Full version of an extended abstract published in Proceedings of Asiacrypt 2010, Springer-Verlag, 2010.

Limitations on Transformations from Composite-Order to Prime-Order Groups: The Case of Round-Optimal Blind Signatures

Sarah Meiklejohn
UC San Diego
smeiklej@cs.ucsd.edu

Hovav Shacham
UC San Diego
hovav@cs.ucsd.edu

Abstract

Beginning with the work of Groth and Sahai, there has been much interest in transforming pairing-based schemes in composite-order groups to equivalent ones in prime-order groups. A method for achieving such transformations has recently been proposed by Freeman, who identified two properties of pairings using composite-order groups — cancelling and projecting — on which many schemes rely, and showed how either of these properties could be obtained using prime-order groups.

In this paper, we show that there are limits to such transformations. Specifically, we show that Freeman's properties, cancelling and projecting, cannot simultaneously be obtained using prime-order groups when subgroup hiding is provided by the Decisional Linear assumption in a natural way. We present a natural cryptosystem whose proof of security makes use of a pairing that is both cancelling and projecting, as evidence that these properties can be helpful together as well as individually.

Our example cryptosystem is a simple round-optimal blind signature scheme that is secure in the common reference string model, without random oracles, and based on mild assumptions; it is of independent interest.

1 Introduction

Composite-order groups were introduced for pairing-based cryptography in 2005, in the work of Boneh, Goh, and Nissim [12], and have since been used to realize a large number of cryptographic systems (see, e.g., the schemes surveyed by Freeman [25]). At the same time, the limited number of elliptic curve families on which composite-order groups can be instantiated and the larger parameter sizes associated with composite-order groups (cf. [24, 13]) has motivated research on translating these schemes to or obtaining similar ones in the prime-order setting.

In one of the first papers to unify the composite- and prime-order settings, Groth and Sahai [32] developed non-interactive zero-knowledge schemes that not only can be instantiated either in composite- or prime-order groups, but are in fact described identically in either instantiation. What facilitates this is a new abstraction for pairing-based crypto in terms of modules over finite commutative rings with an associated bilinear map; this abstraction allows for the simultaneous treatment of three different cryptographic assumptions: the Subgroup Hiding (SGH) assumption of Boneh, Goh, and Nissim [12], in composite-order groups; the Decisional Linear (DLIN) assumption of Boneh, Boyen, and Shacham [10] or its k-Linear family of generalizations [47, 34],[1] in prime-

[1] A family of progressively strictly weaker decisional assumptions, where 1-Linear is DDH and 2-Linear is DLIN.

order groups; and the so-called Symmetric External Diffie-Hellman assumption, also in prime-order groups.

More recently, Freeman [25] and Garg, Sahai, and Waters [28] have proposed methods for transforming schemes secure in the composite-order setting into ones secure (under different but analogous assumptions) in the prime-order setting. Freeman, in particular, identifies two properties of pairings on composite-order groups, *projecting* and *cancelling*, and shows how either can be obtained in prime-order groups. He then demonstrates how to tranform several known cryptosystems that rely on one of these properties from composite- to prime-order groups.

Our contribution: limits on transformations from composite to prime order. In this paper, we show limits to the feasibility of composite-to-prime transforms such as those mentioned above, suggesting that some schemes inherently require composite-order groups and cannot be transformed mechanically from one setting to the other. In our main theorem, Theorem 6.5, we show that no pairing over prime-order groups can, in Freeman's terminology, be both projecting and cancelling when subgroup indistinguishability relies in a natural way on k-Linear, where "natural" means that, for k-Linear, B consists of $k + 1$ copies of G and not some larger number of copies.

If no cryptosystem required a pairing that is both projecting and cancelling, however, our Theorem 6.5 would not be particularly interesting. As such, we present a new cryptosystem — a natural pairing-based blind signature scheme that is of independent interest, and discussed below — whose proof of security calls for a pairing that is both projecting and cancelling.[2]

Blind signatures were introduced by Chaum in 1982 [18]. In a blind signature scheme, a user interacts in a protocol with a signer to obtain a signature on a message of the user's choice. When the protocol execution ends, the user obtains the signature but the signer learns nothing about the message that was signed. Blind signatures have been used as a building block in a variety of applications, including electronic cash [21] and electronic voting [20].

One useful feature of a blind signature scheme is *concurrency*. For example, if a blind signature used to build an electronic cash system does not retain its security even when the signer engages in multiple protocol executions concurrently, it leaves the issuing bank susceptible to denial-of-service attacks. Concurrency turns out to be as difficult to achieve for blind signatures as it is for other cryptographic protocols. Many blind signature schemes have proofs of security only for sequential executions of the protocol, but the problem is not just with proofs. In one example, Martinet, Poupard, and Sola [40] show that signatures in a partially blind signature due to Cao, Lin and Xue [17] are forgeable if the signer interacts with two users concurrently.

Our contribution: a round-optimal blind signature scheme. In a second contribution, as mentioned above, we present a new pairing-based blind signature scheme. Our blind signing protocol is round-optimal: it consists of only two moves (a request and a response); this means that it is secure even in the presence of concurrent signing protocol executions. Our scheme is practical, has a proof of security (without random oracles) in the common reference string model, and relies for its security on falsifiable and non-interactive assumptions: computational Diffie-Hellman and Subgroup Hiding.

The assumptions we rely on are milder than those used in any previous practical concurrently secure blind signature, including those in the random oracle model. ("Practical" means not relying

[2] We emphasize that it is the security proof, not the statement of the scheme, that uses the two pairing properties. We thus do not rule out the possibility that a *different* proof strategy will show our scheme secure in prime-order groups.

on general NIZKs for NP as a building block.) Our scheme extends in a natural way to give a partially blind signature scheme [3] with the same properties.

Our blind signatures combine the Waters signature [48] with non-interactive witness-indistinguishable proofs developed in a line of papers by Groth, Ostrovsky, and Sahai [31, 30, 32]. In this our scheme is related to the group signature scheme of Boyen and Waters [15]. The primary disadvantage of our scheme, as with the Boyen-Waters group signature, is its bit-at-a-time nature, which makes the user's blind signing request large: some 40 kilobytes at the 1024-bit security level; the signer's response and the resulting signatures, however, are short.

Related work. The blind signature literature is extensive and varied. Below, we briefly survey the most closely related schemes with concurrent security; see [5, 4] for more complete recent treatments.

In the random oracle model, there exist elegant round-optimal blind signatures, due to Chaum [19] and Boldyreva [9], that feature short public keys, short signatures, and an efficient blind signing protocol. Unfortunately the security proofs for these schemes rely on strong interactive assumptions: the RSA known-target inversion assumption [8] and the chosen-target CDH assumption (by contrast, the underlying ordinary signatures can be shown secure using RSA and CDH, respectively).

In the common reference string model, several practical concurrently secure blind signature schemes have been proposed. Unlike our scheme, these schemes rely on assumptions that are interactive or whose statement size grows with the number of queries in the reduction (i.e., "q-type"). Kiayias and Zhou [36] give four-move blind and partially-blind signature schemes secure under the (interactive) LRSW assumption [39], the Paillier assumption [43], and DLIN. Okamoto [41] gives four-move blind and partially blind signature schemes based on the (q-type) Two-Variable Strong Diffie-Hellman assumption and Paillier. Fuchsbauer [26] gives two-move blind signature schemes based on the (q-type) Asymmetric Double Hidden Strong Diffie-Hellman assumption, the Asymmetric Weak Flexible CDH assumption, and DLIN. And Abe, Haralambiev, and Ohkubo [4] give two-move blind signature schemes based on the (q-type) Simultaneous Flexible Pairing assumption and DLIN. (The last two papers appeared together as [2].)

Also in the common reference string model, blind signatures that use general NIZKs for NP (and are therefore not practical) were given by Juels, Luby, and Ostrovsky [35], Fischlin [23], and Abe and Ohkubo [5]. The Fischlin and Abe-Ohkubo schemes are round-optimal.

Okamoto [41] first observed that the Waters signature can be combined with witness-indistinguishable proofs for a simple NP language to yield blind and partially blind signatures. Our scheme could be viewed as an instantiation of Okamoto's framework (though we blind the message differently) where we take advantage of Groth-Ostrovsky-Sahai proofs to eliminate a round of interaction.

Until recently, no concurrently-secure blind signature schemes were known in the plain public-key model. The first such scheme was given by Hazay et al. [33]; it relies on general NIZKs, and its round complexity is poly-logarithmic in the number of concurrent executions for which security must be guaranteed.

Applications and extensions. Finally, as an application of our techniques, we show (in Appendix E) how our blind signatures may be used within the Waters IBE system [48] to yield a blind IBE scheme, as introduced by Green and Hohenberger [29]. Compared to Green and Hohenberger's blind extraction protocol, our protocol achieves concurrent security but adds a common reference string and a reliance on the SGH assumption.[3] Furthermore, the Waters signature naturally extends into a hierarchical identity-based signature (cf. [44]); applying our construction at level 2 of the resulting signature gives an identity-based blind signature [49] concurrently secure in the common reference string model.[4] Or, using the Boyen-Waters group signature [15] at level 1 of the hierarchy and our blind signature at level 2 gives a group blind signature [38] concurrently secure in the common reference string model.

2 Mathematical Background

In this paper, we work with bilinear groups: cyclic groups G of some finite order that admit a bilinear map $e : G \times G \to G_T$. Because we generalize the concept of a group and work with modules, we are able to describe our scheme without relying on any particular properties of the underlying group (with the caveat, as mentioned above, that the scheme is provably secure only for composite-order groups).

2.1 Modules

First, we recall the definition of a module; this serves as the foundation for our blind signature scheme, and more specifically for the Groth-Sahai commitments used in our scheme.

Definition 2.1. *Let $(\mathcal{R}, +, \cdot, 0, 1)$ be a finite commutative ring. An $\mathcal{R}$-*module *A is an abelian group $(A, +, 0)$ such that there exists an operator (namely scalar multiplication) $\mathcal{R} \times A \to A$ such that $(r, x) \mapsto rx$ for $r \in \mathcal{R}$ and $x, rx \in A$. In addition, the following four properties are satisfied for all $r, s \in \mathcal{R}$ and $x, y \in A$:*

- $(r + s)x = rx + sx$.
- $r(x + y) = rx + ry$.
- $r(sx) = (rs)x$.
- $1x = x$.

This definition can also be written in more familiar multiplicative notation, where our operator becomes exponentiation rather than multiplication and the requirements are written as $x^{r+s} = x^r \cdot x^s$, $(xy)^r = x^r y^r$, $(x^r)^s = x^{rs}$, and $x^1 = x$ for all $r, s \in \mathcal{R}$ and $x, y \in A$.

In cryptography, we are most used to working with $\mathbb{Z}/n\mathbb{Z}$- and $\mathbb{F}_p$-modules; for example, any finite group of prime order p can be viewed as a $\mathbb{F}_p$-module. In addition, the concept of a module generalizes the concept of an abelian group, as any abelian group can be viewed as a $\mathbb{Z}$-module.

2.2 Groth-Sahai commitments

Groth and Sahai support two kinds of commitments: commitments to elements in an $\mathcal{R}$-module A, and commitments to exponents in the ring $\mathcal{R}$. For our purposes, we will need only commitments to bits; we can simplify things even further by always setting $A = G$ for our bilinear group G.

[3] Note that the efficient range proofs due to Boudot [14] rely on the Strong RSA assumption (due to Baric and Pfitzmann [7]) and require a common reference string. If the scheme of Green and Hohenberger is instantiated with these range proofs then its assumptions and setup model are comparable to those of our scheme, but without providing concurrent security.

[4] One could also obtain an identity-based blind signature through generic composition of our blind signature and an ordinary signature [27].

To form commitment to the module elements, Groth and Sahai define two homomorphisms $\tau : A \to B$ and $\rho : B \to A$.[5] These maps are defined such that, for some elements $h_1, \ldots, h_m$ in B, $\rho(h_i) = 1$ for all i and ρ is non-trivial for all x that are not contained in $B_1 = \langle h_1, \ldots, h_m \rangle$. A commitment to $x \in A$ is then defined as $c(x) = \tau(x) \prod_{i=1}^{m} h_i^{r_i}$ for random values $r_1, \ldots, r_m \leftarrow \mathcal{R}$. This means that the h_i elements act as keys for the commitment scheme, and that the CRS is $(\mathcal{R}, A, B, \tau, \rho, h_1, \ldots, h_m)$. There are two possible cases:

- Hiding keys: in this case, the h_i elements generate the whole module B; in other words $B_1 = \langle h_1, \ldots, h_m \rangle = B$. This implies that $\tau(A) \subseteq B_1$, which means that $c(x)$ will be perfectly hiding (as each commitment will just be a random element of B).
- Binding keys: in this case, $B_1 \neq B$ and $\rho(c) = \rho(\tau(x)h^r) = \rho \circ \tau(x)$ for some restricted space of x. To determine what this restricted space is, we see that c will generally reveal the coset of B_1 where $\tau(x)$ lives. So, in order for the commitment to be perfectly binding we must restrict the space of x to be the inverse image of $B_2 \simeq B/B_1$; because we know that $B_1 \neq B$, both B_2 and $\tau^{-1}(B_2)$ are non-trivial and so this domain restriction is actually meaningful. One final thing to note is that in order for the quotient module to be well-defined B_1 must be a normal submodule of B; because modules are by definition abelian every submodule is normal, and so the quotient module is always well-defined.

It is clear from these definitions that a set of keys cannot be both hiding and binding, as the settings require very different properties of the commitment keys $h_1, \ldots, h_m$. To get any meaningful blindness properties, however, we need these two settings to be indistinguishable. We therefore require an assumption that implies this; the choice of assumption depends on the instantiation being used.

3 Security Notions for Blind and Partially Blind Signatures

In what follows, we will define a blind signature scheme in the common reference string model as a collection of four protocols: a $\mathsf{Setup}(1^k)$ algorithm which outputs the CRS σ_{CRS}, a $\mathsf{KeyGen}(\sigma_{CRS})$ algorithm which outputs the signing keypair (pk, sk), a $\mathsf{BlindSign}$ protocol, which consists of an interaction of the form $\mathsf{User}(\sigma_{CRS}, pk, M) \leftrightarrow \mathsf{Signer}(\sigma_{CRS}, sk)$ (in which the signer outputs $\mathsf{success}$ if the protocol is successful, and the user outputs $\mathsf{success}$ and the unblinded signature σ), and finally a $\mathsf{Verify}(\sigma_{CRS}, pk, M, \sigma)$ algorithm which outputs accept if the signature is valid and fail if not.

In general, there are two properties that blind and partially blind signatures must satisfy: blindness and one-more unforgeability. Informally, the blindess requirement says that in the process of signing a user's message, the signer does not learn anything abut the message he is signing. The one-more unforgeability requirement says that if the user interacts with the signer ℓ times, then he should be able to produce ℓ signatures and no more (so in particular, he cannot produce $\ell + 1$). We will give more formal definitions of these properties later.

3.1 Blind signatures

Formal definitions of blind signatures were introduced by Juels, Luby, and Ostrovsky [35], although both properties were considered informally in Chaum's original paper on the subject [18], and one-more unforgeability was considered formally in Pointcheval and Stern's work on security arguments for signatures [45].

[5] Our notation is a bit different from the original Groth-Sahai notation, but the ideas are the same.

In the Juels-Luby-Ostrovsky formalization, the blindness property is defined as follows: the adversary is given a public-private keypair and outputs two messages M_0 and M_1. He then engages in two signing protocols with honest users: the first user requests a signature on message M_b and the second on message M_{1-b}, where b is a random bit unknown to the adversary. The adversary is then given the resulting signatures σ_0 and σ_1, but only if both interactions are successful, and his goal is to guess the bit b (given the messages, the corresponding signatures, and the signing protocol transcripts).

In this paper, we use a stronger version of the blindness property which allows the adversary to generate the signing keypair himself, possibly in a malicious manner. This strengthening was proposed independently in several recent papers [1, 42, 36].

The one-more unforgeability property can be defined as follows: the adversary is given a public key and engages in multiple executions of the blind signing protocol with a signer; the adversary is able to choose how to interleave the executions. At the end, the adversary is considered successful if he is able to output $\ell + 1$ distinct message-signature pairs $(M_1, \sigma_1), \ldots, (M_{\ell+1}, \sigma_{\ell+1})$, where ℓ is the number of executions in which the signer output $\mathsf{success}$.

In this definition, two message-signatures pairs (M_i, σ_i) and (M_j, σ_j) are considered distinct even if $M_i = M_j$ (so if σ_i and σ_j are just two different signatures on the same message) for $i \neq j$. Unfortunately, this means that any signature scheme in which signatures can be re-randomized (like our signature scheme, as we will see in Section 4) will automatically be unable to satisfy one-more unforgeability. We therefore weaken the property by requiring that the adversary be unable to output $\ell + 1$ message-signature pairs in which the *messages* are all distinct.[6] This modified definition was also considered recently by Okamoto [42].

Putting all this information together, we give a formal definition of our security definition for blind signature schemes in Appendix A.

3.2 Partially blind signatures

The properties for blind signatures can also be extended to partially blind signatures; these formalizations are due to Abe and Okamoto [6]. For partially blind signatures, the adversary outputs two info strings $\mathit{info}^{(0)}$ and $\mathit{info}^{(1)}$ in addition to its messages M_0 and M_1. It then interacts with two separate users in the same manner as before, except this time the first user requests a signature on M_b using $\mathit{info}^{(0)}$ and the second requests a signature on M_{1-b} with info $\mathit{info}^{(1)}$. The adversary is given the resulting signatures σ_0 and σ_1 if both interactions were successful and if $\mathit{info}^{(0)} = \mathit{info}^{(1)}$. As before, his goal is to guess the bit b.

The one-more unforgeability property is also quite similar to the property for blind signatures; here, the adversary is allowed to choose the info string for each interaction with the signer. The goal is then for the adversary to output an info string info^* as well as $\ell + 1$ message-signature pairs $(M_1, \sigma_1), \ldots, (M_{\ell+1}, \sigma_{\ell+1})$, where ℓ represents the number of interactions in which the signer output $\mathsf{success}$ while using the info string info^*.

In our security definitions, we extend the modifications from blind signatures to partially blind signatures as well; this means we strengthen the blindness game to allow the adversary to generate the signing keys, and we weaken the one-more unforgeability game to require that the messages M_i must all be distinct. These modifications can be extended in a natural way and so we omit the formal definition.

[6]We observe that blind signatures satisfying this weakened unforgeability property are still sufficient for e-cash and other standard constructions based on blind signatures.

4 Underlying Signature Scheme

As our underlying signature scheme, we make only a slight modification to the Waters signature scheme. Essentially, we just need to generalize the Waters signature scheme by bringing it into the language of modules so that we can use it in combination with GS commitments to create our blind signature scheme.

4.1 CRS setup

For the Waters signature, the required elements for the CRS are a bilinear group G, the target group G_T and the bilinear map $e : G \times G \to G_T$, as well as generators $g, u', u_1, \ldots, u_k$ for G, where k denotes the length of the messages we will be signing. We now add in the elements discussed in Section 2.2: we start with a ring $\mathcal{R}$ such that G can be interpreted as an $\mathcal{R}$-module. We then add in an $\mathcal{R}$-module B, a map $\tau \colon G \to B$, a map $\rho \colon B \to G$, and a bilinear map $E \colon B \times B \to B_T$, which also requires us to specify the target module B_T and the resulting τ_T and ρ_T maps. This means that the CRS will be $\sigma_{sig} = (\mathcal{R}, G, G_T, B, B_T, e, E, \tau, \tau_T, \rho, \rho_T, g, u', u_1, \ldots, u_k)$; the relations between all these maps can be summarized in the following figure:

$$\begin{tikzcd}
G \arrow[d, shift right, "\tau"'] & \times & G \arrow[d, shift right, "\tau"'] \arrow[r, "e"] & G_T \arrow[d, shift right, "\tau_T"'] \\
B \arrow[u, shift right, "\rho"'] & \times & B \arrow[u, shift right, "\rho"'] \arrow[r, "E"] & B_T \arrow[u, shift right, "\rho_T"']
\end{tikzcd}$$

Figure 1: Commutative diagram for our modules.

4.2 Signing protocol

In our generalized Waters signature, the size of the message space will be $\{0,1\}^k$ for some value k (for example, if we use hash-and-sign with SHA-1 as the hash function, we would use $k = 160$). As noted above, the CRS will contain $k+1$ random generators of G, and the CRS will be shared between the user and the issuer.

- $\mathsf{Setup}(1^k)$: Output a tuple σ_{sig} that has been computed as described in the previous section.
- $\mathsf{KeyGen}(\sigma_{sig})$: Pick a random value $\alpha \leftarrow \mathcal{R}$ and set $A = E(\tau(g), \tau(g))^\alpha$. The public key will be $pk = A$ and the secret key will be $sk = \alpha$ (actually, $\tau(g)^\alpha$ will suffice).
- $\mathsf{Sign}(\sigma_{sig}, sk, M)$: Write the message out bitwise as $M = b_1 \ldots b_k$, and write $sk = \tau(g)^\alpha$. Pick a random $r \leftarrow \mathcal{R}$ and compute

  $$S_1 = \tau(g)^\alpha \Big(\tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\Big)^r \quad \text{and} \quad S_2 = \tau(g)^{-r}.$$

  Output the signature $\sigma = (S_1, S_2)$.
- $\mathsf{Verify}(\sigma_{sig}, pk, M, \sigma)$: Again, write the message out bitwise as $M = b_1 \ldots b_k$; also write the signature as $\sigma = (S_1, S_2)$ and the public key as $pk = A$. Check that these values satisfy the

following equation:

$$E(S_1, \tau(g)) \cdot E\Big(S_2, \tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\Big) = A. \tag{1}$$

If they do, output accept; else, output fail.

One nice property of the Waters signature (and our extended Waters signature) is that anyone can re-randomize a signature by choosing $s \leftarrow \mathcal{R}$ and computing $S_1' = S_1 \cdot \left(\tau(u') \prod_i \tau(u_i)^{b_i}\right)^s$ and $S_2' = S_2 \cdot \tau(g)^{-s}$. Since this results in $S_1' = \tau(g)^\alpha \left(\tau(u') \prod_i \tau(u_i)^{b_i}\right)^{r+s}$ and $S_2' = \tau(g)^{-(r+s)}$, the re-randomization process really does give us a valid signature. In particular, the randomness in the resulting signature (S_1', S_2') will be information-theoretically independent from the randomness r chosen by the signer in the signature (S_1, S_2).

We recall the computational Diffie-Hellman (CDH) assumption used for the Waters signature:

Assumption 4.1. *Assuming there exists a generation algorithm $\mathcal{G}$ that outputs a tuple (q, G, g), where G is of order q (not necessarily prime) with a generator g, it is computationally infeasible to compute the value g^{ab} given the tuple (g, g^a, g^b). More formally, for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that the following holds for all $k > k_0$:*

$$Pr\big[(q, G, g) \leftarrow \mathcal{G}(1^k);\ a, b \leftarrow \mathbb{Z}_q;\ c \leftarrow g^{ab}\ :\ \mathcal{A}(g, g^a, g^b) = c\big] = \nu(k).$$

The Waters signature scheme is existentially unforgeable if the CDH assumption holds on G; in our extended version, the signature scheme will be existentially unforgeable if the CDH assumption holds on B. As the proof is just a trivial extension of the proof from Waters, we will not include it here.

5 Our Blind Signature

In this section, we describe our blind signature scheme. Although we describe only the partially blind setting, our description also encapsulates the fully blind setting (as it just corresponds to the case where we set $k_0 = 0$).

5.1 CRS setup

In our CRS, we must include all the necessary elements for GS commitments, as well as values in the σ_{sig} in the previous section. This means the CRS here will be $\sigma_{CRS} = (\sigma_{sig}, h_1, \ldots, h_m)$, where the h_i elements are binding keys for Groth-Sahai commitments (under the given instantiation).

5.2 The partially blind protocol

In the following protocol, the user and signer both have access to an info string *info*. At the end of the protocol, the user obtains a signature on $info||M$ for a message M, while the signer learns nothing beyond the fact that the message M followed the guidelines laid out in *info*. In addition, the user and the signer will have agreed upon the length of the *info* string; call it k_0 for $0 \leq k_0 \leq k$. Setting $k_0 = 0$ corresponds to a fully blind signature, while setting $k_0 = k$ corresponds to an ordinary run of the signature scheme described in the previous section.

- $\mathsf{Setup}(1^k)$: Output σ_{CRS} as described in the previous section (Section 5.1).
- $\mathsf{KeyGen}(\sigma_{CRS})$: Same as KeyGen from Section 4.2.

- $\mathsf{User}(\sigma_{CRS}, pk, info, M)$: First write the info string out bitwise, so as $info = b_1 \dots b_{k_0}$, and similarly write the message out as $M = b_{k_0+1} \dots b_k$. Now, for each i such that $k_0 < i \le k$, pick random values $t_{i1}, \dots, t_{im} \leftarrow \mathcal{R}$ and compute

$$c_i = \tau(u_i)^{b_i} \cdot \prod_{j=1}^{m} h_j^{t_{ij}} \quad \text{and} \quad \pi_{ij} = \Big(\tau(u_i)^{2b_i - 1} \cdot \prod_{j=1}^{m} h_j^{t_{ij}}\Big)^{t_{ij}},$$

where c_i acts as a GS commitment to the bit b_i and $\vec{\pi}_i = \{\pi_{ij}\}_{j=1}^m$ acts as a proof that the value contained in c_i is in fact a 0 or a 1. Send the tuple $req = (c_{k_0+1}, \vec{\pi}_{k_0+1}, \dots, c_k, \vec{\pi}_k)$ as a request to the issuer (and save some state information $state$).

- $\mathsf{Signer}(\sigma_{CRS}, sk, info, req)$: First, write $info = b_1 \dots b_{k_0}$ and $sk = \tau(g)^\alpha$. Upon receiving the request, check that each c_i is indeed a commitment to a 0 or 1 by checking that

$$E\big(c_i, \tau(u_i)^{-1} c_i\big) = \prod_{j=1}^{m} E(h_j, \pi_{ij}) \tag{2}$$

for each $k_0 < i \le k$. If this equation fails to hold for any value of i, abort the protocol and output $\perp$. Otherwise, compute the value

$$c = \tau(u')\Big(\prod_{i=1}^{k_0} \tau(u_i)^{b_i}\Big)\Big(\prod_{i=k_0+1}^{k} c_i\Big).$$

Finally, pick a random value $r \leftarrow \mathcal{R}$ and compute

$$K_1 = \tau(g)^\alpha \cdot c^r, \quad K_2 = \tau(g)^{-r}, \quad \text{and} \quad K_{3j} = h_j^{-r} \quad \text{for } 1 \le j \le m.$$

Denote $\vec{K}_3 = \{K_{3j}\}_{j=1}^m$, send the tuple $(K_1, K_2, \vec{K}_3)$ back to the user, and output $\mathsf{success}$ and $info$.

- $\mathsf{User}(state, (K_1, K_2, \vec{K}_3))$: First, check that K_2 and $\vec{K}_3$ were formed properly by checking satisfiability of

$$E\big(K_{3j}, \tau(g)\big) = E(K_2, h_j) \tag{3}$$

for each $1 \le j \le m$. If this equation does not verify for some j, abort and output $\perp$. Otherwise, unblind the signature by computing

$$S_1 = K_1 \prod_{i=k_0+1}^{k} \prod_{j=1}^{m} K_{3j}^{t_{ij}} \quad \text{and} \quad S_2 = K_2. \tag{4}$$

Now, verify that this is a valid signature on $info||M$ by running $\mathsf{Verify}(\sigma_{CRS}, pk, info||M, (S_1, S_2))$. If this outputs fail, abort the protocol and output $\perp$. If it outputs accept, however, re-randomize the signature by choosing a random value $s \leftarrow \mathcal{R}$ and computing

$$S_1' = S_1\Big(\tau(u')\prod_{i=1}^{k} \tau(u_i)^{b_i}\Big)^s \quad \text{and} \quad S_2' = S_2 \cdot \tau(g)^{-s}.$$

The final signature will then be $\sigma = (S_1', S_2')$; output σ, as well as $info$ and $\mathsf{success}$.

- $\mathsf{Verify}(\sigma_{CRS}, pk, M, \sigma)$: Same as Verify from Section 4.2.

We give a proof of the following theorem in Appendix B:

Theorem 5.1. *The blind signature scheme outlined above is correct and partially blind, under the assumption that the h_i values in the hiding and binding settings are indistinguishable.*

This theorem demonstrates correctness and (partial) blindness, but it does not show one-more unforgeability. In order to prove this, we need to first define two properties of pairings, adapted from Freeman [25] for our purposes:

Definition 5.2. *A pairing $E \colon B \times B \to B_T$ is* cancelling *if there exists a decomposition $B = B_1 \times B_2$ such that $E(b_1, b_2) = 1$ for all $b_1 \in B_1$, $b_2 \in B_2$.*

Definition 5.3. *A pairing $E \colon B \times B \to B_T$ is* projecting *if there exists a decomposition $B = B_1 \times B_2$, a submodule $B'_T \subset B_T$, and maps $\pi \colon B \to B$ and $\pi_T \colon B_T \to B_T$, such that $B_1 \subseteq \ker(\pi)$, $\pi(x) = x$ for $x \in B_2$, $B'_T \subseteq \ker(\pi_T)$, and $\pi_T(E(x, y)) = E(\pi(x), \pi(y))$ for all $x, y \in B$.*

Observe that because π leaves values in B_2 unchanged, neither π nor π_T can be the trivial map (i.e., the map that is uniformly 1). As we will see in the next section, these properties are both trivially provided in the instantiation under the SGH assumption. Because SGH also provides the necessary indistinguishability properties, we can prove the following theorem, a proof of which can be found in Appendix C:

Theorem 5.4. *The blind signature scheme outlined above is one-more unforgeable under the SGH assumption and the assumption that the modified Waters signature scheme in Section 4 is existentially unforgeable on the submodule $B_2 \subseteq B$.*

5.2.1 Instantiation under the SGH assumption

We first recall the Subgroup Hiding (SGH) assumption:

Assumption 5.5 (Boneh-Goh-Nissim [12])**.** *Assuming a generation algorithm $\mathcal{G}$ that outputs a tuple (p, q, G, G_T, e) such that $e \colon G \times G \to G_T$ and G and G_T are both groups of order $n = pq$, it is computationally infeasible to distinguish between an element of G and an element of G_q. More formally, we have that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that the following holds for all $k > k_0$:*

$$\Big| Pr\big[(p, q, G, G_T, e) \leftarrow \mathcal{G}(1^k); n = pq; x \leftarrow G : \mathcal{A}(n, G, G_T, e, x) = 0\big] - Pr\big[(p, q, G, G_T, e) \leftarrow \mathcal{G}(1^k); n = pq; x \leftarrow G : \mathcal{A}(n, G, G_T, e, x^p) = 0\big] \Big| < \nu(k),$$

where $\mathcal{A}$ will output 1 if it believes x is in G_q and 0 otherwise.

To instantiate the scheme under this assumption, we will work with a group G of order $n = pq$ for p, q prime. We then define $B = G$ and τ such that $\tau(x) = x$ (so τ is just the identity); this means that we can use $E = e$. We need only one h_i element, namely an h_1 such that h_1 generates G_q in the binding setting and h_1 generates the whole group G in the hiding setting. The SGH assumption tells us that these choices of h_1 are indistinguishable. We can also describe our ρ map as $\rho(c_i) = c_i^q = (u_i^q)^{b_i}$ since h_1 has order q. Because the u_i are all generators for G and therefore $u_i^q \neq 1$, we can see that the ρ map will indeed reveal the bit b_i.

Because h_1 will generate either G or G_q, we have $B = G_p \times G_q$. To see that the pairing e is cancelling, note that every element of G_p can be written as $a = g^{\alpha q}$ for some $\alpha \in \mathbb{F}_p$ and every element of G_q can be written as as $b = g^{\beta p}$ for some $\beta \in \mathbb{F}_q$. Then $e(a, b) = e(g^{\alpha q}, g^{\beta p}) =$

$e(g^{\alpha\beta pq}, g) = e\big((g^n)^{\alpha\beta}, g\big) = 1$ because G has order n. Furthermore, e is projecting. To see this, note that there exists a value λ such that $\lambda \equiv 1 \bmod p$ and $\lambda \equiv 0 \bmod q$, and that furthermore this value is efficiently computable (given the factorization of n) using the Chinese Remainder Theorem. By computing x^λ for some $x \in B$, we cancel out the G_q component of x, while leaving the G_p component unchanged. This allows us to define $\pi(z) = \pi_T(z) = z^\lambda$, which can be easily seen to satisfy the projecting property.

Finally, to actually compute the value h_1, we can set $h_1 = g$ in the hiding setting and $h_1 = g^p$ in the binding setting. This means that, as with the map ρ, the factorization of n will be required as a trapdoor to compute h_1.

The obvious downside of using our scheme under the SGH assumption is the use of a composite-order group, which necessitates a common reference string generated by a trusted third party.[7] The upside, on the other hand, is that the scheme is as efficient as possible under this assumption, as each part of the signature involves only one group element.[8]

6 Converting to a Prime-Order Setting

In this section, we would like to argue why our scheme, with its current set of security requirements, cannot be instantiated under an assumption for prime-order groups, in particular for the k-Linear family of assumptions. While any scheme based on Groth-Sahai proofs requires the projecting property from Definition 5.3 and the indistinguishability of elements in B_1 and B (i.e., the indistinguishability of hiding and binding commitment keys), our scheme requires the extra cancelling property from Definition 5.2 and thus cannot be instantiated under the k-Linear family of assumptions. In the following series of lemmas, we will actually prove a stronger statement, namely that *any* scheme that requires these three properties (projecting, cancelling, and key indistinguishability) cannot be instantiated under a natural use of the k-Linear assumption for any k.

Lemma 6.1. *If B is a finitely-generated $\mathcal{R}$-module, the order of B divides the order of $\mathcal{R}^\ell$ for some $\ell \geq 1$.*

Proof. Because B is finitely generated, there exists a natural embedding $\psi\colon B \to \mathcal{R}^\ell$, which also means there exists a surjective homomorphism $\phi\colon \mathcal{R}^\ell \to B$. This immediately implies that the order of B divides the order of $\mathcal{R}^\ell$. □

Lemma 6.2. *If the order of G is a prime p, then $B = G^k = G \times G \times \ldots \times G$ for some $k \geq 1$.*

Proof. If the order of G is p, then G can be interpreted as an $\mathbb{F}_p$-module, and so B can also be interpreted as an $\mathbb{F}_p$-module. Similarly, because we assume that the h_i elements are able to generate all of B (in the hiding setting), we know that B is finitely generated. Then the previous lemma tells us that B has order p^k for some value $k \in \mathbb{Z}$; combining this with the structure theorem for finitely-generated modules over principal ideal domains (a generalization of the structure theorem for finite abelian groups), we see that B can be decomposed into components of order p, which implies that $B = G^k = G \times G \times \ldots \times G$. □

[7] It is an open problem to replace the trusted third party with an efficient secure multiparty computation protocol for computing the CRS.

[8] Of course, the number of bits taken to represent the group element is much larger than it would be for a prime-order setting, in which moduli are about 160 bits vs. the 1024 required for composite moduli (at the 80-bit security level).

Lemma 6.3. *If the order of G is a prime p, then for a symmetric pairing $e\colon G \times G \to G_T$ the order of G_T is p as well.*

Proof. If G has order p, we also know that it has exponent p, as the exponent must divide the order of the group. This implies that G_T has exponent p as well; to see this, note that $e(x, y)^p = e(x^p, y) = e(1, y) = 1$ for any $x, y \in G$. Because G_T has exponent p, its order must be a power of p. To determine which power, we first observe that every element of G_T can be written as $e(x, y)$ for $x, y \in G$, which implies that there are p^2 possible elements in G_T, as there are p choices for x and p choices for y. Because the pairing is symmetric, however, $e(x, y) = e(y, x)$ and thus the order of G_T must be at most $p^2/2$; combining this with the fact that it has exponent p, we see that its order must be p. $\square$

We would now like to show that, in the prime-order setting, our indistinguishability restrictions on B and its submodules yield a pairing E that can be either projecting or cancelling, but not both at the same time. Our approach is to construct a cancelling pairing and then show that it implies that B_T contains only one copy of G_T; as we will see, this implies that B_T is too small to satisfy the projecting property.

In general, there are two possible ways that we have observed being used to cancel elements. As seen in Section 5.2.1, the cancelling in the composite setting is fairly straightforward; we essentially just use the respective (and, importantly, relatively prime) orders of the G_p and G_q subgroups, but in a prime-order setting this is not an option, as every component (i.e., G, G_T, B, B_1, B_2, B_T) has exponent p. We therefore need to use certain linear combinations of exponents in order to successfully cancel elements. As our next lemma will show, forming these linear combinations will require us to combine elements in the pairing and thus shrink the size of the target module.

Lemma 6.4. *If our commitment keys are indistinguishable under a natural use of the $(k-1)$-Linear assumption and E is a cancelling pairing, then $|B_T| = p$.*

Because this proof is rather long and technical, it can be found in Appendix D. Putting all this together, we can finally prove our main theorem:

Theorem 6.5. *If our commitment keys are indistinguishable using the $(k-1)$-Linear assumption in a natural way, the pairing $E\colon B \times B \to B_T$ cannot be both projecting and cancelling.*

Proof. By Lemma 6.4, we know that if E is cancelling then $|B_T| = p$. This means that B_T is cyclic, and thus its only submodules are itself and the trivial submodule $\{1\}$. If we look back at our requirements for a projecting pairing in Definition 5.3, we see that we need a proper submodule B_T' such that $B_T' \subseteq \ker(\pi_T)$; this implies that we need $B_T' = \{1\}$. Observe, however, that for any $x_1 \in B_1$ we have $\pi_T(E(x_1, y)) = E(\pi(x_1), \pi(y)) = 1$ for all $y \in B$ (because $B_1 \subseteq \ker(\pi)$ by definition). Therefore, having $B_T' = \{1\}$ would imply that $E(x_1, y) = 1$ for all $x_1 \in B_1$, $y \in B$; as this would imply that our pairing was degenerate, however, it cannot be the case and so E cannot be projecting. $\square$

7 Conclusions and Open Problems

In this paper, we have shown that there are limitations on transformations of pairing-based cryptosystems from composite- to prime-order groups. In particular, we have shown that two properties of composite-order pairings identified by Freeman — cancelling and projecting — cannot be simultaneously obtained in prime-order groups when subgroup hiding is provided by the Decisional Linear assumption in a natural way: when the module B consists of 3 copies of the group G (or, more generally, $k+1$ copies of G for k-Linear).

As evidence that both properties are sometimes called for simultaneously, we have presented a natural cryptographic scheme whose proof of security calls for a pairing that is both cancelling and projecting. This scheme is a practical round-optimal blind (and partially blind) signature secure in the common reference string model, under mild assumptions and without random oracles.

Many open questions remain. First, we would of course like to generalize our result about using projecting and cancelling in prime-order groups so it does not rely on the "natural" use of Decisional Linear, but would instead rely solely on the properties of prime-order groups. Similarly, it would be interesting to see if there are other schemes (or even entire classes of functionality!) that can be achieved in composite-order but not prime-order settings. Finally, in terms of our blind signature scheme, it would be interesting to either find an attack demonstrating that an instantiation under Decisional Linear was in fact *insecure* (as opposed to just not provably secure) or construct a different, ad-hoc proof that would instead prove the scheme secure in some prime-order setting.

Acknowledgements

We are grateful to Melissa Chase and David Mandell Freeman for discussions about this work, and to our anonymous reviewers for their helpful comments. This work was supported by by a MURI grant administered by the Air Force Office of Scientific Research and by a graduate fellowship from the Charles Lee Powell Foundation.

References

[1] M. Abdalla, C. Namprempre, and G. Neven. On the (im)possibility of blind message authentication codes. In D. Pointcheval, editor, *Proceedings of CT-RSA 2006*, volume 3860 of *LNCS*, pages 262–79. Springer-Verlag, Feb. 2006.

[2] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In T. Rabin, editor, *Proceedings of Crypto 2010*, volume 6223 of *LNCS*, pages 209–36. Springer-Verlag, Aug. 2010.

[3] M. Abe and E. Fujisaki. How to date blind signatures. In K. Kim and T. Matsumoto, editors, *Proceedings of Asiacrypt 1996*, volume 1163 of *LNCS*, pages 244–51. Springer-Verlag, Nov. 1996.

[4] M. Abe, K. Haralambiev, and M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010. `http://eprint.iacr.org/`.

[5] M. Abe and M. Ohkubo. A framework for universally composable non-committing blind signatures. In M. Matsui, editor, *Proceedings of Asiacrypt 2009*, volume 5912 of *LNCS*, pages 435–50. Springer-Verlag, Dec. 2009.

[6] M. Abe and T. Okamoto. Provably secure partially blind signatures. In M. Bellare, editor, *Proceedings of Crypto 2000*, volume 1880 of *LNCS*, pages 271–86. Springer-Verlag, Aug. 2000.

[7] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In W. Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 480–494. Springer-Verlag, May 1997.

[8] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In P. Syverson, editor, *Proceedings of Financial Cryptography 2001*, volume 2339 of *LNCS*, pages 319–38. Springer-Verlag, 2002.

[9] A. Boldyreva. Threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *Proceedings of PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer-Verlag, Jan. 2003.

[10] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, Aug. 2004.

[11] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001*.

[12] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *Proceedings of TCC 2005*, number 3378 in LNCS, pages 325–41. Springer-Verlag, Feb. 2005.

[13] D. Boneh, K. Rubin, and A. Silverberg. Finding composite order ordinary elliptic curves using the Cocks-Pinch method. Cryptology ePrint Archive, Report 2009/533, 2009. `http://eprint.iacr.org/2009/533`.

[14] F. Boudot. Efficient proofs that a committed number lies in an interval. In B. Preneel, editor, *Proceedings of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 431–44. Springer-Verlag, May 2000.

[15] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 427–44. Springer-Verlag, May 2006.

[16] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–22. Springer-Verlag, May 2004.

[17] T. Cao, D. Lin, and R. Xue. A randomized RSA-based partially blind signature scheme for electronic cash. *Computers and Security*, 24(1):44–49, Feb. 2005.

[18] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. Rivest, and A. Sherman, editors, *Proceedings of Crypto 1982*, pages 199–204. Plenum Press, 1983.

[19] D. Chaum. Blind signature system (abstract). In D. Chaum, editor, *Proceedings of Crypto 1983*, page 153. Plenum Press, 1984.

[20] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In C. Günther, editor, *Proceedings of Eurocrypt 1988*, volume 330 of *LNCS*, pages 177–82. Springer-Verlag, May 1988.

[21] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Proceedings of Crypto 1988*, volume 403 of *LNCS*, pages 319–27. Springer-Verlag, 1990.

[22] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Proceedings of Crypto 1986*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, Aug. 1986.

[23] M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In C. Dwork, editor, *Proceedings of Crypto 2006*, volume 4117 of *LNCS*, pages 60–77. Springer-Verlag, Aug. 2006.

[24] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–80, Apr. 2010.

[25] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *Proceedings of Eurocrypt 2010*, LNCS, pages 44–61. Springer-Verlag, May 2010.

[26] G. Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009. `http://eprint.iacr.org/`.

[27] D. Galindo, J. Herranz, and E. Kiltz. On the generic construction of identity-based signatures with additional properties. In X. Lai and K. Chen, editors, *Proceedings of Asiacrypt 2006*, volume 4284 of *LNCS*, pages 178–93. Springer-Verlag, Dec. 2006.

[28] S. Garg, A. Sahai, and B. Waters. Efficient fully collusion-resilient traitor tracing scheme. Cryptology ePrint Archive, Report 2009/532, 2009. `http://eprint.iacr.org/2009/532`.

[29] M. Green and S. Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *Proceedings of Asiacrypt 2007*, volume 4833 of *LNCS*, pages 265–282. Springer-Verlag, 2007.

[30] J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In C. Dwork, editor, *Proceedings of Crypto 2006*, volume 4117 of *LNCS*, pages 97–111. Springer-Verlag, Aug. 2006.

[31] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 339–58. Springer-Verlag, May 2006.

[32] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proceedings of Eurocrypt 2008*, volume 4965 of *LNCS*, pages 415–432. Springer-Verlag, 2008.

[33] C. Hazay, J. Katz, C.-Y. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In S. Vadhan, editor, *Proceedings of TCC 2007*, volume 4392 of *LNCS*, pages 323–341. Springer-Verlag, 2007.

[34] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In A. Menezes, editor, *Proceedings of Crypto 2007*, volume 4622 of *LNCS*, pages 553–71. Springer-Verlag, Aug. 2007.

[35] A. Juels, M. Luby, and R. Ostrovsky. Security of blind digital signatures. In B. Kaliski, Jr., editor, *Proceedings of Crypto 1997*, volume 1294 of *LNCS*, pages 150–64. Springer-Verlag, Aug. 1997.

[36] A. Kiayias and H.-S. Zhou. Concurrent blind signatures without random oracles. In M. Yung, editor, *Proceedings of SCN 2006*, volume 4116 of *LNCS*, pages 49–62. Springer-Verlag, Sept. 2006.

[37] H. Lipmaa. On Diophantine complexity and statistical zero-knowledge arguments. In *Proc. Asiacrypt '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415. Springer-Verlag, 2003.

[38] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In R. Hirschfeld, editor, *Proceedings of Financial Cryptography 1998*, volume 1465 of *LNCS*, pages 184–97. Springer-Verlag, Feb. 1998.

[39] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H. Heys and C. Adams, editors, *Proceedings of SAC 1999*, volume 1758 of *LNCS*, pages 184–99. Springer-Verlag, Aug. 1999.

[40] G. Martinet, G. Poupard, and P. Sola. Cryptanalysis of a partially blind signature scheme, or How to make $100 bills with $1 and $2 ones. In G. D. Crescenzo and A. Rubin, editors, *Proceedings of Financial Cryptography 2006*, volume 4107 of *LNCS*, pages 171–76. Springer-Verlag, 2006.

[41] T. Okamoto. Efficient blind and partially blind signatures without random oracles. Cryptology ePrint Archive, Report 2006/102, 2006. `http://eprint.iacr.org/`.

[42] T. Okamoto. Efficient blind and partially blind signatures without random oracles. In S. Halevi and T. Rabin, editors, *Proceedings of TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer-Verlag, Mar. 2006.

[43] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 223–38. Springer-Verlag, May 1999.

[44] K. Paterson and J. Schuldt. Efficient identity-based signatures secure in the standard model. In L. Batten and R. Safavi-Naini, editors, *Proceedings of ACISP 2006*, volume 4058 of *LNCS*, pages 207–22. Springer-Verlag, July 2006.

[45] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–96, 2000.

[46] C. Schnorr. Efficient signature generation for smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[47] H. Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[48] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 114–27. Springer-Verlag, May 2005.

[49] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In Y. Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 533–47. Springer-Verlag, Dec. 2002.

A Formal Security Definition for Blind Signatures

Definition A.1. *A blind signature scheme is considered* concurrently secure *if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ and a security parameter k_0 such that for all $k > k_0$ the following three properties hold:*

1. *Correctness: For all $\sigma_{CRS} \leftarrow \mathsf{Setup}(1^k)$ and $(pk, sk) \leftarrow \mathsf{KeyGen}(\sigma_{CRS})$, if σ is the output of $\mathsf{User}(\sigma_{CRS}, pk, m) \leftrightarrow \mathsf{Signer}(\sigma_{CRS}, sk)$ for an honest user and signer, then $\mathsf{Verify}(\sigma_{CRS}, pk, m, \sigma)$ outputs* accept *with probability* 1*.*

2. *Blindness: Let $b \leftarrow \{0, 1\}$ be unknown to $\mathcal{A}$. Define the following game:*

   - *Step 1. $\sigma_{CRS} \leftarrow \mathsf{Setup}(1^k)$.*
   - *Step 2. $(M_0, M_1, pk) \leftarrow \mathcal{A}(\sigma_{CRS})$.*
   - *Step 3. $\mathcal{A}$ engages in two arbitrarily interleaved signing protocols; one with $\mathsf{User}(\sigma_{CRS}, pk, M_b)$ and one with $\mathsf{User}(\sigma_{CRS}, pk, M_{1-b})$ (where both users act honestly).*
   - *Step 4. If the first user outputs σ_b and the second user outputs σ_{1-b} (i.e., both users succeed) then $\mathcal{A}$ is given σ_0 and σ_1.*
   - *Step 5. In the end, $\mathcal{A}$ outputs a bit b'.*

   The signature scheme is considered blind *if the probability (over the choices of b, the randomness used in* Setup*, and the randomness used by the users and $\mathcal{A}$) that $b' = b$ is at most $1/2 + \nu(k)$.*

3. *One-more unforgeability: Define the following game for our adversary $\mathcal{A}$:*

   - *Step 1. $\sigma_{CRS} \leftarrow \mathsf{Setup}(1^k)$.*
   - *Step 2. $(pk, sk) \leftarrow \mathsf{KeyGen}(\sigma_{CRS})$.*
   - *Step 3. $\mathcal{A}$, on input σ_{CRS} and pk, engages in poly(k) arbitrarily interleaved executions of the signing protocol with polynomially many copies of $\mathsf{Signer}(\sigma_{CRS}, sk)$ (on messages of its choice). Let ℓ denote the number of executions in which the signer outputs* success *at the end.*
   - *Step 4. $\mathcal{A}$ outputs a collection of message-signature pairs $\{(M_i, \sigma_i)\}_{i=1}^m$ such that $M_i \neq M_j$ for all $i \neq j$, and $\mathsf{Verify}(\sigma_{CRS}, pk, M_i, \sigma_i) = \mathsf{success}$ for all $1 \leq i \leq m$.*

   The signature scheme is considered one-more unforgeable *if the probability (again, taken over the randomness used in* Setup, KeyGen*, $\mathcal{A}$, and* Signer*) that $m > \ell$ is at most $\nu(k)$.*

B Proof of (Partial) Blindness

Here we provide a proof of Theorem 5.1 from Section 5.2, which asserts that our signature protocol is partially blind under the assumption that keys are indistinguishable in the hiding and binding settings.

Proof. First, we show correctness of the protocol. This argument is based on the observation (inspired by Groth, Ostrovsky, and Sahai [31]) that if b is equal to 0 or 1, then $E(\tau(u_i)^{b_i}, \tau(u_i)^{b_i - 1}) = 1$.

Using this observation, we see that a correctly formed commitment c_i will pass the test in Equation 2, as we have that

$$
\begin{aligned}
E(c_i, \tau(u_i)^{-1}c_i) &= E\left(\tau(u_i)^{b_i} \cdot \prod_{j=1}^{m} h_j^{t_{ij}}, \tau(u_i)^{b_i-1} \prod_{j=1}^{m} h_j^{t_{ij}}\right) \\
&= E\left(\tau(u_i)^{b_i}, \tau(u_i)^{b_i-1}\right) \cdot E\left(\prod_{j=1}^{m} h_j^{t_{ij}}, \tau(u_i)^{b_i-1} \prod_{j=1}^{m} h_j^{t_{ij}}\right) \cdot E\left(\tau(u_i)^{b_i}, \prod_{j=1}^{m} h_j^{t_{ij}}\right) \\
&= 1 \cdot \prod_{j=1}^{m} E\left(h_j^{t_{ij}}, \tau(u_i)^{-1}c_i\right) \prod_{j=1}^{m} E\left(\tau(u_i)^{b_i}, h_j^{t_{ij}}\right) \\
&= \prod_{j=1}^{m} E\left(h_j, (c_i/\tau(u_i))^{t_{ij}}\right) \cdot E\left(h_j, \tau(u_i)^{b_i t_{ij}}\right) \\
&= \prod_{j=1}^{m} E\left(h_j, c^{t_{ij}} \cdot \tau(u_i)^{-t_{ij}+b_i t_{ij}}\right) \\
&= \prod_{j=1}^{m} E\left(h_j, (\tau(u_i)^{b_i-1} \cdot c_i)^{t_{ij}}\right) \\
&= \prod_{j=1}^{m} E\left(h_j, (\tau(u_i)^{2b_i-1} \cdot \prod_{j} h_j^{t_{ij}})^{t_{ij}}\right) \\
&= \prod_{j=1}^{m} E(h_j, \pi_{ij}),
\end{aligned}
$$

so that the two sides of the equation are equal and the check will pass. In addition, each of the checks in Equation 3 will pass, as

$$E(K_{3j}, \tau(g)) = E(h_j^{-r}, \tau(g)) = E(\tau(g), h_j^{-r}) = E(\tau(g)^{-r}, h_j) = E(K_2, h_j)$$

for $1 \leq j \leq m$. We also know that $\prod_i \prod_j K_{3j}^{t_{ij}} = \prod_i \prod_j (h_j^{-r})^{t_{ij}} = (\prod_i \prod_j h_j^{t_{ij}})^{-r}$, so that $S_1 = K_1 \prod_i \prod_j K_{3j}^{t_{ij}} = \tau(g)^{\alpha}(\tau(u') \prod_i \tau(u_i)^{b_i})^r$. Combining this with the fact that $K_2 = \tau(g)^{-r}$, we see that forming S_1 and S_2 as described in Equation 4 will give us a properly formed signature for our signature scheme. Finally, by the argument at the end of Section 4.2, the re-randomization process will not alter the validity of the signature, so the user really will end up with a valid signature.

Now, we need to argue that if the h_i in the hiding setting are indistinguishable from the h_i in the binding setting, this protocol is partially blind. To start, we run a series of protocol interactions in the *hiding* setting rather than the binding setting; note that our assumption about the keys implies that an adversary $\mathcal{A}$ cannot perform more than negligibly differently in this setting than in the actual protocol (in which the keys are binding). To argue this more explicitly, we see that, if $\mathcal{B}$ represents an adversary trying to distinguish between the keys and we use $\mathsf{Adv}_{\mathcal{B}}$ to denote $\mathcal{B}$'s advantage over a random guess, we have that

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}} &= |\Pr[\mathcal{A} = 1|\mathit{binding}] - \Pr[\mathcal{A} = 1|\mathit{hiding}]| \\
&= \frac{1}{2}\Big|(2\Pr[\mathcal{A} = 1|\mathit{binding}] - 1) - (2\Pr[\mathcal{A} = 1|\mathit{hiding}] - 1)\Big| \\
&= \frac{1}{2}\Big|(\Pr[\mathcal{A} = 1|\mathit{binding}] - \frac{1}{2}) - (\Pr[\mathcal{A} = 1|\mathit{hiding}] - \frac{1}{2})\Big| \\
&= \frac{1}{2}|\mathsf{Adv}_{\mathcal{A},\mathit{binding}} - \mathsf{Adv}_{\mathcal{A},\mathit{hiding}}|,
\end{aligned}$$

where $\mathsf{Adv}_{\mathcal{A},\mathit{binding}}$ denotes $\mathcal{A}$'s advantage in the binding setting and $\mathsf{Adv}_{\mathcal{A},\mathit{hiding}}$ denotes $\mathcal{A}$'s advantage in the hiding setting. By assumption, $\mathcal{B}$'s advantage must be negligible; this implies that $\mathcal{A}$'s advantage in the hiding setting must be negligibly different from its advantage in the binding setting.

To try to find the ways in which $\mathcal{A}$ could attempt to learn information about the user's message, we remind ourselves of the game in Definition A.1: $\mathcal{A}$ picks two messages and two strings $\mathit{info}^{(0)}$ and $\mathit{info}^{(1)}$, as well as a signing keypair; it then engages in one interaction with a user on message M_b and info string $\mathit{info}^{(b)}$ and one interaction with a user on message M_{1-b} and info string $\mathit{info}^{(1-b)}$ (where b is a random bit unknown to $\mathcal{A}$). Finally, if both users output $\mathsf{success}$ and $\mathit{info}^{(0)} = \mathit{info}^{(1)}$, $\mathcal{A}$ gets to see the corresponding unblinded signatures and in the end must output a bit b' which acts as its guess for b. In what follows, we argue that, in the hiding setting, $\mathcal{A}$ cannot do even negligibly better than a random guess in this game. To do this, we discuss three potential sources of information: 1) the protocol interaction, 2) whether or not the users accept, and 3) the output signatures (if $\mathcal{A}$ is given them).

Protocol interaction. Because our blind signature scheme is two-move, the only opportunity $\mathcal{A}$ has to learn any information about the underlying message is in the request req (which we assume to be computed honestly). The first thing we can notice about req is that it does not depend at all on the info string being used, so that any information $\mathcal{A}$ learns must be about the message itself. Unfortunately for $\mathcal{A}$, however, the hiding setting guarantees that each c_i or π_{ij} value will just be a random element of B (because the h_i are chosen to generate all of B) and therefore will contain no information about the message bits b_i.

Whether users accept. It also turns out that $\mathcal{A}$ cannot learn any new information by observing whether or not the users accept the blinded signatures $(K_1, K_2, \vec{K_3})$. Without loss of generality, let's assume $\mathcal{A}$ tries to learn information from the user working with M_b. Since the user is honest, the request tuple is formed properly, so that in particular each commitment is of the form $c_i = \tau(u_i)^{b_i} \prod_j h_j^{t_{ij}}$ for randomness $t_{ij} \leftarrow \mathcal{R}$. This means that the value c formed by $\mathcal{A}$ will be

$$c = \tau(u') \left(\prod_{i=1}^{k_0} \tau(u_i)^{b_i} \right) \left(\prod_{i=k_0+1}^{k} c_i \right) = \left(\tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i} \right) \left(\prod_{i=k_0+1}^{k} \prod_{j=1}^{m} h_j^{t_{ij}} \right).$$

We now observe that $\mathcal{A}$ can use this value to determine for itself whether or not the user will accept the blinded signature formed (though not necessarily formed properly) using c and the unblinded signature (S_1, S_2). To see this, we look at the set of checks the user performs upon receiving the blinded signature. The first set, which is run in Equation 3 for all $1 \leq j \leq m$, can clearly be run by $\mathcal{A}$. We ignore the re-randomization process (as we have already argued that it does not affect the validity of the signature) and move on to the final check in Equation 1. We first note that if the check in Equation 3 passed for all values of j, then we can multiply together the left-hand sides and right-hand sides of each of these equations to see that $\prod_j E(K_{3j}, \tau(g)) = E(K_2, \prod_j h_j)$. Using this fact and rearranging terms on the left-hand side of the equation, we see that

$$
\begin{aligned}
\text{LHS of (1)} &= E(S_1, \tau(g)) \cdot E\left(S_2, \tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\right) \\
&= E\left(K_1 \prod_{i=k_0+1}^{k} \prod_{j=1}^{m} K_{3j}^{t_{ij}}, \tau(g)\right) \cdot E\left(K_2, \tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\right) \\
&= E(K_1, \tau(g)) \cdot E\left(K_2, \tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\right) \cdot \prod_{i=k_0+1}^{k} \prod_{j=1}^{m} E(K_{3j}, \tau(g))^{t_{ij}} \\
&= E(K_1, \tau(g)) \cdot E\left(K_2, \tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\right) \cdot \prod_{i=k_0+1}^{k} E\left(K_2, \prod_{j=1}^{m} h_j^{t_{ij}}\right) \\
&= E(K_1, \tau(g)) \cdot E(K_2, c),
\end{aligned}
$$

which are all values that $\mathcal{A}$ computed itself. Therefore, $\mathcal{A}$ can check Equation 3 and then check its own verification equation $E(K_1, \tau(g)) \cdot F(K_2, c) = A$ (where recall A is the public signing key) to determine on its own whether or not the user will accept, thus learning no information from this stage of the protocol either.

Resulting signatures. Finally, if both users accept and if $info^{(0)} = info^{(1)}$, $\mathcal{A}$ will be given the resulting signatures σ_0 and σ_1 on M_0 and M_1. Because these signatures have been re-randomized by the user, they will both be uniformly distributed signatures (on $info||M_0$ and $info||M_1$ respectively) and will therefore give $\mathcal{A}$ no information about the underlying message.

Thus, we have argued that there is no part of the blind signing protocol (run in the hiding setting) in which $\mathcal{A}$ can learn any information about the messages being used by the honest users (even if $\mathcal{A}$ has adversarially generated the signing keypair; in fact, even if $\mathcal{A}$ is computationally unbounded) and therefore cannot do even negligibly better than a random guess for the bit b'. Combining this with the discussion at the beginning of the proof that $\mathcal{A}$ cannot perform more than negligibly differently in this setting than in the binding setting (i.e., the one used in the actual protocol) means we are done. □

C Proof of One-More Unforgeability

First, we prove a lemma that follows almost directly from Definition 5.2:

Lemma C.1. *If E is cancelling, then $E(q_1h_1, q_2h_2) = E(q_1, q_2) \cdot E(h_1, h_2)$ for $q_1, q_2 \in B_2$ and $h_1, h_2 \in B_1$.*

Proof. This follows immediately from the properties of a cancelling pairing; namely we see that

$$
\begin{aligned}
E(q_1h_1, q_2h_2) &= E(q_1, q_2h_2) \cdot E(h_1, q_2h_2) \\
&= E(q_1, q_2) \cdot E(q_1, h_2) \cdot E(h_1, q_2) \cdot E(h_1, h_2) \\
&= E(q_1, q_2) \cdot 1 \cdot 1 \cdot E(h_1, h_2) \\
&= E(q_1, q_2) \cdot E(h_1, h_2). \qquad \square
\end{aligned}
$$

We now prove the full theorem for one-more unforgeability, as stated in Theorem 5.4 in Section 5.2.

Proof. To show this, we will take an adversary $\mathcal{A}$ that breaks the one-more unforgeability property on B and use it construct an adversary $\mathcal{B}$ that breaks the existential unforgeability of our modified Waters signature on B_2. Our approach is to essentially use two maps $\phi : G \to B_2$ and $\psi : G \to B_1$; these maps serve to split up B into its separate components and allow $\mathcal{B}$ to manipulate values in one submodule while leaving unchanged the values in the other submodule.

To start, our adversary $\mathcal{B}$ will receive as input a CRS computed as described in Section 4.1; i.e., one that specifies the group G, the module B, the ring $\mathcal{R}$ such that G and B can be interpreted as $\mathcal{R}$-modules, as well as all the other maps and generators, and in particular a map τ' such that $\tau' : G \to B_2$. Note that some trapdoor will be required to compute τ', as it reveals the submodule B_1. The CRS also specifies elements $g, u', u_1, \ldots, u_k$ which are all generators for G. We now describe the behavior of $\mathcal{B}$ in terms of the following steps:

Setup: $\mathcal{B}$ will set the map $\phi = \tau'$ and construct a map $\psi : G \to B_1$ such that $\tau = \phi \cdot \psi$ for the τ specified by the protocol. It will then give to $\mathcal{A}$ the same groups and modules it received, with the exception that it will exchange its input map τ' for τ, so that τ now maps to the full module B. It will then use its trapdoor to construct elements $h_1, \ldots, h_m$ that generate B_1, and publish these elements as well.

KeyGen: $\mathcal{B}$ was also given a public key $A' = E(\tau'(g), \tau'(g))^\alpha = E(\phi(g), \phi(g))^\alpha$ for some unknown $\alpha \in \mathcal{R}$. To output its own set of keys, $\mathcal{B}$ will pick $\beta \leftarrow \mathcal{R}$ and compute $A = A' \cdot E(\psi(g), \psi(g))^\beta$. This value will be output to the forger $\mathcal{A}$.

Signing: For each of the executions of the blind signing protocol, $\mathcal{A}$ will start by giving $\mathcal{B}$ the *req* tuple. For each c_i in *req*, $\mathcal{B}$ will compute $\rho(c_i) = \rho \circ \tau(u_i)^{b_i}$ and thus recover the message $M = b_{k_0+1} \ldots b_k$.[9] $\mathcal{B}$ will also perform the check in Equation 2 for each pair (c_i, π_i) and abort and output $\perp$ if any of these pairs fails to pass the check. Otherwise, $\mathcal{B}$ will then query its own signing oracle on M and receive a signature of the form (S_1, S_2), where

$$S_1 = \phi(g)^\alpha \left(\phi(u') \prod_{i=1}^{k} \phi(u_i)^{b_i} \right)^r \quad \text{and} \quad S_2 = \phi(g)^{-r}$$

for a random $r \leftarrow \mathcal{R}$. To transform this to a blind signature in the full module B, $\mathcal{B}$ will choose a random $s \leftarrow \mathcal{R}$ and compute

$$K_1 = S_1 \cdot \psi(g)^\beta \left(\psi(u') \prod_{i=1}^{k} \psi(u_i)^{b_i} \right)^s \cdot \left(\prod_{i=k_0+1}^{k} \frac{c_i}{\tau(u_i)^{b_i}} \right)^s, \tag{5}$$

$$K_2 = S_2 \cdot \psi(g)^{-s}, \quad \text{and} \tag{6}$$

$$K_{3j} = h_j^{-s} \tag{7}$$

for $1 \leq j \leq m$. $\mathcal{B}$ will then send the tuple $(K_1, K_2, \vec{K}_3)$ back to $\mathcal{A}$ and output *info* and $\mathsf{success}$.

[9]Again we remember that a trapdoor may be required to make the ρ map efficiently computable.

Output: Finally, $\mathcal{A}$ will output a tuple of the form $((M_1, \sigma_1), \ldots, (M_\ell, \sigma_\ell), (M_{\ell+1}, \sigma_{\ell+1}))$ such that $\mathsf{Verify}(\sigma_{CRS}, pk, M_i, \sigma_i) = \mathsf{accept}$ for all i, but $\mathcal{B}$ output $\mathsf{success}$ for only ℓ iterations of the signing protocol. By the pigeonhole principle, then, there must be at least one message M^* such that $\mathcal{A}$ did not obtain a signature from $\mathcal{B}$ on message M^*. In particular, since $\mathcal{A}$ did not get a signature from $\mathcal{B}$ on message M^*, we know that $\mathcal{B}$ also did not get a signature from its own signing oracle on M^*. This means that $\mathcal{B}$ can use the message M^* and its corresponding signature (S_1^*, S_2^*) to output its own forgery. To convert this signature in B into a signature in B_2, $\mathcal{B}$ uses the projecting maps π and π_T (from Definition 5.3) to compute $S_1 = \pi(S_1^*)$ and $S_2 = \pi(S_2^*)$; because B_1 is in the kernel of π, this will map the signature to its B_2 component. Finally, $\mathcal{B}$ will output the pair $(M^*, \sigma^* = (S_1, S_2))$.

Analysis: Now we need to analyze the behavior of $\mathcal{B}$ and argue that it is indistinguishable from the behavior of an honest signer; in addition, we need to argue that the output pair really is a valid forgery for the signature scheme on the submodule B_2. We'll start with the former, and work step by step.

In the setup phase, the commitment keys $h_1, \ldots, h_m$ are computed honestly. In fact, the only difference in what $\mathcal{B}$ gives $\mathcal{A}$ is that it constructs the map τ to hide the submodule B_1. Because τ was constructed to match exactly the τ expected by $\mathcal{A}$, however, this will also look indistinguishable to $\mathcal{A}$, as it will in fact be identical to the output of the honest CRS algorithm.

In the key generation phase, we argue that the key A is a random element of B_T and therefore will be distributed identically to a properly formed public key. To show this, we remember that an honestly formed key A will be of the form $E(\tau(g), \tau(g))^c$ for some random $c \leftarrow \mathcal{R}$. The key formed by $\mathcal{B}$, however, looks like $E(\phi(g), \phi(g))^a \cdot E(\psi(g), \psi(g))^b$, again for random $a, b \leftarrow \mathcal{R}$. Because we are using SGH, we know that τ maps to the full module B and so $E(\tau(g), \tau(g))^c$ will represent a random element of the full target module B_T. Similarly, the only two submodules of B_T are the module generated by pairing elements in B_1 and the modular generated by pairing elements in B_2; because $\phi(g)$ and $\psi(g)$ generate B_1 and B_2 respectively, $\mathcal{B}$ is effectively just multiplying together random elements of each of these submodules to generate a random element of the full target module B_T, meaning the two distributions are in fact identical.

We now come to the signing interactions with $\mathcal{A}$. Although the blind signature that $\mathcal{B}$ sends to $\mathcal{A}$ is not computed according to the Signer algorithm specifications, the SGH assumption again guarantees that the values will be distributed identically to their honest counterparts. A bit more formally, we recall that in the honest game, the elements K_1 and K_2 are both elements of the form $\tau(g)^t$, where t is some random value. Here, however, K_1 and K_2 are both elements of the form $\phi(g)^{t_1}\psi(g)^{t_2}$ for random values $t_1, t_2 \in \mathbb{R}$. Because $\tau(G) = B$ while $\psi(G) = B_1$ and $\phi(G) = B_2$ (in other words, all three maps are surjective), in both these cases K_1 and K_2 will just be random elements in B and so the distributions are again identical. In addition, we can argue that the values sent will also pass the two checks performed by the user.

We start by examining the checks performed by $\mathcal{A}$ in Equations 3 and 1. In the first of these checks, we look back at Equation 6 to remind ourselves that $K_2 = \phi(g)^{-r} \cdot \psi(g)^{-s}$ for $r, s \leftarrow \mathcal{R}$. Using this decomposition, we see that

$$E(K_{3j}, \tau(g)) = E(h_j^{-s}, \phi(g)\psi(g)) = E(\phi(g), h_j^{-s}) \cdot E(\psi(g), h_j)^{-s} = 1 \cdot E(\psi(g)^{-s}, h_j),$$

where this last equality follows from the cancelling property of E and the fact that $\phi(g) \in B_2$. Similarly, we find that

$$E(K_2, h_j) = E(\phi(g)^{-r}\psi(g)^{-s}, h_j) = E(\phi(g), h_j)^{-r} \cdot E(\psi(g)^{-s}, h_j) = 1 \cdot E(\psi(g)^{-s}, h_j)$$

so that the two sides of Equation 3 are equal for all $1 \leq j \leq m$ and this first set of checks will pass. For the last check, we first go back to Equation 5 to see how K_1 is computed. Because $\mathcal{B}$ did not abort in the signing phase, the zero-knowledge property of the proofs (as well as the derivation in Appendix B) tell us that the commitments must be correctly formed, meaning they are formed as $c_i = \tau(u_i)^{b_i} \prod_j h_j^{t_{ij}}$; computing the product $\prod_i c_i/\tau(u_i)^{b_i}$ will in fact give us the desired product $\prod_i \prod_j h_j^{t_{ij}}$. This means that, writing $U_\psi = \psi(u') \prod_{i=1}^{k} \psi(u_i)^{b_i}$, we have

$$K_1 \prod_{i=k_0+1}^{k} \prod_{j=1}^{m} K_{3j}^{t_{ij}} = S_1 \cdot \psi(g)^\beta \cdot U_\psi^s.$$

We now write out the left-hand side of Equation 1 using $U_\tau = \tau(u') \prod_i \tau(u_i)^{b_i}$ and $U_\phi = \phi(u') \prod_i \phi(u_i)^{b_i}$ to see that

$$\begin{aligned}
\text{LHS of (1)} &= E\left(K_1 \prod_{i=k_0+1}^{k} \prod_{j=1}^{m} K_{3j}^{t_{ij}}, \tau(g)\right) \cdot E(K_2, U_\tau) \\
&= E\left(\phi(g)^\alpha \psi(g)^\beta U_\phi^r U_\psi^s, \tau(g)\right) \cdot E\left(\phi(g)^{-r} \psi(g)^{-s}, U_\tau\right) \\
&= E\left(\phi(g)^\alpha \psi(g)^\beta \cdot U_\phi^r U_\psi^s, \phi(g)\psi(g)\right) \cdot E(\phi(g)^{-r} \psi(g)^{-s}, U_\phi U_\psi) \\
&= E\left(\phi(g)^\alpha, \phi(g)\right) \cdot E\left(\psi(g)^\beta, \psi(g)\right) \cdot E\left(U_\phi^r U_\psi^s, \phi(g)\psi(g)\right) \cdot E\left(\phi(g)^{-r}, U_\phi\right) \cdot E\left(\psi(g)^{-s}, U_\psi\right) \\
&= A \cdot E\left(U_\phi^r, \phi(g)\right) \cdot E\left(U_\psi^s, \psi(g)\right) \cdot E\left(U_\phi, \phi(g)^{-r}\right) \cdot E\left(U_\psi, \psi(g)^{-s}\right) \\
&= A \cdot E\left(U_\phi^r \cdot U_\phi^{-r}, \phi(g)\right) \cdot E\left(U_\psi^s \cdot U_\psi^{-s}, \psi(g)\right) \\
&= A,
\end{aligned}$$

so that Equation 1 will in fact verify using the values $\mathcal{B}$ formed and $\mathcal{A}$ will output $\mathsf{success}$ (note that the derivation makes use of Lemma C.1, specifically between lines 3 and 4 and lines 4 and 5).

Finally, we can turn to the output of $\mathcal{B}$. Because $\mathcal{A}$'s forgery is valid, we know that

$$E(S_1^*, \tau(g)) \cdot E(S_2^*, U_\tau) = A. \tag{8}$$

Furthermore, because $B = B_1 \times B_2$, where B_1 is in the kernel of the projecting map π from Definition 5.3, computing $\pi(S_1^*)$ and $\pi(S_2^*)$ will cancel out the B_1 component of S_1^* and S_2^* and leave us with values in B_2. Similarly, computing $\pi_T(A)$ yields

$$\begin{aligned}
\pi_T(A) &= \pi_T(E(\phi(g), \phi(g))^\alpha E(\psi(g), \psi(g))^\beta) \\
&= \pi_T(E(\phi(g), \phi(g))^\alpha) \cdot \pi_T(E(\psi(g), \psi(g))^\beta) \\
&= E(\pi \circ \phi(g), \pi \circ \phi(g))^\alpha \cdot E(\pi \circ \psi(g), \pi \circ \psi(g))^\beta \\
&= E(\phi(g), \phi(g))^\alpha \cdot E(1, 1)^\beta \\
&= A',
\end{aligned}$$

since by definition π cancels elements in B_1 and leaves elements in B_2 alone (and $\phi(g) \in B_2$ and $\psi(g) \in B_1$, again just by definition).

Finally, we use the projecting map π_T applied to the left-hand side of Equation 8 to see that

$$\begin{aligned}
\pi_T(E(S_1^*, \tau(g)) \cdot E(S_2^*, U_\tau)) &= E(\pi(S_1^*), \pi(\tau(g))) \cdot E(\pi(S_2^*), \pi(U_\tau)) \\
&= E(\pi(S_1^*), \phi(g)) \cdot E(\pi(S_2^*), U_\phi),
\end{aligned}$$

where we use projecting, cancelling, and Lemma C.1 to derive this last line of our equation. If we now recall that $\mathcal{B}$'s original input map τ' is in fact identical to what we are calling ϕ, we can see that we have values S_1 and S_2, as well as a value U corresponding to a message M, such that $E(S_1, \tau'(g)) \cdot E(S_2, U) = A'$, and so we are done, as the output will pass the verification check and is therefore a valid forgery. $\square$

D Proof of Lemma 6.4

In this section, we prove Lemma 6.4 from Section 6, which states that for a cancelling pairing instantiated using $(k-1)$-Linear we must have $|B_T| = p$.

Proof. The $(k-1)$-Linear assumption states that tuples of the form $(g^{\alpha_1}, g^{\alpha_2}, \ldots, g^{\alpha_1 r_1}, g^{\alpha_2 r_2}, \ldots, g^{\sum_i r_i})$ are indistinguishable from ones of the form $(g^{\alpha_1}, g^{\alpha_2}, \ldots, g^{\alpha_1 r_1}, g^{\alpha_2 r_2}, \ldots, g^{\alpha_k})$ for $\alpha_i, r_i \leftarrow \mathbb{F}_p$. Therefore, a natural choice for B (and the one used by Groth and Sahai [32] for the $k = 3$ case) is all k-tuples, with commitment keys $h_1 = (g^{\alpha_1}, 1, \ldots, 1, g)$, $h_i = (1, 1, \ldots, g^{\alpha_i}, 1, \ldots, 1, g)$, and $h_k = (g^{\alpha_1 s_1}, g^{\alpha_2 s_2}, \ldots, g^{\alpha_j s_j}, \ldots, g^{\sum_i s_i})$ for some $s_1, \ldots, s_{k-1} \leftarrow \mathbb{F}_p$ (in the binding case, and in the hiding case h_k is chosen to be linearly independent from all the previous h_i elements). If these h_i generate B_1, then elements of B_1 are of the form $(g^{\alpha_1 r_1}, \ldots, g^{\sum_i r_i})$, where the values $r_1, \ldots, r_{k-1}$ are allowed to range over all of $\mathbb{F}_p$ and thus B_1 has order p^{k-1}. As $B = B_1 \times B_2$ and B has order p^k, this implies that B_2 has order p and so we can write elements in B_2 as $(g^{\beta_1 t}, g^{\beta_2 t}, \ldots, g^{\beta_k t})$ for some fixed $\beta_1, \ldots, \beta_k \in \mathbb{F}_p$ (and t allowed to range over all $\mathbb{F}_p$ values).

To start, we write elements in B as either $a = (a_1, \ldots, a_k)$ or $b = (b_1, \ldots, b_k)$. We will generally use $a \in B_1$ and $b \in B_2$, which means we can write them in their $k-1$-Linear forms; namely as $a = (g^{\alpha_1 r_1}, \ldots, g^{\alpha_{k-1} r_{k-1}}, g^{\sum_i r_i})$ for some $r_1, \ldots, r_{k-1} \in \mathbb{F}_p$ and $b = (g^{\beta_1 t}, \ldots, g^{\beta_k t})$ for some $t \in \mathbb{F}_p$. We can furthermore observe that the α_i values are hidden, and that none of them can be equal to 0, as this would give us an easy way to distinguish B_1 from B; more specifically, if $\alpha_i = 0$, then given a random element in either B_1 or B, we can check if the i-th value in the tuple is 1; if it is, output B_1 and otherwise output B. By similar logic, no α_i can be related to another α_j in some known way, as this would again give us a way to distinguish between elements of B_1 and elements of B.

In general, elements of B_T will be tuples, where each entry is of the form $T = e(a_i, b_j)^{e_{ij}} \cdot \ldots \cdot e(a_\ell, b_m)^{e_{\ell m}}$, so that any a_i value can be paired with any (and possibly many) b_j values, using any coefficient e_{ij}. This is not quite true, however, as the e_{ij} values cannot depend on the α_i values (as they are assumed to be hidden) and furthermore cannot depend on the β_j values. To see this last part, suppose that the β_j values were efficiently computable (as they would be if they were related in some known way to the given e_{ij} values). Then given an element x in either B_1 or B, we could compute an element in B_2 using the β_j values and pair it with x; if the resulting value is 1 then we can conclude $x \in B_1$ and otherwise that $x \in B$.

Now, we suppose that $a \in B_1$ and $b \in B_2$ and see what we require in order to have $T = 1$. In full generality (and cancelling the t values, as the result needs to hold for all t and so in particular for $t \neq 0$), our requirement becomes having

$$\sum_i r_i \left(\sum_j e_{ij} \alpha_i \beta_j + \sum_j e_{kj} \beta_j \right) = 0,$$

where the terms in the first inner sum correspond to the cases in which we pair a_i with b_j for some j and $i \neq k$, and the terms in the second inner sum correspond to the cases in which we pair a_k with

b_j for some j. We start by rewriting this equation as $\sum_i r_i(\alpha_i(\sum_j e_{ij}\beta_j) + \sum_j e_{kj}\beta_j) = 0$. Now, suppose some a_i term does not appear anywhere in the pairing, in other words that there exists an ℓ such that $e_{\ell j} = 0$ for all j. Then the term for ℓ becomes $r_\ell(\sum_j e_{kj}\beta_j) = 0$, which implies that $\sum_j e_{kj}\beta_j = 0$. Because this term exists for all i, however, we end up with the requirement that

$$\alpha_i(\sum_j e_{ij}\beta_j) + \sum_j e_{kj}\beta_j = \alpha_i(\sum_j e_{ij}\beta_j) = 0,$$

and so we require $\sum_j e_{ij}\beta_j = 0$ for all values of i. If this were true, however, then consider pairing an arbitrary element $c = (g^{\gamma_1}, \ldots, g^{\gamma_k})$ with b. Then we have $T = e(g,g)^{\sum_i \gamma_i(\sum_j e_{ij}\beta_j)} = e(g,g)^{\sum_i \gamma_i(0)} = 1$, which implies that this tuple element will be 1 when b is paired with *any* value in B, and not just values in B_1. Therefore, if the B_T tuple consisted only of elements of this form, our pairing would be degenerate and so we conclude that this type of element cannot be the only one appearing in the B_T tuple.

Next, suppose that we do have each a_i term appear in the product; this means that each r_i term does in fact appear in the sum. We can again group around each α_i to see that we require

$$\alpha_i \sum_j e_{ij}\beta_j + \sum_j e_{kj}\beta_j = 0. \tag{9}$$

If we had $\sum_j e_{kj}\beta_j = 0$, then the only way for this equation to be satisfied would be to have $\sum_j e_{ij}\beta_j = 0$, which we also saw as a possibility earlier and will discuss later on. Assuming this doesn't happen, we end up with $k-1$ linear equations (one for each i) over k variables (the β_j); we would now like to argue that these equations are in fact linearly independent. To see this, just note that the i-th equation is the only equation containing the α_i value, and that furthermore it contains no α_ℓ value for $\ell \neq i$. Therefore, there is no way to write the other equations in terms of the i-th equation, as doing so would require us to introduce an α_i term, thus introducing an α_i term into the $\sum_j e_{kj}\beta_j$ term and violating the specified form of the a_k term (namely, that $a_k = g^{\sum_i r_i}$ and so no α_i terms appear in the exponent). As this is true for all i, the equations must be linearly independent.

Now that we know our $k-1$ equations over the β_j are all linearly independent, we can conclude that the solution space (i.e., the space of β_j values) must be at most one-dimensional. Because B_2 has dimension 1, however, we know that the space of β_j values is exactly 1, which means that there can be at most $k-1$ equations over the β_j variables before the system becomes overdetermined.

We must now consider the case when we have another element in the B_T tuple, call it T'. We can define the set $\{eq_i\}_{i=1}^{k-1}$ to be the set of constraints imposed by T (of the form in Equation 9), and see that T' comes with its own set of constraints $\{eq'_i\}_{i=1}^{k-1}$; i.e., the requirement that $\alpha_i \sum_j e'_{ij}\beta_j + \sum_j e'_{kj}\beta_j = 0$ for all i (and for some different coefficients e'_{ij}, e'_{kj} from the ones used to compute T). By the same argument as before, we conclude that these equations must all be linearly independent. Because we already have $k-1$ linearly independent equations $\{eq_i\}$ over the β_j, however, we know we cannot add any more without overconstraining the variables, and so we know that each equation eq'_i in the T' set must be linearly dependent on the $\{eq_i\}$ ones from T. Therefore, we look at the i-th equation, $\alpha_i \sum_j e'_{ij}\beta_j + \sum_j e'_{kj}\beta_j = 0$, and consider how to write it in terms of the equations $\{eq_i\}$. As before, however, we know we cannot introduce any new α_i variables when constructing our linear dependence, and so the only choice for this equation eq'_i is for it to depend on the i-th equation eq_i from the T set, as it is the only one that also already contains an α_i term. So, we can write $eq'_i = c_i eq_i$ for some constant c_i; as this was true for an arbitrary i, we can repeat it for all i to end up with a series of dependencies of the form $eq'_1 = c_1 eq_1, \ldots, eq'_{k-1} = c_{k-1} eq_{k-1}$. Although at first glance the c_i terms might all be distinct, we observe that each equation eq'_i contains the term

$c_i \sum_j e_{kj}\beta_j$, and that this value does in fact need to be equal across all equations, so that we do end up with $c = c_1 = \ldots = c_{k-1}$. This further implies that $T' = T^c$, meaning that any additional terms will be dependent on T and so, although we can add in more elements to the tuple, B_T will still contain only one copy of G_T.

We have one final step left in our proof, namely showing that the $\sum_j e_{ij}\beta_j = 0$ case can never come up. As mentioned, this case can occur only if the tuple also contains some other type of element, as otherwise the pairing would be degenerate. By what we have just shown, however, the only other type of tuple element involves constraining the β_j variables using the maximum number of equations, and so it is not possible to add the extra constraint that $\sum_j e_{ij}\beta_j = 0$. Therefore, we must conclude that these two types of elements cannot occur at the same time; as the first type could only occur if the second did as well, however, we conclude that only the second type can exist. Finally, we have argued that if we use this type then B_T can contain only one copy of G_T, which using Lemma 6.3 means that $|B_T| = p$ and so we are done. □

E Blind Identity-Based Encryption

In this section, we briefly outline our blind IBE scheme based on our blind signature. The notion of a blind IBE scheme was introduced by Green and Hohenberger [29]; here we use their definitions for the scheme and its security properties.

A blind IBE scheme consists of four algorithms: the $\mathsf{Setup}(1^k)$ algorithm which is run by the master authority to output *params* and the master secret key *msk*, an interactive protocol BlindExtract run between a user with identity *id* and the master authority in which the user obtains a secret key sk_{id} for his identity *id*, an $\mathsf{Encrypt}(params, id, m)$ algorithm in which a user computes a ciphertext c, and finally a $\mathsf{Decrypt}(params, id, sk_{id}, c)$ algorithm which uses sk_{id} to decrypt the ciphertext c and output m.

There are three security properties that a blind IBE scheme must satisfy. The first, adaptive-identity security, requires us to show that the underlying IBE scheme is IND-ID-CPA secure [11], a strengthening of IND-sID-CPA security [16] that allows the adversary to adaptively pick the identities. The second, leak-free extraction, is related to the one-more unforgeability property of blind signatures in that it requires that a malicious user cannot learn anything more from BlindExtract than it could learn from an unblinded extraction protocol.[10] Finally, the third property, selective-failure blindness, is related to the blindness property of our signature scheme in that a malicious authority cannot learn anything about the user's identity during the BlindExtract protocol; in particular, it cannot choose to fail based on the user's choice of identity.

Because our signature scheme is a generalization of the Waters signature scheme, our blind IBE will also be a straightforward generalization of the Waters IBE. This means that the Encrypt and Decrypt algorithms should look very familiar, as they are generalizations of the same algorithms from Waters. Furthermore, we remind ourselves that the Waters IBE requires the Decisional Bilinear Diffie Hellman (DBDH) assumption for security, and so the security of our blind IBE will be based on the same assumption (in addition to whatever assumption we use for the blindness property).

- $\mathsf{Setup}(1^k)$: Here, we output the CRS from our Setup algorithm for the blind signature scheme in Section 5, as well as the keypair $(pk = A, sk = \tau(g)^\alpha)$ (where we remind ourselves that $A = F(\tau(g), \tau(g))^\alpha)$ from the Keygen algorithm. The authority will use $msk = \tau(g)^\alpha$, and the identity space will be $\mathcal{I} = \{0,1\}^k$.

[10] As Green and Hohenberger note, leak-free extraction is stronger in that it implies one-more unforgeability.

- $\mathsf{BlindExtract}$: In this protocol, we will run $\mathsf{User}(\sigma_{CRS}, pk, v) \leftrightarrow \mathsf{Signer}(\sigma_{CRS}, sk)$ from the blind signature scheme, where v represents the user's identity and the output signature (S_1, S_2) will be interpreted as the secret key sk_v for this identity.

- $\mathsf{Encrypt}(pk, M, v)$: We first write $pk = A$ and $v = b_1 \dots b_k$. Then the ciphertext C will be

$$C = \left(A^t M, \tau(g)^{-t}, \left(\tau(u') \prod_{i=1}^{k} \tau(u_i)^{b_i}\right)^{-t}\right)$$

  for some random value $t \leftarrow \mathcal{R}$.

- $\mathsf{Decrypt}(sk_v, C)$: Here we write $sk_v = (S_1, S_2)$ and $C = (C_1, C_2, C_3)$. Then we compute

$$M = C_1 \cdot E(S_2, C_3) \cdot E(S_1, C_2).$$

Theorem E.1. *Under the DBDH and SGH assumptions, the above protocol is a concurrently secure blind IBE system that satisfies leak-free extraction and selective-failure blindness.*

Proof. (Sketch) First, we need to argue correctness of the $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ protocols (as the correctness of the $\mathsf{BlindExtract}$ phase has already been argued in Theorem 5.1). To show that $\mathsf{Decrypt}$ completely recovers the message M, we write $U = \tau(u') \prod_i \tau(u_i)^{b_i}$ and see that

$$\begin{aligned}
C_1 \cdot E(S_2, C_3) \cdot E(S_1, C_2) &= (E(\tau(g), \tau(g))^{\alpha})^t \cdot M \cdot E(\tau(g)^{-r}, U^{-t}) \cdot E(\tau(g)^{\alpha} U^r, \tau(g)^{-t}) \\
&= M \cdot E(\tau(g), \tau(g))^{\alpha t} \cdot E(\tau(g), U)^{rt} \cdot E(\tau(g)^{\alpha}, \tau(g)^{-t}) \cdot E(U, \tau(g))^{-rt} \\
&= M \cdot E(\tau(g), \tau(g))^{\alpha t} \cdot E(\tau(g), \tau(g))^{-\alpha t} \\
&= M.
\end{aligned}$$

The $\mathsf{IND\text{-}ID\text{-}CPA}$ security of the scheme under the DBDH assumption has already been argued in Waters' original security proof, and so we won't reproduce it here and instead move on to leak-free extraction and selective-failure blindness. Because of the similarities between the properties, our proof of leak-free extraction will use the same techniques as our proof of Theorem 5.4 and our proof of selective-failure blindness will use the same techniques as our proof of Theorem 5.1. In Theorem 5.4, we have already constructed the ideal adversary $\mathcal{S}$ needed for leak-free extraction: it is just the adversary $\mathcal{B}$. In the analysis of $\mathcal{B}$, we argue that its behavior is indistinguishable from that of an honest signer (we do this to make sure that $\mathcal{A}$ cannot distinguish between the two and intentionally fail when it knows it is talking to $\mathcal{B}$) and so if we define the behavior of $\mathcal{S}$ to be identical to the behavior of $\mathcal{B}$ this implies that no efficient algorithm D can distinguish between $\mathcal{A}$ interacting with an honest signer in the blind signature scheme and $\mathcal{A}$ interacting with an ideal simulator $\mathcal{S}$ that has access to a signer for the underlying signature scheme.

For selective-failure blindness, we have also done all the work in our proof of Theorem 5.1. In fact, if we look at the definition of selective-failure blindness given by Green and Hohenberger we can see that it is identical to our strengthened blindness property in Definition A.1 and so our proof of blindness in Theorem 5.1 immediately implies the proof of selective-failure blindness here. □

Although our scheme might, on the surface, seem similar to the original one proposed by Green and Hohenberger, we highlight here some advantages of our scheme. In the Green-Hohenberger blind IBE, they use general discrete-log-based techniques for zero-knowledge proofs. In particular, they require a protocol to prove knowledge of a discrete logarithm [46] and a protocol to prove that a committed value lies in a public interval [14, 37]. These protocols are typically interactive, so Green and Hohenberger either require the Fiat-Shamir heuristic [22] to make them non-interactive (and thus secure only in the random oracle model) or require a round-complexity for the $\mathsf{BlindExtract}$ protocol that is greater than two, which also implies that their scheme is not concurrently secure.