Revisiting Oblivious Signature-Based Envelopes

Samad Nasserian and Gene Tsudik
Computer Science Department
University of California, Irvine
samad.nasserian@rwth-aachen.de, gts@ics.uci.edu

Abstract

Secure, anonymous and unobservable communication is becoming increasingly important due to the gradual erosion of privacy in many aspects of everyday life. This prompts the need for various anonymity- and privacy-enhancing techniques, e.g., group signatures, anonymous e-cash and secret handshakes.

In this paper, we investigate an interesting and practical cryptographic construct – Oblivious Signature-Based Envelopes (OSBEs) recently introduced in [15]. OSBEs are very useful in anonymous communication since they allow a sender to communicate information to a receiver such that the receiver's rights (or roles) are unknown to the sender. At the same time, a receiver can obtain the information only if it is authorized to access it. This makes OSBEs a natural fit for anonymity-oriented and privacy-preserving applications, such as Automated Trust Negotiation and Oblivious Subscriptions.

Previous results yielded three OSBE constructs: one based on RSA and two based on Identity-Based Encryption (IBE). Our work focuses on the ElGamal signature family: we succeed in constructing practical and secure OSBE schemes for several well-known signature schemes, including: Schnorr, Nyberg-Rueppel, ElGamal and DSA. As experiments with the prototype implementation illustrate, our schemes are more efficient than previous techniques. Furthermore, we show that some OSBE schemes, despite offering affiliation privacy for the receiver, introduce no additional cost over schemes that do not offer this feature.

NOTE*: An earlier incarnation of this paper was previously submitted to another security conference. The present version is a result of a major revision. In particular, it includes a thorough proof of security for the Schnorr-OSBE scheme as well as an in-depth discussion of implementation and efficiency/experiments.*

1 Introduction

The surge in popularity of electronic communication and electronic transaction prompts many natural concerns about anonymity and, more generally, privacy of communicating entities. In the last decade, there has been a lot interest in privacy-enhancing tools and techniques. Prominent topics include advanced cryptographic constructs, such as: blind signatures [10], group signatures (e.g., [4]), identity escrow (e.g., [14]), secret handshakes [2] and privacy-preserving trust negotiation [7].

In a recent paper [15], Li, Du and Boneh introduced a simple and interesting cryptographic concept termed OSBE: Oblivious Signature-Based Envelopes. One motivating scenario for OSBE is as follows: suppose that Bob is an agent of the British Secret Service and has a digitally signed certificate asserting his membership. The rules of the trade stipulate that a secret agent must only reveal his certificate to another party if that party is also a secret agent. Thus, if Bob and Alice (who is also a secret agent) want to communicate securely, they are seemingly at an impasse since someone must reveal their certificate first. A simpler, and perhaps more appealing, scenario occurs if Alice is a regular entity without any specific affiliation. However, she has some information that she is only willing to reveal to another party (who claims to be named Bob) if that party has certain credentials, for example, Alice might be a potential informant and Bob might be an FBI or DEA agent. At the same time, Bob is unwilling – or not allowed – to reveal his credentials. In this case, Alice and Bob are also stuck since neither wants to be the first to reveal information. Note that, in both examples, Bob has a signed credential which he cannot reveal; specifically, Bob needs to keep the signature secret, whereas, the message covered by the signature is not secret at all.

An OSBE scheme can help in the aforementioned situations since it allows Alice to communicate information to Bob in such a way that: (1) Bob only obtains the information if he possesses appropriate credentials, (2) Alice does not determine whether Bob possesses such credentials, and (3) no other party learns anything about Alice's information and Bob's possession, or lack of, the credentials. A more detailed discussion about application of OSBEs can be found in Appendix C.

Besides introducing the OSBE concept, [15] presented three concrete OSBE schemes: RSA-OSBE, BLS-OSBE and Rabin-OSBE. The last two use Identity-Based Encryption (Boneh-Franklin [5] and Cocks [11] schemes, respectively) and don't require interaction, while RSA-OSBE is essentially a 2-round protocol[1] with some very interesting properties (a discussion of these properties can be found in Section 9). Notably, no OSBE schemes for any signature schemes derived from ElGamal [13] have been developed. (In fact, OSBE for DSA is explicitly mentioned as an open problem.) In this paper, we begin where the work of [15] left off.

Contributions: Our main result is the development of a series of OSBE schemes for the ElGamal family of signature schemes, including Schnorr, Nyberg-Rueppel and DSA. We rigorously prove the security of Schnorr-OSBE and discuss the security of the other schemes. We analyze and compare their respective costs (as well as that of RSA-OSBE) and present the results of our implementation of the OSBE schemes. Our proposed schemes are very efficient and, in fact, demonstrate that, in some cases, added privacy (provided by OSBE schemes) introduces **no additional costs**. We also consider some natural extensions of OSBEs to constructing provably secure and practical secret handshake protocols.

Organization: This paper is organized as follows: the next section contains the necessary OSBE definitions. Section 3 presents the roadmap and a summary of our contributions. Then, Section 4 shows the construction of OSBE for Schnorr signatures, followed by Section 5 which does the same for Nyberg/Rueppel signatures. Section 6 presents OSBE schemes for other ElGamal family signatures, including one for the Digital Signature Standard (DSA). The costs of all OSBE schemes and the results of the implementation of the OSBE schemes are analyzed in Section 8. We discuss certain other security features apart from the security features required for OSBEs in Section 9. We then sketch some new approaches to building secret handshake protocols in Section 10 and conclude in Sections 11 and 12 with an overview of some related work and future research directions.

2 Preliminaries

This section presents some background material, including definitions of OSBE components and OSBE security properties.[2]

An OSBE scheme enables a sender to encrypt a message and send it to a receiver who can decrypt the message **if and only if** the receiver has the right third party's signature (e.g., a signature from a certification authority) on a previously agreed upon message. However, the sender is not allowed to know – not even at the end – if the receiver has the right third party's signature. The sender is assured only that the encrypted message will only be decrypted if the receiver has the right signature. The components of an OSBE scheme are (1) a setup algorithm and (2) three parties S, R_1 and R_2 or sender, authorized receiver and unauthorized receiver (adversary), respectively. In other words, S is the party who wants to send message P to the authorized receiver who has the right signature on some authorization string M, e.g., M can be thought of as a certificate. R_1 is the receiver who has the right signature σ on message M and R_2 is the receiver who does not have σ and who might try to impersonate R_1.

An OSBE scheme consists of two phases: *Setup* and *Interaction*.[3]

Setup: This algorithm generates two messages M and P and a public/private key-pair for a given signature scheme. It uses the secret key to generate a signature σ on an input message M. The values M and the public parameters/keys for the signature scheme are known to all parties. Whereas, P is known only to S and σ is known only to R_1. Since the

[1] See Appendix B for a brief re-cap of RSA-OSBE.

[2] Since much of the material in this section is adapted from [15], those familiar with [15] may wish to skip this section with no lack of continuity.

[3] Note that [15] defines an additional *Open* phase. We merged this phase with *Interaction*.

setup algorithm generates the signature, we assume that it also takes the role of a certification authority (CA). (However, this is not a requirement.)

Interaction: In this phase, S communicates with the receiver R, which is either R_1 or R_2. However, in the process, S cannot distinguish between R_1 or R_2. At the end of this phase, if $R = R_1$, R_1 outputs message P; otherwise, R_2 cannot output P.

An OSBE scheme must satisfy three properties: *soundness*, *obliviousness* and *semantic security against the receiver*, which are informally defined as (formal treatment of these properties can be found in [15]):

Soundness: An OSBE scheme is sound if the authorized receiver R_1 (who has the signature σ on message M) can output P with non-negligible probability at the end of the *Interaction* phase.

Obliviousness: An OSBE scheme is oblivious if, at the end of the interaction phase, S does not know whether it is communicating with R_1 or R_2. In other words, if one of: R_1 or R_2 is randomly picked to take part in the Interaction with S, the probability of S correctly guessing the other party is, at best, negligibly over $1/2$.

Semantic Security Against the Receiver: An OSBE scheme is semantically secure against the receiver if R_2 learns nothing about P. Even if P can be only one of two possible messages (selected by R_2), at the end of the Interaction phase, R_2 cannot determine – with probability non-negligibly greater than $1/2$ – which message was actually sent.

In the remainder of this paper we use the term *negligible* to refer to functions with a certain property: a function $f(x)$ is said to be *negligible* if, for each polynomial $p(k)$, there exists a k_0 with $f(x) \leq \frac{1}{|p(k)|}$ for all $x \geq k_0$.

As discussed in Section 9 below, there are other features that could be desired from OSBE schemes.

3 Roadmap

As mentioned earlier, the only prior result in OSBEs is [15]. We consider the main contributions of [15] to be two-fold: (1) the introduction and formalization of the novel OSBE concept, and (2) the construction of, and proof of security for, a practical RSA-OSBE scheme. The results of [15] also include two non-interactive (or one-round) OSBE schemes for BLS and Rabin signatures using Boneh-Franklin and Cocks IBE schemes, respectively. However, their constructions are quite intuitive since, as the authors of [15] show, an OSBE scheme can be built from **any** identity-based encryption scheme.

Our work has roughly the same goals as [15]. Specifically, we are interested in coming up with more practical and secure OSBE schemes that are: (1) based on standard cryptographic assumptions, and (2) utilize popular (or at least well-known) signature schemes. Since RSA-OSBE and Rabin-OSBE schemes have been already demonstrated, we turn to the natural alternative: the family of signature schemes originated by the ElGamal signature scheme [13]. This family includes such notable signature schemes as DSA, Schnorr and Nyberg-Rueppel.

Besides being grounded in different number-theoretic settings, the biggest difference between ElGamal-like and RSA signatures is their physical representation: an ElGamal-like signature is a pair, whereas, an RSA signature is a single value. When building an OSBE scheme for RSA signatures, intuitively, one has no choice but to somehow encrypt (or otherwise hide) the receiver's RSA signature. With ElGamal-like signatures, there is a choice: we can try to encrypt the entire signature or reveal only part of it. In this paper, we elect to do the latter, partly because we observe that half of an ElGamal signature is essentially random and yields no information about the other half.

The above observation leads us to design relatively simple, yet secure and efficient, OSBE schemes for Schnorr, Nyberg/Rueppel, ElGamal and DSA signatures. A valuable, but not entirely unexpected, side-effect is that each proposed OSBE scheme can be used as a natural building block for secret handshake protocols [2].

We begin, in the next section, with the Schnorr signature scheme. The Nyberg/Rueppel signature scheme is considered next, followed by other ElGamal schemes, including DSA.

4 OSBE with Schnorr Signatures

Recall that Schnorr's signature scheme works as follows [21]:

> Let p be a large prime and q be a large prime factor of $p-1$. Let g be an element of order q in $\mathbb{Z}_p^*$, $\mathcal{M}$ be the message space and $H: \mathcal{M} \rightarrow \mathbb{Z}_q^*$ be a suitable cryptographic hash function. The signer's secret key is:

$a \in_R\ Z_q^*$ and the corresponding public key is: $y = g^a \bmod p$. The values: p, q and y are public, while a is only known to the signer. A signature $\sigma = (e, s)$ on input message M is computed as follows:

1. select a random value $k \in_R \mathbb{Z}_q^*$
2. compute $e = H(M, g^k \bmod p)$
3. compute $s = ae + k \mod q$

A Schnorr signature is verified by checking that $H(M, g^s y^{-e} \bmod p)$ matches e.

Similar to RSA-OSBE [15] and all other non-IBE-based OSBE schemes, the interaction in Schnorr-OSBE is essentially a Diffie-Hellman style key agreement protocol. It is run between S and either R_1 or R_2 where the former is a legitimate certificate holder and the latter is an adversarial party. If S and R_1 take part in the protocol, then – at the end – both parties agree on the same shared secret key. Whereas, if S and R_2 run the protocol, then they compute distinct values and R_2 is unable to derive the key computed by S. Since the very nature of OSBE prohibits R_1 (or R_2) from authenticating to S, no key confirmation flows in either direction.

Once S computes the Diffie-Hellman secret K_S, it sends its message (P) to the other party (either R_1 or R_2) encrypted under a key derived from K_S.

$R_{1/2}$ starts the protocol by sending to S one part of its signature: $X = g^s y^{-e} \bmod p = g^k \bmod p$. S then generates a random $z \in_R Z_q^*$ and computes its version of the secret as:

$$K_s \ = \ [y^{H(M,X)} X]^z = g^{(ae+k)z}$$

and sends $Z = g^z \bmod p$ to $R_{1/2}$. If $R_{1/2}$ is indeed R_1, it knows the other half of the signature: $s = ae+k \bmod q$. It can thus easily compute $K_r = Z^s = g^{z(ae+k)}$. Both S and R_1 employ a function $H'()$ for deriving from the Diffie-Hellman secret, the actual key to be used for the symmetric encryption of message P. In more detail, Schnorr-OSBE is as follows:

Setup: On input of a security parameter t, this algorithm creates a Schnorr key: (p, q, g, a, y), selects a suitable cryptographic hash function H, a function H' for key derivation and two security parameters t_1 and t_2, which are linear in t. It also chooses a semantically secure symmetric encryption scheme $\mathcal{E}$, two messages M and P. It computes a Schnorr signature $\sigma = (e, s)$ on message M. Finally, it gives $M,\ \sigma$ and (p, q, g, y) to R_1, M and (p, q, g, y) to R_2 and M, P and (p, q, g, y) to S.

Interaction:

Step 1a: $R_1 \ \longrightarrow \ S: \quad X = g^s \cdot y^{-e} \bmod p = g^k \bmod p$

OR

Step 1b: $R_2 \ \longrightarrow \ S: \quad X = g^{k'} \bmod p$ for some $k' \in \mathbb{Z}_q^*$

Step 2: S receives X, checks that: $(X)^{(p-1)/q} \bmod p \notin \{0, 1\}$, picks a random $z \in \{1..2^{t_1} q\}$ with $z \bmod q \neq 0$, computes $K_s = [y^{H(M,X)} X]^z$, $k_s = H'(K_s)$ and:

Step 3: $S \ \longrightarrow \ R_1$ or $R_2: \quad Z = g^z \bmod p,\ C = \mathcal{E}_{k_s}[P]$

Step 4: R_1 receives (Z, C), computes $K_r = Z^s \bmod p$, derives $k_r = H'(K_r)$ and finally decrypts C with k_r.

We now prove that Schnorr-OSBE is sound, oblivious and semantically secure against the receiver.

Soundness: To see that Schnorr-OSBE is sound, at the end of *Interaction*, $K_r = K_s$ has to hold. This is easily established, since:

$$K_s \ = \ [y^{H(M,X)} X]^z = [g^{ae} g^s y^{-e}]^z = [g^{ae+ae+k} g^{-ae}]^z = g^{sz} = g^{zs} = Z^s = K_r$$

Showing that Schnorr-OSBE is oblivious is similar to the proof of obliviousness for RSA-OSBE in [15]. We first re-state the notion of statistical indistinguishability. Two distribution families $D^1(t_1)$ and $D^2(t_1)$ are said to be **statistically indistinguishable** if:

$$\Sigma_y |Pr_{x \in D^1(t_1)}[x = y] - Pr_{x \in D^2(t_1)}[x = y]|$$

is negligible in t.

If two distribution families are statistically indistinguishable, then there exists no algorithm that can distinguish between the two with non-negligible advantage by sampling from them. We use this to prove the following theorem.

Theorem 1. Schnorr-OSBE is oblivious.

Proof (sketch): Two distribution families:

$$\begin{aligned} D^1(t_1) &= \{g^s y^{-e} \bmod p = g^k \bmod p \mid k \in \{1..2^{t_1}q\}\} \\ &\quad \text{and} \\ D^2(t_1) &= \{g^{k'} \bmod p \mid k' \in \{1..2^{t_1}q\}\} \end{aligned}$$

range over the same values and, during each run of *Interaction*, a random value from one of these distribution families is sent by a communicating partner (either R_1 or R_2). Since these values are chosen at random and the two distribution families range over the same values, S cannot decide whether the other party is R_1 or R_2. Consequently, Schnorr-OSBE is oblivious.

In more detail, since q is the order of g, $D^1(t_1)$ and $D^2(t_1)$ (for a fixed t_1) each have q points. The probability difference on any point is at most $\frac{1}{2^{t_1}q}$, therefore, the total difference is at most $\frac{q}{2^{t_1}q} = \frac{1}{2^{t_1}}$. Since this quantity is negligible in t_1 and t_1 is linear in t, the total difference is also negligible in t. Thus, the two distribution sets are statistically indistinguishable.

Theorem 2. Assuming the non-existence of a polynomial time algorithm for solving the CDH Problem and that H and H' are modelled as random oracles, Schnorr-OSBE is secure against the receiver.

Proof. Schnorr-OSBE uses a semantically secure symmetric encryption algorithm and H' is modelled as a random oracle. Therefore, Schnorr-OSBE is semantically secure against the receiver if no polynomially bounded adversary, who does not possess the signature $\sigma = (e, s)$ on M, can compute with non-negligible probability the OSBE key $K_s = g^{z(ae+k)} \bmod p$. More precisely, Schnorr-OSBE is semantically secure against the receiver if there is no polynomially bounded adversary who can win with non-negligible probability the following game:

1. $\mathcal{A}$ is given a message M and the public key (p, q, g, y) with $y = g^a \bmod p$.
2. $\mathcal{A}$ chooses a value $X = g^k \bmod p$.[4]
3. $\mathcal{A}$ is given the value $Z = g^z \bmod p$.
4. $\mathcal{A}$ outputs a value K.

A wins the game if and only if $K = g^{z(ea+k)} \bmod p$ with $e = H(M, g^k)$. We prove our claim by contradiction. We show that if there is a polynomial adversary who wins the above game with non-negligible probability, then such an adversary can also solve every instance (g^a, g^z) of the CDH Problem in polynomial time. Assume there is an adversary $\mathcal{A}$ who does not have a signature $\sigma = (e, s)$ but who nevertheless wins the above game (i.e. computes the value $g^{z(ae+k)} \bmod p$) with non-negligible probability ϵ. Using the forking lemma [20], we know that then, $\mathcal{A}$ can be executed twice in a row with the same value $X = g^k \bmod p$ and different random oracles H and H' (such that $e = H(M, g^k) \neq H'(M, g^k) = e'$) and $\mathcal{A}$ wins both games with non-negligible probability of at least $\frac{\epsilon^2}{q_H}$, where q_H is the number of queries $\mathcal{A}$ makes to

[4] Note that we do not make any assumptions about $\mathcal{A}$'s knowledge about k. $\mathcal{A}$ might know the value of k, it might have partial knowledge of k or k might even be completely unknown to $\mathcal{A}$.

the hash function. This means, $\mathcal{A}$ can compute with non-negligible probability the values $K = g^{z(ea+k)} \bmod p$ and $K' = g^{z(e'a+k)} \bmod p$ with $e \neq e'$. Consequently, $\mathcal{A}$ can also efficiently compute g^{az}:

$\left(\frac{K}{K'}\right)^{(e-e')^{-1}} = \left(g^{zea-ze'a}\right)^{(e-e')^{-1}} = \left(g^{za(e-e')}\right)^{(e-e')^{-1}} = g^{az} \bmod p.$

However, this means that $\mathcal{A}$ can solve the CDH Problem in polynomial time, which is a contradiction to our assumption.

Note that the Schnorr signature scheme is existentially unforgeable assuming there is no polynomial time algorithm which can solve the Discrete Logarithm (DL) Problem [20]. Since we assume that there is no polynomial time algorithm for solving the CDH Problem, this also implies that there is no polynomial time algorithm which can solve the DL Problem. Therefore, assuming there is no polynomial time algorithm which can solve the DL Problem, a polynomial time adversary $\mathcal{A}$ cannot forge a signature on M in order to be able to compute the OSBE key.

5 Nyberg/Rueppel OSBE

We now turn to the Nyberg/Rueppel signature scheme. Recall that the Nyberg/Rueppel signature scheme [18] is as follows:

> Let p be a large prime and q be a large prime factor of $p-1$. Let g be an element of order q in $\mathbb{Z}_p^*$, $\mathcal{M}$ be the message space and $H : \mathcal{M} \rightarrow \mathbb{Z}_p$ be a suitable cryptographic hash function. (Note that the *textbook* description of Nyberg-Rueppel scheme does not require a hash function, since the scheme provides the *message recovery* feature.) The signer's secret key is: $a \in_R \mathbb{Z}_q^*$ and the corresponding public key is: $y = g^a \bmod p$. The values: p, q and y are public, while a is only known to the signer. A signature $\sigma = (e, s)$ on input message M is computed as follows:
>
> 1. select a random value $k \in_R \mathbb{Z}_q^*$ and set $h = H(M)$
> 2. compute $e = hg^{-k} \mod p$
> 3. compute $s = ae + k \mod q$
>
> A Nyberg-Rueppel signature is verified by checking that: (1) $0 < e < p$, $0 < s < q$, and (2) $h' = g^s y^{-e} e \bmod p$ matches $h' = H(M)$.

NR-OSBE is very similar to Schnorr-OSBE presented above:

Setup: This algorithm takes as input a security parameter t and creates a Nyberg-Rueppel key: (p, q, g, a, y), selects a suitable cryptographic hash function H, a function H' for key derivation and two security parameters t_1 and t_2, which are linear in t. It also chooses a semantically secure symmetric encryption scheme $\mathcal{E}$, two messages M and P and computes $\sigma = (e, s)$ where:[5]

$$e = hg^{-k} \mod p \text{ where } k \in_R \mathbb{Z}_q^*,$$

$$e \mod q \neq 0 \text{ and } s = ae + k \mod q$$

It then gives M, σ and (p, q, g, y) to R_1, M and (p, q, g, y) to R_2 and M, P and (p, q, g, y) to S.

Interaction: Although in the following, we describe actions for S, R_1 and R_2, it is understood that only one of $[R_1, R_2]$ actually participates in the protocol. (The term *participates* means: **sends a message in Step 1a/b below**.)

Step 1a: $R_1 \longrightarrow S : \quad e = hg^{-k} \bmod p$

OR

Step 1b: $R_2 \longrightarrow S : \quad e = hg^{-k'} \bmod p$ for some $k' \in \mathbb{Z}_q^*$

[5] We need the property $e \mod q \neq 0$ for the proof of security of our scheme. Note that this is only a restriction on the signer and in particular no restriction on S or $R_{1/2}$.

Step 2: S receives e, checks that: $(e/h)^{(p-1)/q} \bmod q \notin \{0,1\}$, picks a random $z \in \{1..2^{t_1}q\}$, computes $K_s = [y^e(e/h)^{-1}]^z$ derives $k_s = H'(K_s)$ and:

Step 3: $S \longrightarrow R_1$ or R_2 : $Z = g^z \bmod p,\ C = \mathcal{E}_{k_s}[P]$

Step 4: R_1 receives (Z, C), computes $K_r = Z^s = g^{z(ae+k)}$, derives $k_r = H'(K_r)$ and finally decrypts C with k_r.

Note that, in Step 1b, the value e sent by R_2 must be such that: $(e/h)^{(p-1)/q} \bmod p \notin \{0,1\}$. In other words, e/h has to be in the unique group of order q, which is generated by g. If this is not the case, e is immediately rejected by S. Therefore, there must be $k' \in \{1..2^{t_1}q\}$ such that: $e = hg^{-k'} \bmod p$.

Soundness: To show that NR-OSBE is sound, S and R_1 must share the same symmetric key when the protocol completes successfully, i.e., $K_r = K_s$. It is easy to see that:

$$K_s \quad = \quad [y^e(e/h)^{-1}]^z = g^{(ae+k)z} = g^{zs} = Z^s = K_r$$

Theorem 3. NR-OSBE is oblivious.

Proof (sketch): Two distribution families:

$$\begin{aligned} D^1(t_1) \quad &= \quad \{e = hg^{-k} \bmod p | k \in \{1..2^{t_1}q\}\} \\ &\quad\ \text{and} \\ D^2(t_1) \quad &= \quad \{e = h \cdot g^{-k'} \bmod p | k \in \{1..2^{t_1}q\}\} \end{aligned}$$

range over the same values and, during each execution of the Diffie-Hellman key exchange protocol, a random value from one of the two families is sent by each party. Since these values are chosen at random and, since the two distribution families have the same values, S does not know if the other party is R_1 or R_2. Consequently NR-OSBE is oblivious. More concretely, since q is the order of g, both $D^1(t_1)$ and $D^2(t_1)$ (for a fixed t_1) have q points. The probability difference on any point is at most $\frac{1}{2^{t_1}q}$ and, therefore, the total difference is at most $\frac{q}{2^{t_1}q} = \frac{1}{2^{t_1}}$. Since the total difference is negligible in t_1 and t_1 is linear in t, the total difference is also negligible in t. Thus, two distribution sets are statistically indistinguishable.

Semantic security against the receiver: While proving the semantic security of Schnorr-OSBE is straightforward, the proof of the semantic security of Nyberg/Rueppel-OSBE and also that of ElGamal-OSBE and DSA-OSBE is far from easy, if standard cryptographic assumptions are to be made. Further discussion of semantic security for Nyberg/Rueppel-OSBE, ElGamal-OSBE and DSA-OSBE can be found in the appendix.

6 ElGamal and DSA OSBE

A number of ElGamal variants are known in the literature. The following 6 are taken from [16] (note that none of them corresponds to either Schnorr or Nyberg-Rueppel schemes):

1. $s = (h - ag^k)k^{-1}$
2. $s = (h - kg^k)a^{-1}$
3. $s = ag^k + kh$
4. $s = ah + kg^k$
5. $s = (g^k - kh)a^{-1}$
6. $s = (g^k - ah)k^{-1}$

All computation is done modulo $(p-1)$ and existence of q is not required, although $(p-1)$ cannot be a product of small factors (to prevent the so-called Pohlig-Hellman attack [19]). In each case, the other part of the signature is $e = g^k \bmod p$.

It is easy to construct OSBE schemes for variants (3) and (4) above. In each case, the interaction component is as follows (the Setup component is trivial):

Step 1: $R_1 \longrightarrow S: \ \ e = g^k \bmod p$

Step 2: S receives e, generates $z \in_R \{1..2^{t_1}q\}$, computes:

$$\begin{aligned} K_s &= [y^e \cdot e^h]^z = g^{z(ag^k+kh)} \text{ for variant (3)} \\ K_s &= [y^h \cdot e^e]^z = g^{z(ah+kg^k)} \text{ for variant (4)} \end{aligned}$$

and derives $k_s = H'(K_s)$

Step 3: $S \longrightarrow R_1;: \ Z = g^z \bmod p, \ C = \mathcal{E}_{k_s}[P]$

Step 4: R_1 receives (Z, C), computes $K_r = Z^s$, derives $k_r = H'(K_r)$ and decrypts C with k_r.

To avoid repetition, we omit proofs of obliviousness and semantic security against the receiver for the above OSBE variants. Suffice it to say that the proofs are almost identical to those for Schnorr-OSBE and NR-OSBE.

OSBE constructs for variants (1), (2), (5) and (6) are less trivial since the signing equation (computation of s) involves either a^{-1} or k^{-1}. We now focus on variant (1) since it represents the original ElGamal signature scheme [13] and also naturally leads to an OSBE scheme for DSA.

The Interaction Component in EG-OSBE is as follows:

Interaction:

Step 1: $R_1 \longrightarrow S: \ \ e = g^k \bmod p$

Step 2: S receives e, generates $z \in_R \{1..2^{t_1}p\}$ with $e \bmod (p-1) \neq 0$, computes $K_s = y^{ez} \cdot g^{-hz} \bmod p$ and derives $k_s = H'(K_s)$

Step 3: $S \longrightarrow R_1;: \ Z = e^z \bmod p, \ C = \mathcal{E}_{k_s}[P]$

Step 4: R_1 receives (Z, C), computes $K_r = Z^{-s}$, derives $k_r = H'(K_r)$ and decrypts C with k_r.

Soundness. It is easy to see that:

$$K_s = y^{ez}g^{-hz} = g^{(ae-h)z} = g^{k(ae-h)k^{-1}z} = e^{-sz} = e^{-zs} = Z^{-s} = K_r$$

Theorem 4. EG-OSBE is oblivious.

Proof. Almost identical to that for Schnorr-OSBE.

The Digital Signature Algorithm (DSA) [17] was developed by NIST as a more efficient alternative to ElGamal. The DSA signature scheme works as follows [17]:

> Let p be a prime such that $p-1$ has a large prime divisor q, let g be an element of order q in $\mathbb{Z}_p^*$, $\mathcal{M}$ be the message space and $H : \mathcal{M} \rightarrow \mathbb{Z}_q^*$ be a cryptographic hash function. Furthermore, let $a \in_R \mathbb{Z}_q^*$ and $y = g^a \bmod p$ be signer secret and public keys, respectively. A DSA signature $\sigma = (e, s)$ on input message M is computed as follows:
>
> 1. generate $k \in_R \mathbb{Z}_q^*$ and set $h = H(M)$.
> 2. compute $e = (g^k \bmod p) \bmod q$ and $s = k^{-1}(h + ea) \bmod q$.

A DSA signature is verified by checking that: $(g^{s^{-1}h}y^{es^{-1}} \bmod p) \bmod q$ matches e.

To avoid unnecessary repetition, due to similarities between ElGamal and DSA, we omit the full description of DSA-OSBE. The only details worth mentioning involve the arithmetic of computing the secret:

1. $K_s = (y^e g^h)^z = g^{(ae+h)z}$
2. $K_r = Z^s = e^{zs} = g^{k(ae+h)k^{-1}z} = g^{(ae+h)z}$

Semantic security against the receiver: as mentioned in Section 5, proving the semantic security of Nyberg/Rueppel-OSBE, ElGamal-OSBE and DSA-OSBE is far from straightforward, if standard cryptographic assumptions are to be used. A discussion of the semantic security of these schemes can be found in the appendix.

7 Cost Analysis and Comparison

We now consider the communication and computation costs of the five schemes discussed in this paper, including RSA-OSBE (which is presented in Appendix B). Table 1 below summarizes the results.[6] We collapse EG-OSBE and DSA-OSBE since they are substantially similar. (However, we keep in mind that exponentiations and other modular arithmetic in DSA are appreciably cheaper than in ElGamal).

The number of rounds and the number of messages exchanged between the parties are the same (two rounds and two messages of constant length) for all OSBE schemes.

Schnorr-OSBE involves the most total exponentiations, while NR and EG/DSA have the fewest. Interestingly, all schemes require S to perform 3 exponentiations, whereas, R_1 performs between 1 and 3 exponentiations. Although we show the number of exponentiations for R_1 in RSA-OSBE as 2, this can be reduced to 1 by observing that R_1 does not need to generate a new blinding factor for each OSBE run.[7] Other cost factors (inverses and multiplications) are relatively minor and we do not elaborate on them further.

	OSBE Schemes:			
Costs:	NR	Schnorr	EG/DSA	RSA
protocol rounds	2	2	2	2
protocol messages	2	2	2	2
mod exps. for S	3	3	3	3
mod exps. for R_1	1	3	1	2
inverses for S	2	0	0	1
inverses for R_1	0	0	0	0
mod mults. for S	2	1	1	1
mod mults. for R_1	0	1	0	2

Table 1. Cost Factors for Various OSBE Schemes

To put the costs of OSBE schemes into perspective, we consider the hypothetical scenario whereby S and R_1 communicate securely without OSBEs (i.e., without the obliviousness factor). If R_1's affiliation privacy were not an issue, S would expect R_1 to first supply a valid signed certificate. Verifying a certificate would involve a cryptographic operation (e.g., one exponentiation for RSA and two for DSA). This would be in addition to cryptographic operations necessary to compute a Diffie-Hellman session key: two for S and at least one for R_1. (Here we are assuming that S always computes a new Diffie-Hellman exponent, whereas R_1 does not; to mimic their respective actions in all of the above OSBE schemes.) It becomes clear that the total costs (three exponentiations for S and one for R_1) would be the same as these for $NR - OSBE$, $EG/DSA - OSBE$ and $RSA - OSBE$. This is an interesting observation demonstrating that, for some OSBE schemes, there is **no extra cost for added privacy**.

[6] We do not include IBE-based OSBE schemes since they are non-interactive and, also, because BLS-IBE involves cryptographic operations in a very different setting, while Rabin-OSBE is very space-inefficient.

[7] Re-using blinding factors would sacrifice the impostor obliviousness property (see Section 9) but would not affect other security properties.

8 Implementation

Section 7 presented a comparison of the cost factors for different OSBE schemes. However, this only gives us a rough overview of the efficiency of OSBE schemes since arithmetic operations are performed in different algebraic structures. Consequently, a comparison of the number of multiplications alone does not provide a fair overall cost comparison. For instance, modular operations in ElGamal-OSBE are performed in $\mathbb{Z}_p^*$ while modular operations in DSA-OSBE are performed in a subgroup of $\mathbb{Z}_p^*$ (of order q) and are appreciably cheaper. Apart from that, arithmetic operations in BLS-IBE-OSBE scheme [6] – which were not considered in Section 7 – are performed in a different algebraic setting and thus a fair comparison becomes more difficult.

To provide a more accurate comparison, we implemented all OSBE schemes (including BLS-IBE-OSBE) in 'C'. We used popular OpenSSL cryptographic library for modular arithmetic with long integers and Miracl library for the implementation of BLS-OSBE. We modified some functions in the Miracl IBE package and introduced some new data structures. The following modifications were made:

- The original IBE implementation uses AES to encrypt the message (using a randomly chosen session key) while all other our OSBE implementations use RC4. For the sake of consistency we replaced AES with RC4.
- Miracle saves the parameters of the IBE scheme, the extracted key corresponding to an ID string (which, in our case, is the signature), the ciphertext and the decrypted cleartext in separate files and loads them every time they are needed. Whereas, other OSBE schemes use user-defined data structures kept in memory. Once again, for consistency's sake, we modified Miracl to work with user-defined data structures.

We also consider two implementation flavors of RSA-OSBE: plain and optimized. Recall that in RSA-OSBE, R_1 sends to S a blinded RSA signature h^{x+d} using the blinding factor h^x. Since it is chosen anew for each interaction, RSA-OSBE provides two extra properties (discussed in Section 9): Perfect Forward Secrecy (PFS) and Impostor Obliviousness. However, it also requires one extra modular exponentiation and one extra modular multiplication for each interaction. This slows down the scheme. Since neither property is, strictly-speaking, required and since none of the other OSBE schemes provide them[8] we can improve the performance of RSA-OSBE by re-using the same blinding factor. The optimized version is referred to as "RSA(optimized)" in Table 2 below.

We measured the schemes' performance with the following settings:

- For all ElGamal-family schemes we set $|p| = 1024$ and $q = 160$
- For RSA schemes we set $|n| = 1024$
- For BLS-IBE scheme we set $|p| = 512$ and $q = 160$

Our results (in milliseconds) are illustrated in Table 2. All of them were obtained on an IBM Thinkpad R40 with Pentium M processor running at 1.3Ghz with 256MB of RAM. The experimental OS platform was Debian Linux (Kernel Version $2.4.27$). Timings for each scheme represent average values taken over $1,000$ executions. The results illustrate that the most efficient scheme is Schnorr-OSBE while BLS-IBE-OSBE is the least efficient one. We also observe that three ElGamal family schemes (DSA-, NR- and Schnorr-OSBE) are more efficient than even the optimized RSA sheme.

# Runs:	RSA	RSA(optimized)	BLS-IBE(Miracl)	EG	DSA	NR	Schnorr
1,000	60.29	45.21	181.53	57.68	22.71	23.37	27.27

Table 2. Average running time for different OSBE Schemes

[8] Note, however, that PFS can be provided in all of these schemes using a blinding factor as described in Section 9.

9 Additional Features

In addition to the two security properties specified in [15] and in Section 2 (sender obliviousness and semantic security against the receiver), another interesting and useful feature is Perfect Forward Secrecy (PFS). PFS is a well-known property particularly desirable in key distribution and key agreement protocols. Informally, PFS means that compromise of a long-term secret (or secrets) does not result in compromise of short-term (session or ephemeral) secrets. In [15], this feature is considered but neither recognized nor referred to as PFS. Instead, it is called *inability to recover a shared secret even if the adversary knows the signature* and is treated as a useful but not mandatory feature.

Another way to motivate PFS in OSBE is to re-state it as: security against the original signer (TTP or CA), i.e., the party who originally issued σ to R_1. Since the signer is assumed to know all such signatures, it can successfully eavesdrop on all communication between S and R_1, unless, of course, PFS is provided.

RSA-OSBE (see Appendix B) offers PFS, as proven in [15]. In contrast, none of the OSBE schemes presented in this paper offer PFS. This can be easily seen by observing that, in all variants, Interaction involves R_1 computing, in Step 4, $K_r = Z^s$ or $K_r = Z^{-s}$. An adversary – who at some point discovers $\sigma = (e, s)$ – can thus trivially compute $K_r = K_s$.

While we recognize and acknowledge lack of PFS as a shortcoming, a small change to each of our OSBE schemes would enable PFS. The change involves adding a new quantity: $g^b \bmod p$ (where R_1 picks b at random from $\mathbb{Z}_p^*$) to Step 1 of the Interaction. Then, S computes K'_s as $K_s \cdot\ g^{bz}$ where K_s is the secret as computed by S in each protocol as presented above. Similarly, R_1 computes K'_r as $K_r \cdot\ g^{bz}$. This change does not influence any OSBE security properties for our schemes.

Showing that PFS is attained entails proving that the adversary (who knows σ and can compute $K_r = K_s$) cannot compute K'_s or K'_r, or equivalently, cannot compute g^{bz}. But, computing g^{bz} from g^z and g^b represents a solution to the CDH problem which is assumed to be intractable.

Another closely related feature is what we refer to as: **impostor obliviousness**. This, incidentally, is a new feature, not considered either in prior work. Suppose that S runs two different instances of OSBE Interaction, each time with someone who claims to be R_1 and claims to possess the necessary credentials (i.e., σ). We call an OSBE scheme *impostor-oblivious* if, after running both Interaction instances, S is unable to determine[9] whether one of the counter-parties was an impostor. (This definition can be trivially extended to cover more than two Interaction instances.)

Clearly, since our goal is to maximize anonymity, impostor-obliviousness is a very useful feature. Its main advantage is that, over multiple OSBE Interactions, the sender remains totally unaware of the genuineness of the population of receivers. It is easy to see that RSA-OSBE is impostor-oblivious, owing to the very same feature that provides PFS: randomized encryption of the RSA signature in Step 1 of RSA-OSBE Interaction. (Recall that in RSA-OSBE, for each Interaction, R_1 chooses a new random x and encrypts its RSA signature as $h^{x+d} \bmod n$. See Appendix B for the summary of RSA-OSBE.)

None of the OSBE schemes presented in this paper are impostor-oblivious. To see this, consider what happens if S engages in two instances of OSBE Interaction (using any of our proposed OSBE schemes): once with R_2 (the impostor) and once with R_1. Since each of our schemes involves revealing g^k in Step 1 of the Interaction component and this value is constant for a given signature σ, only R_1 reveals the correct g^k. Whereas, R_2 reveals some other value – $g^{k'}$. At that point, S would determine, with certainty, that one of the parties is an impostor. One of the items for our future work is the further investigation of impostor obliviousness for the proposed OSBE schemes.

10 Towards Secret Handshakes

As a concept, OSBEs are very closely related to Secret Handshakes [2, 22]. Briefly, a secret handshake scheme allows two parties to authenticate each other in an anonymous, unlinkable and unobservable manner such that one's membership is not revealed unless every other party's membership is also ensured. In more detail, a secure handshake allows two members of the same group to identify each other *secretly*, such that each party reveals its affiliation to the other if and only if the latter is also a group member. For example, an FBI agent (Alice) wants to authenticate to Bob only if Bob is

[9] With probability which is non-negligibly greater than $1/2$.

also an FBI agent. If Bob is *not* an FBI agent, he should be unable to determine whether Alice is one (and vice versa). This property can be further extended to ensure that affiliations are revealed only to members who hold specific *roles* in the group. For example, Alice might want to authenticate herself as an agent with a certain clearance level *only if* Bob is also an agent with at least the same clearance level.

OSBEs can be viewed as a sort of a one-sided or asymmetric secret handshakes. Based on the required security properties alone, it is easy to see that OSBEs are a simpler concept than secret handshakes. It is thus natural to wonder how to construct a secret handshake scheme out of an OSBE scheme. To conserve space, we limit our discussion to sketching out a general method and leave further details for future work.

The most naive approach is to simply combine two OSBE Interactions (S, R_1) and (R_1, S) to obtain a secret handshake. For clarity, we rename the two communicating parties as A and B (since neither is a sender or receiver in a secret handshake). Assuming that A has $\sigma_a = (e_a, s_a)$ and B has $\sigma_b = (e_b, s_b)$ we can build a simple secret handshake protocol composed of a single (assuming synchronized clocks) round:

$$A \longrightarrow B : \ g^{k_1}, \ Z_a$$

$$B \longrightarrow A : \ g^{k_2}, \ Z_b$$

where k_1 and k_2 are the respective randomizers in σ_a and σ_b.

Note that the single-round protocol is possible with RSA-OSBE, NR-OSBE, Schnorr-OSBE and OSBEs from ElGamal variants (3) and (4). However, it does not work for EG- and DSA-OSBE since they compute $Z = e^z = (g^k)^z$. (Thus an extra initial round to exchange g^{k_1}, g^{k_2} would be necessary.)

Thereafter, A computes K_s^a (acting as S with respect to A-to-B OSBE) from g^{k_2} and Z_a as well as K_r^a (acting as R_1 with respect to B-to-A OSBE) from Z_b and g^{k_1}. Finally, A sets $K_a = H'(K_s^a || K_r^a)$. B does the same by computing K_s^b, K_r^b and setting $K_b = H'(K_s^b || K_r^b)$.

We claim (albeit, without proof) that the above construct is a secure secret handshake protocol. In particular, no observer derives any information from the exchange and A and B compute $K_a = K_b$ if and only if each possesses a valid signature produced by the same authority (signer). Clearly, much more work is needed to properly assess our claim of security. We leave this as a major item for future research.

11 Related Work

The OSBE schemes introduced in this paper are closely related to those in [15] and secret handshakes [2, 9]. Since both of these topics have been amply discussed earlier, we do not elaborate on them further.

Another related cryptographic concept is Fair Exchange of Signatures (FES) [1, 3]. FES enables two parties to exchange signatures such that either both parties obtain the other party's signature or none of them obtains the other party's signature. There are several differences between OSBE and FES. First, OSBE does not require fair exchange. In OSBE, S sends its encrypted message to R_1 without receiving R_1's signature. S is assured only that R_1 will obtain the message if it has a required signature. This allows for efficient OSBE protocols, without the involvement of any third parties (other than the original signer, of course). Another difference is that in FES, signatures are generated by the parties involved in the protocol. Whereas, in OSBE the signatures are generated by a certification authority or some other trusted third party. Also, at some stage in FES protocols, one party finds out that the other party has a signature without obtaining that signature. This does not hold for OSBE and would in fact violate one of the OSBE security requirements (sender obliviousness).

12 Summary and Future Work

In this paper we began where the initial OSBE work had left off. Our main result is a collection of OSBE schemes for all signature schemes in the ElGamal family, including Nyberg-Rueppel, Schnorr and DSA. Proposed OSBE schemes are simple and quite efficient. Furthermore, they can be easily used to build secret handshake protocols. There are several directions for future work:

- Further exploration of the impostor-awareness feature.
- Design of Secret Handshake protocols from OSBE schemes presented in this paper.
- Investigation of other OSBE applications, including Automated Trust Negotiation.

References

[1] N. Asokan, V. Shoup and M. Waidner, *Optimistic Fair Exchange of Digital Signatures* IEEE Journal on Selected Areas in Communications, Vol. 18, No. 4, April 2000.

[2] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong, *Secret Handshakes from Pairing-Based Key Agreements*, In Proceedings of *IEEE Symposium on Research in Security and Privacy*, May 2003.

[3] F. Bao, R. Deng and W. Mao, *Efficient and Practical Fair Exchange Protocols with Off-line TTP*, In Proceedings of 1998 IEEE Symposium on Security and Privacy, May 1998.

[4] M. Bellare, D. Micciancio, and B. Warinschi, *Foundations of Group Signatures: Formal Definitions, Simplified Requirements and a Construction Based on General Assumptions*, In Proceedings of *EUROCRYPT 2003*.

[5] D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing*, SIAM Journal of Computing, Vol. 32, No. 3, pp. 586-615, 2003.

[6] D. Boneh, B. Lynn, and H. Shacham, *Short Signatures from the Weil Pairing*, In Proceedings of Asiacrypt 2001, volume 2248 of LNCS, pages 51432. Springer-Verlag, 2001.

[7] R. Bradshaw, J. Holt, and K. Seamons, *Concealing Complex Policies with Hidden Credentials*, In Proceedings of *ACM CCS 2004*.

[8] S. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*, MIT Press, August 2000.

[9] C. Castelluccia, S. Jarecki, and G. Tsudik, *Secret handshakes from ca-oblivious encryption*, In Proceedings of *ASIACRYPT 2004*.

[10] D. Chaum, *Blind Signatures for Untraceable Payments*, In Proceedings of CRYPTO 1982.

[11] C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, In Proceedings of Cryptography and Coding: 8th IMA International Conference, Springer-Verlag, LNCS Vol. 2260/2001, December 2001.

[12] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, *IEEE ToIT*, Vol. 22 pp. 644–654, November 1976.

[13] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, No. 4, 1985.

[14] J. Kilian and E. Petrank, *Identity Escrow* In Proceedings of CRYPTO 1998.

[15] N. Li, W. Du and D. Boneh, *Oblivious Signature-Based Envelopes*, In Proceedings of ACM Symposium on Principles of Distributed Computing (PODC'2003), 2003. Extended version to appear in of Distributed Computing, 2005.

[16] A. Menezes, P. Van Oorschot and S. Vanstone, *Handbook of Applied Cryptography, Chapter 11*, CRC press, 2nd Edition, 2001.

[17] National Institute of Standards and Technology, *Digital Signature Standard*, *NIST FIPS PUB 186*, U.S. Department of Commerce, 1994.

[18] K. Nyberg and R. Rueppel, *A New Signature Scheme Based on DSA Giving Message Recovery*, In Proceedings of ACM Conference on Computer and Communications Security, November 1993.

[19] S. Pohlig and M. Hellman, *An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance*, IEEE Transactions on Information Theory, Vol. 24 pp. 106-110, January 1978.

[20] D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes*, In Eurocrypt'96, pp. 387 - 398, 1996.

[21] C. Schnorr, *Efficient Signature Generation by Smart Cards*, Journal of Cryptology, Vol. 4, pp. 161-174, 1991.

[22] S. Xu and M. Yung. *k-Anonymous Secret Handshakes with Reusable Credentials*, In Proceedings of *ACM CCS 2004*.

Appendix A: Semantic Security of NR-OSBE, EG-OSBE and DSA-OSBE

In Section 4 we used the forking lemma to prove that Schnorr-OSBE is semantically secure against the receiver. More precisely, we prove that assuming there is no polynomial time algorithm which can solve the CDH-Problem and assuming that the hash functions H and H' are modelled as random oracles, Schnorr-OSBE is semantically secure against the receiver.

Recall that in the Schnorr signature scheme, the hash value e is not computed on the message M alone but on the concatenation of M and g^k which is determined by the signer ($e = H(M, g^k)$). We use this fact to prove semantic security for Schnorr-OSBE as follows:

Using the forking lemma in the ROM we can show that if an adversary, who has only access to the public data, can compute the OSBE key, we can construct for one g^k two values $e = H(M, g^k)$ and $e' = H'(M, g^k)$ with $e \neq e'$ for two different random oracles H and H'.

This in turn, enables us to solve the CDH problem (see proof of Theorem 2 in Section 4). However, we cannot use the same "trick" for OSBE schemes based on Nyberg/Rueppel, ElGamal or DSA signatures. This is because, in all these signature schemes, the hash is computed only over the message M. Therefore, the forking lemma does not help us prove semantic security of these OSBE schemes. In fact, proving semantic security of these schemes based on the Decision Diffie Hellman (DDH), the Computational Diffie Hellman (CDH) or the Discrete Logarithm (DL) assumption alone would be a major progress not only in proving the security of the corresponding OSBE schemes but also in proving the security of the underlying signature schemes. (We note the security of none of these signature schemes has been shown to be based on efficiently solving DDH, CDH or DL problem.)

Another possibility would be to prove semantic security by showing that an adversary who can compute the OSBE key can be used to forge signatures in the underlying signature scheme. Proving this also seems difficult: the signatures in all three signature schemes have the form (e, s), where e is computed mod p and s is computed mod $p - 1$ in ElGamal and mod q in Nyberg/Rueppel and DSA. In each OSBE scheme, S sends Z to the receiver who (if he is R_1) computes the OSBE secret as $Z^s \bmod p$. Therefore, the signature and the OSBE key are computed in different algebraic structures.

Proving that an adversary who can break the OSBE scheme can also forge signatures in the underlying signature scheme means being able to choose Z such that Z^s yields the signature or a part of the signature. In particular, if the desired part of the signature the adversary wants to forge is s, extracting the value s from Z^s is equivalent to computing the discrete logarithm of Z^s to base Z, which is assumed to be intractable.

Therefore, we make the following assumptions about the security of the three OSBE schemes:

Nyberg/Rueppel-OSBE: We assume that there is no polynomially bounded adversary $\mathcal{A}$, who can win with non-negligible probability the following game:

1. $\mathcal{A}$ is given a message M and a public key (p, q, g, y) with $y = g^a$, where a is only known to the signer.
2. $\mathcal{A}$ chooses a value $e = hg^{-k}$ with $h = H(M)$ and outputs it.[10]
3. $\mathcal{A}$ is given the value g^z.
4. $\mathcal{A}$ outputs its OSBE key K.

$\mathcal{A}$ wins the above game if and only if $K = g^{z(ea+k)}$.

The above means that $\mathcal{A}$ knows the values (p, q, g, y), M, e, $g^k = \left(\frac{e}{h}\right)^{-1}$ and g^z. Using y, g^k and e, $\mathcal{A}$ computes g^{ea+k}. In the best case, $\mathcal{A}$ also knows k. However, even in that case, $ea + k$ can assume any value since a is only known to the signer and, for given e, k and $s \in \mathbb{Z}_q^*$, there exists an $a = e^{-1}(s - k)$ with $s = ea + k$.

More precisely, it holds for $S_{e,k} = \left\{s \mid \exists\, a \in \mathbb{Z}_q^* :\; s = ea + k\right\} : |S_{e,k}| = q - 1$. Therefore, $\mathcal{A}$, who wins the above game with non-negligible probability, can compute $g^{z(ea+k)}$ from g^z and g^{ea+k} without knowing z or $ea + k$ and z and $ea + k$ are completely unrelated. Therefore, he can solve **an instance** of the CDH problem. Note that this is quite different from solving the CDH problem (since doing so would mean solving it for **all** instances).

Our security assumption for Nyberg/Rueppel is that a polynomially bounded adversary cannot win in the above game, i.e., solve the instance (g^z, g^{ea+k}) of the CDH problem. If this assumption holds, we can make the following statement:

> *If H' is modeled as a random oracle and the Nyberg/Rueppel signature scheme is existentially unforgeable*[11]*, NR-OSBE is semantically secure against the receiver.*

ElGamal-OSBE/DSA-OSBE: Here, we also assume that there is no polynomially bounded adversary $\mathcal{A}$ who can win with non-negligible probability the following game:

[10] Recall from Section 5 that e must have the form $e = hg^{-k}$, otherwise, it is rejected by S.

[11] Note that we apply a hash function H on the message M before we apply the signature generation algorithm on it. If no such hash function is applied on M, the Nyberg/Rueppel signature scheme is existentially forgeable.

1. $\mathcal{A}$ is given a message M and a public key (p, g, y) for ElGamal-OSBE and (p, q, g, y) for DSA-OSBE with $y = g^a$, where a is only known to the signer.
2. $\mathcal{A}$ chooses $e = g^k$ and outputs it.
3. $\mathcal{A}$ is given $e^z = g^{zk}$.
4. $\mathcal{A}$ outputs its OSBE key K.

$\mathcal{A}$ wins the above game if and only if $K = g^{z(-h+ea)}$ with $h = H(M)$. This means that $\mathcal{A}$ knows (p, g, y) for ElGamal-OSBE ((p, q, g, y) for DSA-OSBE), g^k, g^{kz} and $g^{(-h+ea)}$.

If $\mathcal{A}$ wins the above game with non-negligible probability, it can also compute $g^{z(-h+ea)}$ from g^k, g^{zk} and g^{-h+ea}. Similarly, we assume that this is impossible for a polynomially bounded adversary $\mathcal{A}$ who does not have a valid signature on M. If this assumption holds, we can conclude the following:

> *If H' is modeled as a random oracle and the ElGamal (DSA) signature scheme is existentially unforgeable, then ElGamal-OSBE (DSA-OSBE) is semantically secure against the receiver.*

Appendix B: OSBE for RSA

We present the RSA-OSBE scheme from [15] in order make the paper self-contained. Also, RSA-OSBE is a good “measuring stick” for the proposed OSBE schemes. RSA-OSBE is defined as follows:

Setup: This algorithm takes as input a security parameter t and creates an RSA key: (n, e, d), selects a suitable cryptographic hash function H, a function H' for key derivation and two security parameters t_1 and t_2, which are linear in t. It then chooses a semantically secure symmetric encryption scheme $\mathcal{E}$, two messages M and P and computes the signature $\sigma = h^d \bmod n$. It then gives M, σ and (n, e) to R_1, M and (n, e) to R_2 and M, P and (n, e) to S.

Interaction:

Step 1a: $R_1 \longrightarrow S: \quad Y = h^x \sigma \bmod n$

OR

Step 1b: $R_2 \longrightarrow S: \quad Y = h^{x'} \bmod n$ for some $x' \in \mathbb{Z}_n^*$

Step 2: S receives Y, picks a random $z \in \{1..2^{t_1} n\}$, computes $K_s = (Y^e/h)^z = (h^{(x+d)ez}/h) = h^{zxe} \bmod n$, derives $k_s = H'(K_s)$ and:

Step 3: $S \longrightarrow R_1$ or $R_2: \quad Z = h^z \bmod n,\ C = \mathcal{E}_{k_s}[P]$

Step 4: R_1 receives (Z, C), computes $K_r = Z^{xe} = h^{zxe}$, derives $k_r = H'(K_r)$ and finally decrypts C with k_r.

As shown in [15], RSA-OSBE is sound, oblivious and semantically secure against the receiver.

Appendix C: Potential Applications

We briefly discussed the motivation for OSBEs in Section 1. In this section, we consider using OSBEs in client-server interactions where client anonymity plays an important role. In general, we believe that OSBEs can be used in any application where a client needs to access a resource anonymously, but, with authorization. We consider two types of such applications: (1) distribution of Blogs and (2) distribution of content in peer-to-peer (P2P) networks.

Suppose that Alice is a member of a dissident group G in an authoritarian state. Since membership in this group is illegal and anyone suspected of being a member can risk serious consequences, Alice has to hide her membership with respect to non-members. In particular, she cannot disclose her membership to anyone she does not know. However, as an active member of G, Alice wants to follow the activities of the group G and, in particular, to be informed about secret group meetings.

Some members of G have weblogs, where they provide information about secret meetings. However, there is an essential difference between these and ordinary weblogs. Only members of G should be allowed to read the weblog contents. Clearly, if the secret police were to read weblog contents, it would find out the times and places of secret meetings. By using OSBEs, we can solve the problem as follows:

Each member A with identification information ID_A gets from the group authority a certificate attesting to its membership. This certificate is essentially a signature on a message M=“ID_A is a member of G”. Each time A requests a weblog, it executes the OSBE protocol with the Blog server. The Blog server encrypts the data such that only someone who has a valid signature on M, can decrypt it. Even if the Blog server is ever seized (say, by the police) and the identification information from past OSBE transactions is obtained, there would be no proof of membership for anyone who may have taken part in past transactions.

A very similar OSBE application is in P2P content-sharing networks. Here, both the content provider and the party requesting the content are peers. A peer might need a certain license for accessing some files but might not want the provider to prove to a third party that it has that license. As with Blogs, a client can use OSBEs to ensure anonymity. We are currently working on the implementation of an OSBE plug-in for a popular P2P client library.