious gate, it is possible to have policies that are combinations of: conjunction, disjunction, and threshold-based. Note that the circuit size is still $O(\rho n)$.
4. More complex policies can be represented by combining several of the above-defined oblivious circuits together and then connecting them with other circuits. Of course, when this is done, the structure of the policy is revealed slightly (as the topography of the circuit is revealed). However, the structure is at a high level and the individual pieces of the policy are hidden (thus, the evaluator does not learn things like whether the binary tree being used is for conjunction/disjunction, the thresholds that values are being compared to, and which attributes are being used).
5. Another option is to use a universal circuit, as described by Valiant [30]. Recall that a universal circuit for a specific size $s$ and depth $d$ is a circuit that can be used to evaluate any circuit with these dimensions. Furthermore, there is such a universal circuit of size $O(ds \log s)$ and depth $O(d \log s)$. Thus, when such a universal circuit is used, it could represent any circuit of a certain size and depth.

3. We treat the values from the credential hiding phase as cryptographic keys; however, they could be seeds to a pseudorandom generator that is used to generate these keys.

4. As the function can be arbitrary, it could be any problem (including those not in $P$, such as the reachability problem for vector addition systems). Clearly, supporting any computable function requires super-polynomial computation.

Clearly, a large class of useful policies can be used with this technique and, while arbitrary functions cannot be computed, the communication requirements are polynomial in the number of attributes in Bob's policy.

### 3.3 Protocol Summary

In summary, there are two primary phases in our protocols: 1) a credential hiding phase and 2) a blinded policy evaluation phase. For the credential hiding phase, we gave three protocols that have a constant number of rounds and respective communication complexities: $O(\rho n)$, $O(\rho mn)$, and $O(\rho^2 mn)$. Furthermore, we gave two protocols for computing the blinded policy evaluation phase: 1) for arbitrary policies (using exponential communication) and 2) a limited, but useful, class of policies that can be computed with polynomial communication. In the next section, we discuss the security of our protocols in more detail.

## 4 Security Proofs

We begin by defining our security model (Section 4.1) and then prove that our system is secure in that model.

### 4.1 Security Definitions

In this section, we discuss our security model, which is that of [7], [16]. At a high level, a protocol securely implements a function $f$ if the information that can be learned by engaging in the protocol could be learned in an ideal implementation of the protocol where the functionality was provided by a trusted oracle. For more details, we refer the reader to [7], [16]. Throughout this paper, we assume that the adversary is a probabilistic polynomial-time algorithm, hence, the security of our problem is defined in a "computationally infeasible" sense. Having protocols that leak nothing other than what can be computed by the input and output alone are enough in Bob's case, but there are further restrictions on Alice. We must also show that Alice gets the message iff she satisfies Bob's policy.

**Composition Theorem.** While it is not always true that two secure protocols can be composed to create another secure protocol, the work in [7] gave conditions where composition is valid. If a protocol $\Pi'$ securely implements $g$ by making calls to trusted oracles (i.e., a trusted third party) for functions $f_1, \ldots, f_n$, then the protocol $\Pi$ that replaces the calls to the oracles by secure implementations of $f_1, \ldots, f_n$ will securely implement $g$. This composition theorem allows us to prove that many complex protocols are secure. There are composition theorems for semi-honest adversaries, malicious adversaries, and adaptive adversaries.

**Concurrent Execution.** There has been a large amount of work on developing protocols that are secure even if they are run in parallel with many other protocols, where the participants can be different in each protocol. While it has been shown that this is impossible for two-party computation, it was shown by [8] that, in the Common Reference String Model (i.e., where each party has a string common to all) that this is achievable. Of course, this model is unrealistic in many environments since the parties would have to agree on such a string beforehand and, so, no new parties could be added. Our protocols do not address such concerns, but it may be possible to extend our protocols with these techniques.

### 4.2 Assumptions/Security of Building Blocks

In this section, we introduce our assumptions about the security of various building blocks.

1. *Strong Chooser OT*. We assume the existence of an OT scheme where the chooser receives information about at most one value.
2. *Strong Sender OT*. We assume the existence of an OT scheme where the sender receives no (in a computational sense) information about which value the chooser picked.
3. *Strong Evaluator SCE*. We assume that Yao's Scrambled Circuit Evaluation has the following property: Assuming that the generator is honest, then a malicious evaluator will learn at most one encoding per wire for the circuit. This result assumes the *Strong Chooser OT* property.
4. *Secure Set Intersection*. We assume there exists a set intersection protocol $SetInt(x_B, S_A, E_A)$, where Bob inputs a value $x_B$ and Alice inputs a set $S_A$ and a semantically secure homomorphic encryption scheme $E_A$. In this protocol, Bob will learn a value $E_A(\gamma)$, where $\gamma$ will be 0 iff Alice's set $S_A$ contains $x_B$ and, otherwis,e $\gamma$ will be a value that is uniformly distributed over the base of the homomorphic scheme that Alice has no information about. It is worth pointing out that such a protocol easily follows from [14].

### 4.3 Proof of Credential Hiding Phase

We first prove that the protocols correctly implement input and output requirements given for the credential hiding phase (Section 3.1).

**Theorem 1.** *For each attribute, Alice can learn at most one value $r_i[0]$ or $r_i[1]$.*

**Proof.** For protocols CHP 1 and CHP 2, the only time that these values are used is in the last step of the protocol (i.e., the oblivious transfer). By the *Strong Chooser OT* property, the chooser (i.e., Alice) learns information about at most one value. This theorem holds for protocol CHP 3 because of the *Strong Evaluator SCE* property because this property implies that the evaluator (i.e., Alice) learns at most one encoding for the output wire of the circuit, which is the only place in the protocol where the values in question are used. □

**Theorem 2.** *For each attribute $attr_i$, Alice can learn $r_i[1]$ iff she has $attr_i$.*

**Proof.** We first prove this theorem for protocol CHP 1. Suppose Alice can obtain $r_i[1]$ with nonnegligible probability. Bob only uses $r_i[1]$ in the value $Enc(r_i[1], k_i)$ and, so, Alice must be able to learn $k_i$ with nonnegligible probability. However, the only other place where $k_i$ is used is in $I(k_i, nym \parallel attr_i)$ and, since Alice can learn $k_i$ with nonnegligible probability, she must be able to invert $I(k_i, nym \parallel attr_i)$ with nonnegligible probability. Thus, this implies that she must have the private key corresponding to $nym \parallel attr_i$, which implies that she has $attr_i$.

We now prove this theorem for protocol CHP 2. Again suppose Alice can obtain $r_i[1]$ with nonnegligible probability. As it does not affect the correctness of the proof, we ignore the permutation in Step 2 of the protocol. Bob uses $r_i[1]$ only in the value $Enc(r_i[1], k_i[1])$ and, so, Alice must be able to learn $k_i[1]$ with nonnegligible probability. However, for Alice to learn any information about $k_i[1]$, the value $\gamma_i$ must be 0 (if it is a randomly chosen value, then $\delta_i$ reveals no information about $k_i[1]$). Thus, by the *Secure Set Intersection* property, Alice must be able to have a set of values $\beta_{i,1}, \ldots, \beta_{i,m}$ that contains $k_i[0]$ with nonnegligible probability, but this implies that Alice knows $k_i[0]$ with nonnegligible probability. The rest of the proof is identical to end of the last case.

Finally, we prove this theorem for protocol CHP 3. Since Bob generated the circuit in Step 3 of the protocol, Alice will only obtain $r_i[1]$ iff her value $\eta_i$ is equal to $k_i[1]$. And, thus, if Alice can learn $r_i[1]$ with nonnegligible probability, then she can learn $k_i[1]$ with nonnegligible probability. The rest of the proof is identical to the end of the last case. □

We now prove that a malicious Alice or malicious Bob does not learn additional information about the other party's private input from each of the CHP protocols.

**Theorem 3.** *In protocol CHP 1, Alice learns only a superset of the attributes in Bob's policy and Bob learns no information about which credentials Alice possesses.*

**Proof.** The second part of the theorem is trivial since Bob's only communication from Alice is from the OT protocol, which reveals no information to Bob by the *Strong Sender OT* property. The first part is almost as trivial because the only information about Bob's policy used in this protocol is a superset of the attributes in his policy, but this is exactly the information that Alice learns. □

**Theorem 4.** *In protocol CHP 2, Alice learns how many attributes that she has in Bob's policy and Bob learns only the number of credentials that Alice uses.*

**Proof.** In Step 1, Bob sends to Alice $I(k_i[0], nym \parallel attr_i)$ for each attribute in his policy. Since $k_i[0]$ is a randomly chosen value, a correct decryption is indistinguishable from an incorrect decryption (assuming that decryption in IBE with the wrong key is a pseudorandom function). Thus, this first step reveals no information to Alice about which credentials are in Bob's policy. After Step 2, Bob sends to Alice in a permuted order a set of ordered pairs of the form $(\delta_j, Enc(q, k_j[1]))$, where $\delta_j = E_A(k_j[1])$ iff Alice has this attribute. From this information, Alice clearly learns when she has an attribute in Bob's policy, but the permuted order prevents her from learning which of her attributes is responsible for this. Finally, Alice sees the interaction from the OT protocol, but, since Bob's policy does not affect his input into this protocol, this reveals no information about his policy.

We now show the second part of the theorem. Bob's only communication from Alice is the set intersection protocol and the OT protocol. The first part of this is a set of values encrypted with a semantically secure homomorphic encryption scheme (which reveals no information to Bob by definition of semantic security) and the second set of values reveals nothing by the *Strong Sender OT* property. □

**Theorem 5.** *In protocol CHP 3, Alice learns how many attributes are in Bob's policy and Bob learns only the number of credentials that Alice uses.*

**Proof.** This proof is very similar to that of the last case; we only highlight the differences. The set of values $\delta_1, \ldots, \delta_n$ is given to Alice. By the *Secure Set Intersection* property, these values will either be $E_A(k_i[1])$ or $E_A(r_i + k_i[1])$, where $r_i$ is a uniformly chosen value. However, since $k_i[1]$ is a uniformly chosen value, these cases are indistinguishable. Her only other interaction is from being the evaluator in SCE, but, by the *Secure Evaluator SCE* property, she will learn at most one encoding for the output wire and, since the encodings are both randomly chosen values, she will not learn when she has an attribute.

We now show the second part of the theorem. The difference between this protocol and the last protocol is that Bob sees the communication from the Scrambled Circuit Evaluation, but this is just several OT protocols which reveal no information to him about Alice's values. □

## 4.4 Proof of Blinded Policy Evaluation

As both implementations are just SCE with the first protocol having a single $n$-ary gate and the second protocol revealing a topology of a circuit but hiding the values, all that needs to be shown is that Alice learns the message only when she has a set of values that correspond to satisfying the policy. As long as Alice knows only one value per input wire, then she cannot learn anything other than at most one value for the output wire. Since Bob generates the circuits to match his policy, the correctness is guaranteed. It is worth pointing out that, in the second case, the notion of policy indishtinguishability being used is that two policies are indistinguishable if and only if they can be represented by the circuit topology that is being used and they evaluate to the same value on the client's input.

## 4.5 Proof of Composition

We now show that the protocol is secure according to the definition presented in Problem 1.

**Theorem 6.** *Given trusted oracles $CHP$ and $BPE$, where $CHP$ provides the Credential Hiding Phase functionality and $BPE$ provides the Blinded Policy Evaluation functionality, the protocol is secure (in as strong of a model as the weakest of $CHP$ and $BPE$).*

**Proof.** The $CHP$ oracle requires that Bob input $n$ triples into the system $(attr_i, r_i[0], r_i[1])$, that Alice learns one of Bob's $r_i$ values per tuple, and that she learns the $r_i[1]$ only if she has a credential satisfying $attr_i$. Now, $BPE$ requires that Bob inputs pairs of the form $(r_i[0], r_i[1])$ and a policy $P$, that Alice inputs a single value from the pair $r_i[x_i]$, and that she gets $M$ iff she $P(x_1, \ldots, x_m) = 1$. In order to make multiple distinct probes at $BPE$, she must be able to learn two values from the same attribute pair, which is not possible since the $CHP$ oracle allows Alice to learn at most one value (the values can be chosen to be from a large enough space do as to make guessing computationally infeasible). Also, she obtains the message only when she has a set of attributes that matches the policy. Up to this point, we have been informal and have not discussed the effect of the information leaked by our protocols (e.g., an upper bound on the number of attributes in the policy). As this is a fairly simple extension of the above proof, we do not give all of the details here, but this information does not give the adversary an advantage in breaking the above system.□

## 5 Conclusion

We gave efficient protocols for Alice to access a resource from Bob such that Alice does not learn Bob's policy and Bob does not learn Alice's credentials. The only information Alice learns is whether she got access. Our protocols support different types of policies, from simple to arbitrarily complex. For many simple policies, our protocols require polynomial communication. However, for arbitrary policies, our protocols require communication exponential in the size of the policy.

## Appendix A
## Scrambled Circuit Evaluation

We review an efficient scheme for secure two-party circuit simulation in constant rounds (as in [35], for more details about this work and extensions of this work, see the excellent survey [15]). In this protocol, one party is a *generator* of a scrambled circuit and the other party is an *evaluator*. The generator creates a scrambled circuit where each wire of the circuit has two encodings (one for each possible value of the wire) and the gates contain information that allows an evaluator to obtain the encoding of the output wire when given the encoding for the gate's input wires. This is a private circuit evaluation because the evaluator learns only one encoding per wire. We now describe in more detail a protocol for Scrambled Circuit Evaluation.

**Circuit Generation.** For each wire in the circuit $w_1, \dots, w_n$, the generator creates random encodings for the wires (in this case, the encodings are a key for a trapdoor function or are a random seed that can be used by a pseudorandom number generator to generate such a key). We denote the encodings of 0 and 1 for wire $w_i$ by $w_i[0]$ and $w_i[1]$, respectively. To create a 2-ary gate for a function $f$ with input wires $w_i$ and $w_j$ and with output wire $w_k$, the gate consists of four messages (where $m$ is a publicly defined marker, used to recognize when an item has been successfully decrypted):

1. $Enc(Enc(m \parallel w_k[f(0,0)], w_j[0]), w_i[0])$,
2. $Enc(Enc(m \parallel w_k[f(0,1)], w_j[1]), w_i[0])$,
3. $Enc(Enc(m \parallel w_k[f(1,0)], w_j[0]), w_i[1])$, and
4. $Enc(Enc(m \parallel w_k[f(1,1)], w_j[1]), w_i[1])$.

Note that the scrambled gate consists of the above messages in a randomly permuted order. Clearly, a scrambled circuit with fan-in 2 can be represented in size proportional to the size of the original circuit. It is a natural extension of this to create an $n$-ary gate with $m$ outputs which can be encoded with size $2^n m$ and, while this has an exponential blowup in gate size, there are situations where this is useful.

**Learning Input Wires.** In order to evaluate a circuit, the evaluator must know the values of the input wires. For input wires corresponding to the generator's inputs, the generator simply sends the evaluator the encoding of each of his inputs. For input wires corresponding to the evaluator's inputs, the two parties engage in a 1-out-of-2 Oblivious Transfer(OT), where the two "messages" are the generator's encodings of 1 and 0 and the evaluator gets the encoding corresponding to his input for that wire.

**Evaluating the Circuit.** To evaluate a gate, the evaluator decrypts each message in the gate with the keys that it has for the input wires. Only one of these decrypted messages will contain the marker $m$ (the others will look random) and, thus, the evaluator will learn exactly one encoding for the output wire (he will know it is the correct value for that wire, but, of course, he cannot tell whether it corresponds to a 0 or a 1).

**Learning the Result.** If the goal is to have the evaluator simply learn the result, then it is enough for the generator to tell the evaluator both encodings of the output wire.

With passive adversaries, it is enough for the trapdoor function used for gate construction to be a strong encryption function (such as AES [10], which we assume to be a strong encryption function). In [19], this was extended to the malicious model and, so, our protocols are secure in a malicious-adversary sense. However, we now argue that Yao's passive construction is sufficient for our application. Essentially, the evaluator (Alice in our protocols) has no advantage in being malicious as long as the OT is secure. However, the generator can make the circuit for anything that matches the topology, but this is not a concern because the generator will never see the result of the circuit computation and, thus, this just gives more expressive power to the policy holder.

## Acknowledgments

Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the US National Science Foundation, Contract N00014-02-1-0364 from the US Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

## References

[1] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.-C. Wong, "Secret Handshakes from Pairing-Based Key Agreements," *Proc. IEEE Symp. Security and Privacy,* pp. 180-196, May 2003.
[2] M. Bellare and S. Micali, "Non-Interactive Oblivious Transfer and Applications," *Advances in Cryptology—CRYPTO 1989,* pp. 547-557, 1989.

[3] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," *Proc. IEEE Symp. Security and Privacy,* pp. 164-173, May 1996.

[4] P. Bonatti and P. Samarati, "Regulating Service Access and Information Release on the Web," *Proc. Seventh ACM Conf. Computer and Comm. Security,* pp. 134-143, Nov. 2000.

[5] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Advances in Cryptology—Proc. CRYPTO 2001,* pp. 213-229, 2001.

[6] R. Bradshaw, J. Holt, and K. Seamons, "Concealing Complex Policies with Hidden Credentials," *Proc. 11th ACM Conf. Computer and Comm. Security,* pp. 146-157, Oct. 2004.

[7] R. Canetti, "Security and Composition of Multiparty Cryptographic Protocols," *J. Cryptology,* vol. 13, no. 1, pp. 143-202, 2000.

[8] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, "Universally Composable Two-Party and Multi-Party Secure Computation," *Proc. 24th ACM Symp. Theory of Computing,* pp. 494-503, 2002.

[9] C. Cocks, "An Identity Based Encryption Scheme Based on Quadratic Residues," *Proc. Eighth IMA Int'l Conf. Cryptography and Coding,* pp. 360-363. Dec. 2001.

[10] J. Daemen and V. Jijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard.* Springer, 2002.

[11] I. Damgård and M. Jurik, "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System," *Proc. Fourth Int'l Workshop Practice and Theory in Public Key Cryptosystems,* pp. 119-136, 2001.

[12] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, *SPKI Certificate Theory.* IETF RFC 2693, Sept. 1999.

[13] S. Even, O. Goldreich, and A. Lempel, "A Randomized Protocol for Signing Contracts," *Comm. ACM,* vol. 28, no. 6, pp. 637-647, 1985.

[14] M. Freedman, K. Nissim, B. Pinkas, "Efficient Private Matching and Set Intersection," *Advances in Cryptology—Proc. EUROCRYPT 2004,* pp. 1-19, May 2004.

[15] O. Goldreich, "Cryptography and Cryptographic Protocols," *Distributed Computing,* vol. 16, nos. 2-3, pp. 177-199, 2003.

[16] O. Goldreich, *Foundations of Cryptography: Volume II Basic Application.* Cambridge Univ. Press, 2004.

[17] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority," *Proc. 19th ACM Symp. Theory of Computing,* pp. 218-229, May 1986.

[18] J.E. Holt, R.W. Bradshaw, K.E. Seamons, and H. Orman, "Hidden Credentials," *Proc. Second ACM Workshop Privacy in the Electronic Soc.,* pp. 1-8, Oct. 2003.

[19] J. Katz and R. Ostrovsky, "Round-Optimal Secure Two-Party Computation," *Advances in Cryptology—Proc. CRYPTO 2004,* pp. 335-354, 2004.

[20] N. Li, W. Du, and D. Boneh, "Oblivious Signature-Based Envelope," *Proc. 22nd ACM Symp. Principles of Distributed Computing,* pp. 182-189, July 2003.

[21] N. Li, J.C. Mitchell, and W.H. Winsborough, "Design of a Role-Based Trust Management Framework," *Proc. IEEE Symp. Security and Privacy,* pp. 114-130, May 2002.

[22] N. Li, W.H. Winsborough, and J.C. Mitchell, "Distributed Credential Chain Discovery in Trust Management," *J. Computer Security,* vol. 11, no. 1, pp. 35-86, Feb. 2003.

[23] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—A Secure Two-Party Computation System," *Proc. Usenix Security,* pp. 287-302, Aug. 2004.

[24] M. Naor and B. Pinkas, "Efficient Oblivious Transfer Protocols," *Proc. SIAM Symp. Discrete Algorithms,* pp. 448-457, Jan. 2001.

[25] T. Okamoto, S. Uchiyama, and E. Fujisaki, *Epoc: Efficient Probabilistic Public-Key Encryption Submission to IEEE p1363a.,* 1998.

[26] P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," *Advances in Cryptology—Proc. EUROCRYPT 1999,* pp. 223-238, 1999.

[27] R.L. Rivest and B. Lampson, *SDSI—A Simple Distributed Security Infrastructure,* Oct. 1996, http://theory.lcs.mit.edu/~rivest/sdsi11.html.

[28] K.E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation," *Proc. Symp. Network and Distributed System Security,* Feb. 2001.

[29] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," *Advances in Cryptology—Proc. CRYPTO 1984,* pp. 47-53, 1984.

[30] L. Valiant, "Universal Circuits (Preliminary Report)," *Proc. Eighth ACM Symp. Theory of Computing,* pp. 196-203, 1976.

[31] W.H. Winsborough and N. Li, "Towards Practical Automated Trust Negotiation," *Proc. Third Int'l Workshop Policies for Distributed Systems and Networks,* pp. 92-103, June 2002.

[32] W.H. Winsborough and N. Li, "Safety in Automated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy,* pp. 147-160, May 2004.

[33] W.H. Winsborough, K.E. Seamons, and V.E. Jones, "Automated Trust Negotiation," *Proc. DARPA Information Survivability Conf. and Exposition,* vol. I, pp. 88-102, Jan. 2000.

[34] M. Winslett, T. Yu, K.E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating Trust on the Web," *IEEE Internet Computing,* vol. 6, no. 6, pp. 30-37, Nov. 2002.

[35] A. Yao, "How to Generate and Exchange Secrets," *Proc. 27th IEEE Symp. Foundations of Computer Science,* pp. 162-167, 1986.

[36] A.C. Yao, "How to Generate and Exchange Secrets," *Proc. 27th IEEE Symp. Foundations of Computer Science,* pp. 162-167, 1986.

[37] T. Yu and M. Winslett, "A Unified Scheme for Resource Protection in Automated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy,* pp. 110-122, May 2003.

[38] T. Yu, M. Winslett, and K.E. Seamons, "Interoperable Strategies in Automated Trust Negotiation," *Proc. Eighth ACM Conf. Computer and Comm. Security,* pp. 146-155, Nov. 2001.

**Keith Frikken** received the BS degree in computer science and mathematics from Winona State University in 2000 and he received the PhD degree in 2005. His research interests include information security, cryptographic protocols, access control, privacy, and digital rights management. In the fall of 2006, he joined Miami University as an assistant professor. He is a member of the IEEE.

**Mikhail ("Mike") Atallah** received the PhD degree in 1982 from the Johns Hopkins University and joined the Computer Sciences Department at Purdue University, where he currently holds the rank of Distinguished Professor. He has served on the editorial boards of many top journals (including *SIAM Journal on Computing*, *Journal of Parallel and Distributed Computig*, *IEEE Transactions on Computers*, etc.), and on the program committees of many top conferences and workshops (including PODS, SODA, SoCG, WWW, PET, DRM, SACMAT, etc.). He has been the keynote and invited speaker at many national and international meetings and a speaker in the Distinguished Colloquium Series of six top Computer Science Departments. He was selected in 1999 as one of the best teachers in the history of Purdue University and included in Purdue's Book of Great Teachers, a permanent wall display of Purdue's best teachers past and present. He is a cofounder of Arxan Technologies Inc. He is a fellow of the IEEE.

**Jiangtao Li** received the BS degree from the University of Science and Technology of China in 1999. He received the MS and PhD degrees in 2002 and 2006, respectively, both from Purdue University. His research interests are in the area of information security, in particular, access control, applied cryptography, and privacy. He will join Intel Corporation as a security architect. He is a student member of the IEEE.