

OACerts: Oblivious Attribute Certificates

Jiangtao Li and Ninghui Li

CERIAS and Department of Computer Science, Purdue University
250 N. University Street, West Lafayette, IN 47907
{jtli, ninghui}@cs.purdue.edu

Abstract. We propose Oblivious Attribute Certificates (OACerts), an attribute certificate scheme in which a certificate holder can select which attributes to use and how to use them. In particular, a user can use attribute values stored in an OACert obliviously, i.e., the user obtains a service if and only if the attribute values satisfy the policy of the service provider, yet the service provider learns nothing about these attribute values. This way, the service provider’s access control policy is enforced in an oblivious fashion.

To enable the oblivious access control using OACerts, we propose a new cryptographic primitive called Oblivious Commitment-Based Envelope (OCBE). In an OCBE scheme, Bob has an attribute value committed to Alice and Alice runs a protocol with Bob to send an envelope (encrypted message) to Bob such that: (1) Bob can open the envelope if and only if his committed attribute value satisfies a predicate chosen by Alice, (2) Alice learns nothing about Bob’s attribute value. We develop provably secure and efficient OCBE protocols for the Pedersen commitment scheme and predicates such as $=$, \geq , \leq , $>$, $<$, \neq as well as logical combinations of them.

1 Introduction

In Trust Management and certificate-based access control systems [3, 19, 11, 35, 34], access control decisions are based on attributes of requesters, which are established by digitally signed certificates. Each certificate associates a public key with the key holder’s identity and/or attributes such as employer, group membership, credit card information, birth-date, citizenship, and so on. Because these certificates are digitally signed, they can serve to introduce strangers to one another without online contact with the attribute authorities. Privacy becomes an important concern in the use of Internet and web services. When the attribute information in a certificate is sensitive, the certificate holder may want to disclose only the information that is absolutely necessary to obtain services. Consider the following example.

Example 1. A senior citizen Bob requests from a service provider Alice a document that can be accessed freely by senior citizens. Bob wants to use his digital driver license to prove that he is entitled to free access. Bob’s driver license certificate has fields for an identification number, expiration date, name, address, birth-date, and so on; and Bob would like to reveal as little information as possible.

In the above example, it might seem that Bob needs to reveal at least the fact that he is a senior citizen, i.e., his birth-date is before a certain date. However, even this seemingly minimal amount of information disclosure can be avoided. Suppose that the

document is encrypted under a key and the encrypted document is freely available to everyone. Further suppose a protocol exists such that after the protocol is executed between Alice and Bob, Bob obtains the key if and only if the birth-date in his driver license is before a certain date and Alice learns nothing about Bob’s birth-date. Under these conditions, Alice can perform access control based on Bob’s attribute values while being oblivious about Bob’s attribute information.

We call this *oblivious access control*, because Alice’s access control policies for her resources are enforced without Alice learning any information about Bob’s certified attribute values, not even whether Bob satisfies her policy or not. To enable such oblivious access control, we propose Oblivious Attribute Certificates (OACerts), a scheme for using certificates to document sensitive attributes. The basic idea of OACerts is quite simple. Instead of storing attribute values directly in the certificates, a certificate authority (CA) stores the cryptographic commitments [39, 23, 12, 16] of these values in the certificates. Using OACerts, a user can select *which* attributes to use as well as *how* to use them. An attribute value in an OACert can be used in several ways: (1) by opening a commitment and revealing the attribute value, (2) by using zero-knowledge proof protocols [13, 37, 18, 5] to prove that the attribute value satisfies a condition without revealing other information, and (3) by running a protocol so that the user obtains a message only when the attribute value satisfies a condition, without revealing any information about the attribute value. The idea of storing cryptographic commitments of attribute values in certificates was used in anonymous credentials [10, 7, 36, 9, 8]; however, we are not aware of prior work on the oblivious usage of such attribute values.

In Example 1, suppose that the driver-license certificate that Bob has is an OACert. With attribute values committed rather than stored in the clear in her certificates, Bob can send his certificate to Alice without revealing his birth-date or any other attribute information. Using zero-knowledge proof protocols [13, 37, 18, 5], Bob can prove to Alice that his committed birth-date is before a certain date without revealing any other information. However, our goal is that Alice should learn nothing about Bob’s birth-date, not even whether Bob is a senior citizen or not. To enable oblivious access control, we need to solve the following two-party Secure Function Evaluation (SFE) problem:

Problem 1. Let commit be a commitment algorithm, let Params be public parameters for commit , and Pred be a public predicate. Let a be a private number (Bob’s attribute value), $c = \text{commit}_{\text{Params}}(a, r)$ be a commitment of a under the parameters Params with a random number r , and M be a private message (Alice wants Bob to see M if and only if a satisfies Pred). Alice and Bob jointly compute a family F of functions, parameterized by commit and Pred . Both parties have commit , Pred , Params , and c . Alice has private input M . Bob has private input a and r . The function F is defined as follows.

$$\begin{aligned} F[\text{commit}, \text{Pred}]_{\text{Alice}}(\text{Params}, c, M, a, r) &= 0 \\ F[\text{commit}, \text{Pred}]_{\text{Bob}}(\text{Params}, c, M, a, r) &= \begin{cases} M & \text{if } c = \text{commit}_{\text{Params}}(a, r) \wedge \text{Pred}(a) = \text{true}; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

where $F[\text{commit}, \text{Pred}]_{\text{Alice}}$ represents Alice’s output, $F[\text{commit}, \text{Pred}]_{\text{Bob}}$ represents Bob’s output. In other words, our goal is that Alice learns nothing and Bob learns M only when his committed attribute value satisfies the predicate Pred .

The preceding problem can be solved using general solutions to two-party SFE [44, 26, 25]; however, the general solutions are inefficient, as commitment verification is done within the SFE. We propose an Oblivious Commitment Based Envelope (OCBE) scheme that solves Problem 1 efficiently. Formal definition of OCBE will be given in Section 4. Informally, an OCBE scheme enables a sender Alice to send an envelope (encrypted message) to a receiver Bob, such that Bob can open the envelope if and only if his committed value satisfies the predicate. An OCBE scheme is *oblivious* if at the end of the protocol the sender cannot learn any information about the receiver’s committed value. An OCBE scheme is *secure against the receiver* if a receiver whose committed value does not satisfy the predicate cannot open the envelope.

We develop efficient OCBE protocols for the Pedersen commitment scheme [39] and six kinds of comparison predicates: $=$, \neq , $<$, $>$, \leq , \geq , as well as conjunctions and disjunctions of multiple predicates. These predicates seem to be the most useful ones for testing attribute values in access control policies. We present a protocol (called EQ-OCBE) for equality predicates and a protocol (called GE-OCBE) for greater-than-or-equal-to predicates and prove that these protocols are provably secure in the Random Oracle Model [2]. These protocols use cryptography hash functions to efficiently derive symmetric encryption keys from a shared secret, and random oracles are used to model such usage of hash functions. We also show that it is easy to construct OCBE protocols for other comparison predicates using variants of EQ-OCBE, GE-OCBE. The contributions of this paper are as follows.

- We introduce the notions of OACerts and OCBE, which together enable oblivious access control. OACerts and OCBE may be of interests in other applications as well.
- We present efficient and provably secure OCBE protocols for the Pedersen commitment scheme [39] and several kinds of comparison predicates.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the architecture and application of OACerts. Section 4 gives a formal definition of OCBE. Section 5 reviews the Pedersen commitment scheme. Section 6 presents several efficient and provably secure OCBE protocols. Section 7 describes our implementation and performance measurements. Section 8 concludes our paper.

2 Related Work

Recent works on using cryptographic protocols for certificate-based access control include Hidden Credentials [28, 6, 22], Secret Handshakes [1], and Oblivious Signature Based Envelope [32]. Using any of these schemes, the service provider Alice could send an encrypted message to a client Bob such that Bob can decrypt if and only if he has certificates whose contents are the same as those identified by Alice’s policy; at the same time, Alice does not know whether Bob has those certificates or not. (In Secret handshakes [1], Alice computes a key such that Bob can compute if and only if Bob has the required certificate.) These schemes can implement oblivious access control when Alice’s policies have very specific forms. In Example 1, if Alice’s policy is that Bob’s birth-date is April 1st, 1974, then oblivious access control can be achieved using these existing schemes, as Alice could identify the contents of the certificates that

would enable Bob to satisfy her policy. However, for the policy in Example 1 (birth-date in a certain range) where many possible attribute values would satisfy a policy, these schemes do not work well.

Our work is also closely related to anonymous credentials [10, 7, 36, 9, 8]. Indeed, the ideas of storing commitments of attribute values in certificates and using zero-knowledge proofs to prove properties of these values appeared in the literature on anonymous credentials, e.g. [7]. These schemes differ from OACerts in that they provide orthogonal privacy protections. None of the existing anonymous credential schemes enables oblivious access control as the verifier learns whether the prover satisfies her policy or not. On the other hand, anonymous credentials enable Bob to use a credential anonymously, i.e., Alice and other service providers cannot link together transactions in which Bob's credential is used. For such protection to make sense, anonymous communication channels are required. The OACerts scheme does not provide anonymity protection and therefore does not require anonymous communication channels. Furthermore, anonymous credential schemes tend to involve protocols dramatically different from existing public-key infrastructure standards, whereas the OACerts scheme is compatible with existing standards, such as X.509 [29].

Crescenzo et al. [15] introduced a variant of oblivious transfer called Conditional Oblivious Transfer (Conditional OT), in which Alice and Bob each has a private input and shares with each other a public predicate that is evaluated over the private inputs. In the Conditional OT of a bit b from Alice to Bob, Bob receives the bit only when the predicate holds; furthermore, Alice learns nothing about Bob's private input or the output of the predicate. Crescenzo et al. [15] developed an efficient protocol for a special case of Conditional OT where the predicate is greater-than-or-equal-to. OCBE can be viewed as another special case of the Conditional OT problem, in which Alice has no private inputs, the commitment c of Bob's private input a is made public, and the public predicate for this Conditional OT is a conjunction of two conditions: (1) Bob's private input a must be the value he committed in c , and (2) Bob's private input a must also satisfy a predicate (e.g., greater-than-or-equal-to some value). The additional requirement of (1) makes OCBE quite different from Conditional OT for greater-than-or-equal-to predicate; therefore, the solution in [15] cannot apply to OCBE.

Crépeau [14] introduced the notion of Committed Oblivious Transfer (COT). In COT, Alice commits two bits: a_0 and a_1 , and Bob commits a bit b . All three committed values are public knowledge. The goal of COT is enable Bob to learn a_b without learning anything else, while Alice learns nothing. Garay et al. [24] gave an efficient construction of COT in the universal composability framework. OCBE differs from COT in that Bob's input in OCBE is an integer whereas Bob's input in COT is a single bit. Furthermore, because the predicate in OCBE could be arbitrary, results in COT cannot apply directly to our OCBE protocol.

Our work is related to zero-knowledge proof protocols [13, 37, 18, 5] that prove a committed value satisfies some property. Our GE-OCBE protocol has similarities with the range proof protocol in [37, 18], which proves that a committed value lies within a range. Also, the details of our GE-OCBE protocol are reminiscent of the techniques used in the oblivious transfer protocols [38, 41] and the comparison method for millionaires [21].

3 Architecture and Applications of OACerts and OCBE

In this section, we present the architecture of OACerts and OCBE and outline their applications.

Architecture There are three kinds of parties in the OACerts scheme: certificate authorities (CA's), certificate holders, and service providers. A CA issues OACerts for certificate holders. Each CA and each certificate holder has a unique public-private key pair. A service provider, when providing services to a certificate holder, performs access control based on the attributes of the certificate holder, as certified in OACerts.

An OACert is a digitally signed assertion about the certificate holder by a CA. Each OACert contains one or more attributes. We use $attr_1, \dots, attr_m$ to denote the m attribute names in an OACert, and v_1, \dots, v_m to denote the corresponding m attribute values. Let $c_i = \text{commit}_{\text{Params}}(v_i, r_i)$ be the commitment of attribute value v_i for $1 \leq i \leq m$ with r_i being the secret random number. The attribute part of the certificate consists of a list of m entries, each entry is a tuple $\langle attr_i, c_i \rangle$. When the commitment scheme used is secure, the certificate itself does not leak any information about the sensitive attributes. Thus, an OACert's content can be made public. A certificate holder can show his OACerts to others without worrying about the secrecy of his attributes.

OACerts can be implemented on existing public-key infrastructure standards, such as X.509 Public Key Infrastructure Certificate [4, 29] and X.509 Attribute Certificate [20]. The commitments can be stored in X.509v3 extension fields, in which case a certificate includes also the following fields: serial number, validity period, issuer name, user name, certificate holder's public key, and so on. The distribution and revocation of OACerts can be handled using existing infrastructure and techniques. See Section 7 for our implementation and performance measurements of OACerts.

There are four basic protocols in the OACerts scheme:

- **CA-Setup:** A CA picks a signature scheme Sig with a public-private key pair (K_{CA}, K_{CA}^{-1}) , and a commitment scheme commit with public parameters Params. The public parameters of the CA are $\{\text{Sig}, K_{CA}, \text{commit}, \text{Params}\}$.
- **Issue Certificate:** A CA uses this protocol to issue an OACert to a user. A user Bob generates a public-private key pair (K_B, K_B^{-1}) and sends to the CA a certificate request that includes his public key K_B and attributes information $(attr_1, v_1), \dots, (attr_m, v_m)$, and is signed by K_B^{-1} . After the CA verifies the correctness of v_1, \dots, v_m (most likely using off-line methods), it issues an OACert for Bob. In this process, the CA computes $c_i = \text{commit}_{\text{Params}}(v_i, r_i)$ and sends the certificate along with the secrets r_1, \dots, r_m to Bob. Bob stores the certificate and stores the values $(v_1, r_1), \dots, (v_m, r_m)$ together with his private key K_B^{-1} . The role of the CA here is similar to the role of a CA in the traditional Public Key Infrastructure.
- **Alice-Bob initialization:** Bob, a certificate holder, establishes a secure communication channel with Alice, a service provider, and at the same time proves to Alice the ownership of an OACert. In this protocol, Alice checks the signature and the validity period of the certificate, then verifies that the certificate has not been revoked (using, e.g., standard techniques in [29]). Alice also verifies that Bob possesses the

private key corresponding to K_B in the OACert. All these can be done using standard protocols such as TLS/SSL [40]. Bob then requests the decryption key for an encrypted document, and Alice sends Bob her policy.

- **Alice-Bob Interaction:** Alice and Bob runs an interactive protocol so that in the end Bob obtains the decryption key if and only if his committed attribute satisfies Alice’s policy.

Applications of OACerts In addition to enabling oblivious access control, OACerts and OCBE can also be used to break policy cycles (see [32] for definition) in automated trust negotiation [43, 45, 42]. Consider the following scenario where Alice and Bob want to exchange their salary certificates. Alice’s policy says that she can show her salary certificate only to those whose salary is great than \$100k. Similarly, Bob will reveal his certificate only to other who earns more than \$80k a year. Using current trust negotiation techniques, neither Alice nor Bob is willing to present her/his certificate first. The technique developed in [32] does not work well here neither, because the salary requirement in the policies is a range, not a specific value. Such problems can be solved using OACerts and OCBE.

4 Definition of Oblivious Commitment-Based Envelope (OCBE)

We now give a formal definition of OCBE. While the definition follows the usage scenario described in Section 3 in general, it abstracts away some of the details in the scenario that have been solved using OACerts and focuses on the parts that still need to be solved by the OCBE protocol.

Definition 1 (OCBE). An Oblivious Commitment-Based Envelope (OCBE) scheme is parameterized by a commitment scheme `commit`. It involves a sender S , a receiver R , and a trusted CA, and has the following phases:

CA-Setup CA takes a security parameter t and outputs the following: the public parameters `Params` for `commit`, a set \mathcal{V} of possible values, and a set \mathcal{P} of predicates. Each predicate in \mathcal{P} maps an element in \mathcal{V} to either true or false. The domain of `commit[Params]` contains \mathcal{V} as a subset.

CA-Commit R chooses a value $a \in \mathcal{V}$ (R ’s attribute value) and sends to CA. CA picks a random number r and computes the commitment $c = \text{commit}_{\text{Params}}(a, r)$. CA gives c and r to R , and c to S .

Recall that in the actual usage scenario, CA does not directly communicate with R . Instead, CA stores the commitment c in R ’s OACert certificate. The certificate is then sent by R to S , enabling S to have c as if it is sent from CA. Here we abstract these steps away to have CA sending c to S . We stress that CA does *not* participate in the interactions between S and R .

Initialization S chooses a message $M \in \{0, 1\}^*$. S and R agree¹ on a predicate $\text{Pred} \in \mathcal{P}$.

Now S has Pred , c , and M . R has Pred , c , a , and r .

¹ The main effect of having both the sender and the receiver to affect the predicate is that in the security definitions both an adversarial sender and an adversarial receiver can choose the predicate they want to attack on.

Interaction S and R run an interactive protocol, during which an envelope containing an encryption of M is delivered from S to R .

Open After the interaction phase, if $\text{Pred}(a)$ is true, R outputs the message M ; otherwise, R does nothing.

Observe that the receiver R 's attributed value a is committed by a trusted CA. This is natural (and necessary) in our intended usage scenarios for OCBE.

4.1 Basic Cryptographic Assumptions

We say that a function f is *negligible* in the security parameter t if, for every polynomial p , $f(t)$ is smaller than $1/|p(t)|$ for large enough t ; otherwise, it is *non-negligible*. The security of our OCBE protocols is based on two standard assumptions in cryptography and the random oracle model.

- *Discrete Logarithm (DL) Assumption*. Given a finite cyclic group G , a generator $g \in G$, and a group element y , there exists no polynomial-time algorithm that can compute $\log_g y$ with non-negligible probability.
- *Computational Diffie-Hellman (CDH) Assumption*. Given a finite cyclic group G , a generator $g \in G$, and group elements g^a, g^b , there exists no polynomial-time algorithm that can compute g^{ab} with non-negligible probability.
- *Random Oracle Model*. The random oracle model is an idealized security model introduced by Bellare and Rogaway [2]. Roughly speaking, a random oracle is a function $H: X \rightarrow Y$ chosen uniformly at random from the set of all functions $\{h: X \rightarrow Y\}$. Random oracles are used to model cryptographic hash functions such as SHA-1.

4.2 Security Definitions

Let an *adversary* be a probabilistic interactive Turing Machine [27]. An OCBE scheme must satisfy the following three properties. It must be sound, oblivious, and semantically secure against the receiver.

Sound An OCBE scheme is *sound* if in the case that $\text{Pred}(a)$ is true, the receiver can output the message M with overwhelming probability, i.e., the probability that the receiver cannot output M is negligible.

Oblivious An OCBE scheme is *oblivious* if the sender learns nothing about a , i.e., no adversary \mathcal{A} has a non-negligible advantage against the challenger in the game described in Figure 1 where the challenger emulates CA and the receiver, and the adversary emulates the sender. In other words, an OCBE scheme is *oblivious* if for every probabilistic interactive Turing Machine \mathcal{A} , $|\Pr[\mathcal{A} \text{ wins the game in Figure 1}] - \frac{1}{2}| \leq f(t)$, where f is a negligible function in t .

Secure against the receiver An OCBE scheme is *secure against the receiver* if the receiver learns nothing about M when $\text{Pred}(a)$ is false, i.e., no adversary \mathcal{A} has a non-negligible advantage against the challenger in the game described in Figure 2 where the challenger emulates CA and the sender, and the adversary emulates the receiver.

We now argue that OCBE is an adequate solution to the two-party SFE problem in Problem 1, by showing intuitively that the security properties defined for OCBE suffice to prove that the scheme protects the privacy of the participants in the malicious

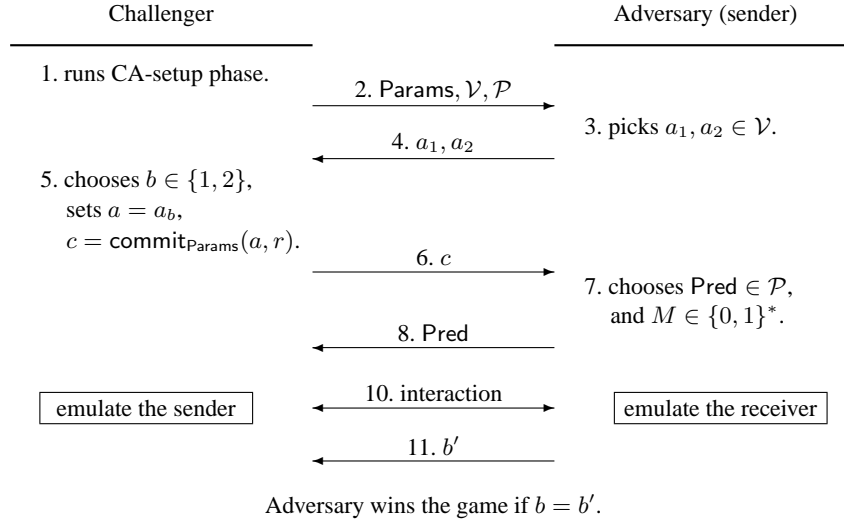


Fig. 1. The attacker game for OCBE’s oblivious property. We allow the adversary to pick a predicate Pred and two attribute values a_1, a_2 of her choice; yet the adversary still should not be able to distinguish a receiver with attribute a_1 from one with attribute a_2 .

model [25]. Observe that our definitions allow arbitrary adversaries, rather than just those following the protocol (semi-honest adversaries). The oblivious property guarantees that the sender’s view of any protocol run can be simulated using just the sender’s input, because one can simulate a protocol run between the sender and receiver, and no polynomially bounded sender can figure out the receiver’s input. Soundness and security against the receiver guarantee that the receiver’s view can be simulated using just the receiver’s input and output. If the receiver’s committed value a satisfies Pred , then the message M is in the output, one can therefore simulate the sender S . If the receiver’s committed value a does not satisfy Pred , one can simulate the sender with an arbitrary message M' and no polynomially bounded receiver can tell the difference.

The security properties defined for OCBE guarantee also the correctness [25] of the OCBE protocol against malicious receivers. Our security definitions do not cover the correctness of the protocol against malicious senders, i.e., if the receiver’s value does not satisfy the predicate, a malicious sender may trick the receiver to output the message M which violates the correctness of the protocol². However, this malicious behavior does not make sense in the applications. If a malicious sender does not want to send the message M , she can choose not to participate in the protocol; on the other hand, if a malicious sender wants the receiver to see M without satisfying her policy; she can choose to send M directly rather than participating in the protocol.

We assume that the interaction phase of the OCBE scheme is executed on top of a previously established private communication channel between the sender and the receiver. Recall that the certificate holder establishes an SSL channel with the service provider using OACerts described in Section 3.

² In such case, the views of the sender and receiver cannot be simulated in the ideal model.

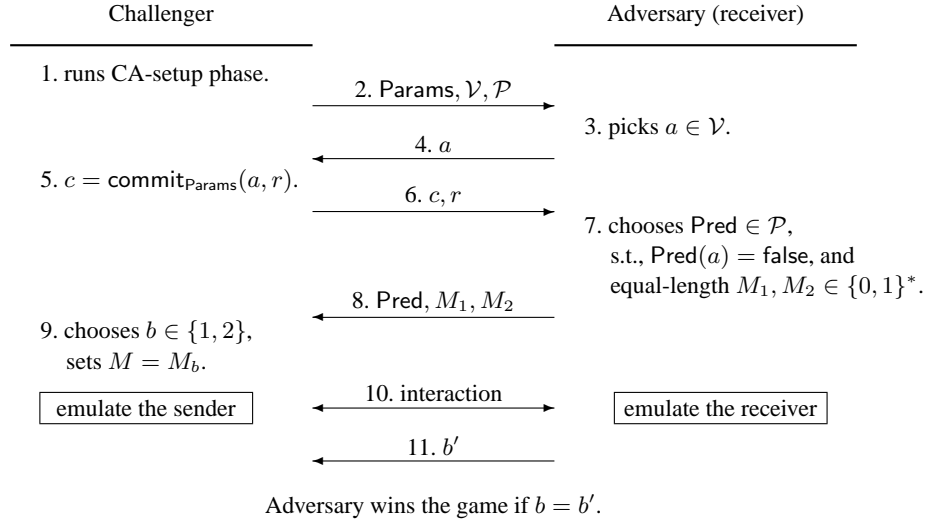


Fig. 2. The attacker game for OCBE’s security property against the receiver. Even if we give the adversary the power to pick two equal-length messages M_1 and M_2 of her choice, she still cannot distinguish an envelope containing M_1 from one containing M_2 . This formalizes the intuitive notion that the envelope leaks no information about its content.

5 The Pedersen Commitment Scheme

We now review the Pedersen commitment scheme [39], which will be used in the OCBE protocols.

Definition 2 (The Pedersen Commitment Scheme).

Setup A trusted third party T chooses two large prime numbers p and q such that q divides $p - 1$. It is typical to have p be 1024 bits and q be 160 bits. Let g be a generator of G_q , the unique order- q subgroup of \mathbb{Z}_p^* . We use $x \leftarrow \mathbb{Z}_q$ to denote that x is uniformly randomly chosen from \mathbb{Z}_q . T picks $x \leftarrow \mathbb{Z}_q$ and computes $h = (g^x \bmod p)$. T keeps the value x secret and makes the values p, q, g, h public.

Commit The domain of the committed values is \mathbb{Z}_q . For a party A to commit an value $a \in \mathbb{Z}_q$, A chooses $r \leftarrow \mathbb{Z}_q$ and computes the commitment $c = (g^a h^r \bmod p)$.

Open To open a commitment c , A reveals a and r , and a verifier verifies whether $c = (g^a h^r \bmod p)$.

The above setting is slightly different from the standard setting of commitment schemes, in which the verifier runs the setup program and does a zero-knowledge proof to convince A that the parameters are constructed properly. We use a trusted third party to generate the parameters, because this is done by a trusted CA in the OACerts scheme.

The Pedersen commitment scheme is *unconditionally hiding*: Even with unlimited computational power it is impossible for an adversary to learn any information about the value a from c , because the commitments of any two numbers in \mathbb{Z}_q have exactly the same distribution. This commitment scheme is *computationally binding*: Under the DL assumption, it is computationally infeasible for an adversarial committer to open a value

a' other than a in the open phase of the commitment scheme. Suppose an adversary finds a' (other than a) and r' such that $g^{a'}h^{r'} \equiv g^a h^r \pmod{p}$, then she can compute $\frac{a'-a}{r-r'} \pmod{q}$, which is $\log_g(h)$, the discrete logarithm of h with respect to the base g .

6 OCBE Protocols

In this section, we present two OCBE protocols using the Pedersen commitment scheme, one for equality predicates, the other for greater-than-or-equal-to predicates. We then sketch how to construct OCBE protocols for other comparison predicates. All arithmetic in this section is assumed to be mod p unless otherwise specified.

6.1 EQ-OCBE: an OCBE protocol for equality predicates

Our EQ-OCBE protocol runs a Diffie-Hellman style key-agreement protocol [17] with the twist that the receiver can compute the shared secret if and only if the receiver's committed value a is equal to a_0 .

Protocol 1 (EQ-OCBE) Let \mathcal{E} be a semantically secure symmetric encryption scheme with keyspace $\{0, 1\}^s$. Let $H : G_q \rightarrow \{0, 1\}^s$ be a cryptographic hash function that extracts a key for \mathcal{E} from an element in the group G_q , the order- q subgroup of \mathbb{Z}_p^* . EQ-OCBE involves a sender S , a receiver R , and a trust CA.

CA-Setup CA takes a security parameter t and runs the setup algorithm of the Pedersen commitment scheme to create $\text{Params} = \langle p, q, g, h \rangle$. CA also outputs $\mathcal{V} = \mathbb{Z}_q$ and $\mathcal{P} = \{\text{EQ}_{a_0} \mid a_0 \in \mathcal{V}\}$, where $\text{EQ}_{a_0} : \mathcal{V} \rightarrow \{\text{true}, \text{false}\}$ is a predicate such that $\text{EQ}_{a_0}(a)$ is true if $a = a_0$ and false if $a \neq a_0$.

CA-Commit R chooses an integer $a \in \mathcal{V}$ and sends to CA. CA picks $r \leftarrow \mathbb{Z}_q$ and computes the commitment $c = g^a h^r$. CA gives c and r to R , and c to S .

Initialization S chooses a message $M \in \{0, 1\}^*$. S and R agree on a predicate $\text{EQ}_{a_0} \in \mathcal{P}$. Now S has EQ_{a_0} , c , and M . R has EQ_{a_0} , c , a , and r .

Interaction S picks $y \leftarrow \mathbb{Z}_q^*$, computes $\sigma = (cg^{-a_0})^y$, and then sends to R the pair $\langle \eta = h^y, C = \mathcal{E}_{H(\sigma)}[M] \rangle$.

Open R receives $\langle \eta, C \rangle$ from the interaction phase. If $\text{EQ}_{a_0}(a)$ is true, R computes $\sigma' = \eta^r$, and decrypts C using $H(\sigma')$.

To see that EQ-OCBE is sound, observe that when $\text{EQ}_{a_0}(a)$ is true,

$$\sigma = (cg^{-a_0})^y = (g^a h^r g^{-a_0})^y = (g^{a-a_0} h^r)^y = (h^r)^y = (h^y)^r = \eta^r = \sigma'.$$

Therefore the sender and receiver share the same symmetric key.

Also observe that the interaction phase of the EQ-OCBE protocol is one-round; it involves only one message from the sender to the receiver. In the interaction and open phases, the sender does two exponentiations and the receiver does one exponentiation.

The key idea of EQ-OCBE is that if the receiver's committed value a is equal to a_0 , the sender can compute $cg^{-a_0} = g^{a-a_0} h^r = h^r$. The sender now holds h^r such that the receiver knows the value r . This achieves half of the Diffie-Hellman key-agreement protocol [17], with h as the base. The sender then does the other half by sending h^y to the receiver. Thus both the sender and receiver can compute $\sigma = (cg^{-a_0})^y = h^{ry}$. If the receiver's committed value a is not equal to a_0 , then it is presumably hard for him

to compute $\sigma = (cg^{-a_0})^y$ from h^y and cg^{-a_0} . The receiver cannot effectively compute $\log_h(cg^{-a_0})$, because if the receiver is able to find a number $r' = \log_h(cg^{-a_0})$, he can break the binding property of the commitment scheme, i.e., he finds a (a_0, r') pair such that $g^{a_0}h^{r'} = g^a h^r$.

Theorem 1. *EQ-OCBE is oblivious.*

Theorem 2. *Under the CDH assumption on G_q , the order- q subgroup of \mathbb{Z}_p^* , and when H is modeled as a random oracle, EQ-OCBE is secure against the receiver.*

Due to space limitation, all the proofs are given in the full version of this paper [31].

6.2 GE-OCBE: an OCBE protocol for greater-than-or-equal-to predicates

In this section, we present an OCBE protocol (GE-OCBE) for the Pedersen commitment scheme with greater-than-or-equal-to predicates. The basic idea of the GE-OCBE protocol is as follows. Let ℓ be an integer such that $2^\ell < q/2$. Let a and a_0 be two numbers in $[0..2^\ell - 1]$, and let $d = ((a - a_0) \bmod q)$. Let $c = g^a h^r$ be a commitment of a where r is known to the receiver, then $cg^{-a_0} = g^{a-a_0} h^r = g^d h^r$ is a commitment of d that the receiver knows how to open. Notice that if $a \geq a_0$ then $d \in [0..2^\ell - 1]$, otherwise $d \notin [0..2^\ell - 1]$.

If $a \geq a_0$, the receiver generates ℓ new commitments $c_0, \dots, c_{\ell-1}$, one for each of the ℓ bits of d . The sender picks a random encryption key k and split it into ℓ secrets $k_0, \dots, k_{\ell-1}$. Then the sender and receiver run a “bit-OCBE” protocol for each commitment, i.e., if c_i is a bit-commitment, the receiver obtains k_i , otherwise he gets nothing, while the sender learns nothing about the value committed under c_i .

Protocol 2 (GE-OCBE) Let \mathcal{E} be a semantically secure symmetric encryption scheme with keyspace $\{0, 1\}^s$. Let $H : G_q \rightarrow \{0, 1\}^s$ and $H' : \{0, 1\}^{s\ell} \rightarrow \{0, 1\}^s$ be two cryptographic hash functions. Our GE-OCBE protocol involves a sender S , a receiver R , and a trust CA.

CA-Setup CA takes two parameters, a security parameter t and a parameter ℓ (which specifies the desired range of the attribute values). CA runs the setup algorithm of the Pedersen commitment scheme to create $\text{Params} = \langle p, q, g, h \rangle$ such that $2^\ell < q/2$. CA also outputs $\mathcal{V} = [0..2^\ell - 1]$ and $\mathcal{P} = \{\text{GE}_{a_0} \mid a_0 \in \mathcal{V}\}$, where $\text{GE}_{a_0} : \mathcal{V} \rightarrow \{\text{true}, \text{false}\}$ is a predicate such that $\text{GE}_{a_0}(a)$ is true if $a \geq a_0$ and false otherwise.

CA-Commit R chooses an integer $a \in \mathcal{V}$ and sends to CA. CA picks $r \leftarrow \mathbb{Z}_q$ and computes the commitment $c = g^a h^r$. CA gives c and r to R , and c to S .

Initialization S chooses a message $M \in \{0, 1\}^*$. S and R agree on a predicate $\text{GE}_{a_0} \in \mathcal{P}$. Now S has GE_{a_0}, c , and M . R has GE_{a_0}, c, a , and r .

Interaction Let $d = ((a - a_0) \bmod q)$, $\text{GE}_{a_0}(a) = \text{true}$ if and only if $d \in [0..2^\ell - 1]$. Note that $cg^{-a_0} = g^d h^r$ is a commitment of d that R can open.

1. R picks $r_1, \dots, r_{\ell-1} \leftarrow \mathbb{Z}_q$ and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i \bmod q$. When $\text{GE}_{a_0}(a) = \text{true}$, let $d_{\ell-1} \dots d_1 d_0$ be the binary representation of d , i.e., $d = d_0 2^0 + d_1 2^1 + \dots + d_{\ell-1} 2^{\ell-1}$. When $\text{GE}_{a_0}(a) = \text{false}$, R randomly picks $d_1, d_2, \dots, d_{\ell-1} \leftarrow \{0, 1\}$, and sets $d_0 = d - \sum_{i=1}^{\ell-1} 2^i d_i \bmod q$. R computes, for $0 \leq i \leq \ell - 1$, the commitment $c_i = \text{commit}(d_i, r_i) = g^{d_i} h^{r_i}$. R sends $c_0, \dots, c_{\ell-1}$ to S .

2. S verifies that $cg^{-a_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$. S randomly chooses ℓ symmetric keys $k_0, \dots, k_{\ell-1} \in \{0, 1\}^t$ and sets $k = H'(k_0 || \dots || k_{\ell-1})$. S picks $y \leftarrow \mathbb{Z}_q^*$, computes $\eta = h^y$ and $C = \mathcal{E}_k[M]$. For each $0 \leq i \leq \ell - 1$, S computes $\sigma_i^0 = (c_i)^y$, $\sigma_i^1 = (c_i g^{-1})^y$, $C_i^0 = H(\sigma_i^0) \oplus k_i$, and $C_i^1 = H(\sigma_i^1) \oplus k_i$. S sends to R the tuple $\langle \eta, C_0^0, C_0^1, \dots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$.

Open R receives $\langle \eta, C_0^0, C_0^1, \dots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$ from the interaction phase. If $\text{GE}_{a_0}(a)$ is true, $d = \sum_{i=0}^{\ell-1} 2^i d_i$ where $d_i \in \{0, 1\}$. For each $0 \leq i \leq \ell - 1$, R computes $\sigma_i' = \eta^{r_i}$, and obtains $k_i' = H(\sigma_i') \oplus C_i^{d_i}$. R then computes $k' = H'(k_0' || \dots || k_{\ell-1}')$, and decrypts C using k' .

To see that the GE-OCBE protocol is sound, observe that when $\text{GE}_{a_0}(a)$ is true, $d_0, \dots, d_{\ell-1}$ are either 0 or 1. If the receiver follows the protocol, the sender will succeed in verifying $\prod_{i=0}^{\ell-1} (c_i)^{2^i} = \prod_{i=0}^{\ell-1} (g^{d_i} h^{r_i})^{2^i} = g^d h^r = cg^{-a_0}$. For each $0 \leq i \leq \ell - 1$, if $d_i = 0$, $\sigma_i^0 = (c_i)^y = (g^{d_i} h^{r_i})^y = (h^y)^{r_i} = \eta^{r_i} = \sigma_i'$, the receiver can compute $k_i = C_i^0 \oplus H(\sigma_i')$; if $d_i = 1$, $\sigma_i^1 = (c_i g^{-1})^y = (g^{d_i-1} h^{r_i})^y = (h^y)^{r_i} = \eta^{r_i} = \sigma_i'$, the receiver can compute $k_i = C_i^1 \oplus H(\sigma_i')$. As $k = H'(k_0 || \dots || k_{\ell-1})$, the receiver can successfully obtain k . Thus the sender and receiver share the same symmetric key k if $\text{GE}_{a_0}(a)$ is true.

The interaction phase of the GE-OCBE protocol is two rounds. The receiver does about 2ℓ exponentiations. The sender does about ℓ exponentiations (observe that σ_i^1 can be computed as $\sigma_i^0 g^{-y}$, where g^{-y} needs to be computed only once).

We briefly sketch the idea why the receiver cannot obtain M if $\text{GE}_{a_0}(a)$ is false. If the receiver follows the protocol, then $d_1, \dots, d_{\ell-1} \in \{0, 1\}$ and $d_0 \notin \{0, 1\}$. The receiver can successfully compute $k_1, \dots, k_{\ell-1}$, but fails to compute k_0 because he can compute neither $\sigma_0^0 = (c_0)^y = (g^{d_0} h^{r_0})^y$ nor $\sigma_0^1 = (c_0 g^{-1})^y = (g^{d_0-1} h^{r_0})^y$. Even if the receiver does not follow the protocol, it is impossible for him to find $d_0, \dots, d_{\ell-1} \in \{0, 1\}$ and $r_0, \dots, r_{\ell-1}$ such that $cg^{-a_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$ and $c_i = g^{d_i} h^{r_i}$. Suppose the receiver finds such $d_0, \dots, d_{\ell-1} \in \{0, 1\}$ and $r_0, \dots, r_{\ell-1}$; let $d' = \sum_{i=0}^{\ell-1} d_i 2^i \in [0..2^\ell - 1]$ and $r' = \sum_{i=0}^{\ell-1} r_i 2^i \pmod{q}$, then

$$\begin{aligned} g^{a-a_0} h^r &= cg^{-a_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i} = \prod_{i=0}^{\ell-1} (g^{d_i} h^{r_i})^{2^i} \\ &= g^{\sum_{i=0}^{\ell-1} d_i 2^i} h^{\sum_{i=0}^{\ell-1} r_i 2^i} = g^{d'} h^{r'}. \end{aligned}$$

As $a - a_0 \notin [0..2^\ell - 1]$ and $d' \in [0..2^\ell - 1]$, $d' \neq a - a_0$, the receiver is able to find $a - a_0, r, d'$, and r' such that $g^{a-a_0} h^r = g^{d'} h^{r'}$, which breaks the binding property of the Pedersen commitment scheme.

Theorem 3. *GE-OCBE is oblivious.*

Theorem 4. *Under the CDH assumption on G_q , the order- q subgroup of \mathbb{Z}_p^* , and when H and H' are modeled as random oracles, GE-OCBE is secure against the receiver.*

6.3 OCBE protocols for other predicates

In this section, we first present two logical combination OCBE protocols, one for \wedge (AND-OCBE), the other for \vee (OR-OCBE). Then we describe OCBE protocols for

comparison predicates: $>$ (GT-OCBE), \leq (LE-OCBE), $<$ (LT-OCBE), \neq (NE-OCBE). Finally, we present an OCBE protocol for range predicates (RANGE-OCBE). Due to space limitation, we only sketch the ideas. Note that similar techniques have been used before in [7, 33]. In what follows, we use $OCBE(\text{Pred}, a, M)$ to denote an OCBE protocol with predicate Pred and committed value a , the receiver outputs M if $\text{Pred}(a)$ is true.

1. **AND-OCBE:** Suppose there exists OCBE protocols for Pred_1 and Pred_2 , the goal is to build an OCBE protocol for the new predicate $\text{Pred} = \text{Pred}_1 \wedge \text{Pred}_2$. An $OCBE(\text{Pred}_1 \wedge \text{Pred}_2, a, M)$ can be constructed as follows: In the interaction phase, the sender picks two random keys k_1 and k_2 and sets $k = k_1 \oplus k_2$. The sender then runs the interaction phases of $OCBE(\text{Pred}_1, a, k_1)$ and $OCBE(\text{Pred}_2, a, k_2)$ with the receiver. Finally, the sender sends $\mathcal{E}_k[M]$ to the receiver. The receiver can recover M in the open phase only if both $\text{Pred}_1(a)$ and $\text{Pred}_2(a)$ are true.
2. **OR-OCBE:** An $OCBE(\text{Pred}_1 \vee \text{Pred}_2, M)$ can be constructed as follows: In the interaction phase, the sender picks a random key k . The sender then runs the interaction phases of $OCBE(\text{Pred}_1, a, k)$ and $OCBE(\text{Pred}_2, a, k)$ with the receiver. Finally, the sender sends $\mathcal{E}_k[M]$ to the receiver. The receiver can recover M in the open phase if either $\text{Pred}_1(a)$ or $\text{Pred}_2(a)$ is true.
3. **GT-OCBE:** For integer space, $a > a_0$ is equivalent to $a \geq a_0 + 1$. An $OCBE(>_{a_0}, a, M)$ protocol is equivalent to an $OCBE(\geq_{a_0+1}, a, M)$ protocol.
4. **LE-OCBE:** Observe that $a \leq a_0$ if and only if $d = ((a_0 - a) \bmod q) \in [0..2^\ell - 1]$. Let $c = g^a h^r$ be a commitment of a , then $g^{a_0} c^{-1} = g^{(a_0 - a) \bmod q} h^{-r \bmod q}$ is a commitment of d such that the receiver knows how to open. The LE-OCBE protocol uses the same method as in GE-OCBE.
5. **LT-OCBE:** For integer space, $a < a_0$ is equivalent to $a \leq a_0 - 1$. An $OCBE(<_{a_0}, a, M)$ protocol is equivalent to an $OCBE(\leq_{a_0-1}, a, M)$ protocol.
6. **NE-OCBE:** $a \neq a_0$ is equivalent to $(a > a_0) \vee (a < a_0)$. Therefore, an $OCBE(\neq_{a_0}, a, M)$ can be built as $OCBE(>_{a_0} \vee <_{a_0}, a, M)$.
7. **RANGE-OCBE:** $a_0 \leq a \leq a_1$ is equivalent to $(a \geq a_0) \wedge (a \leq a_1)$. Therefore, a RANGE-OCBE can be built as $OCBE(\geq_{a_0} \wedge \leq_{a_1}, a, M)$.

7 Implementation and Performance

We have implemented a toolkit that generates X.509 certificates [29] that are also OACerts using Java v1.4.2 SDK and JCSI PKI Server Library [30]. In our implementation, both the parameters of the Pedersen commitment scheme and commitments of certificate holder's attributes are encoded in the X.509v3 extension fields. We assign each attribute in the certificate a unique object identifier (OID). We convert each attribute value into an octet string and place it together its corresponding OID in the extension fields as a non-critical extension. The CA can publish a list of attribute names and their corresponding OID, so that the service providers know which commitment corresponds to which attribute. Our OACerts can be recognized by OpenSSL.

We also implemented the OCBE protocols and some zero-knowledge proof protocols [12, 16, 13, 37] using Java 2 Platform v1.4.2 SDK. We use the Pedersen commitment scheme with security parameters $p = 1024$ bits and $q = 160$ bits. The size of a

commitment is 128 bytes. In the implementation of the OCBE protocols, we use MD5 as the cryptographic hash function, AES as the symmetric key encryption scheme. In our setting, M is typically a 16-byte symmetric key.

We ran our implementation on a 2.53GMz Intel Pentium 4 machine with 384MB RAM running RedHat Linux 9.0. We simulate the certificate holder and the service provider on the same machine. The performance of two zero-knowledge proof protocols and two OCBE protocols is summarized in Table 1.

	execution time	communication size
Zero-knowledge proof that $a = a_0$	28 ms	168 bytes
Zero-knowledge proof that $a \geq a_0$	2.2 s	15 KB
EQ-OCBE	75 ms	144 bytes
GE-OCBE	0.9 s	5.1 KB

Table 1. Running time and size of communication on a 2.53GMz Intel Pentium 4 running RedHat Linux. Security parameters are $\ell = 32$, $p = 1024$ bits, and $q = 160$ bits.

8 Conclusion

In this paper, we proposed OACerts, an attribute certificate scheme that enables oblivious access control. We introduced the notion of OCBE, and developed provably secure and efficient OCBE protocols for the Pedersen commitment scheme and predicates such as $=, \geq, \leq, >, <, \neq$ as well as logical combinations of them. Future work includes developing efficient OCBE protocols for predicates other than comparison predicates.

Acknowledgement

This work is supported by NSF ITR grant CCR-0325951 and by sponsors of CERIAS. We would like to thank Dan Boneh for helpful discussions. We thank the anonymous reviewers for their helpful comments. We thank also Ziad Bizri, Ji-Won Byun, Klorida Miraj, and Mahesh V. Tipunitara for reading drafts of the paper and making suggestions that have improved the paper’s presentation.

References

1. Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *Proceedings of the IEEE Symposium and Security and Privacy*, pages 180–196, May 2003.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.
3. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.
4. Sharon Boeyen, Tim Howes, and Patrick Richard. Internet X.509 Public Key Infrastructure LDAPc2 Schema. IETF RFC 2587, June 1999.
5. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology: EUROCRYPT ’00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer, May 2000.

6. Robert Bradshaw, Jason Holt, and Kent Seamons. Concealing complex policies with hidden credentials. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, October 2004.
7. Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, August 2000.
8. Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS '02*, pages 21–30. ACM, nov 2002.
9. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology: EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.
10. David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.
11. Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.
12. Ronald Cramer and Ivan Damgård. Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? In *Advances in Cryptology: CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441. Springer, 1998.
13. Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology: EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1996.
14. Claude Crépeau. Verifiable disclosure of secrets and applications (abstract). In *Advances in Cryptology: EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 150–154. Springer, 1990.
15. Giovanni Di Crescenzo, Rafail Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 74–89, March 1999.
16. Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In *Advances in Cryptology: ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142. Springer, December 2002.
17. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.
18. Glenn Durfee and Matt Franklin. Distribution chain security. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 63–70. ACM Press, 2000.
19. Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory. IETF RFC 2693, September 1999.
20. Stephen Farrell and Russell Housley. An internet attribute certificate profile for authorization. IETF RFC 3281, April 2002.
21. Marc Fischlin. A cost-effective pay-per-multiplication comparison method for millionaires. In *CT-RSA 2001: Proceedings of the 2001 Conference on Topics in Cryptology*, volume 2020 of *Lecture Notes in Computer Science*, pages 457–472. Springer, 2001.
22. Keith B. Frikken, Mikhail J. Atallah, and Jiangtao Li. Hidden access control policies with hidden credentials. In *Proceedings of the 3rd ACM Workshop on Privacy in the Electronic Society*, October 2004.
23. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology: CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1997.
24. Juan Garay, Philip MacKenzie, and Ke Yang. Efficient and universally composable committed oblivious transfer and applications. In *Theory of Cryptography, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 297–316. Springer, 2004.

25. Oded Goldreich. Secure multi-party computation, October 2002.
26. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, May 1987.
27. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18:186–208, feb 1989.
28. Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and Hilarie Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society*, October 2003.
29. Russell Housley, Warwick Ford, Tim Polk, and David Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF RFC 2459, January 1999.
30. JCSI. Java cryptographic secure implementation. Wedgetail Communications, 2004.
31. Jiangtao Li and Ninghui Li. OACerts: Oblivious attribute certificates. Technical Report CERIAS-TR-2005-26, Center for Education and Research in Information Assurance and Security, Purdue University, April 2005.
32. Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*. ACM Press, July 2003.
33. Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. *Distributed Computing*, 2005. to appear.
34. Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, May 2002.
35. Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.
36. Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography, 6th Annual International Workshop, SAC '99*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1999.
37. Wenbo Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Public Key Cryptography: PKC'98*, volume 1431 of *Lecture Notes in Computer Science*, pages 60–71. Springer, February 1998.
38. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of SODA 2001 (SIAM Symposium on Discrete Algorithms)*, pages 448–457, January 2001.
39. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.
40. Eric Rescorla. *SSL, TLS: Designing, and Building Secure Systems*. Addison-Wesley, 2001.
41. Wen-Guey Tzeng. Efficient 1-out-n oblivious transfer schemes. In *PKC '02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, number 2274 in *Lecture Notes in Computer Science*, pages 159–171. Springer, 2002.
42. William H. Winsborough and Ninghui Li. Safety in automated trust negotiation. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 147–160, May 2004.
43. William H. Winsborough, Kent E. Seamons, and Vicki E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pages 88–102. IEEE Press, January 2000.
44. Andrew C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, 1986.
45. Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, February 2003.