Dual Form Signatures: An Approach for Proving Security from Static Assumptions

Michael Gerbush
University of Texas at Austin
mgerbush@cs.utexas.edu

Allison Lewko *
University of Texas at Austin
alewko@cs.utexas.edu

Adam O'Neill
Boston University
amoneill@bu.edu

Brent Waters †
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

In this paper, we introduce the abstraction of Dual Form Signatures as a useful framework for proving security (existential unforgeability) from static assumptions for schemes with special structure that are used as a basis of other cryptographic protocols and applications. We demonstrate the power of this framework by proving security under static assumptions for close variants of pre-existing schemes:

- the LRSW-based Camenisch-Lysyanskaya signature scheme
- the identity-based sequential aggregate signatures of Boldyreva, Gentry, O'Neill, and Yum.

The Camenisch-Lysyanskaya signature scheme was previously proven only under the interactive LRSW assumption, and our result can be viewed as a static replacement for the LRSW assumption. The scheme of Boldyreva, Gentry, O'Neill, and Yum was also previously proven only under an interactive assumption that was shown to hold in the generic group model. The structure of the public key signature scheme underlying the BGOY aggregate signatures is quite distinctive, and our work presents the first security analysis of this kind of structure under static assumptions. We view our work as enhancing our understanding of the security of these signatures, and also as an important step towards obtaining proofs under the weakest possible assumptions.

Finally, we believe our work also provides a new path for proving security of signatures with embedded structure. Examples of these include: attribute-based signatures, quoteable signatures, and signing group elements.

1 Introduction

Digital signatures are a fundamental technique for verifying the authenticity of a digital message. The significance of digital signatures in cryptography is also amplified by their use as building blocks for more complex cryptographic protocols. Recently, we have seen several pairing based signature schemes (e.g., [22, 18, 29, 71]) that are both practical and have added structure which has been used to build other primitives ranging from Aggregate Signatures [20, 58] to Oblivious Transfer [30, 44]. Ideally, for such a fundamental cryptographic primitive we would like to have security proofs from straightforward, static complexity assumptions.

*Supported by a Microsoft Research Ph.d. Fellowship.

†Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

Meeting this goal for certain systems is often challenging. For instance, the Camenisch and Lysyanskaya signature scheme [29][1] has been very influential as it is used as the foundation for a wide variety of advanced cryptographic systems, including anonymous credentials [29, 9, 8], group signatures [29, 7], ecash [27], uncloneable functions [26], batch verification [28], and RFID encryption [6]. While the demonstrated utility of CL signatures has made them desirable, it has been difficult to reduce their security to a static security assumption. Currently, the CL signature scheme is proven secure under the LRSW assumption [60], an interactive complexity assumption that closely mirrors the description of the signature scheme itself. In addition, the interactive assumption transfers to the systems built around these signatures.

The identity-based sequential aggregate signatures of Boldyreva, Gentry, O'Neill, and Yum [14, 15] were also proven in the random oracle model under a interactive assumption (justified in the generic bilinear group model), which again closely mirrors the underlying signature scheme itself. (This can be viewed as providing a proof of the scheme only in the generic group model.) Proofs of complicated interactive assumptions in the generic group model have several disadvantages. First, they are themselves complex and prone to error. In fact, the original version of the BGOY identity-based sequential aggregate signature scheme [14] relied on an assumption that was shown to be false, and the scheme was insecure [50]. This scheme and proof were corrected in [15]. Secondly, such proofs do not tend to provide much insight into the security of the scheme. This lack of insight tends to hinder transferring schemes to other settings. For example, many schemes developed in bilinear groups now have lattice-based analogs, and these transformations reused high-level ideas from the original security proofs in the bilinear group setting. Techniques from [71] were used in the lattice setting in [25], techniques from [31] were used in [32], and techniques from [17] were used in [2]. This kind of transference of ideas from the bilinear setting to the lattice setting is unlikely to be achieved through generic group proofs.

In this work, we develop techniques that can be applied to prove security from static assumptions for new signature schemes as well as (slight variants of) pre-existing schemes. Providing new proofs for these existing schemes provides a meaningful sanity check as well as new insight into their security. This kind of sanity check is valuable not only for schemes proven in the generic group model, but also for signatures (CL signatures included) that require extra checks to rule out trivial breaks (e.g. not allowing the message signed to be equal to 0), since these subtleties can easily be missed at first glance. Having new proofs from static assumptions for variants of schemes like CL signatures and BGOY signatures gives us additional confidence in their security without having to sacrifice the variety of applications built from them. Ultimately, this provides us with a fuller understanding of these kinds of signatures, and is a critical step towards obtaining proofs under the simplest and weakest assumptions.

Dual Form Signatures Our work is centered around a new abstraction that we call Dual Form Signatures. Dual Form Signatures have similar structure to existing signature schemes, however they have two signing algorithms, Sign_A and Sign_B, that respectively define two forms of signatures that will both verify under the same public key. In addition, the security definition will categorize forgeries into two *disjoint* types, Type I and Type II. Typically, these forgery types will roughly correspond with signatures of form A and B.

In a Dual Form system, we will demand three security properties (stated informally here):

A-I Matching. If an attacker is only given oracle access to Sign_A, then it is hard to create any forgery that is not of Type I.

B-II Matching. If an attacker is only given oracle access to Sign_B, then it is hard to create any forgery that is not of Type II.

Dual-Oracle Invariance. If an attacker is given oracle access to both Sign_A and Sign_B and a "challenge signature" which is either from Sign_A or Sign_B, the attacker's probability of producing a Type I forgery is approximately the same when the challenge signature is from Sign_A as when the challenge signature is from Sign_B.

[1] Throughout, we will be discussing the CL signatures based on the LRSW assumption, which should not be confused with those based on the strong RSA assumption.

A Dual Form Signature scheme immediately gives a secure signature scheme if we simply set the signing algorithm $\text{Sign} = \text{Sign}_A$. Unforgeability now follows from a hybrid argument. Consider any EUF-CMA [43] attacker $\mathcal{A}$. By the A-I matching property, we know that it might have a noticeable probability ϵ of producing a Type I forgery, but has only a negligible probability of producing any other kind of forgery. We then show that ϵ must also be negligible. By the dual-oracle invariance property, the probability of producing a Type I forgery will be close to ϵ if we gradually replace the signing algorithm with Sign_B, one signature at a time. Once all of the signatures the attacker receives are from Sign_B, the B-II Matching property implies that the probability of producing a Type I forgery must be negligible in the security parameter.

We demonstrate the usefulness of our framework with two main applications, using significantly different techniques. This illustrates the versatility of our framework and its adaptability to schemes with different underlying structures. In particular, while dual form signatures are related to the dual system encryption methodology introduced by Waters [72] for proving full security of IBE schemes and other advanced encryption functionalities, we demonstrate that our dual form framework can be applied to signature schemes that have no known encryption or IBE analogs. Though all of the applications given here use bilinear groups, the dual form framework can be used in other contexts, including proofs under general assumptions.

Our first application is a slight variant of the Camenisch-Lysyanskaya signature scheme, set in a bilinear group $\mathbb{G}$ of composite order $N = p_1p_2p_3$. This application is surprising, since these signatures do not have a known IBE analog. We let $\mathbb{G}_{p_i}$ for each $i = 1, 2, 3$ denote the subgroup of order p_i in the group. The Sign_A algorithm produces signatures which exhibit the CL structure in the $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_2}$ subgroups and are randomized in the $\mathbb{G}_{p_3}$ subgroup. The Sign_B algorithm produces signatures which exhibit the CL structure in the $\mathbb{G}_{p_1}$ subgroup and are randomized in the $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ subgroups. Type I and II forgeries roughly mirror signatures of form A and B. The verification procedure in our scheme will verify that the signature is well formed in the $\mathbb{G}_{p_1}$ subgroup, but not "check" the other subgroups.

We prove security in the dual form framework based on three static subgroup decision-type assumptions, similar to those used in [56]. The most challenging part of the proof is dual-oracle invariance, which we prove by developing a backdoor verification test (performed by the simulator) which acts as an almost-perfect distinguisher between forgery types. Here we face a potential paradox, which is similar to that encountered in dual system encryption [72, 56]: we need to create a simulator that does not know whether the challenge signature it produces is distributed as an output of Sign_A or Sign_B, but it also must be able to test the type of the attacker's forgery. To arrange this, we create a "backdoor verification" test, which the simulator can perform to test the form of all but a small space of signatures. Essentially, this backdoor verification test acts an almost-perfect type distinguisher which fails to correctly determine the type of only a very small set of potential forgeries.

The challenge signature of unknown form produced by the simulator will fall within the untestable space; however, with very high probability a forgery by an attacker will not, because some information about this space is information-theoretically hidden from the attacker. This is possible because the elements of the verification key are all in the subgroup $\mathbb{G}_{p_1}$, and the space essentially resides in $\mathbb{G}_{p_2}$. Thus the verification key reveals no information about the hidden space. The only information about the space that the attacker receives is contained in the single signature of unknown type, and we show that this is insufficient for the attacker to be able to construct a forgery that falls inside the space for a *different message*. This is reminiscent of the concept of nominal semi-functionality in dual system encryption (introduced in [56]): in this setting, the simulator produces a key of unknown type which is correlated in its view with the ciphertext it produces, but this correlation is information-theoretically hidden from the attacker. This correlation prevents the simulator from determining the type of the key for itself by testing decryption against a ciphertext.

We believe that the existing techniques and systems built around CL signatures should apply to our variant with modest modifications. While our assumptions are not strictly comparable to the LRSW assumption, they have the qualitative feature of being static. Moreover, the security of our modified scheme can "fall back" on the LRSW assumption in the $\mathbb{G}_{p_1}$ subgroup, so security cannot be any worse. One can also view our proofs as a heuristic argument for gaining more confidence in the security of the prior scheme and the LRSW assumption.

As a second application of our dual form framework, we prove security from static assumptions for

a variant of the BGOY identity-based sequential aggregate signature scheme. Aggregate signatures are useful because they allow signature "compression," meaning that any n individual signatures by n (possibly) different signers on n (possibly) different messages can be transformed into an aggregate signature of the same size as an individual one that nevertheless allows verifying that all these signers signed their messages. However, aggregate signatures do not provide compression of the public keys, which are needed for signature verification. In the identity-based setting, only the identities of the signers are needed – this is a big savings because identities are much shorter than randomly generated keys. However, identity-based aggegate signatures have been notoriously difficult to realize.

We first prove security for a basic public-key version of the scheme, and then show that security for its identity-based sequential aggregate analog reduces to security of the basic scheme (in the random oracle model, as for the original proof). Our techniques here are significantly different, and reflect the different structure of the scheme (it is this structure that allows for aggregation). The core structure of the underlying public key scheme is composed of three group elements of the form $g^{a+bm}g^{r_1 r_2}, g^{r_1}, g^{r_2}$, where m is a message (or a hash of the message), a, b are fixed parameters, and r_1, r_2 are randomly chosen for each signature. There are significant differences between this and the core structure of other notable signatures, like CL and Waters signatures [29, 71]. Here, the message term is not multiplicatively randomized, but rather additively randomized by the quadratic term $r_1 r_2$. It is the quadratic nature of this term that allows verification via application of the bilinear map while thwarting attackers who try to combine received signatures by taking linear combinations in the exponents. This unique structure presents a challenge for static security analysis, and we develop new techniques to achieve a proof for a variant of this scheme in our dual form framework.

We still employ composite order subgroups, with the main structure of the scheme reflected in the $\mathbb{G}_{p_1}$ subgroup and the other two subgroups used for differentiating between signature and forgery types. However, to prove dual-oracle invariance, we rely on the fact that the scheme has the basic structure of a one-time signature scheme embedded in it, in addition to the quadratic mechanism to prevent an attacker from forming new signatures by taking combinations of received signatures. We capture the security resulting from this combination of structures through a static assumption for our dual-oracle invariance proof, and we show that this assumption holds in the generic group model. Though we do employ the generic group model as a check on our static assumptions, we believe that our proof provides valuable intuition into the security of the scheme that is not gleaned from a proof based on an interactive assumption or given solely in the generic group model. Also, checking the security of a static assumption in the generic group model is much easier (and less error-prone) than checking the security of an interactive assumption or scheme. We believe that the techniques and insights provided by our proof are an important step toward finding a prime order variant of the scheme that is secure under more standard assumptions, such as the decisional linear assumption.

In Appendix C, we provide one more application: a signature scheme using the private key structure in the Lewko-Waters IBE system [56]. The LW system itself can be viewed as a composite order extension of the Boneh-Boyen selectively secure IBE scheme [16], although the structure of the proofs of these systems are very different (LW achieves adaptive security). For this reason, we call these "BB-derived" signatures. While the existing LW IBE system can be transformed into a signature scheme using Naor's [2] general transformation, our scheme checks the signature "directly" without going through an IBE encryption. The resulting signature has a constant number of elements in the public key and signatures consist of two group elements.

Given that several interesting applications have arisen from CL signatures, a natural question is to ask if one can follow in the same path using BB-derived signatures. BB-derived signatures share desirable structural features with CL signatures, making them prime candidates for use in building the many applications that have been built from CL signatures. Using BB-derived signatures as a base scheme would also have several advantages. First, BB-derived signatures contain only two group elements, while CL signatures contain three. In addition, CL signatures require additional checks to rule out degenerate cases (i.e. the message cannot be 0, and the first signature element cannot be the identity). These additional checks percolate up to the applications that are built upon CL signatures, and might require more diligence to avoid errors easily made, whereas the BB-derived signature verification does not need to check for any such degenerate cases.

[2] Naor's observation was noted in [19].

Further Directions While we have focused here on applying our techniques for core short signatures, we envision that dual form signatures will be a framework for proving security of many different signature systems that have to this point been difficult to analyze under static assumption Some examples include embed additional structure, such as Attribute-Based signatures [61] and Quoteable signatures [3]. Attribute-based signatures allow a signer to sign a message with a predicate satisfied by his attributes, without revealing any additional information about his attribute set. Our framework could potentially be applied to obtain stronger security proofs for ABS schemes, such as the schemes of [61] proved only in the generic group model. Quoteable signatures enable derivation of signatures from each other under certain conditions, and current constructions are proved only selectively secure [3]. Another future target is signatures that "natively" sign group elements [1].

The primary goal of our work is providing techniques for realizing security under static assumptions, and we leverage composite order groups as a convenient setting for this. A natural future direction is to complement our work by discovering prime order analogs of our techniques. Many previous systems were originally constructed in composite order groups and later transferred into prime order groups [21, 46, 23, 45, 24, 70, 47, 52, 51, 38, 39, 55, 66]. The general techniques presented in [38, 54] do not seem directly applicable here, but we emphasize that our dual form framework is not tied to composite order groups and could also be used in the prime order setting.

1.1 Related Work

Goldwasser, Micali, and Rivest [43] gave the first formal definitions of security for signature schemes. Most early signature systems proved secure in the standard model were "tree-based". This included schemes built from general assumptions [11, 63, 68] as well as some that proposed more efficient solutions [37, 34, 35] that traded off the tree depth with the size of the public key.

The term "hash and sign" has been informally associated with more efficient signature schemes where the signature is compact and not built in a tree form. There exist several hash and sign systems in the random oracle model from different assumptions such as [69, 65, 12, 67, 22, 41], among others. In the standard model, there have been signature schemes with short signatures based on the Strong RSA assumption by Gennaro, Halevi, and Rabin [40] and Cramer-Shoup [36] and the RSA assumption by Hohenberger and Waters [48, 49].

Using bilinear groups, there are notable signature constructions by Boneh and Boyen [18], Camenisch and Lysyanskaya [29], and Waters [71]. The proofs of each of these schemes do not apply the random oracle heuristic. Boneh and Boyen [18] introduce a non-static "q-type" assumption called q-Strong Diffie-Hellman to prove security and Camenisch and Lysyanskaya prove their security based on the interactive LRSW assumption. The Waters' signature scheme proves security on the weaker Computational Diffie-Hellman assumption, but requires a larger public key in the scheme. As noted above, several advanced cryptosystems have been built using these signature schemes as building blocks.

Aggregate signatures were first introduced and realized in [20]. Due to the savings on bandwidth and storage that signature compression permits, aggregate signatures have proven useful in many practical applications, including PKI certificate chains and route attestation in Secure BGP. There has been a significant amount of work on aggregate signatures in the public-key setting [20, 59, 58, 64, 4]. In the identity-based setting, the first progress was made by Gentry and Ramzan [42], who presented a "synchronized" (using the terminology of [4]) identity-based aggregate signature scheme, meaning that signers are required to have synchronized clocks. However, it is desirable to avoid this requirement. The BGOY scheme that we modify [15] employs "sequential" aggregation (first introduced by [59]) instead of requiring synchronization. This means that the aggregation process cannot be carried out by anyone, but it must be done by the signers themselves one-by-one at the same time they produce their signatures. This turns out to be sufficient for many practical applications.

As previously noted, our dual form abstraction bears some resemblance to the dual system encryption technique introduced by Waters [72]; however, there are several important distinctions. While Naor showed that the private keys for IBE systems give immediate rise to signature schemes, the converse is not known to be true and several dual form signature systems such as CL signatures have no known IBE analog. In addition, even the signatures derived from IBE systems will typically have different verification algorithms

than the generic one derived from Naor's technique. Finally, we note that our dual form signature system is formally given as an abstraction. Our techniques are also related to techniques which have been employed recently in the area of leakage-resilience, e.g. [53, 62, 5]. In these works, the form of signatures or ciphertexts changes as the security proof progresses, similar to our strategy of changing signatures from one form to the other. There is also some relation to the work of Bellare and Goldwasser [10], who present signatures containing a (simulation-sound) NIZK proof of knowledge, and one can think of using the NIZK simulator as an alternate signing algorithm. The main contribution of our work is formalizing such techniques into an abstraction which can be broadly applied to signature schemes, particularly useful schemes like CL signatures.

2 Dual Form Signatures

We now define dual form signatures and their security properties. We then show that creating a secure dual form signature system naturally yields an existentially unforgeable signature scheme. We emphasize that the purpose of the dual form signature framework is to provide a template for creating security proofs from static assumptions, but the techniques employed to prove the required properties can be tailored to the structure of the particular scheme.

2.1 Definition

We define a dual form signature system to have the following algorithms:

KeyGen(λ)**:** Given a security parameter λ, generate a public key, VK, and a private key, SK.

Sign$_A$(SK, M)**:** Given a message, M, and the secret key, output a signature, σ.

Sign$_B$(SK, M)**:** Given a message, M, and the secret key, output a signature, σ.

Verify(VK, M, σ)**:** Given a message, the public key, and a signature, output 'true' or 'false'.

We note that a dual form signature scheme is identical to a usual signature scheme, except that it has two different signing algorithms. While only one signing algorithm will be used in the resulting existentially unforgeable scheme, having two different signing algorithms will be useful in our proof of security.

2.2 Forgery Classes

In addition to having two signature algorithms, the dual form signature framework also considers two disjoint classes of forgeries. Whether or not a signature verifies depends on the message that it signs as well as the verification key. For a fixed verification key, we consider the set of pairs, $\mathcal{S} \times \mathcal{M}$, over the message space, $\mathcal{M}$, and the signature space, $\mathcal{S}$. Consider the subset of these pairs for which the Verify algorithm outputs 'true': we will denote this set as $\mathcal{V}$. [3] We let $\mathcal{V}_I$ and $\mathcal{V}_{II}$ denote two disjoint subsets of $\mathcal{V}$, and we refer to signatures from these sets as *Type I* and *Type II* forgeries, respectively. In our applications, we will have the property $\mathcal{V} = \mathcal{V}_I \cup \mathcal{V}_{II}$ in addition to $\mathcal{V}_I \cap \mathcal{V}_{II} = \emptyset$, but only the latter property is necessary.

We will use these classes to specify two different types of forgeries received from an adversary in our proof of security. In general, these classes are not the same as the output ranges of our two signing algorithms. However, Type I forgeries will be related to signatures output by the Sign$_A$ algorithm and Type II forgeries will be related to signatures output by the Sign$_B$ algorithm. The precise relationships between the forgery types and the signing algorithms are explicitly defined by the following set of security properties for the dual form system.

[3] Here we will assume that the Verify algorithm is deterministic. If we consider a nondeterministic Verify algorithm, we could simply take the subset of ordered pairs that are accepted by Verify with non-negligible probability.

2.3 Security Properties

We define the following three security properties for a dual form signature scheme. We consider an attacker $\mathcal{A}$ who is initially given the verification key VK produced by running the key generation algorithm. The value SK is also produced, and *not* given to $\mathcal{A}$.

***A-I Matching*:** Let $\mathcal{O}_A$ be an oracle for the algorithm Sign_A. More precisely, this oracle takes a message as input, and produces a signature that is identically distributed to an output of the Sign_A algorithm (for the SK produced from the key generation). We say that a dual form signature is *A-I matching* if for all probabilistic polynomial-time (PPT) algorithms, $\mathcal{A}$, there exists a negligible function, $negl(\lambda)$, in the security parameter λ such that:

$$Pr[\mathcal{A}^{\mathcal{O}_A}(\text{VK}) \notin \mathcal{V}_I] = negl(\lambda).$$

This property guarantees that if an attacker is only given oracle access to Sign_A, then it is hard to create anything but a Type I forgery.

***B-II Matching*:** Let $\mathcal{O}_B$ be an oracle for the algorithm Sign_B (which takes in a message and outputs a signature that is identically distributed an output of the Sign_B algorithm). We say that a dual form signature is *B-II matching* if for all PPT algorithms, $\mathcal{A}$:

$$Pr[\mathcal{A}^{\mathcal{O}_B}(\text{VK}) \notin \mathcal{V}_{II}] = negl(\lambda).$$

This property guarantees that if an attacker is only given oracle access to Sign_B, then it is hard to create anything but a Type II forgery.

***Dual-Oracle Invariance* (DOI):** First we define the dual-oracle security game.

1. The key generation algorithm is run, producing a verification key VK and a secret key SK.
2. The adversary, $\mathcal{A}$, is given the verification key VK and oracle access to $\mathcal{O}_0 = \text{Sign}_A(\cdot)$ and $\mathcal{O}_1 = \text{Sign}_B(\cdot)$.
3. $\mathcal{A}$ outputs a challenge message, m.
4. A random bit, $b \leftarrow \{0, 1\}$, is chosen, and then a signature $\sigma \leftarrow \mathcal{O}_b(m)$ is computed and given to $\mathcal{A}$. We call σ the challenge signature.
5. $\mathcal{A}$ continues to have oracle access to $\mathcal{O}_0$ and $\mathcal{O}_1$.
6. $\mathcal{A}$ outputs a forgery pair (m^*, σ^*), where $\mathcal{A}$ has not already received a signature for m^*.

We say that a dual form signature scheme has dual-oracle invariance if, for all PPT attackers $\mathcal{A}$, there exists a negligible function, $negl(\lambda)$, in the security parameter λ such that

$$|Pr[(m^*, \sigma^*) \in \mathcal{V}_I | b = 1] - Pr[(m^*, \sigma^*) \in \mathcal{V}_I | b = 0]| = negl(\lambda).$$

We say that a dual form signature scheme is *secure* if it satisfies all three of these security properties.

2.4 Secure Signature Scheme

Once we have developed a secure dual form signature system, $(\text{KeyGen}^{DF}, \text{Sign}_A^{DF}, \text{Sign}_B^{DF}, \text{Verify}^{DF})$, this system immediately implies a secure signature scheme. The secure scheme is constructed as follows:

Construction 1.

- $\text{KeyGen} = \text{KeyGen}^{DF}$

- Sign $= \text{Sign}_A^{DF}$
- Verify $= \text{Verify}^{DF}$

Our new secure signature scheme is identical to the dual form system except that we have arbitrarily chosen to use Sign_A as our signing algorithm. We could have equivalently elected to use Sign_B. (In which case, we would modify the dual-oracle invariance property to be with respect to Type II forgeries instead of Type I forgeries. Alternatively, we could strengthen the property to address both forgery types.) Now we will prove that this signature scheme is secure.

2.5 Proof of Security

We prove:

Theorem 2.1. If $\Pi = (\text{KeyGen}^{DF}, \text{Sign}_A^{DF}, \text{Sign}_B^{DF}, \text{Verify}^{DF})$ is a secure dual form signature scheme, then Construction 1= $(\text{KeyGen}^{DF}, \text{Sign}_A^{DF}, \text{Verify}^{DF})$ is existentially unforgeable under an adaptive chosen message attack.

We begin by defining some additional games. We denote the first game as $Game_{Real}$. $Game_{Real}$ is the usual existential unforgeability security game for the digital signature scheme described in Construction 1. We denote the advantage of an algorithm $\mathcal{A}$ in $Game_{Real}$ as $Adv_{\mathcal{A}}^{Real}$. The next game, $Game_{OnlyI}$, is the same as $Game_{Real}$, except the attacker is limited to producing Type I forgeries. Finally, $Game_{OnlyI,B}$, will be defined in exactly the same way as $Game_{OnlyI}$ except that all of the attacker's queries are answered with signatures from the Sign_B algorithm.

We now introduce an additional property, oracle invariance, which we will show is implied by dual-oracle invariance through a hybrid argument.

***Oracle Invariance*:** We say that a dual form signature scheme satisfies *oracle invariance* if for all PPT algorithms, $\mathcal{A}$:

$$|Pr[\mathcal{A}^{\mathcal{O}_A}(\text{VK}) \in \mathcal{V}_I] - Pr[\mathcal{A}^{\mathcal{O}_B}(\text{VK}) \in \mathcal{V}_I]| = negl(\lambda).$$

In other words, given access to an oracle for either signature algorithm, an attacker is nearly equally likely to output a Type I forgery. Here, we must also restrict the output of $\mathcal{A}$ to be a forgery in the traditional sense of existential unforgeability: this means that $\mathcal{A}$ is not allowed to output a signature on a message it has queried to the signing oracle. We now prove that dual-oracle invariance implies oracle invariance.

Theorem 2.2. If a dual form signature scheme, $\Pi = (\text{KeyGen}^{DF}, \text{Sign}_A^{DF}, \text{Sign}_B^{DF}, \text{Verify}^{DF})$, has dual-oracle invariance, then it must have oracle invariance.

To prove Theorem 2.2, we will use a hybrid argument over a sequence of games. Suppose that there exists an attacker, $\mathcal{A}$, on $Game_{OnlyI}$, and that $\mathcal{A}$ makes a polynomial number of queries, q, to the signature oracle. Then, for $0 \leq k \leq q$, we define $Game_k$ to be the same as $Game_{OnlyI}$, except the first k signatures are answered by the Sign_B oracle and the last $q-k$ signatures are answered by the Sign_A oracle. We will denote the attacker's advantage in $Game_k$ by $Adv_{\mathcal{A}}^k$.

Lemma 2.1. Suppose that there exists an algorithm, $\mathcal{A}$, such that $Adv_{\mathcal{A}}^k - Adv_{\mathcal{A}}^{k-1} = \epsilon$ for some $0 \leq k \leq q$. Then, we can construct an algorithm, $\mathcal{B}$, that breaks dual-oracle invariance.

Proof. $\mathcal{B}$ is given oracle access to $\mathcal{O}_A = \text{Sign}_A(\cdot)$ and $\mathcal{O}_B = \text{Sign}_B(\cdot)$ and passes the corresponding public key to $\mathcal{A}$. Suppose that $\mathcal{A}$ requests a signature for message m_i for $0 \leq i \leq q$. If $i < k$, $\mathcal{B}$ will return $\mathcal{O}_B(m_i)$. If $i > k$, $\mathcal{B}$ will return $\mathcal{O}_A(m_i)$. However, if $i = k$, then $\mathcal{B}$ will pass m_i to the dual-oracle challenger and receive a challenge signature for m_i. $\mathcal{B}$ can then pass the challenge signature to $\mathcal{A}$. $\mathcal{A}$ then outputs a forgery, (m^*, σ^*), which $\mathcal{B}$ forwards to the challenger.

If the challenge signature that $\mathcal{B}$ receives is from the Sign_A algorithm, then $\mathcal{A}$ will be playing $Game_{k-1}$, and if the challenge signature is from the Sign_B algorithm then $\mathcal{A}$ will be playing $Game_k$. Therefore, if the challenge signature is from Sign_A, then the probability that $\mathcal{A}$ creates a valid, Type I forgery is exactly $Adv_{\mathcal{A}}^{k-1}$. Since $\mathcal{B}$ simply forwards this signature to the challenger, the probability that $\mathcal{B}$ outputs a forgery in $\mathcal{V}_I$ is equal to $Adv_{\mathcal{A}}^{k-1}$. Likewise, if the challenge signature is from Sign_B, then the probability that $\mathcal{B}$ outputs a forgery in $\mathcal{V}_I$ is equal to $Adv_{\mathcal{A}}^{k}$. Hence,

$$Pr[(m^*, \sigma^*) \in \mathcal{V}_I | b = 1] - Pr[(m^*, \sigma^*) \in \mathcal{V}_I | b = 0] = Adv_{\mathcal{A}}^{k} - Adv_{\mathcal{A}}^{k-1} = \epsilon.$$

This proves the lemma. □

Using Lemma 2.1, we can prove Theorem 2.2.

Proof of Theorem 2.2. Suppose that there exists an attacker, $\mathcal{A}$, that breaks the oracle invariance of the dual form system, attaining a probability difference of ϵ. We will construct an algorithm $\mathcal{B}$, such that $Adv_{\mathcal{B}}^{0} - Adv_{\mathcal{B}}^{q} = \epsilon$.

$\mathcal{B}$ is given access to an oracle for either the Sign_A algorithm or the Sign_B algorithm, depending on whether it is playing $Game_0$ or $Game_q$, respectively. $\mathcal{B}$ can use the oracle to sign any queries made by algorithm $\mathcal{A}$. $\mathcal{A}$ will return a forgery (m^*, σ^*), that $\mathcal{B}$ can output as a forgery. If $\mathcal{B}$ is playing $Game_0$, then the probability that it is successful is exactly equal to the probability that $\mathcal{A}$ produces a Type I forgery given access to a Sign_A oracle. Likewise, if $\mathcal{B}$ is playing $Game_q$, then the probability that it is successful is exactly equal to the probability that $\mathcal{A}$ produces a Type I forgery given access to a Sign_B oracle. That is,

$$Pr[\mathcal{A}^{\mathcal{O}_A}(\text{VK}) \in \mathcal{V}_I] - Pr[\mathcal{A}^{\mathcal{O}_B}(\text{VK}) \in \mathcal{V}_I] = Adv_{\mathcal{B}}^{0} - Adv_{\mathcal{B}}^{q} = \epsilon.$$

However, if there exists an attacker, namely $\mathcal{B}$, that can distinguish between $Game_0$ and $Game_q$ with advantage ϵ, then there is some $0 \leq k \leq q$ such that $\mathcal{B}$ can distinguish between $Game_k$ and $Game_{k-1}$ with advantage at least ϵ/q. However, we know from Lemma 2.1 that if there exists an attacker that can distinguish between $Game_k$ and $Game_{k-1}$ with advantage ϵ/q, then we can create an attacker that breaks dual-oracle invariance with advantage ϵ/q.

Since q is polynomial, if ϵ is non-negligible, then ϵ/q is also non-negligible. Thus, if there exists an attacker that breaks oracle invariance, then there exists an attacker that breaks dual-oracle invariance. This proves the theorem. □

We now prove Theorem 2.1. Our proof consists of a hybrid argument over a series of games. Between each step, we use the attacker against Construction 1 to break A-I matching, B-II matching, or oracle invariance. We begin by correctly simulating the signature scheme and limiting the attacker to Type I forgeries using the A-I matching property of the dual form system. Then, using the oracle invariance property of the dual form system, we change the simulation of the signature scheme to use the Sign_B algorithm to respond to the attacker's queries. However, since the attacker is still limited to Type I forgeries, if it produces a forgery then it violates the B-II matching property of the dual form system.

Lemma 2.2. Suppose there exists an algorithm $\mathcal{A}$, such that $Adv_{\mathcal{A}}^{Real} - Adv_{\mathcal{A}}^{OnlyI} = \epsilon$. Then, we can construct an algorithm, $\mathcal{B}$, with advantage ϵ in breaking the A-I matching property of the dual form system.

Proof. This follows directly from the definition of A-I matching. Given oracle access to Sign_A, $\mathcal{B}$ can simulate $Game_{Real}$ with $\mathcal{A}$. With probability ϵ, $\mathcal{A}$ produces a valid forgery, $m^* \notin \mathcal{V}_I$. $\mathcal{B}$ can pass this forgery to the A-I matching challenger. □

Lemma 2.3. Suppose that there exists an algorithm, $\mathcal{A}$, such that $Adv_{\mathcal{A}}^{OnlyI} - Adv_{\mathcal{A}}^{OnlyI,B} = \epsilon$. Then, we can construct an algorithm $\mathcal{B}$, that breaks the oracle invariance of the dual form system with advantage ϵ.

Proof. This follows from the definition of the oracle invariance of a dual form system. $\mathcal{B}$ will be given oracle access to either Sign_A or Sign_B from the oracle invariance challenger. $\mathcal{B}$ will use this oracle to answer all of the signing queries made by algorithm $\mathcal{A}$. If $\mathcal{B}$ receives an oracle for Sign_A from the challenger, then it will be simulating $Game_{OnlyI}$, and if $\mathcal{B}$ receives an oracle for Sign_B, it will be simulating game $Game_{OnlyI,B}$.

Eventually, $\mathcal{B}$ will return the forgery output by $\mathcal{A}$. When the output of $\mathcal{A}$ is a Type I forgery, the output of $\mathcal{B}$ must be a Type I forgery and in particular,

$$Pr[\mathcal{B}^{\mathcal{O}_A}(\text{VK}) \in \mathcal{V}_I] - Pr[\mathcal{B}^{\mathcal{O}_B}(\text{VK}) \in \mathcal{V}_I] = Adv_{\mathcal{A}}^{OnlyI} - Adv_{\mathcal{A}}^{OnlyI,B} = \epsilon.$$

Thus, $\mathcal{B}$ breaks the oracle invariance of the dual form system with advantage ϵ. □

Finally, we now show that the advantage of any attacker against $Game_{OnlyI,B}$ is negligible.

Lemma 2.4. Suppose that there exists an algorithm, $\mathcal{A}$, such that $Adv_{\mathcal{A}}^{OnlyI,B} = \epsilon$. Then, we can construct an algorithm, $\mathcal{B}$, with advantage ϵ in breaking the B-II matching property of the dual form system.

Proof. Again, this follows from the definition of B-II matching for the dual form system. If we simply let $\mathcal{B} = \mathcal{A}$, we know that with probability ϵ, $\mathcal{A}$ will create a valid Type I forgery given only oracle access to the Sign_B algorithm. This means that $\mathcal{B}$ will succeed in breaking B-II matching with probability ϵ. □

Theorem 2.1 now follows.

3 Background on Composite Order Bilinear Groups

Composite order bilinear groups were first introduced in [21]. We define a group generator $\mathcal{G}$, an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group $\mathbb{G}$. In our case, we will have $\mathcal{G}$ output $(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e)$ where p_1, p_2, p_3 are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order $N = p_1p_2p_3$, and $e : \mathbb{G}^2 \to \mathbb{G}_T$ is a map such that:

1. (Bilinear) $\forall g, h \in \mathbb{G},\ a, b \in \mathbb{Z}_N,\ e(g^a, h^b) = e(g, h)^{ab}$
2. (Non-degenerate) $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order N in $\mathbb{G}_T$.

Computing $e(g, h)$ is also commonly referred to as "pairing" g with h.

We assume that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are computable in polynomial time with respect to λ, and that the group descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ include generators of the respective cyclic groups. We let $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$, and $\mathbb{G}_{p_3}$ denote the subgroups of order p_1, p_2, and p_3 in $\mathbb{G}$ respectively. We note that when $h_i \in \mathbb{G}_{p_i}$ and $h_j \in \mathbb{G}_{p_j}$ for $i \neq j$, $e(h_i, h_j)$ is the identity element in $\mathbb{G}_T$. To see this, suppose we have $h_1 \in \mathbb{G}_{p_1}$ and $h_2 \in \mathbb{G}_{p_2}$. Let g denote a generator of $\mathbb{G}$. Then, $g^{p_1p_2}$ generates $\mathbb{G}_{p_3}$, $g^{p_1p_3}$ generates $\mathbb{G}_{p_2}$, and $g^{p_2p_3}$ generates $\mathbb{G}_{p_1}$. Hence, for some α_1, α_2, $h_1 = (g^{p_2p_3})^{\alpha_1}$ and $h_2 = (g^{p_1p_3})^{\alpha_2}$. Then:

$$e(h_1, h_2) = e(g^{p_2p_3\alpha_1}, g^{p_1p_3\alpha_2}) = e(g^{\alpha_1}, g^{p_3\alpha_2})^{p_1p_2p_3} = 1.$$

This orthogonality property of $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ is a useful feature of composite order bilinear groups which we leverage in our constructions and proofs.

If we let g_1, g_2, g_3 denote generators of the subgroups $\mathbb{G}_{p_1}$, $\mathbb{G}_{p_2}$, and $\mathbb{G}_{p_3}$ respectively, then every element h in $\mathbb{G}$ can be expressed as $h = g_1^a g_2^b g_3^c$ for some $a, b, c \in \mathbb{Z}_N$. We refer to g_1^a as the "$\mathbb{G}_{p_1}$ part" or "$\mathbb{G}_{p_1}$ component" of h. If we say that an h has no $\mathbb{G}_{p_2}$ component, for example, we mean that $b \equiv 0 \mod p_2$. Below, we will often use g to denote an element of $\mathbb{G}_{p_1}$ (as opposed to writing g_1).

The original Camenisch-Lysyanskaya scheme and BGOY identity-based sequential aggregate signature scheme both use prime order bilinear groups, i.e. groups $\mathbb{G}$ and $\mathbb{G}_T$ are each of prime order q with an efficiently computable bilinear map $e : \mathbb{G}^2 \to \mathbb{G}_T$.

4 Camenisch-Lysyanskaya Signatures

Now we use the dual form framework to prove security of a signature scheme similar to the one put forward by Camenisch and Lysyanskaya [29]. The Camenisch-Lysyanskaya signature scheme was already shown to be secure under the LRSW assumption. However, the scheme can be naturally adapted to our framework, allowing us to prove security under static, non-interactive assumptions. Our result is not strictly comparable to the result under the LRSW assumption because our signature scheme is not identical to the original. However, this is the first proof of security for a scheme similar to the Camenisch-Lysyanskaya signature scheme from static, non-interactive assumptions.

Our signature scheme will use bilinear groups, $\mathbb{G}$ and $\mathbb{G}_T$, of composite order $N = p_1p_2p_3$, where p_1, p_2, and p_3 are all distinct primes. Our construction is identical to the original Camenisch-Lysyanskaya signature scheme in the $\mathbb{G}_{p_1}$ subgroup, but with additional components in the subgroups $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$. The signatures produced by the Sign_A algorithm will have random components in $\mathbb{G}_{p_3}$ and components in $\mathbb{G}_{p_2}$ which mirror the structure of the scheme in $\mathbb{G}_{p_1}$. The signatures produced by the Sign_B algorithm will have random components in both $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$. Type I forgeries are those that are distributed exactly like Sign_A signatures in the $\mathbb{G}_{p_2}$ subgroup, while Type II forgeries encompass all other distributions.

To prove dual-oracle invariance, we develop a *backdoor verification* test that the simulator can use to determine the type of the attacker's forgery. We leverage the fact that the simulator will know the discrete logarithms of the public parameters, which will allow it to strip off the components in $\mathbb{G}_{p_1}$ in the forgery and check the distribution of the $\mathbb{G}_{p_2}$ components. This check will fail to determine the type correctly only with negligible probability. In more detail, we create a simulator which must solve a subgroup decision problem and ascertain whether an element T is in $\mathbb{G}_{p_1p_3}$ or in the full group $\mathbb{G}$. It will use T to create a challenge signature which is either distributed as an output of the Sign_A algorithm or as an output of the Sign_B algorithm, depending on the nature of T. It will be unable to determine the nature of this signature for itself because this will fall into the negligible error space of its backdoor verification test. When the simulator receives a forgery from the attacker, it will perform the backdoor verification test and correctly determine the type of the forgery, unless the attacker manages to produce a forgery for which this test fails. This will occur only with negligible probability, because the attacker will have only limited information about the error space from the challenge signature, and it needs to forge on a *different* message. This is possible because the public parameters are in $\mathbb{G}_{p_1}$, and so reveal no information about the error space of the backdoor test modulo p_2. We use a pairwise independent argument to show that the limited amount of information the attacker can glean from the challenge signature on a message m is insufficient for it to produce a forgery for a different message m^* that causes the backdoor test to err.

4.1 Our Dual Form Scheme

KeyGen(λ)**:** The key generation algorithm chooses two groups, $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$, of order $N = p_1p_2p_3$ (where p_1, p_2, and p_3 are all distinct primes of length λ) that have a non-degenerate, efficiently computable bilinear map, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. It then selects uniformly at random $g \in \mathbb{G}_{p_1}$, $g_3 \in \mathbb{G}_{p_3}$, $g_{2,3} \in \mathbb{G}_{p_2p_3}$, and $x, y, x_e, y_e \in \mathbb{Z}_N$. It sets

$$\text{SK} = (x, y, x_e, y_e, g_3, g_{2,3}),$$

and

$$PK = (N, \mathbb{G}, g, X = g^x, Y = g^y).$$

Sign$_A(\text{SK}, m)$**:** Given a secret key $(x, y, x_e, y_e, g_3, g_{2,3})$, a public key $(N, \mathbb{G}, g, X, Y)$, and a message $m \in Z_N^*$, the algorithm chooses a random $r, r' \in \mathbb{Z}_N$, $R_{2,3} \in \mathbb{G}_{p_2p_3}$, and random R_3, R_3', and $R_3'' \in \mathbb{G}_{p_3}$, and outputs the signature

$$\sigma = (g^r R_{2,3}^{r'} R_3,\ (g^r)^y (R_{2,3}^{r'})^{y_e} R_3',\ (g^r)^{x+mxy} (R_{2,3}^{r'})^{x_e + m x_e y_e} R_3'').$$

Note that the random elements of $\mathbb{G}_{p_3}$ can be obtained by raising g_3 to random exponents modulo N. Likewise, the random elements of $\mathbb{G}_{p_2p_3}$ can be obtained by raising $g_{2,3}$ to random exponents modulo

N. The random exponents modulo N will be uncorrelated modulo p_2 and modulo p_3 by the Chinese Remainder Theorem.

$\mathbf{Sign}_B(\mathrm{SK}, m)$: Given a secret key $(x, y, x_e, y_e, g_3, g_{2,3})$, a public key $(N, \mathbb{G}, g, X, Y)$, and a message $m \in Z_N^*$, the algorithm chooses a random $r \in \mathbb{Z}_N$ and random $R_{2,3}$, $R'_{2,3}$, and $R''_{2,3} \in \mathbb{G}_{p_2p_3}$, and outputs the signature

$$\sigma = (g^r R_{2,3},\ (g^r)^y R'_{2,3},\ (g^r)^{x+mxy} R''_{2,3}).$$

The random elements can be generated in the same way as in Sign_A.

$\mathbf{Verify}(\mathrm{VK}, m, \sigma)$: Given a public key $pk = (N, \mathbb{G}, g, X, Y)$, message $m \neq 0$, and a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, the verification algorithm checks that:

$$e(\sigma_1, g) \neq 1$$

(which ensures that $\sigma_1 \notin \mathbb{G}_{p_2p_3}$), and

$$e(\sigma_1, Y) = e(g, \sigma_2) \text{ and } e(X, \sigma_1) \cdot e(X, \sigma_2)^m = e(g, \sigma_3).$$

As in the original CL scheme, messages must be chosen from $\mathbb{Z}_N^*$, so that $m \neq 0$. If we allow $m = 0$, then an adversary can easily forge a valid signature using the public key elements (g, Y, X). Also like the original scheme, the Verify algorithm will not accept a signature where all the elements are the identity in $\mathbb{G}_{p_1}$. It suffices to check that the first element is not the identity in $\mathbb{G}_{p_1}$ and that the other verification equations are satisfied. If σ_1 is the identity in $\mathbb{G}_{p_1}$, then it will be an element of the subgroup $\mathbb{G}_{p_2p_3}$. To determine if $\sigma_1 \in \mathbb{G}_{p_2p_3}$, we pair σ_1 with the public key element g under the bilinear map and verify that it does not equal the identity in $\mathbb{G}_T$. Without this check, a signature where all three elements are members of the subgroup $\mathbb{G}_{p_2p_3}$ would be valid for any message with the randomness $r' = 0 \bmod p_1$.

Notice, until Sign_A is called, no information about the exponents x_e and y_e is given out. Once Sign_A is called, these exponents behave exactly like the secret key exponents x and y, except in the $\mathbb{G}_{p_2}$ subgroup. These exponents will be used to verify that a forgery is of Type I. The additional randomization with the $\mathbb{G}_{p_3}$ elements guarantees that there will be no correlation in the $\mathbb{G}_{p_3}$ subgroup between the three signature elements. Unlike the signatures given out by the Sign_A algorithm, signatures from the Sign_B algorithm will be completely randomized in the $\mathbb{G}_{p_2}$ subgroup as well.

Forgery Classes We will divide verifiable forgeries according to their correlation in the $\mathbb{G}_{p_2}$ subgroup, similar to the way we have defined the signatures from the Sign_A and Sign_B algorithms. We let z be an exponent in $\mathbb{Z}_N$. By the Chinese Remainder Theorem, we can represent z as an ordered tuple $(z_1, z_2, z_3) \in \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times \mathbb{Z}_{p_3}$, where $z_1 = z \bmod p_1$, $z_2 = z \bmod p_2$, and $z_3 = z \bmod p_3$. Letting $(z_1, z_2, z_3) = (0 \bmod p_1, 1 \bmod p_2, 0 \bmod p_3)$ and g_2 be a generator of $\mathbb{G}_{p_2}$, we define the forgery classes as follows:

Type I. $\mathcal{V}_I = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_1^*)^z = g_2^{r'}, (\sigma_2^*)^z = g_2^{r'y_e}, \text{ and } (\sigma_3^*)^z = g_2^{r'(x_e + m^* x_e y_e)} \text{ for some } r'\}$

Type II. $\mathcal{V}_{II} = \{(m^*, \sigma^*) \in \mathcal{V} | (m^*, \sigma^*) \notin \mathcal{V}_I\}$

Essentially, Type I forgeries will be correlated in the $\mathbb{G}_{p_2}$ subgroup exactly in the same way as they are correlated in the $\mathbb{G}_{p_1}$ subgroup, with the exponents x_e and y_e playing the same role in the $\mathbb{G}_{p_2}$ subgroup that x and y play in the $\mathbb{G}_{p_1}$ subgroup. Type I forgeries will align with the Sign_A algorithm, to guarantee that our scheme is A-I matching. Type II forgeries include any other verifiable signatures, i.e. those not correctly correlated in the $\mathbb{G}_{p_2}$ subgroup. Unlike the signatures produced by the Sign_B algorithm, Type II forgeries need not be completely random in the $\mathbb{G}_{p_2}$ subgroup. However, we will show in our proof of security that this is enough to guarantee B-II matching.

4.2 Complexity Assumptions

We now state our complexity assumptions. We let $\mathbb{G}$ and $\mathbb{G}_T$ denote two cyclic groups of order $N = p_1 p_2 p_3$, where p_1, p_2, and p_3 are distinct primes, and $e : \mathbb{G}^2 \to \mathbb{G}_T$ is an efficient, non-degenerate bilinear map. In addition, we will denote the subgroup of $\mathbb{G}$ of order $p_1 p_2$ as $\mathbb{G}_{p_1 p_2}$, for example.

The first two of these assumptions were introduced in [56], where it is proven that these assumptions hold in the generic group model, assuming it is hard to find a non-trivial factor of the group order, N. These are specific instances of the General Subgroup Decision Assumption described in [13]. The third assumption is new, and in Appendix B we prove that it also holds in the generic group model, assuming it is hard to find a non-trivial factor of the group order, N.

Assumption 4.1. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, X_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3 \xleftarrow{R} \mathbb{G}_{p_3}$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, X_1 X_2, X_3)$$

$$T_1 \xleftarrow{R} \mathbb{G}_{p_1 p_2}, T_2 \xleftarrow{R} \mathbb{G}_{p_1}$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 4.1 to be:

$$Adv^{4.1}_{\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 1. We say that $\mathcal{G}$ satisfies Assumption 4.1 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{4.1}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 4.2. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3, Y_3 \xleftarrow{R} \mathbb{G}_{p_3},$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, X_1 X_2, X_3, Y_2 Y_3),$$

$$T_1 \xleftarrow{R} \mathbb{G}, T_2 \xleftarrow{R} \mathbb{G}_{p_1 p_3}$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 4.2 to be:

$$Adv^{4.2}_{\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 2. We say that $\mathcal{G}$ satisfies Assumption 4.2 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{4.2}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 4.3. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$a, r \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} \mathbb{G}_{p_1}, X_2, X_2', X_2'', Z_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3 \xleftarrow{R} \mathbb{G}_{p_3},$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g^a, g^r X_2, g^{ra} X_2', g^{ra^2} X_2'', Z_2, X_3),$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 4.3 to be:

$$Adv^{4.3}_{\mathcal{A}}(\lambda) := Pr[\mathcal{A}(D) = (g^{r'a^2} R_3, g^{r'} R_3') \text{ and } r' \neq 0 \bmod p_1],$$

where R_3 and R_3' are any values in the subgroup $\mathbb{G}_{p_3}$.

Definition 3. We say that $\mathcal{G}$ satisfies Assumption 4.3 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{4.3}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.

4.3 Proof of Security

We will show that our signature scheme is secure under these assumptions by showing that it satisfies the three properties of a secure dual form signature scheme.

Lemma 4.1. If Assumption 4.1 holds, then our signature scheme is A-I matching.

Proof. Suppose that there exists an attacker, $\mathcal{A}$, that can create a Type II forgery (i.e. a non-Type I forgery) with probability ϵ given access to an oracle for the Sign_A algorithm. Then we can create an algorithm $\mathcal{B}$ that breaks Assumption 4.1 with advantage ϵ.

$\mathcal{B}$ first receives g, X_1X_2, X_3, T. $\mathcal{B}$ chooses random exponents $x, y \in \mathbb{Z}_N$ and passes (g, g^x, g^y) to $\mathcal{A}$. $\mathcal{B}$ then acts as an oracle for the Sign_A algorithm, using (x, y, X_3) as the secret key, where $x_e = x$ and $y_e = y$. Note that by the Chinese Remainder Theorem, $x \bmod p_1$ and $x \bmod p_2$ are uncorrelated, so the value x modulo p_1 will not be correlated to the value of x_e modulo p_2, and likewise, y modulo p_1 will not be correlated to y_e modulo p_2. Also, notice that $\mathcal{B}$ does not actually need to know the secret key element in $\mathbb{G}_{p_2p_3}$ because it will be able to simulate the Sign_A algorithm without it.

To sign a message m, $\mathcal{B}$ chooses random exponents $r, t, u, v \in \mathbb{Z}_N$ and returns

$$\sigma = ((X_1X_2)^r X_3^t,\ (X_1X_2)^{ry} X_3^u,\ (X_1X_2)^{r(x+mxy)} X_3^v).$$

$\mathcal{A}$ produces a forgery (m^*, σ^*), where $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$. First $\mathcal{B}$ must check that $\text{Verify}(m^*, \sigma^*) =$ 'true', if not then $\mathcal{B}$ will abort. In addition, $\mathcal{B}$ must verify that it has received a Type II forgery. To do this, $\mathcal{B}$ can check the correlation in both the $\mathbb{G}_{p_2}$ components by pairing each signature element with X_1X_2 as follows. It checks if:

$$e(\sigma_1^*, (X_1X_2)^y) = e(\sigma_2^*, X_1X_2)$$
$$e(\sigma_1^*, (X_1X_2)^{x+mxy}) = e(\sigma_3^*, X_1X_2).$$

If and only if (m^*, σ^*) is a Type I forgery, both of these equations will hold. Therefore, if both of these are true, then $\mathcal{B}$ will abort. Notice that $\mathcal{B}$ can only create Type I forgeries, and it needs the output of $\mathcal{A}$ to be a Type II forgery in order to defeat the Assumption 4.1 challenger.

Finally, if one or both of these equations fail, $\mathcal{B}$ will be guaranteed that it has a Type II forgery, which it can use to determine what subgroup T is in. $\mathcal{B}$ can simply use T in place of X_1X_2 in the previous two equations and checks if:

$$e(\sigma_1^*, T^y) = e(\sigma_2^*, T)$$
$$e(\sigma_1^*, T^{x+mxy}) = e(\sigma_3^*, T).$$

By the contrapositive of the previous argument, since (m^*, σ^*) is a Type II forgery, if $T \in \mathbb{G}_{p_1p_2}$, then one of these equations must fail. On the other hand, since the forgery must be verifiable, if $T \in \mathbb{G}_{p_1}$, the $\mathbb{G}_{p_2}$ components will not affect the bilinear map and the correlation will correctly hold in the $\mathbb{G}_{p_1}$ components. Thus, if $(m^*, \sigma^*) \in \mathcal{V}_{II}$ then $\mathcal{B}$ can determine what subgroup T is in with probability 1.

Hence, if $\mathcal{A}$ succeeds in creating a Type II forgery with non-negligible probability ϵ, then $\mathcal{B}$ can achieve a non-negligible advantage against Assumption 4.1. □

Lemma 4.2. If Assumption 4.2 holds, then our signature scheme has the dual-oracle invariance property.

Proof. Suppose that there exists an attacker, $\mathcal{A}$, that can break dual-oracle invariance with a non-negligible advantage. Then we can create an algorithm $\mathcal{B}$ that breaks Assumption 4.2 with non-negligible advantage.

$\mathcal{B}$ receives $\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3, T$. $\mathcal{B}$ chooses random exponents $x, y, x_e, y_e \in \mathbb{Z}_N$ and sets $\text{SK} = (x, y, x_e, y_e, X_3, Y_2Y_3)$ and $PK = (\mathbb{G}, g, X = g^x, Y = g^y)$. Using the secret key, $\mathcal{B}$ can answer queries from $\mathcal{A}$ to either oracle, by simply deciding whether to randomize the second two pieces of the signature in the $\mathbb{G}_{p_2}$ subgroup.

For a query, m, to the Sign_A oracle, $\mathcal{B}$ chooses $r, r', t, u, v \in \mathbb{Z}_N$ and responds with:

$$\sigma = (g_{p_1}^r (Y_2Y_3)^{r'} X_3^t,\ g_{p_1}^{ry} (Y_2Y_3)^{r'y_e} X_3^u,\ g_{p_1}^{r(x+mxy)} (Y_2Y_3)^{r'(x_e+mx_ey_e)} X_3^v).$$

For query, m, to the Sign_B oracle, $\mathcal{B}$ chooses $r, t, u, v \in \mathbb{Z}_N$ and responds with:

$$\sigma = (g_{p_1}^r (Y_2Y_3)^t,\ g_{p_1}^{ry} (Y_2Y_3)^u,\ g_{p_1}^{r(x+mxy)} (Y_2Y_3)^v).$$

Finally, for the challenge message, m, $\mathcal{B}$ chooses $r, r', t, u, v \in \mathbb{Z}_N$ and returns:

$$\sigma = (T^r (Y_2Y_3)^{r'} X_3^t,\ T^{ry} (Y_2Y_3)^{r'y_e} X_3^u,\ T^{r(x+mxy)} (Y_2Y_3)^{r'(x_e+mx_ey_e)} X_3^v).$$

If $T \in \mathbb{G}_{p_1p_3}$, then $\mathcal{B}$ has correctly simulated the Sign_A oracle on the challenge message, and If $T \in \mathbb{G}$, then $\mathcal{B}$ has properly simulated the Sign_B oracle. We will now argue that $\mathcal{B}$ can use the output of $\mathcal{A}$ to determine whether T has a $\mathbb{G}_{p_2}$ component or not.

In order to use the output of $\mathcal{A}$ to determine which subgroup T belongs to, $\mathcal{B}$ must be able to determine whether $\mathcal{A}$ returns a Type I or a Type II forgery. First, $\mathcal{B}$ will check that $\mathcal{A}$ has not seen a signature for m^* before and that the forgery (m^*, σ^*) verifies correctly, if not then $\mathcal{B}$ will guess randomly. Next, $\mathcal{B}$ will try to distinguish between the two forgery types by checking the correlation in the $\mathbb{G}_{p_2}$ components of the signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$. We call this our *backdoor verification test.*

If we let g_i be a generator of the subgroup $\mathbb{G}_{p_i}$, we can rewrite the signature elements as follows:

$$\begin{aligned}
\sigma_1^* &= g^r g_2^{r'} g_3^u \\
\sigma_2^* &= g^{ry} g_2^{r'y_e + k_2} g_3^v \\
\sigma_3^* &= g^{r(x+m^*xy)} g_2^{r'(x_e+m^*x_ey_e)+k_3} g_3^w,
\end{aligned}$$

where r, r', k_2, k_3, u, v, and w are all constants from $\mathbb{Z}_N$. From the definition of our forgery types, in this new representation, (m^*, σ^*) is a Type I forgery if and only if $k_2 = k_3 = 0$.

To determine if $k_2 = k_3 = 0$, first $\mathcal{B}$ will need to isolate the $\mathbb{G}_{p_2}$ components of σ_2^* and σ_3^* by removing their $\mathbb{G}_{p_1}$ components. We will refer to σ_2^* with its $\mathbb{G}_{p_1}$ component removed as s_2, and σ_3^* with its $\mathbb{G}_{p_1}$ component removed as s_3. $\mathcal{B}$ is able to find s_2 and s_3 by using the known correlation in the $\mathbb{G}_{p_1}$ components as follows:

$$\begin{aligned}
s_2 &= \frac{\sigma_2^*}{(\sigma_1^*)^y} = \frac{g^{ry} g_2^{r'y_e+k_2} g_3^v}{g^{ry} g_2^{r'y} g_3^{uy}} = g_2^{r'(y_e-y)+k_2} g_3^{v'} \\
s_3 &= \frac{\sigma_3^*}{(\sigma_1^*)^{x+m^*xy}} = \frac{g^{r(x+m^*xy)} g_2^{r'(x_e+m^*x_ey_e)+k_3} g_3^w}{g^{r(x+m^*xy)} g_2^{r'(x+m^*xy)} g_3^{u(x+m^*xy)}} \\
&= g_2^{r'((x_e+m^*x_ey_e)-(x+m^*xy))+k_3} g_3^{w'},
\end{aligned}$$

where $v' = v - uy$ and $w' = w - u(x + m^*xy)$. These constants will be unimportant since we only care about the correlation in the $\mathbb{G}_{p_2}$ components. Next, we will use the expected correlation for a Type I forgery in the $\mathbb{G}_{p_2}$ components of s_2 and s_3 to try remove them from the remaining $\mathbb{G}_{p_3}$ components. Let

$$c^* = \frac{(x_e + m^*x_ey_e) - (x + m^*xy)}{y_e - y},$$

then

$$s = \frac{s_3}{s_2^{c^*}} = g_2^{k_3 - c^*k_2} g_3^{w''},$$

where $w'' = w' - c^*v'$. Finally, $\mathcal{B}$ will check if s is indeed an element of $\mathbb{G}_{p_3}$ by computing the pairing of s with X_1X_2 and checking if it is the identity element in $\mathbb{G}_T$,

$$e(s, X_1X_2) \stackrel{?}{=} 1 \in \mathbb{G}_T.$$

Clearly, if (m^*, σ^*) is a Type I forgery, then $k_2 = k_3 = 0$, so $s \in \mathbb{G}_{p_3}$. However, s will be an element of $\mathbb{G}_{p_3}$ as long as $k_3 - c^* k_2 = 0 \bmod p_2$. This gives us the criterion for a Type II forgery to pass this backdoor verification test. In fact, notice that our challenge signature will meet exactly this requirement, regardless of what subgroup T is in. This is important, because it means that $\mathcal{B}$ cannot break the assumption without the output of $\mathcal{A}$.

We claim that an attacker can only create a Type II forgery that satisfies this criterion with negligible probability. We know that in a Type II forgery, k_2 or k_3 will be nonzero. However, in order to pass the backdoor verification test, if k_2 or k_3 is nonzero, they both must be nonzero. Thus, the attacker must be able to produce a forgery with exponents k_2 and k_3 that non-trivially satisfy the equation $k_3 - c^* k_2 = 0$ (modulo p_2). This requires the attacker to implicitly find the value of c^* modulo p_2.

During the query phase, x and y modulo p_2 are not given out by either signing oracle. The only place they appear is in the challenge signature. The exponents x_e and y_e modulo p_2 are information-theoretically revealed by $\mathcal{B}$ during the query phase, and the challenge message m will be known to the attacker. However, in the challenge signature, there remains the freshly chosen random exponents r and r' and the values of x and y modulo p_2. No information could have previously been revealed about these exponents modulo p_2. (Note that the r and r' from the challenge signature are not the same as the r and r' from the forgery σ^*.)

The challenge signature establishes three equations in $\mathbb{Z}_{p_2}$ from the exponents of the three $\mathbb{G}_{p_2}$ components. In these three equations, there are four unknown values, x, y, r, and r' modulo p_2. We will also need to consider the constant exponents contributed by T and $Y_2 Y_3$. We will represent these elements by their subgroup components,

$$\begin{aligned} T &= g^{t_1} g_2^{t_2} g_3^{t_3} \\ Y_2 Y_3 &= g_2^{y_2} g_3^{y_3}, \end{aligned}$$

where g_i is a generator for the subgroup $\mathbb{G}_{p_i}$.

Finally, if we define the three exponents of the three challenge signature elements to be a_1, a_2, a_3, respectively, then the equations available to the attacker are

$$t_2 r + y_2 r' = a_1 \tag{1}$$

$$t_2 r y + y_2 r' y_e = a_2 \tag{2}$$

$$t_2 r(x + mxy) + y_2 r'(x_e + m x_e y_e) = a_3. \tag{3}$$

Recall that all these equations hold modulo p_2. We will argue that from these equations, an attacker has only a negligible chance of implicitly determining

$$c^* = \frac{(x_e + m^* x_e y_e) - (x + m^* xy)}{y_e - y}.$$

We will relabel some of our variables by defining $s := t_2 r + y_2 r'$, $v := t_2 r$ (which are distributed uniformly randomly). We can then rewrite our three equations as:

$$\begin{aligned} s &= a_1 \\ s y_e + v(y - y_e) &= a_2 \\ s(x_e + m x_e y_e) + v(x - x_e + m(xy - x_e y_e)) &= a_3. \end{aligned}$$

Now, information-theoretically, the attacker knows the values of $a_1, a_2, a_3, x_e, y_e, m$, so it additionally learns the values $v(y - y_e)$ and $v(x - x_e + m(xy - x_e y_e))$ here, and nothing more. We introduce one more change of variable, setting $v' := v(y - y_e)$, $b := v(x - x_e)$, and $d := v(xy - x_e y_e)$. We argue that the joint distribution of v', b, d modulo p_2 is negligibly close to the uniform distribution on $\mathbb{Z}_{p_2}^3$. To see this, note that given a

triple of values for v', b, d in $\mathbb{Z}_{p_2}$ (with x_e, y_e fixed), one can (almost always) solve uniquely for v, x, y as:

$$v = \frac{v'b}{d - by_e - x_e v'}$$
$$y = \frac{v'}{v} + y_e$$
$$x = \frac{b}{v} + x_e.$$

With all but negligible probability, the denominators will be invertible. Now, information-theoretically, the attacker knows only v' and $b + md$ modulo p_2 (and also knows m). If it correctly determines c^* modulo p_2, then it can also determine $v'c^* = b + dm^*$. This can only happen with negligible probability when $m^* \neq m$ modulo p_2, since $b + dm$ is a pairwise independent function of m. Now, it is possible that m^* is equal to m modulo p_2, but unequal to m modulo N. If this occurs with non-negligible probability, then $\mathcal{B}$ can extract a non-trivial factor of N by computing the g.c.d of N and $m - m^*$. This non-trivial factor can be used to break Assumption 4.2 with non-negligible advantage, as proven in [56]. Thus, we may assume that $m \neq m^*$ modulo p_2, except with negligible probability.

Since the attacker has a negligible chance of creating a Type II forgery that will pass the backdoor verification test, if the forgery does pass, then with probability negligibly close to 1, it must be a Type I forgery. We already know that if the forgery fails the test, it must be a Type II forgery. Therefore, $\mathcal{B}$ will be able to determine the forgery type with probability negligibly close to 1, and thus can break Assumption 4.2 with non-negligible advantage. □

Lemma 4.3. If Assumption 4.3 holds, then our signature scheme is B-II matching.

Proof. Suppose that there exists an attacker, $\mathcal{A}$, that can create a Type I forgery with non-negligible probability ϵ given access to an oracle for the Sign_B algorithm. Then we can create an algorithm $\mathcal{B}$ that breaks Assumption 4.3 with probability negligibly close to ϵ.

$\mathcal{B}$ first receives $g, g^a, g^r X_2, g^{ra} X_2', g^{ra^2} X_2'', Z_2, X_3$. $\mathcal{B}$ chooses random exponents $b, x_e, y_e \in \mathbb{Z}_N$, and sets $PK = (g, g^a, (g^a)^b)$ and $\text{SK} = (x = a, y = ab, x_e, y_e, X_3, Z_2 X_3)$. Clearly, $\mathcal{B}$ does not know a, so parts of the secret key are set implicitly. Also, notice that the exponents x and y are distributed uniformly at random modulo p_1.

$\mathcal{B}$ can now simulate the Sign_B oracle. If $\mathcal{A}$ queries on a message m, $\mathcal{B}$ chooses $\tilde{r}, u_2, v_2, w_2, u_3, v_3, w_3 \in \mathbb{Z}_N$ and signs,

$$\sigma = ((g^r X_2)^{\tilde{r}} Z_2^{u_2} X_3^{u_3},\ (g^{ra} X_2')^{b\tilde{r}} Z_2^{v_2} X_3^{v_3},\ (g^{ra} X_2' (g^{ra^2} X_2'')^{mb})^{\tilde{r}} Z_2^{w_2} X_3^{w_3}).$$

The $\mathbb{G}_{p_1}$ components of the signature will all have the correct correlation, where $x = a$ and $y = ab$ and the randomness is $r\tilde{r}$. Also, it is easy to verify that the $\mathbb{G}_{p_2}$ and the $\mathbb{G}_{p_3}$ components will all appear uniformly distributed.

After it has finished making queries, $\mathcal{A}$ will return a forgery (m^*, σ^*). First $\mathcal{B}$ will check that (m^*, σ^*) verifies, and if not $\mathcal{B}$ will abort. Then, $\mathcal{B}$ must check that (m^*, σ^*) is a Type I forgery. To do this, $\mathcal{B}$ can compute

$$s_2 := \frac{\sigma_2^*}{(\sigma_1^*)^{y_e}},\ s_3 := \frac{\sigma_3^*}{(\sigma_1^*)^{x_e + m^* x_e y_e}}.$$

It then checks if $e(s_2, Z_2) = 1 = e(s_3, Z_2)$. If these checks pass, then it is a Type I forgery. Otherwise, it is a Type II forgery, and $\mathcal{B}$ will abort.

However, with non-negligible probability ϵ, $\mathcal{A}$ will return a Type I forgery. This implies that the forgery has the correct correlation in x_e and y_e modulo p_2. We observe that x_e and y_e are information-theoretically hidden from $\mathcal{A}$. Therefore, $\mathcal{A}$ can only produce a Type I forgery with the correct correlation in x_e and y_e and nonzero $\mathbb{G}_{p_2}$ components with negligible probability, say δ'.

Thus, with probability at least $\epsilon - \delta'$, $\mathcal{B}$ will receive a Type I forgery that has *no* $\mathbb{G}_{p_2}$ components. This would mean creating a signature of the form:

$$\sigma^* = (g^{r'} X_3^{u'},\ g^{r'y} X_3^{v'},\ g^{r'(x + m^* xy)} X_3^{w'}).$$

If $\mathcal{A}$ returns a forgery of this type and $r' \neq 0 \bmod p_1$, then $\mathcal{B}$ can return,

$$\sigma_1^* = g^{r'} X_3^{u'}$$
$$\left(\frac{\sigma_3^*}{(\sigma_2^*)^{1/b}}\right)^{1/m^*b} = \left(\frac{g^{r'(a+m^*a^2b)} X_3^{w'}}{(g^{r'a} X_3^{v'/b})}\right)^{1/m^*b} = g^{r'a^2} X_3^{w''},$$

where $w'' = \frac{1}{m^*b}(w' - \frac{v'}{b})$. This will defeat the Assumption 4.3 challenger.

If $\mathcal{A}$ returns a signature where $r' = 0 \bmod p_1$, then by the definition of the verification algorithm, it will not be accepted. (Recall this was required in the definition of our original verification algorithm.)

Thus, with probability at least $\epsilon - \delta'$, the forgery returned will allow $\mathcal{B}$ to defeat the Assumption 4.3 challenger. Therefore, $\mathcal{B}$ will be successful in defeating the Assumption 4.3 challenger with probability negligibly close to ϵ. □

5 BGOY Signatures

Here we give a public key variant of the BGOY signatures and prove existential unforgeability using our dual form framework. In Appendix A, we show how this base scheme can be built into an identity-based sequential aggregate signature scheme and reduce the security of the aggregate scheme to the security of this base scheme, in the random oracle model. We will also employ the random oracle model in our proof for the base scheme, although this use of the random oracle can be removed (see below for discussion of this).

Our techniques here are quite different than those employed for the BB-derived and CL signature variants, and they reflect the different structure of this scheme. There are some basic commonalities, however: we again employ a bilinear group of order $N = p_1p_2p_3$, and the main structure of the scheme occurs in the $\mathbb{G}_{p_1}$ subgroup. The signatures produced by the Sign_A algorithm contain group elements which are only in $\mathbb{G}_{p_1}$, while the signatures produced by the Sign_B algorithm additionally have components in $\mathbb{G}_{p_3}$. These components in $\mathbb{G}_{p_3}$ are not fully randomized each time and do not occur on all signature elements: they occur only on three signature elements, and the ratio between two of their exponents is the same for all Sign_B signatures. Our forgery types will be defined in terms of the subgroups present on two of the elements in the forgery.

We design our proof to reflect the structure of the scheme, which essentially combines a one-time signature with a mechanism to prevent an attacker from producing new signatures from linear combinations of old signatures in the exponent. In proving dual-oracle invariance, we leverage these structures by first changing the challenge signature from an output of Sign_A to a signature that has components in $\mathbb{G}_{p_2}$, and then changing it to an output of Sign_B. It is crucial to note that as we proceed through this intermediary step, the challenge signature is the *only* signature which has any non-zero components in $\mathbb{G}_{p_2}$. This allows us to argue that as we make this transition, an attacker cannot change from producing Type I forgeries (which do not have $\mathbb{G}_{p_2}$ components on certain elements) to producing forgeries which do have non-zero $\mathbb{G}_{p_2}$ components in the relevant locations. Intuitively, such an attacker would violate the combination of one-time security and inability to combine signatures, since the attacker has only received one signature with $\mathbb{G}_{p_2}$ elements, and it cannot combine this with any other signatures to produce a forgery on a new message. These aspects seem hard to capture when working directly in a prime order rather than composite order group. (We note, however, that the one-time aspect was also implicit in the security proof of the Gentry-Ramzan scheme [42] on which the BGOY scheme was based; however, differences in the schemes prevent capturing it in the same way for the latter.) The techniques here are also quite different from those used in our proofs for CL and BB-derived signatures: here there is no backdoor verification test or pairwise-independence argument.

5.1 The Dual Form Scheme

KeyGen$(\lambda) \rightarrow \text{VK}, \text{SK}$ The key generation algorithm chooses a bilinear group $\mathbb{G}$ of order $N = p_1p_2p_3$. It chooses two random elements $g, k \in \mathbb{G}_{p_1}$, random elements $g_3, g_3^d \in \mathbb{G}_{p_3}$, and random exponents a_1, a_2, b_1, b_2,

$\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{Z}_N$. It also chooses a function $H : \{0,1\}^* \to \mathbb{Z}_N$ which will be modeled as a random oracle. It sets the verification key as

$$\text{VK} := \{N, H, \mathbb{G}, g, k, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}, g^{\alpha_1}, g^{\alpha_2}, g^{\beta_1}, g^{\beta_2}\}$$

and the secret key as

$$\text{SK} := \{N, H, \mathbb{G}, g, k, g^{a_1 a_2}, g^{b_1 b_2}, g^{\alpha_1 \alpha_2}, g^{\beta_1 \beta_2}, g_3, g_3^d\}.$$

Sign$_A(m, \text{SK}) \to \sigma$ The Sign$_A$ algorithm takes in a message $m \in \{0,1\}^*$. It chooses two random exponents $r_1, r_2 \in \mathbb{Z}_N$, and computes:

$$\sigma_1 := g^{a_1 a_2 + b_1 b_2 H(m)} g^{r_1 r_2},\ \sigma_2 := g^{r_1},\ \sigma_3 := g^{r_2},\ \sigma_4 := k^{r_2},\ \sigma_5 := g^{\alpha_1 \alpha_2 + \beta_1 \beta_2 H(m)} k^{r_1 r_2}.$$

It outputs the signature $\sigma := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.

Sign$_B(m, \text{SK}) \to \sigma$ The Sign$_B$ algorithm takes in a message $m \in \{0,1\}^*$. It chooses two random exponents $r_1, r_2, x, y \in \mathbb{Z}_N$, and computes:

$$\sigma_1 := g^{a_1 a_2 + b_1 b_2 H(m)} g^{r_1 r_2} g_3^x,\ \sigma_2 := g^{r_1} g_3^y,\ \sigma_3 := g^{r_2},\ \sigma_4 := k^{r_2},\ \sigma_5 := g^{\alpha_1 \alpha_2 + \beta_1 \beta_2 H(m)} k^{r_1 r_2} (g_3^d)^x.$$

It outputs the signature $\sigma := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.

Verify$(m, \sigma, \text{VK}) \to \{True, False\}$ The verification algorithm first checks that:

$$e(\sigma_1, g) = e(g^{a_1}, g^{a_2}) e(g^{b_1}, g^{b_2})^{H(m)} e(\sigma_2, \sigma_3).$$

It also checks that:

$$e(\sigma_5, g) = e(g^{\alpha_1}, g^{\alpha_2}) e(g^{\beta_1}, g^{\beta_2})^{H(m)} e(\sigma_2, \sigma_4).$$

Finally, it checks that:

$$e(g, \sigma_4) = e(k, \sigma_3).$$

If all of these checks pass, it outputs "True." Otherwise, it outputs "False."

We note that the use of the random oracle H to hash messages in $\{0,1\}^*$ to elements in $\mathbb{Z}_N$ in this public key scheme that forms the base of our identity-based sequential aggregate signatures is not necessary, and can be replaced in the following way. Instead of using $g^{a_1 a_2 + H(m) b_1 b_2}$, we can assume our messages are n-bit strings (denoted $m_1 m_2 \ldots m_n$) and use $g^{a_0 b_0} \prod_{i=1}^n g^{m_i a_i b_i}$. Here, $g^{a_0}, \ldots, g^{a_n}, g^{b_0}, \ldots, g^{b_n}$ will be in the public verification key. In the proof, instead of guessing which random oracle query corresponds to the challenge message, the simulator will guess a bit which differs between the challenge message and the message that will be used in the forgery. This guess will be correct with non-negligible probability. However, the use of the random oracle model to prove security for the full identity-based sequential aggregate scheme is still required. Removing the random oracle model altogether remains an open problem.

Forgery Classes We will divide the forgery types based on whether they have any $\mathbb{G}_{p_2}$ or $\mathbb{G}_{p_3}$ components on σ_1 or σ_5. We let $z_2 \in \mathbb{Z}_N$ denote the exponent represented by the tuple $(0 \bmod p_1, 1 \bmod p_2, 0 \bmod p_3)$, and we let $z_3 \in \mathbb{Z}_N$ denote the exponent represented by the tuple $(0 \bmod p_1, 0 \bmod p_2, 1 \bmod p_3)$. Then we can define the forgery classes as follows:

Type I. $\mathcal{V}_I = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_1^*)^{z_2} = 1, (\sigma_1^*)^{z_3} = 1 \text{ and } (\sigma_5^*)^{z_2} = 1, (\sigma_5^*)^{z_3} = 1\}$

Type II. $\mathcal{V}_{II} = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_1^*)^{z_2} \neq 1 \text{ or } (\sigma_5^*)^{z_2} \neq 1 \text{ or } (\sigma_1^*)^{z_3} \neq 1 \text{ or } (\sigma_5^*)^{z_3} \neq 1\}$

In other words, Type I forgeries have $\sigma_1^*, \sigma_5^* \in \mathbb{G}_{p_1}$, while Type II forgeries have a non-zero component in $\mathbb{G}_{p_3}$ or $\mathbb{G}_{p_2}$ on at least one of these terms. We note that these types are disjoint and exhaustive.

5.2 Complexity Assumptions

We now state the static complexity assumptions which we will use to prove security of our scheme. The first is a weaker version of Assumption 4.1 stated in Section 4.2. (Here, we do not give out the X_1X_2 term.) The third is another instance of the General Subgroup Decision assumption [13], and was previously used in [57] (where it was also proven to hold in the generic group model, assuming it is hard to find a non-trivial factor of the group order). The second and fourth assumptions are new, and we prove in Appendix B that they hold in the generic group model, assuming it is hard to find a non-trivial factor of the group order, N.

Assumption 5.1. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$g \xleftarrow{R} \mathbb{G}_{p_1}, g_3 \xleftarrow{R} \mathbb{G}_{p_3}$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g_3)$$

$$T_1 \xleftarrow{R} \mathbb{G}_{p_1p_2}, T_2 \xleftarrow{R} \mathbb{G}_{p_1}$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 5.1 to be:

$$Adv^{5.1}_{\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 4. We say that $\mathcal{G}$ satisfies Assumption 5.1 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{5.1}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 5.2. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, g^{c_1}, g^{c_2}, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, X_2 \xleftarrow{R} \mathbb{G}_{p_2}, g_3 \xleftarrow{R} \mathbb{G}_{p_3},$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g^{c_1}, g^{c_2}, g_3, X_1X_2)$$

Given D, we assume it is hard to produce group elements $s_1, s_2, s_3 \in \mathbb{G}$ such that $e(s_1, g)/e(s_2, s_3) = e(g^{c_1}, g^{c_2})$ *and* s_1 has a non-zero component in $\mathbb{G}_{p_2}$. More formally, we let $z_2 \in \mathbb{Z}_N$ denote the exponent represented by the tuple $(0 \bmod p_1, 1 \bmod p_2, 0 \bmod p_3)$ and we define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 5.2 to be:

$$Adv^{5.2}_{\mathcal{A}}(\lambda) := Pr[\mathcal{A}(D) = (s_1, s_2, s_3) \text{ s.t. } e(s_1, g)/e(s_2, s_3) = e(g^{c_1}, g^{c_2}) \text{ and } s_1^{z_2} \neq 1].$$

Definition 5. We say that $\mathcal{G}$ satisfies Assumption 5.2 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{5.2}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.

We note that without the requirement of a non-zero $\mathbb{G}_{p_2}$ component on s_1, this is easy – set $s_1 = 1, s_2 = g^{c_1}, s_3 = g^{-c_2}$.

Perhaps surprisingly, we are able to use the above simple assumption to in some sense capture both the "one-time" security of the BGOY scheme and its mechanism to prevent taking linear combinations of old signatures to prevent forgeries. We discuss this more in Section 5.3 where the assumption is used.

Assumption 5.3. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1p_2p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, Y_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3, Y_3 \xleftarrow{R} \mathbb{G}_{p_3}$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, X_1X_3, Y_2Y_3)$$

$$T_1 \xleftarrow{R} \mathbb{G}_{p_1p_3}, T_2 \xleftarrow{R} \mathbb{G}_{p_1p_2}$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 5.3 to be:

$$Adv^{5.3}_{\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 6. We say that $\mathcal{G}$ satisfies Assumption 5.3 if for any polynomial time algorithm, $\mathcal{A}$, $Adv_{\mathcal{A}}^{5.3}(\lambda)$ is a negligible function of λ.

Assumption 5.4. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, g^{c_1}, g^{c_2} \xleftarrow{R} \mathbb{G}_{p_1}, g_3, X_3 \xleftarrow{R} \mathbb{G}_{p_3}$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g^{c_1}, g^{c_2}, g^{c_1 c_2} X_3, g_3)$$

Given D, we assume it is hard to compute $g^{c_1 c_2}$. More formally, We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 5.4 to be:

$$Adv_{\mathcal{A}}^{5.4}(\lambda) := Pr[\mathcal{A}(D) = g^{c_1 c_2}].$$

Definition 7. We say that $\mathcal{G}$ satisfies Assumption 5.4 if for any polynomial time algorithm, $\mathcal{A}$, $Adv_{\mathcal{A}}^{5.4}(\lambda)$ is a negligible function of λ.

For the identity-based, aggregate version of our scheme, we will need one more assumption which is again an instance of the General Subgroup Decision assumption. This is weaker than the Assumption 1 appearing in [57].

Assumption 5.5. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$D = (N, \mathbb{G}, \mathbb{G}_T, e)$$

$$T_1 \xleftarrow{R} \mathbb{G}_{p_1}, T_2 \xleftarrow{R} \mathbb{G}$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 5.3 to be:

$$Adv_{\mathcal{A}}^{5.3}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 8. We say that $\mathcal{G}$ satisfies Assumption 5.5 if for any polynomial time algorithm, $\mathcal{A}$, $Adv_{\mathcal{A}}^{5.5}(\lambda)$ is a negligible function of λ.

5.3 Proof of Security

We now prove that our scheme has the A-I matching, dual-oracle invariance, and B-II matching properties, when H is modeled as a random oracle.

Lemma 5.1. Under Assumption 5.1, our scheme is A-I matching.

Proof. We assume there exists a PPT attacker $\mathcal{A}$ which violates the A-I matching property. We will use $\mathcal{A}$ to create a PPT algorithm $\mathcal{B}$ which breaks Assumption 5.1 (either the way it is written, or with the roles of p_2 and p_3 interchanged). $\mathcal{B}$ receives g, T (we will not need the g_3 term here, so we ignore it). $\mathcal{B}$ chooses random exponents $a_1, a_2, b_1, b_2, \alpha_1, \alpha_2, \beta_1, \beta_2, v \in \mathbb{Z}_N$. It sets the verification key as:

$$\text{VK} := \{N, G, g, k := g^v, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}, g^{\alpha_1}, g^{\alpha_2}, g^{\beta_1}, g^{\beta_2}\}.$$

For this portion of the proof, the random oracle is not really needed, so we can think of H as a publicly available function here or just have $\mathcal{B}$ choose random responses to the random oracle queries and remember its answers to maintain consistency. It is also easy for the simulator to respond to $\mathcal{A}$'s signing queries by calling the Sign_A algorithm, since it knows all of the secret key needed in the Sign_A algorithm.

With non-negligible probability, $\mathcal{A}$ produces a forgery $(\sigma_1^*, \ldots, \sigma_5^*), m^*$ that is not of Type I (i.e it is of Type II). This means that either σ_1^* or σ_5^* has a non-zero component in $\mathbb{G}_{p_2}$ or $\mathbb{G}_{p_3}$. We let h^* denote $H(m^*)$. Whenever it receives a valid forgery, $\mathcal{B}$ computes:

$$S := (\sigma_1^* \cdot g^{-a_1 a_2 - h^* b_1 b_2})^v \cdot (\sigma_5^* \cdot g^{-\alpha_1 \alpha_2 - h^* \beta_1 \beta_2})^{-1}.$$

Since the forgery verifies correctly, this S has a zero $\mathbb{G}_{p_1}$ component. With non-negligible probability, S will have a non-zero $\mathbb{G}_{p_2}$ or $\mathbb{G}_{p_3}$ component, since there is a non-zero $\mathbb{G}_{p_2}$ or $\mathbb{G}_{p_3}$ component on at least one of σ_1^* and σ_5^* whenever $\mathcal{A}$ produces a forgery that is not of Type I. We note that two non-zero $\mathbb{G}_{p_2}$ or $\mathbb{G}_{p_3}$ components will cancel out in the computation of S only with negligible probability, because the value of v modulo p_2 and p_3 is information-theoretically hidden from $\mathcal{A}$.

Either S will have a non-zero $\mathbb{G}_{p_2}$ component with non-negligible probability, or S will have a non-zero $\mathbb{G}_{p_3}$ component with non-negligible probability. If the former happens, we will break Assumption 5.1 as it is written, by having $\mathcal{B}$ compute $e(T, S)$. If the result is 1, then $\mathcal{B}$ guesses randomly. If the result if not 1, then $\mathcal{B}$ knows that $T \in \mathbb{G}_{p_1 p_2}$. Thus, it achieves non-negligible advantage against Assumption 5.1. If S instead has a $\mathbb{G}_{p_3}$ component with non-negligible probability, then we break Assumption 5.1 with the roles of p_2 and p_3 interchanged in the same fashion. □

To prove dual-oracle invariance, we first define an intermediary security game. We let Game_A denote the dual-oracle security game where the challenge signature is produced by calling the Sign_A algorithm, and we let Game_B denote the dual-oracle security game where the challenge signature is produced by calling the Sign_B algorithm. We now define:

Game$_{Mid}$ This is like Game_A and Game_B except that the challenge signature is distributed as:

$$\sigma_1 := g^{a_1 a_2 + b_1 b_2 H(m)} g^{r_1 r_2} R_2,\ \sigma_2 := g^{r_1} R_2',\ \sigma_3 := g^{r_2},\ \sigma_4 := k^{r_2},\ \sigma_5 := g^{\alpha_1 \alpha_2 + \beta_1 \beta_2 H(m)} k^{r_1 r_2} R_2'',$$

where R_2, R_2', R_2'' are random elements of $\mathbb{G}_{p_2}$.

We now prove:

Lemma 5.2. Under Assumption 5.1, for any PPT $\mathcal{A}$, its probability of producing a forgery which has a non-zero component in $\mathbb{G}_{p_3}$ on either σ_1^* or σ_5^* is only negligibly different between Game_A and Game_{Mid}. Also, its total probability of producing a forgery is only negligibly different between Game_A and Game_{Mid}.

Proof. We assume there exists a PPT attacker $\mathcal{A}$ which achieves a non-negligible difference either in its probability of producing a forgery at all or in its probability of producing a forgery which has non-zero component in $\mathbb{G}_{p_3}$ between Game_A and Game_{Mid}. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 5.1. $\mathcal{B}$ receives g, g_3, T, where T is either in $\mathbb{G}_{p_1}$ or $\mathbb{G}_{p_1 p_2}$. It will simulate either Game_A or Game_{Mid} with $\mathcal{A}$, depending on the value of T.

It chooses random exponents $a_1, a_2, b_1, b_2, \alpha_1, \alpha_2, \beta_1, \beta_2, v \in \mathbb{Z}_N$ and forms the verification key as:

$$\text{VK} : \{N, G, g, k := g^v, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}, g^{\alpha_1}, g^{\alpha_2}, g^{\beta_1}, g^{\beta_2}\}.$$

As in the proof of the previous lemma, modeling H as a random oracle is not needed here, so we can think of H as a public function for this step, or simply imagine that $\mathcal{B}$ chooses random responses to the random oracle queries and remembers its answers for consistency. Since $\mathcal{B}$ knows all of the $\mathbb{G}_{p_1}$ elements in the secret key, it can call the Sign_A algorithm to produce signatures. It also chooses $d \in \mathbb{Z}_N$ randomly and sets the rest of the secret key as g_3, g_3^d so it can call the Sign_B algorithm to produce signatures as well.

To produce the challenge signature on a message m with $H(m)$ denoted by h, the simulator chooses a random $r_2 \in \mathbb{Z}_N$ and implicitly sets g^{r_1} to be the $\mathbb{G}_{p_1}$ part of T. It forms the challenge signature as:

$$\sigma_1 = g^{a_1 a_2 + h b_1 b_2} T^{r_2},\ \sigma_2 = T,\ \sigma_3 = g^{r_2},\ \sigma_4 = k^{r_2},\ \sigma_5 = g^{\alpha_1 \alpha_2 + h \beta_1 \beta_2} T^{v r_2}.$$

If $T \in \mathbb{G}_{p_1}$, this is identically distributed to the outcome of calling the Sign_A algorithm, so $\mathcal{B}$ has properly simulated Game_A in this case. If $T \in \mathbb{G}_{p_1 p_2}$, then the $\mathbb{G}_{p_2}$ components on σ_1, σ_2, and σ_5 are uniformly random here, since r_2 and v are random modulo p_2. In this case, $\mathcal{B}$ has properly simulated Game_{Mid}.

$\mathcal{B}$ can always perform the verification test to check whether or not $\mathcal{A}$ has produced a valid forgery. Now, when $\mathcal{A}$ has produced a forgery, $\mathcal{B}$ tests if σ_1^*, σ_5^* have any $\mathbb{G}_{p_3}$ components by pairing each with g_3. We note that one of σ_1^*, σ_5^* has a non-zero $\mathbb{G}_{p_3}$ component if and only if one of the pairings $e(\sigma_1^*, g_3)$, $e(\sigma_5^*, g_3)$ does not yield the group identity. Thus, $\mathcal{B}$ can observe when $\mathcal{A}$ is forging, and also when it is producing a forgery with with a non-zero $\mathbb{G}_{p_3}$ component on either of σ_1^*, σ_5^*. Thus, $\mathcal{B}$ can use the non-negligible difference in $\mathcal{A}$'s probability of satisfying whichever one of these conditions to achieve a non-negligible advantage against Assumption 5.1. □

We next prove dual oracle invariance with respect to forgeries $(\sigma_1^*, \ldots, \sigma_5^*)$ having a non-zero $\mathbb{G}_{p_2}$ component on σ_1^* or σ_5^*. This is where we use Assumption 5.2. As we mentioned, this will in some sense capture both the "one-time" security of the BGOY scheme as well as its mechanism to prevent the adversary from taking linear combinations of old signatures to create forgeries.

We give a bit more intuition about how this is possible. For simplicity, let's ignore the last two elements of the signature (i.e., corresponding exactly to the original scheme [15]) and and consider forgeries having non-zero $\mathbb{G}_{p_2}$ component on σ_1^*; the remaining signature elements are not used in an essential way here. Intuitively, we set up the simulation by embedding g^{c_1}, g^{c_2} in the assumption into $g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}$ in the verification key of the scheme so that, to answer a signing query, the simulator produces $(\sigma_1, \sigma_2, \sigma_3)$ where the $g^{c_1 c_2}$ term embedded in σ_1 is cancelled out by g^{c_1}, g^{c_2} embedded in the σ_2, σ_3 components. However, for exactly *one* signature, the $g^{c_1 c_2}$ term embedded in σ_1 already cancels out, which allows the simulator to embed $X_1 X_2$ into $g^{r_1 r_2}$ instead. Thus, the simulator can produce *one* signature with $X_1 X_2$ embedded in σ_1, but (necessarily) no $g^{c_1 c_2}$ term. However, observe that if the adversary could take a linear combination of both kinds of signatures to produce a new one (or create one from scratch), this would have both $g^{c_1 c_2}$ *and* $X_1 X_2$ embedded in the first component, which is exactly the type of forgery we consider and could be used to contradict the assumption.

Lemma 5.3. Under Assumption 5.2, for any PPT $\mathcal{A}$, its probability of producing a forgery which has a non-zero component in $\mathbb{G}_{p_2}$ on either σ_1^* or σ_5^* is negligible in Game_{Mid}.

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who produces such a forgery with non-negligible probability in Game_{Mid}. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 5.2. $\mathcal{B}$ is given $g, g^{c_1}, g^{c_2}, g_3, X_1 X_2$. $\mathcal{B}$ will simulate Game_{Mid}. Here, we will use that H is modeled as a random oracle. $\mathcal{B}$ first chooses a random value $h \in \mathbb{Z}_N$, which it wants to be $H(m)$ for the challenge message m. However, $\mathcal{B}$ does not know what m will be at the start of the game, so it will guess which oracle query corresponds to the challenge message m and respond with h to this query ($\mathcal{B}$ artificially inserts a random oracle query on the challenge message once the challenge message is received in order to ensure that it queried at least once). It responds to all other random oracle queries by choosing a random value in $\mathbb{Z}_N$ (unless the query is repeated, in which case, it repeats its previous response). Since the attacker $\mathcal{A}$ can only make a polynomial number of random oracle queries, $\mathcal{B}$'s guess will be correct with non-negligible probability. When the guess is incorrect, $\mathcal{B}$ will abort.

$\mathcal{B}$ chooses random exponents $b_1', b_2', \alpha_1', \alpha_2', \beta_1', \beta_2', v, \in \mathbb{Z}_N$. It sets $k = g^v$ and implicitly sets $a_1 = c_1, a_2 = hc_2, b_1 = c_1 + b_1', b_2 = -c_2 + b_2', \alpha_1 = vc_1 + \alpha_1', \alpha_2 = hc_2 + \alpha_2', \beta_1 = vc_1 + \beta_1', \beta_2 = -c_2 + \beta_2'$. It forms the verification key as:

$$\text{VK} : \{N, G, g, k := g^v, g^{a_1} = g^{c_1}, g^{a_2} = (g^{c_2})^h, g^{b_1} = g^{c_1} g^{b_1'}, g^{b_2} = (g^{c_2})^{-1} g^{b_1'},$$

$$g^{\alpha_1} = (g^{c_1})^v g^{\alpha_1'}, g^{\alpha_2} = (g^{c_2})^h g^{\alpha_2'}, g^{\beta_1} = (g^{c_1})^v g^{\beta_1'}, g^{\beta_2} = (g^{c_2})^{-1} g^{\beta_2'}\}.$$

The parameters $b_1', b_2', \alpha_1', \alpha_2', \beta_1', \beta_2'$ here are just used for re-randomization so that the parameters are properly distributed. The important thing to note is that $a_1 a_2 + h b_1 b_2$ and $\alpha_1 \alpha_2 + h \beta_1 \beta_2$ will have no occurrences of $c_1 c_2$, since these will cancel out.

To respond to a Sign_A query on a message m', $\mathcal{B}$ proceeds as follows. We let h' denote $H(m')$. $\mathcal{B}$ chooses random values $r_1', r_2' \in \mathbb{Z}_N$ and implicitly sets $r_1 = c_1(h - h') + r_1', r_2 = -c_2 + r_2'$. Since r_1', r_2' are random,

these are well-distributed. The term $-c_1c_2(h-h')$ in r_1r_2 will be used to the cancel the c_1c_2 terms arising from $a_1a_2 + h'b_1b_2$ and $\alpha_1\alpha_2 + h'\beta_1\beta_2$. We note that

$$a_1a_2 + h'b_1b_2 = (h-h')c_1c_2 - h'b_1'c_2 + h'b_2'c_1 + h'b_1'b_2',$$

$$\alpha_1\alpha_2 + h'\beta_1\beta_2 = v(h-h')c_1c_2 + vc_1\alpha_2' + \alpha_1'hc_2 + \alpha_1'\alpha_2' + h'\beta_1'\beta_2' - h'\beta_1'c_2 + h'vc_1\beta_2'.$$

$$r_1r_2 = -(h-h')c_1c_2 - r_1'c_2 + r_1'r_2' + c_1(h-h')r_2'.$$

The signature is formed as:

$$\sigma_1 = (g^{c_2})^{-h'b_1'-r_1'}(g^{c_1})^{h'b_2'+(h-h')r_2'}g^{h'b_1'b_2'+r_1'r_2'},\ \sigma_2 = (g^{c_1})^{h-h'}g^{r_1'},\ \sigma_3 = (g^{c_2})^{-1}g^{r_2'},$$

$$\sigma_4 = (g^{c_2})^{-v}g^{vr_2'},\ \sigma_5 = (g^{c_1})^{v\alpha_2'+vh'\beta_2'+v(h-h')r_2'}(g^{c_2})^{h\alpha_1'-h'\beta_1'-vr_1'}g^{\alpha_1'\alpha_2'+h'\beta_1'\beta_2'+vr_1'r_2'}.$$

To respond to Sign_B queries, the simulator chooses a random $d \in \mathbb{Z}_N$ and computes g_3^d. For each query, it first forms a signature $(\sigma_1', \ldots, \sigma_5')$ in $\mathbb{G}_{p_1}$ as above, and adds on $\mathbb{G}_{p_3}$ components by choosing random values $x, y \in \mathbb{Z}_N$ and computing:

$$\sigma_1 := \sigma_1' \cdot g_3^x,\ \sigma_2 := \sigma_2' \cdot g_3^y,\ \sigma_3 := \sigma_3', \sigma_4 := \sigma_4', \sigma_5 := \sigma_5' \cdot (g_3^d)^x.$$

It returns $(\sigma_1, \ldots, \sigma_5)$, which is well-distributed as an output of the Sign_B algorithm.

If the simulator has guessed correctly so that the challenge message m satisfies $H(m) = h$, then the simulator creates the challenge signature as follows (otherwise, it aborts). It leverages the fact that the c_1c_2 terms in $a_1a_2 + hb_1b_2$ and $\alpha_1\alpha_2 + h\beta_1\beta_2$ cancel, so it no longer needs to embed c_1 and c_2 into the r_1 and r_2 values. Instead, it chooses random exponents $r_1', r_2 \in \mathbb{Z}_N$ and implicitly sets g^{r_1} to be $X_1^{r_1'}$. It forms the signature as:

$$\sigma_1 = (g^{c_2})^{-hb_1'}(g^{c_1})^{hb_2'}g^{hb_1'b_2'}(X_1X_2)^{r_1'r_2},\ \sigma_2 = (X_1X_2)^{r_1'},\ \sigma_3 = g^{r_2},$$

$$\sigma_4 = k^{r_2},\ \sigma_5 = (g^{c_1})^{v\alpha_2'+hv\beta_2'}(g^{c_2})^{h\alpha_1'-h\beta_1'}g^{\alpha_1'\alpha_2'+h\beta_1'\beta_2'}(X_1X_2)^{vr_1'r_2}.$$

Note that since r_1', r_2, and v are all random modulo p_2, the terms $X_2^{r_1'}$, $X_2^{r_1'r_2}$ and $X_2^{vr_1'r_2}$ are distributed randomly in $\mathbb{G}_{p_2}$. So $\mathcal{B}$ has properly simulated Game_{Mid}. It is crucial to note here that the simulator can only make this *one* signature with $\mathbb{G}_{p_2}$ components - this is a reflection of both the underlying one-time security and the mechanism which prevents combining signatures. By the cancelation technique, the simulator can only sign this one message with p_2 components, and it cannot combine this with other signatures.

Since the event that $\mathcal{A}$ produces a forgery which has a non-zero $\mathbb{G}_{p_2}$ component on σ_1^* or σ_5^* is independent of the event that $\mathcal{B}$ aborts, we have that with non-negligible probability, $\mathcal{B}$ does not abort and $\mathcal{A}$ produces such a forgery (m^*, σ^*). We let h^* denote $H(m^*)$. When $\mathcal{B}$ receives a forgery, it computes:

$$\sigma_1' := \sigma_1^* \cdot (g^{c_2})^{h^*b_1'}(g^{c_1})^{-h^*b_2'}g^{-h^*b_1'b_2'},$$

$$\sigma_5' := \sigma_5^* \cdot (g^{c_1})^{-v\alpha_2'-vh^*\beta_2'}(g^{c_2})^{-h\alpha_1'+h^*\beta_1'}g^{-\alpha_1'\alpha_2'-h^*\beta_1'\beta_2'}.$$

This strips off the extra randomizing terms, and the remaining $\mathbb{G}_{p_1}$ parts of the forgery must be of the form:

$$\sigma_1' = (g^{c_1c_2})^{h-h^*}g^{r_1r_2},\ \sigma_2^* = g^{r_1},\ \sigma_3^* = g^{r_2},\ \sigma_4^* = k^{r_2},\ \sigma_5' = (g^{c_1c_2})^{v(h-h^*)}k^{r_1r_2}$$

for some $r_1, r_2 \in \mathbb{Z}_N$.

Now, if σ_1' also has a non-zero $\mathbb{G}_{p_2}$ component, then $\mathcal{B}$ can compute:

$$s_1 := (\sigma_1')^{(h-h^*)^{-1}},\ s_2 := (\sigma_2^*)^{(h-h^*)^{-1}},\ s_3 = \sigma_3^*,$$

and we then have that

$$e(g, s_1)/e(s_2, s_3) = e(g^{c_1}, g^{c_2}),$$

and s_1 has a non-zero $\mathbb{G}_{p_2}$ component, so this breaks Assumption 5.2. We note that either s_2 or s_3 might also have a non-zero $\mathbb{G}_{p_2}$ component, but both cannot have non-zero $\mathbb{G}_{p_2}$ components, since the original forgery (m^*, σ^*) verifies.

If instead σ_5' has a non-zero $\mathbb{G}_{p_2}$ component, then $\mathcal{B}$ can compute:

$$s_1 : (\sigma_5')^{(h-h^*)^{-1}v^{-1}},\ s_2 := (\sigma_2^*)^{(h-h^*)^{-1}},\ s_3 = \sigma_3^*,$$

which will satisfy

$$e(g, s_1)/e(s_2, s_3) = e(g^{c_1}, g^{c_2}),$$

and s_1 has non-zero $\mathbb{G}_{p_2}$ component. Hence, since at least one of these two events must occur with non-negligible probability, $\mathcal{B}$ can break Assumption 5.2 with non-negligible probability. □

By Lemma 5.2, we see that a PPT attacker must have the same overall probability of forging in Game_A versus Game_{Mid}, up to a negligible difference. By A-I matching, it can only produce Type I forgeries in Game_A with non-negligible probability. By lemmas 5.2 and 5.3, we see that a PPT attacker cannot start producing forgeries of Type II with non-negligible probability as we move from Game_A to Game_{Mid}. This allows us to conclude that any PPT attacker's probability of producing a Type I forgery must differ only negligibly between Game_A and Game_{Mid}. We now show that the same holds for the transition from Game_{Mid} to Game_B, and dual-oracle invariance then follows.

Lemma 5.4. Under Assumption 5.3, for any PPT attacker, its probability of producing a Type I forgery is only negligibly different between Game_{Mid} and Game_B.

Proof. We assume there exists a PPT attacker $\mathcal{A}$ which achieves a non-negligibly different probability of producing a Type I forgery between Game_{Mid} and Game_B. We will use this to create a PPT algorithm $\mathcal{B}$ which breaks Assumption 5.3. $\mathcal{B}$ is given g, X_1X_3, Y_2Y_3, T, and its task is to determine if $T \in \mathbb{G}_{p_1p_3}$ or $T \in \mathbb{G}_{p_1p_2}$. It will simulate either Game_{Mid} or Game_B with $\mathcal{A}$, depending on the value of T. It will use the term Y_2Y_3 to test if $\mathcal{A}$ is produces a Type I forgery.

$\mathcal{B}$ chooses random exponents $a_1, a_2, b_1, b_2, \alpha_1, \alpha_2, \beta_1, \beta_2, v \in \mathbb{Z}_N$ and sets the verification key as:

$$\text{VK} : \{N, G, g, k := g^v, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}, g^{\alpha_1}, g^{\alpha_2}, g^{\beta_1}, g^{\beta_2}\}.$$

For this stage of the proof, modeling H as a random oracle is not needed, so we can again think of H as a public function for this step, or simply imagine that $\mathcal{B}$ chooses random responses to the random oracle queries and remembers its answers for consistency. Since $\mathcal{B}$ knows all of the $\mathbb{G}_{p_1}$ elements in the secret key, it can call the Sign_A algorithm to produce signatures.

To produce signatures which are properly distributed as outputs of the Sign_B algorithm, $\mathcal{B}$ proceeds as follows. To sign a message m' with $H(m') = h'$, it chooses random exponents $r_1', r_2 \in Z_N$ and implicitly sets g^{r_1} equal to $X_1^{r_1'}$. It will implicitly set the value d in the Sign_B algorithm equal to the value of v modulo p_3 (this is uncorrelated from the verification key, which only involves the value of v modulo p_1). It forms the signature as:

$$\sigma_1 = g^{a_1a_2+h'b_1b_2}(X_1X_3)^{r_1'r_2},\ \sigma_2 = (X_1X_3)^{r_1'},\ \sigma_3 = g^{r_2},\ \sigma_4 = k^{r_2},\ \sigma_5 = g^{\alpha_1\alpha_2+h'\beta_1\beta_2}(X_1X_3)^{vr_1'r_2}.$$

This produces signatures identically distributed to outputs of the Sign_B algorithm.

To produce the challenge signature for a message m with $H(m) = h$, $\mathcal{B}$ chooses a random exponent $r_2 \in \mathbb{Z}_N$ and implicitly sets g^{r_1} equal to the $\mathbb{G}_{p_1}$ component of T. It forms the signature as:

$$\sigma_1 = g^{a_1a_2+hb_1b_2}T^{r_2},\ \sigma_2 = T,\ \sigma_3 = g^{r_2},\ \sigma_4 = k^{r_2},\ \sigma_5 = g^{\alpha_1\alpha_2+h\beta_1\beta_2}T^{vr_2}.$$

If $T \in \mathbb{G}_{p_1p_3}$, this produces a signature identically distributed to an output of the Sign_B algorithm. If $T \in \mathbb{G}_{p_1p_2}$, this produces a signature whose $\mathbb{G}_{p_2}$ components on σ_1, σ_2, and σ_5 are uniformly random, since r_2 and v are random modulo p_2. (Note that the value of v modulo p_2 is not involved in either the verification key or the Sign_B signatures.) Thus, if $T \in \mathbb{G}_{p_1p_3}$, then $\mathcal{B}$ has properly simulated Game_B, and if $T \in \mathbb{G}_{p_1p_2}$,

then $\mathcal{B}$ has properly simulated Game_{Mid}. When $\mathcal{B}$ receives a forgery (m^*, σ^*) from $\mathcal{A}$, it can test whether the forgery is of Type I or not by computing $e(Y_2Y_3, \sigma_1^*)$ and $e(Y_2Y_3, \sigma_5^*)$. The forgery is of Type I if and only if both of these pairings produce the identity element. Hence, $\mathcal{B}$ can leverage the non-negligible difference in $\mathcal{A}$'s probability of producing a Type I forgery to achieve non-negligible advantage against Assumption 5.3. □

This completes our proof of dual-oracle invariance. Finally, we prove our scheme is B-II matching.

Lemma 5.5. Under Assumption 5.4, our scheme is B-II matching.

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who produces a non-Type II forgery (i.e. a Type I forgery) with non-negligible probability when it is only given signatures produced by the Sign_B algorithm. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 5.4. $\mathcal{B}$ is given $g, g^{c_1}, g^{c_2}, g^{c_1c_2}X_3, g_3$. It chooses random exponents $b_1, b_2, \alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{Z}_N$. It implicitly sets $a_1 = c_1$ and $a_2 = c_2$. It forms the verification key as:

$$\text{VK} : \{N, G, g, k := g^v, g^{a_1} = g^{c_1}, g^{a_2} = g^{c_2}, g^{b_1}, g^{b_2}, g^{\alpha_1}, g^{\alpha_2}, g^{\beta_1}, g^{\beta_2}\}.$$

As for many of the previous steps,, modeling H as a random oracle is not necessary here, so we can again think of H as a public function for this step, or simply imagine that $\mathcal{B}$ chooses random responses to the random oracle queries and remembers its answers for consistency.

To produce a signature for a message m with $H(m) = h$, $\mathcal{B}$ proceeds as follows. It chooses a random value $d \in \mathbb{Z}_N$ which will be fixed for all signatures. For each signature, it chooses fresh random values r_1', r_2. It will implicitly set g^{r_1} as $(g^{c_1c_2})^{d(v-d)^{-1}r_2^{-1}} g^{r_1'}$. (Note the $v - d$ and r_2 are invertible modulo N with all but negligible probability.) Since r_1', r_2 are random, this makes r_1, r_2 also randomly distributed, as required. It also chooses random values $t, y \in \mathbb{Z}_N$. It computes the signature as:

$$\sigma_1 = g^{hb_1b_2}(g^{c_1c_2}X_3)^{1+d(v-d)^{-1}} g^{r_1'r_2} g_3^t,\ \sigma_2 = (g^{c_1c_2}X_3)^{d(v-d)^{-1}r_2^{-1}} g^{r_1'} g_3^y,\ \sigma_3 = g^{r_2},$$

$$\sigma_4 = k^{r_2},\ \sigma_5 = g^{\alpha_1\alpha_2 + h\beta_1\beta_2}(g^{c_1c_2}X_3)^{vd(v-d)^{-1}} g^{vr_1'r_2} g_3^{dt}.$$

To see that the $\mathbb{G}_{p_3}$ parts are well distributed, note that the ratio between the exponents of X_3 is equal to d:

$$d(1 + d/(v - d)) = vd/(v - d).$$

This equality is the reason we have used the value $d/(v - d)$. To see how we arrived at this choice of parameters, let the variable r denote the coefficient of $g^{c_1c_2}X_3$ inside $g^{r_1r_2}$. To make the $\mathbb{G}_{p_1}$ parts well-distributed and make the ratio between the X_3 exponents equal to d simultaneously, we need r to satisfy:

$$d(1 + r) = vr.$$

Solving for r, we obtain $r = d/(v - d)$, which is the value we have used above. The extra terms g_3^t and g_3^{dt} ensure that the $\mathbb{G}_{p_3}$ components on σ_1 and σ_5 are random up to having this fixed ratio, and the g_3^y term ensures that the $\mathbb{G}_{p_3}$ component of σ_2 is always uniformly random. Thus, $\mathcal{B}$ is producing signatures which are identically distributed to outputs of Sign_B.

When $\mathcal{A}$ produces a forgery (m^*, σ^*) where $H(m^*) = h^*$, $\mathcal{B}$ computes the quantities:

$$\sigma_1' := \sigma_1^* \cdot g^{-h^*b_1b_2},\ \sigma_5' := \sigma_5^* \cdot g^{-\alpha_1\alpha_2 - h^*\beta_1\beta_2}.$$

Then, $\mathcal{B}$ computes:

$$((\sigma_1')^v/(\sigma_5'))^{v^{-1}}.$$

If $\mathcal{A}$ has produced Type I forgery (i.e. $\sigma_1^*, \sigma_5^* \in \mathbb{G}_{p_1}$), then this quantity is equal to $g^{c_1c_2}$. Since this happens with non-negligible probability, $\mathcal{B}$ breaks Assumption 5.4. □

References

[1] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO*, pages 209–236, 2010.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[3] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. TCC 2012, 2012. `http://eprint.iacr.org/`.

[4] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In *ACM Conference on Computer and Communications Security*, pages 473–484, 2010.

[5] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[6] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable rfid tags via insubvertible encryption. In *ACM Conference on Computer and Communications Security*, pages 92–101, 2005.

[7] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/`.

[8] Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *WPES*, pages 40–46, 2005.

[9] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols Workshop*, pages 20–42, 2004.

[10] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interative zero knowledge proofs. In *CRYPTO*, pages 194–211, 1989.

[11] Mihir Bellare and Silvio Micali. How to sign given any trapdoor function. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403, pages 200–215, 1990.

[12] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73, 1993.

[13] Mihir Bellare, Brent Waters, and Scott Yilek. Identity-based encryption secure against selective opening attack. In *TCC*, pages 235–252, 2011.

[14] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In *ACM Conference on Computer and Communications Security*, pages 276–285, 2007.

[15] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. Cryptology ePrint Archive, Report 2007/438, 2007. `http://eprint.iacr.org/`.

[16] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '04*, volume 3027, pages 223–238, 2004.

[17] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[18] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology — EUROCRYPT '04*, volume 3027, pages 54–73, 2004.

[19] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[20] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.

[21] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–341, 2005.

[22] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[23] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, pages 573–592, 2006.

[24] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[25] Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, pages 499–517, 2010.

[26] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In *ACM Conference on Computer and Communications Security*, pages 201–210, 2006.

[27] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT*, pages 302–321, 2005.

[28] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In *EUROCRYPT*, pages 246–263, 2007.

[29] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO '04*, volume 3152, pages 56–72, 2004.

[30] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.

[31] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[32] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[33] Jean-Sébastien Coron. On the exact security of full domain hash. In *CRYPTO*, pages 229–235, 2000.

[34] Ronald Cramer and Ivan Damgård. Secure signature schemes based on interactive protocols. In *Advances in Cryptology — CRYPTO '95*, pages 297–310, 1995.

[35] Ronald Cramer and Ivan Damgård. New generation of secure and practical RSA-based signatures. In *Advances in Cryptology — CRYPTO '96*, pages 173–185, 1996.

[36] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and System Security*, 3(3):161–185, 2000.

[37] Cynthia Dwork and Moni Naor. Universal one-way hash functions and their cryptographic applications. In *Symposium on the Theory of Computation*, pages 33–43, 1989.

[38] David M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. To appear in Advances in Cryptology – EUROCRYPT 2010, 2010.

[39] Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2010.

[40] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592, pages 123–139, 1999.

[41] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Symposium on the Theory of Computing*, pages 197–206, 2008.

[42] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography*, pages 257–273, 2006.

[43] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2), 1988.

[44] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, pages 265–282, 2007.

[45] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *CRYPTO*, pages 97–111, 2006.

[46] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.

[47] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[48] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *EUROCRYPT*, pages 333–350, 2009.

[49] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In *CRYPTO*, pages 654–670, 2009.

[50] Jung Yeon Hwang, Dong Hoon Lee, and Moti Yung. Universal forgery of the identity-based sequential aggregate signature scheme. In *ASIACCS*, pages 157–160, 2009.

[51] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In *Pairing*, pages 75–88, 2008.

[52] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

[53] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[54] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, 2012.

[55] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[56] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–479, 2010.

[57] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.

[58] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT*, pages 465–485, 2006.

[59] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *EUROCRYPT*, pages 74–90, 2004.

[60] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Carlisle Adams and Howard Heys, editors, *6th Selected Areas in Cryptography (SAC)*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1999.

[61] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*, pages 376–392, 2011.

[62] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[63] Moni Naor and Moti Yung. An efficient existentially unforgeable signature scheme and its applications. In *Advances in Cryptology – CRYPTO '94*, volume 839, pages 234–246, 1994.

[64] Gregory Neven. Efficient sequential aggregate signed data. In *EUROCRYPT*, pages 52–69, 2008.

[65] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology – CRYPTO '92*, volume 740, pages 31–53, 1992.

[66] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[67] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070, pages 387–398, 1996.

[68] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Symposium on the Theory of Computing*, pages 387–394. ACM, 1990.

[69] Claus P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[70] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, 2007.

[71] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494, pages 320–329, 2005.

[72] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

A Our Identity-Based Sequential Aggregate Signature Scheme

We show that the public key signature scheme defined in Section 5 gives rise to a provably secure identity-based sequential aggregate signature (IBSAS) scheme. The scheme is an extension of the BGOY scheme [15] in composite order groups.

A.1 Syntax and Security Definitions

We recall from [15] that an identity-based sequential aggregate signature (IBSAS) scheme consists of four algorithms:

Setup$(1^\lambda) \to \text{PP}, \text{MK}$ The setup algorithm takes as input the unary encoding of security parameter λ and outputs the public parameters PP and master secret key MK.

KeyDer$(\text{MK}, I) \to \text{SK}_I$ The key derivation algorithm takes as input the master key K and an identity I, and outputs a signing key SK_I.

AggSign$(\text{SK}_I, m, ((I_1, m_1), \ldots, (I_i, m_i)), \sigma') \to \sigma$ The sequential aggregate signing algorithm takes as input a signing key SK_I, a message m, a list of identity-message pairs $(I_1, m_1), \ldots, (I_i, m_i)$ for $i \in \mathbb{Z}^+$, and an aggregate signature σ', and outputs a new aggregate signature σ or $\perp$.

AggVerify$(\text{PP}, ((I_1, m_n), \ldots, (I_n, m_n)), \sigma) \to \{True, False\}$ The aggregate verification algorithm takes as input public parameters PP, a list of identity-message pairs $(I_1, m_1), \ldots, (I_n, m_n)$ for $n \in \mathbb{Z}^+$, and an aggregate signature σ, and outputs "True" or "False."

The scheme is required to be *correct*, meaning that for any $n \in \mathbb{Z}^+$ and list of identity-message pairs $(I_1, m_1), \ldots, (I_n, m_n)$, the aggregate verification algorithm when run on PP, $((I_1, m_n), \ldots, (I_n, m_n)), \sigma_n$ outputs "True," for any PP, σ_n generated as follows. We first generate PP, MK by running the setup algorithm on 1^λ. For $i = 1$ to n we generate σ_i by running the aggregate signing algorithm run on inputs $\text{SK}_{I_i}, m_i, ((I_1, m_1), \ldots, (I_{i-1}, m_{i-1})), \sigma_{i-1}$, where σ_0 is empty and each SK_{I_i} is itself generated by running the key derivation algorithm on inputs MK, I_i.

Now, the security definition for an IBSAS scheme considers a challenger interacting with an adversary against the scheme as follows:

Setup The challenger runs the Setup algorithm to generate public parameters PP and master secret key MK. It gives PP to the adversary.

Attack Phase The adversary may issue "CreateKey" and "Aggregate" queries to the challenger. For a "CreateKey" query, the adversary provides an identity I, and the challenger responds with a signing key SK_I it generates by running the key derivation algorithm on inputs MK and I. For an "Aggregate" query, the adversary provides an identity I, a message m, a list of identity-message pairs $(I_1, m_1), \ldots, (I_i, m_i)$ for any $i \in \mathbb{Z}^+$ of the adversary's choosing, and an aggregate signature σ'. The challenger responds with an aggregate signature σ it generates by running the aggregate signing algorithm on SK_I (itself generated by running the key derivation algorithm on inputs MK and I) and the remaining inputs provided by the adversary.

Forgery Finally, the adversary outputs an attempted forgery consisting of a list of identity-message pairs $(I_1, m_1), \ldots, (I_n, m_n)$, for any $i \in \mathbb{Z}^+$ of the adversary's choosing, and an aggregate signature σ. We say that the adversary *succeeds* if (1) the verification algorithm returns "True" on inputs $\text{PP}, ((I_1, m_1), \ldots, (I_n, m_n)), \sigma$, (2) all of $I_1, \ldots, I_n$ are distinct, and (3) there is some $1 \leq i^* \leq n$ such that during the attack phase the adversary made neither a CreateKey query on I_{i^*} nor any Aggregate query with the first two components as I_{i^*} and m_{i^*}.

We define the *advantage* of an adversary against an IBSAS scheme to be the probability it succeeds in the above experiment. We say that an IBSAS scheme is *secure* if the advantage of any PPT adversary against it is negligible.

A.2 Our Scheme

We now define our IBSAS scheme.

Setup$(1^\lambda) \to \text{VK}, \text{SK}$ The setup algorithm chooses a bilinear group $\mathbb{G}$ of order $N = p_1p_2p_3$. It chooses a random elements $g, k \in \mathbb{G}$, and random exponents $a, b, \alpha, \beta \in \mathbb{Z}_N$. It also chooses hash functions $H_1, H_2, H_3, H_4 : \{0,1\}^* \to \mathbb{G}$ and $H_5 : \{0,1\}^* \to \mathbb{Z}_N$, which will be modeled as random oracles. It sets the public parameters as

$$\text{PP} := \{N, \{H_i\}_{i=1,..,5}, \mathbb{G}, g, k, g^a, g^b, g^\alpha, g^\beta\}$$

and the master secret key as

$$\text{MK} := \{N, \{H_i\}_{i=1,..,4}, \mathbb{G}, a, b, \alpha, \beta\}.$$

KeyDer$(\text{MK}, I) \to \text{SK}_I$ The key derivation algorithm outputs

$$\text{SK}_I := (H_1(I)^a, H_2(I)^b, H_3(I)^\alpha, H_4(I)^\beta)$$

as the signing key for identity I.

AggSign$(\text{SK}_{I_i}, m_i, ((I_1, m_1), \ldots, (I_{i-1}, m_{i-1})), \sigma') \to \sigma$ If $i > 1$, the sequential aggregate signing algorithm first checks σ' verifies (according to the verification algorithm defined below) relative to PP and $(I_1, m_1), \ldots, (I_{i-1}, m_{i-1})$; if not, it outputs $\perp$. If so, then it chooses two random exponents $s, x \in \mathbb{Z}_N$, and computes:

$$\sigma_1 := \sigma'_1 H_1(I)^a H_2(I)^{bH_5(m)} (\sigma'_3)^{xs},\ \sigma_2 := (\sigma'_2)^{1/s} g^x,\ \sigma_3 := (\sigma'_3)^s,$$
$$\sigma_4 := (\sigma'_4)^s,\ \sigma_5 := \sigma'_5 H_1(I)^\alpha H_2(I)^{\beta H_5(m)} (\sigma'_4)^{xs}$$

where if $i = 1$ we let $\sigma'_1 = 1_\mathbb{G}$, $\sigma'_2 = g$, $\sigma'_3 = g$, $\sigma'_4 = k$, $\sigma'_5 = 1_\mathbb{G}$. It outputs the new aggregate signature $\sigma := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.

Note that if we define $r'_1 = dlog_g(\sigma'_2)$ and $r'_2 = dlog_g(\sigma'_3)$, then the new aggregate signature has randomness $r_1 = s^{-1}r'_1 + x$ and $r_2 = sr'_2$. These values are random and mutually independent.

AggVerify$(\text{PP}, ((I_1, m_n), \ldots, (I_n, m_n)), \sigma) \to \{True, False\}$ The aggregate verification algorithm first checks that $I_1, \ldots, I_n$ are all distinct. It then checks that:

$$e(\sigma_1, g) = e(\prod_{i=1}^{n} H_1(I_i), g^a) e(\prod_{i=1}^{n} H_2(I_i)^{H_5(m_i)}, g^b) e(\sigma_2, \sigma_3).$$

It also checks that:

$$e(\sigma_5, g) = e(\prod_{i=1}^{n} H_3(I_i), g^\alpha) e(\prod_{i=1}^{n} H_4(I_i)^{H_5(m_i)}, g^\beta) e(\sigma_2, \sigma_4).$$

Finally, it checks that:

$$e(g, \sigma_4) = e(k, \sigma_3).$$

If all of these checks pass, it outputs "True." Otherwise, it outputs "False."

We show correctness of the scheme by induction on i in the definition of correctness. For $i = 1$ this is straightforward. For $i > 1$ we derive the first equation checked by aggregate verification algorithm as follows:

$$\begin{aligned} e(\sigma_1, g) &= e(\sigma'_1, g)e(H_1(I_i)^a H_2(I)^{bH_5(m_i)}(\sigma'_3)^{xs}, g) \\ &= e(\prod_{j=1}^{i-1} H_1(I_j), g^a)e(\prod_{j=1}^{i-1} H_2(I_j)^{H_5(m_j)}, g^b)e(\sigma'_2, \sigma'_3)e(H_1(I_i), g^a)e(H_2(I)^{H_5(m_i)}, g^b)e((\sigma'_3)^{xs}, g) \\ &= e(\prod_{j=1}^{i} H_1(I_j), g^a)e(\prod_{j=1}^{i} H_2(I_j)^{H_5(m_j)}, g^b)e((\sigma'_2)^{1/s} g^x, (\sigma'_3)^s) \\ &= e(\prod_{j=1}^{i} H_1(I_j), g^a)e(\prod_{j=1}^{i} H_2(I_j)^{H_5(m_j)}, g^b)e(\sigma_2, \sigma_3) \end{aligned}$$

where on the second line we use that by assumption the equation holds for $(\sigma'_1, \sigma'_2, \sigma'_3)$ if we remove (I_i, m_i) from the list of identity-message pairs. An analogous derivation pertains to the second equation checked by the aggregate verification algorithm. Finally, the last equation checked by the aggregate verification algorithm is easy to derive.

Discussion Our scheme can be viewed as an extension of the prior IBSAS scheme of [15] (BGOY) in composite order rather than prime groups. Namely, the $\sigma_1, \sigma_2, \sigma_3$ components of our aggregate signature correspond exactly to a BGOY aggregate signature (in a composite order group). The additional σ_4, σ_5 elements in our aggregate signature may be viewed as a "second copy" of the BGOY scheme tied together with the first copy by g and by r_1, r_2 (defined as $dlog_g(\sigma_2)$ and $dlog_g(\sigma_3)$ respectively). This second copy is carried over to the corresponding public-key signature scheme in Section 5 and facilitates the security proof of the latter.

However, note that the second copy is not "complete" in the sense that there is no component of the form k^{r_1}, which is crucial for our security proof but introduces a new challenge for aggregation. Namely, it is important for the scheme's security that the aggregation process independently re-randomize both r_1 and r_2. However, we cannot additively re-randomize r_1 and r_2 as $r_1 + s$ and $r_2 + x$ for random s, x as done in [15], since in our case the signing algorithm would then need to form a "cross-term" $k^{r_1 x}$ to multiply into σ_5, which is hard without k^{r_1}. Instead we re-randomize r_2 multiplicatively as sr_2 but r_1 as $s^{-1}r_1 + x$ in order to allow updating the randomness in σ_1 and σ_5 correctly *without* needing k^{r_1}.

We also note that while our security model for IBSAS schemes requires that the signers in an aggregate signature have distinct identities, this is mainly to simplify the security proof of our scheme. Indeed, it is possible to circumvent this issue for our scheme by using a technique introduced in [58]. Namely, if a signer I has previously added a signature on a message m to an aggregate $(\sigma'_1, \ldots, \sigma'_5)$ and wishes to add another signature on a message m', it can set $\sigma_1 := \sigma'_1 H_2(I)^{bH_5(m\|m')}/H_2(I)^{bH_5(m)}$ and similarly for σ_5, and then re-randomize the aggregate signature as above.

A.3 Security Analysis

Unlike the prior BGOY scheme, our IBSAS scheme is provably-secure under static assumptions. This is a corollary of the following.

Theorem A.1. Under Assumption 5.5 and the assumption that the public-key signature scheme (using the Sign_A algorithm) defined in Section 5 is existentially unforgeable, our IBSAS scheme is secure in the sense defined above.

Proof. Assume there is a PPT adversary $\mathcal{A}$ breaking the IBSAS scheme. In the first step of the proof, we change the real IBSAS security game $Game_{real}$ to a game $Game_{real'}$ where the elements g, k in the public parameters as well as the outputs of random oracles $H_1, \ldots, H_4$ belong to the $\mathbb{G}_{p_1}$ subgroup rather than the full group $\mathbb{G}$. For this we use Assumption 5.5 that given $\mathbb{G}, N$ and a challenge T where either $T \in \mathbb{G}$ or

$T \in \mathbb{G}_{p_1}$, it is hard to guess which one is the case. We give a simulator $\mathcal{S}$ that breaks this assumption if there is a non-negligible difference in $\mathcal{A}$'s advantage in $Game_{real}$ versus $Game_{real'}$. Simulator $\mathcal{S}$ given $\mathbb{G}, N$ and T runs $\mathcal{A}$ on public parameters

$$\text{PP} := \{N, \{H_i\}_{i=1,..,5}, \mathbb{G}, g := T, k := T^d, T^a, T^b, T^\alpha, T^\beta\}$$

where it chooses $d, a, b, \alpha, \beta \in \mathbb{Z}_N$ itself at random. When $\mathcal{A}$ makes a random oracle query to H_i for any $1 \leq i \leq 4$, it responds with T^r for a new random $r \in \mathbb{Z}_N$, and when $\mathcal{A}$ makes a random oracle query to H_5, it responds with a new random $x \in \mathbb{Z}_N$. (However, $\mathcal{S}$ maintains consistent answers to any repeat queries.) Since $\mathcal{S}$ knows the corresponding MK to PP it can easily answer any CreateKey or Aggregate queries from $\mathcal{A}$ appropriately. Finally, if $\mathcal{A}$ produces a forgery then $\mathcal{S}$ outputs 1. It is easy to see that if there is a non-negligible difference in $\mathcal{A}$'s advantage in $Game_{real}$ versus $Game_{real'}$ then $\mathcal{S}$ breaks the Assumption 5.5.

It remains to argue that if $\mathcal{A}$'s advantage in $Game_{real'}$ is non-negligible then there is an adversary $\mathcal{B}$ that breaks the public-key signature scheme. We describe $\mathcal{B}$ as follows. On input

$$\text{VK} := \{N, H, \mathbb{G}, g, k, g^{a_1}, g^{a_2}, g^{b_1}, g^{b_2}, g^{\alpha_1}, g^{\alpha_2}, g^{\beta_1}, g^{\beta_2}\}$$

$\mathcal{B}$ runs $\mathcal{A}$ on public parameters

$$\text{PP} := \{N, \{H_i\}_{i=1,..,5}, \mathbb{G}, g, k, g^a := g^{a_1}, g^b := g^{b_1}, g^\alpha := g^{\alpha_1}, g^\beta := g^{\beta_1}\};.$$

Let q_i denote the maximum number of queries that $\mathcal{A}$ makes to H_i for $1 \leq i \leq 4$. Then $\mathcal{B}$ chooses random numbers $1 \leq n_i^* \leq q_i$ for $1 \leq i \leq 4$. To simplify the simulation, we assume that $\mathcal{A}$ never includes an identity I anywhere in its CreateKey or Aggregate queries, or in its forgery, that it has not queried to H_i for all $1 \leq i \leq 4$. We describe how $\mathcal{B}$ responds to the various queries $\mathcal{A}$ can make (it is understood that consistency is maintained for repeat queries):

Queries to $H_1, \ldots, H_4$ When $\mathcal{A}$ makes its j-th query to H_1, which we denote by $I_{1,j}$, if $j = n_1^*$ it responds with g^{a_2} and otherwise with g^t where $t \in \mathbb{Z}_N$ is a new randomly chosen value. The responses given by $\mathcal{B}$ to $\mathcal{A}$'s queries to H_2, H_3, H_4 are analogous where g^{a_2} is replaced with g^{b_2}, g^{α_2}, and g^{β_2} respectively. As before we denote the j-th query to H_i by $I_{i,j}$.

Queries to H_5 When $\mathcal{A}$ makes a query to H_5, $\mathcal{A}$ forwards this query to its own oracle H and gives the response to $\mathcal{B}$.

CreateKey queries When $\mathcal{A}$ makes a CreateKey query on identity I, if $I = I_{i,n_i^*}$ for any $1 \leq i \leq 4$ then $\mathcal{B}$ aborts. Otherwise, let $H_i(I) = g^{t_i}$ for $1 \leq i \leq 4$ where the t_i are known to $\mathcal{B}$ (by maintaining an appropriate list). It sets

$$\text{SK}_I := (g^{a_1 t_1}, g^{b_1 t_2}, g^{\alpha_1 t_3}, g^{\beta_1 t_4})$$

and gives SK_I as the response to $\mathcal{A}$.

Aggregate queries When $\mathcal{A}$ makes an Aggregate query on identity I, message m, list of identity-message pairs $(I_1, m_1), \ldots, (I_k, m_k)$ and aggregate signature σ', if $I_j = I_{i,n_i^*}$ for any $1 \leq j \leq k$ and $1 \leq i \leq 4$ then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ first checks that σ' verifies with respect to PP and $(I_1, m_1), \ldots, (I_k, m_k)$, and otherwise responds to $\mathcal{A}$ with $\perp$. If so, if $I = I_{i,n_i^*}$ for any $1 \leq i \leq 4$ then $\mathcal{B}$ queries m to its own signing oracle to receive $(\sigma_1, \ldots, \sigma_5)$. Otherwise, let $H_i(I) = g^{t_i}$ for $1 \leq i \leq 4$ where the t_i are known to $\mathcal{B}$. Using VK it sets

$$\sigma_1 := g^{a_1 t_1 + b_1 t_2 H_5(m)} g^{r_1 r_2}, \sigma_2 := g^{r_1}, \sigma_3 := g^{r_2}, \sigma_4 := k^{r_2}, \sigma_5 := g^{\alpha_1 t_3 + \beta_1 t_4 H_5(m)} k^{r_1 r_2}$$

where $r_1, r_2 \in \mathbb{Z}_N$ are randomly chosen. Now let $H_i(I_j) = g^{t_{i,j}}$ for $1 \leq i \leq 4$ and $1 \leq j \leq k$ where the $t_{i,j}$ are known to $\mathcal{B}$. It updates σ_1 and σ_5 as follows

$$\sigma_1 := \sigma_1 \prod_{j=1}^{k} g^{a_1 t_{1,j} + b_1 t_{2,j} H_5(m_j)}, \sigma_5 := \sigma_5 \prod_{j=1}^{k} g^{\alpha_1 t_{3,j} + \beta_1 t_{4,j} H_5(m_j)}$$

and gives $(\sigma_1, \ldots, \sigma_5)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs as its attempted forgery an aggregate signature $\sigma^* = (\sigma_1^*, \ldots, \sigma_5^*)$ and a list of identity-message pairs $(I_1, m_1), \ldots, (I_n, m_n)$. If this is not a successful forgery, then we can ignore this attempt. Otherwise, let i^* satisfy condition (3) of the forgery definition. If $I_j = I_{i,n_i^*}$ for any $1 \leq j \neq i^* \leq n$ and $1 \leq i \leq 4$, or if $I_{i^*} \neq I_{i,n_i^*}$ for *any* $1 \leq i \leq 4$, then $\mathcal{B}$ aborts. Otherwise, let $H_i(I_j) = g^{t_{i,j}}$ for $1 \leq i \leq 4$ and $1 \leq j \leq n$ where the $t_{i,j}$ are known to $\mathcal{B}$ Then $\mathcal{B}$ updates σ_1^* and and σ_5^* as follows:

$$\sigma_1^* := \sigma_1^* / \prod_{j \neq i*} g^{a_1 t_{1,j} + b_1 t_{2,j} H_5(m_j)}, \sigma_5^* := \sigma_5^* / \prod_{j \neq i^*} g^{\alpha_1 t_{3,j} + \beta_1 t_{4,j} H_5(m_j)} .$$

It outputs $(\sigma_1^*, \ldots, \sigma_5^*)$ as its own forgery. This concludes the description of $\mathcal{B}$. Observe that unless $\mathcal{B}$ aborts, it provides $\mathcal{A}$ with a perfect simulation of $Game_{real'}$; in particular, this uses the fact that (as previously noted) our aggregation procedure independently re-randomizes both r_1 and r_2. Furthermore, it is not hard to show that $\mathcal{B}$ outputs a forgery of its own whenever $\mathcal{A}$ does; this uses the fact that $I_1, \ldots, I_n$ in $\mathcal{A}$'s forgery are required to be distinct. Furthermore, $\mathcal{B}$ does not abort with probability at least $1/q_1 q_2 q_3 q_4$, which is inverse polynomial. (A somewhat tighter security reduction is possible using Coron's technique [33].) This is because due to the rules of the IBSAS security game, $\mathcal{B}$ will not abort as long as it guesses the correct values of n_i^* for all $1 \leq i \leq 4$, meaning that $\mathcal{A}$ queries I_{i^*} in its forgery as the n_i^*-th query to H_i, and all these guesses are random and independent. The theorem follows. □

B Security of Our Assumptions in the Generic Group Model

Though the assumptions we use are static and compact, some fall outside of the usual class of "subgroup decision" assumptions. As a check on these assumptions, we show they hold in the generic bilinear group model extended to the setting of composite order groups as in [52], when it is hard to find a non-trivial factor of the group order N.

The Model We briefly describe the generic bilinear group model for composite order groups (or just "generic group model") and refer the reader to [52] for more details. Fix groups $\mathbb{G}, \mathbb{G}_T$, each of the same order $p_1 \cdots p_t$, equipped with a bilinear map $e \colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. In the generic group model, a group element $x \in \mathbb{G}$ is represented by a tuple $(x_1, \ldots, x_t)$ where each $x_i \in \mathbb{Z}_{p_i}$ is $dlog_{g_i}(x)$ for a fixed generator g_i of the $\mathbb{G}_{p_i}$ subgroup of $\mathbb{G}$. Similarly, a group element in $y \in \mathbb{G}_T$ is represented by a bracketed tuple $[y_1, \ldots, y_t]$, with respect to the generators $e(g_i, g_i)$. An algorithm is not given access to the groups $\mathbb{G}, \mathbb{G}_T$ directly, but rather is provided with "handles" that correspond to group elements. The algorithm may query for the handle of a new group element in a limited number of ways. First, it may submit two handles for elements both in $\mathbb{G}$ or both in $\mathbb{G}_T$ and it will receive in return the handle corresponding to their product under the group operation. It may also submit a handle and an exponent in $\mathbb{Z}_N$, and it will receive a handle corresponding to that group element raised to that exponent. It can also query for the handle of a result of a pairing operating by submitting the two handles of the elements in $\mathbb{G}$ to be paired. We limit the adversary to querying on handles it has previously received (this can be enforced by choosing random handles from a large enough set that the adversary can only a guess a new handle with negligible probability).

To show an assumption is hard in this model, we consider a game between an adversary and a challenger. The challenger will generate group elements according to the distribution specified in the assumption, and will give the adversary handles to the group elements that are given out in the assumption. The adversary then makes queries for handles of new elements, according to the allowed operations of group multiplication, exponentiation by an element of $\mathbb{Z}_N$, and pairing. It is the adversary's task to break the assumption.

We employ the strategy developed in [52]. We represent any group elements that are randomly chosen from a subgroup by a tuple with indeterminates (i.e., formal variables) in the corresponding positions. For example, a random element $a \in \mathbb{G}_{p_1 p_2}$ would be represented by $A = (A_1, A_2, 0 \ldots, 0)$ where A_1, A_2 are indeterminates. Dependencies between elements are expressed by re-using formal variables. In the original game, these random variables are instantiated by choosing group elements from the appropriate distributions and giving out these handles. However, we pass to an alternate game where the variables are never concretely

instantiated, and the challenger instead keeps track of the formal variables, and only assigns group elements equal handles if they are the same as formal polynomials in the variables. The difference between this and the original game is negligible (assuming the polynomials in the formal variables have constant degree), since two unequal formal polynomials will only happen to be equal when instantiated with negligible probability. We can then make an additional change to the game by doing all computation modulo N instead of modulo p_i in each component, since if this causes a difference, then a non-trivial factor of N can be computed, which we are assuming to be hard.

In addition to the terms explicitly given to the adversary in our assumptions, we will add a randomly distributed element of the full group, $\mathbb{G}$. We denote this element by (Y_1, Y_2, Y_3). This models the fact that the adversary may have the capability to sample randomly from the full group. To see that giving the adversary a single randomly distributed group element suffices to model this sampling ability, consider an alternate game where the adversary is additionally able to query for a fresh random sample from $\mathbb{G}$. If there exists an adversary $\mathcal{A}$ who breaks one of our assumptions in this model, then we can create an adversary $\mathcal{B}$ which breaks the assumption given only one randomly generated group element. This is because $\mathcal{B}$ can respond to the random sampling queries of $\mathcal{A}$ by taking its one group element and raising it to randomly chosen powers. We do not claim that this applies to *all* assumptions, since the internal views of $\mathcal{B}$ and $\mathcal{A}$ will differ. However, in our assumptions, the goal is merely to produce group elements with particular distributions, so $\mathcal{A}$'s success will always imply success for $\mathcal{B}$.

We do not include a random generator for $\mathbb{G}_T$, since all of the assumptions we consider here require the adversary to produce elements in $\mathbb{G}$, and elements of $\mathbb{G}_T$ clearly do not aid the adversary in this task. We note that the theorems in [52] which were used in [56, 57] to show generic security for the subgroup decision assumptions that we also employ do not explicitly include generators of $\mathbb{G}$ or $\mathbb{G}_T$ being given to the adversary, but the theorems still hold if one were to add these terms.

Theorem B.1. Assumption 4.3 holds in the generic group model if it is hard to find a non-trivial factor of N.

Proof. Here the adversary is given

$$(1,0,0), (A_1,0,0), (R_1,X_2,0), (A_1R_1,W_2,0), (R_1A_1^2,V_2,0), (0,1,0), (0,0,1), (Y_1,Y_2,Y_3)$$

and must compute

$$a = (a_1,a_2,a_3), b = (b_1,b_2,b_3)$$

such that

$$a_1 = b_1 A_1^2 \tag{4}$$

and $a_2, b_2 = 0$, $a_1 \neq 0$. Since the adversary can only produce linear combinations in $\mathbb{Z}_N$ of the terms it is given, it must use the $(R_1A_1^2, V_2, 0)$ term to introduce a non-zero A_1^2 term in a. However, this introduces a non-zero contribution from V_2 in a_2, and this cannot be canceled out because V_2 does not appear elsewhere. □

Theorem B.2. Assumption 5.4 holds in the generic group model if it is hard to find a non-trivial factor of N.

Proof. Here the adversary if given

$$(1,0,0), (C_1,0,0), (C_2,0,0), (C_1C_2,0,X_3), (0,0,1), (Y_1,Y_2,Y_3)$$

and must compute $(C_1C_2,0,0)$. Since the adversary can only take linear combinations of the received terms, it must use non-zero contribution from $(C_1C_2,0,X_3)$, but this also yields a non-zero contribution from X_3 in the third coordinate, which cannot be canceled out because X_3 does not appear elsewhere. □

Theorem B.3. Assumption 5.2 holds in the generic group model if it is hard to find a non-trivial factor of N.

Proof. Here is the adversary given

$$(1,0,0),(C_1,0,0),(C_2,0,0),(0,0,1),(X_1,X_2,0),(Y_1,Y_2,Y_3)$$

and must compute

$$a=(a_1,a_2,a_3),b=(b_1,b_2,b_3),d=(d_1,d_2,d_3)$$

such that

$$a_1 - b_1 d_1 = C_1 C_2 \tag{5}$$

and $a_2 \neq 0$. Let's write

$$\begin{aligned} a_1 &= \alpha_a + \beta_a C_1 + \gamma_a C_2 + \zeta_a X_1 + \eta_a Y_1 \\ b_1 &= \alpha_b + \beta_b C_1 + \gamma_b C_2 + + \zeta_b X_1 + \eta_b Y_1 \\ d_1 &= \alpha_d + \beta_d C_1 + \gamma_d C_2 + + \zeta_d X_1 + \eta_d Y_1 \end{aligned}$$

where the coefficients are in $\mathbb{Z}_N$. Consider the coefficients of $X_1C_1, X_1C_2, Y_1C_1, Y_1C_2$ in b_1d_1, which are respectively $\beta_b\zeta_d + \beta_d\zeta_b$, $\gamma_b\zeta_d + \gamma_d\zeta_b$, $\eta_b\beta_d + \eta_d\beta_b$, $\eta_b\gamma_d + \eta_d\gamma_b$. We argue at least one of them is non-zero, which contradicts equation 5. (This uses some ideas from the proof of [15, Lemma C.1].) By equation 5 we have $\beta_b\gamma_d + \gamma_b\beta_d = 1$. In particular, this means at least one the pairs $(\beta_b,\gamma_d),(\gamma_b,\beta_d)$ contains two non-zero coefficients. But since the coefficients of C_1^2 and C_2^2 on the RHS of equation 5 are zero and it is assumed to be hard to find a non-trivial factor of N, *exactly* one of these pairs contains two non-zero coefficients and the other pair has both equal to zero. Moreover, since $a_2 \neq 0$, at least one of ζ_a, η_a must be non-zero. If ζ_a is non-zero then $\zeta_b\alpha_d + \zeta_d\alpha_b = -\zeta_a$ is also non-zero. But since the coefficient of X_1^2 on the RHS of equation 5 is zero, similarly to before we have that *exactly* one of ζ_b, ζ_d is non-zero and the other is equal to zero. If $\eta_a \neq 0$, then an analogous statement holds for η_b, η_d. In all cases we can derive the desired contradiction. □

C BB-Derived Signatures

In this section, we will improve upon a signature scheme implied by a variant of the IBE scheme introduced by Boneh and Boyen [16] as a final example of leveraging the dual form signature framework.

Naor observed that if an IBE scheme is adaptively secure, then the implied signature scheme is also secure under the same complexity assumptions [19]. The IBE scheme originally proposed by Boneh and Boyen was only shown to be selectively secure. However, Waters proposed a variant of the Boneh-Boyen scheme with linear-size public keys that was shown to be adaptively secure [71]. Unfortunately, these linear size public keys would carry over to the implied signature scheme.

Our starting point is a variant of the Boneh-Boyen IBE introduced by Lewko and Waters [56]. By embedding the Boneh-Boyen scheme in composite order groups, Lewko and Waters were able to keep the constant-size public keys and show that the scheme is adaptively secure. This yields an implied signature scheme which has constant-size signatures and is existentially unforgeable. (By "constant-size", we mean a constant number of group elements. The group size will depend on the security parameter.) Our signature scheme is a more efficient version of this, where verification is done directly instead of through IBE encryption. Because of this modification, security for our scheme does not follow from the security of the related IBE scheme.

In our scheme, the Boneh-Boyen structure takes place in $\mathbb{G}_{p_1}$, while $\mathbb{G}_{p_3}$ is used for additional randomization, and the difference between the two signing algorithms appears in $\mathbb{G}_{p_2}$. Namely, signatures produced by Sign_A have no $\mathbb{G}_{p_2}$ components, while signatures produced by Sign_B have random $\mathbb{G}_{p_2}$ components. We define the forgery types similarly: Type I forgeries have no $\mathbb{G}_{p_2}$ components, while Type II forgeries have at least one nonzero $\mathbb{G}_{p_2}$ component. As for CL signatures, we prove dual-oracle invariance by designing a *backdoor verification test* that the simulator can perform to reveal the presence of $\mathbb{G}_{p_2}$ components in the forgery produced by the attacker. To perform this test, the simulator uses knowledge of the discrete

logarithms of some of the public parameters u and h that form the core of the Boneh-Boyen structure. More precisely, in the $\mathbb{G}_{p_1}$ group our signatures are of the form $g^r, g^\alpha(u^m h)^r$, where g, u, h are public parameters, α is secret, and r is chosen randomly each time. In the proof, the simulator will know the discrete logarithms of u and h base g (denoted by a and b respectively), which will allow it to perform a backdoor verification which is like the real verification, except that the base g has been replaced by another a group element of the simulator's choice. Since the public parameters reveal no information about a and b modulo p_2 and $a + bm$ is a pairwise independent function of m, we are able to show that the challenge signature does not reveal enough information about the space on which the backdoor test fails to enable the attacker to produce a forgery for some new message m^* that causes the simulator to mistake its type.

C.1 Our Dual Form Scheme

We now give our dual form version of BB-derived signatures.

KeyGen(λ)**:** The key generation algorithm chooses two groups, $\mathbb{G}$ and $\mathbb{G}_T$ of order $N = p_1p_2p_3$, where p_1, p_2, and p_3 are distinct primes of length λ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate, efficiently computable bilinear map. It selects uniformly at random $u, g, h \in \mathbb{G}_{p_1}$ and $\alpha \in \mathbb{Z}_N$. It sets

$$\text{SK} = (\alpha, g_{2,3}, g_3)$$

and

$$PK = (N, u, g, h, e(g, g)^\alpha),$$

where $g_{2,3}$ and g_3 are generators of the subgroups $\mathbb{G}_{p_2p_3}$ and $\mathbb{G}_{p_3}$, respectively.

Sign$_A(\text{SK}, M)$**:** Given a secret key $\text{SK} = (\alpha, g_{2,3}, g_3)$, a public key $PK = (N, u, g, h, e(g, g)^\alpha)$, and a message $m \in Z_N^*$, the algorithm chooses $r \in \mathbb{Z}_N$ and $R_3, R_3' \in \mathbb{G}_{p_3}$ uniformly at random (we note that random elements in $\mathbb{G}_{p_3}$ can be generated by raising the generator g_3 to random exponents in $\mathbb{Z}_N$). It outputs the signature

$$\sigma = (\sigma_2, \sigma_1) = (g^r R_3,\ g^\alpha(u^m h)^r R_3').$$

Sign$_B(\text{SK}, M)$**:** Given a secret key $\text{SK} = (\alpha, g_{2,3}, g_3)$, a public key $PK = (N, u, g, h, e(g, g)^\alpha)$, and a message $m \in Z_N^*$, the algorithm chooses $r \in \mathbb{Z}_N$ and $R_{2,3}, R_{2,3}' \in \mathbb{G}_{p_2p_3}$ uniformly at random. It outputs the signature

$$\sigma = (\sigma_2, \sigma_1) = (g^r R_{2,3},\ g^\alpha(u^m h)^r R_{2,3}').$$

Verify(PK, m, σ)**:** Given a public key $PK = (N, u, g, h, e(g, g)^\alpha)$, a message m, and a signature $\sigma = (\sigma_2, \sigma_1)$, the verification algorithm checks that

$$\frac{e(\sigma_1, g)}{e(\sigma_2, u^m h)} = \frac{e(g, g)^\alpha e(u^m h, g)^r}{e(g, u^m h)^r} = e(g, g)^\alpha.$$

Forgery Classes We will divide the forgery types based on whether they have a $\mathbb{G}_{p_2}$ component. We let $z \in \mathbb{Z}_N$ denote the exponent represented by the tuple $(0 \bmod p_1, 1 \bmod p_2, 0 \bmod p_3)$. Then we can define the forgery classes as follows:

Type I. $\mathcal{V}_I = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_2^*)^z = 1 \text{ and } (\sigma_1^*)^z = 1\}$

Type II. $\mathcal{V}_{II} = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_2^*)^z \neq 1 \text{ or } (\sigma_1^*)^z \neq 1\}$

C.2 Complexity Assumptions

We employ the same three static assumptions used to prove security for the LW IBE scheme [56]. The first two of these were previously stated in Section 4.2, so here we only state the third.

Assumption C.1. Given a group generator $\mathcal{G}$, we define the following distribution:

$$(N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G},$$

$$\alpha, s \xleftarrow{R} \mathbb{Z}_N, g \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2, Z_2 \xleftarrow{R} \mathbb{G}_{p_2}, X_3 \xleftarrow{R} \mathbb{G}_{p_3},$$

$$D = (\mathbb{G}, g, g^{\alpha} X_2, X_3, g^{s} Y_2, Z_2),$$

$$T_1 = e(g,g)^{\alpha s}, T_2 \xleftarrow{R} \mathbb{G}_T$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption C.1 to be:

$$Adv_{\mathcal{A}}^{C.1}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 9. We say that $\mathcal{G}$ satisfies Assumption C.1 if for any polynomial time algorithm, $\mathcal{A}$, $Adv_{\mathcal{A}}^{C.1}(\lambda)$ is a negligible function of λ.

C.3 Proof of Security

We will show that our new signature scheme is secure under these assumptions by showing that it satisfies the three properties of a secure dual form signature scheme.

Lemma C.1. If Assumption 4.1 holds, then our signature scheme is A-I Matching.

Proof. Suppose that there exists an attacker, $\mathcal{A}$, that can create a forgery that is not of Type I with probability ϵ given access to an oracle for the Sign_A algorithm. Then we can create an algorithm $\mathcal{B}$ that breaks Assumption 4.1 with advantage negligibly close to ϵ.

$\mathcal{B}$ first receives g, X_3, T. $\mathcal{B}$ then randomly chooses exponents $\alpha, a, b \in \mathbb{Z}_N$ and sets $\text{SK} = (\alpha, X_3)$ and $PK = (N, u = g^a, g = g, h = g^b, e(g,g)^{\alpha} = e(g,g)^{\alpha})$. Notice that the Sign_A algorithm does not use $g_{2,3}$, so $\mathcal{B}$ does not need it in the secret key. For a query m, $\mathcal{B}$ chooses $r, u, v \in \mathbb{Z}_N$ uniformly at random and outputs the signature,

$$\sigma = (g^r X_3^u, g^{\alpha} (u^m h)^r X_3^v).$$

Eventually, $\mathcal{A}$ will output a forgery, (σ^*, m^*). First, $\mathcal{B}$ will check that the forgery verifies, if not then $\mathcal{B}$ will output $b \in \{0,1\}$ uniformly at random. If the forgery verifies, then $\mathcal{B}$ will try to use this forgery to determine whether T is in $\mathbb{G}_{p_1}$ or $\mathbb{G}_{p_1 p_2}$.

$\mathcal{B}$ sets

$$C_0 = e(T, g)^{\alpha},\ C_1 = T^{am^* + b},\ C_2 = T.$$

Then $\mathcal{B}$ proceeds with a *backdoor verification test* using the C values as follows,

$$\frac{e(\sigma_1, C_2)}{e(\sigma_2, C_1)} = C_0.$$

Since it is guaranteed that (σ^*, m^*) passes the real verification test, we know that it will pass this verification equation with $T = g^s$ for any exponent $s \in \mathbb{Z}_N$. However, since we do not know if $T \in \mathbb{G}_{p_1}$, we do not know if we can express T simply as g^s. In general, we can break T into its separate subgroup components as follows,

$$T = g^s g_2^{t_2},$$

where g_2 is some generator for $\mathbb{G}_{p_2}$ and the exponents s and t_2 are random, unless $T \in \mathbb{G}_{p_1}$, in which case $t_2 = 0$. Likewise, we can divide the signature elements into their components,

$$\sigma_2^* = g^r g_2^{s_{2,2}} g_3^{s_{2,3}}$$
$$\sigma_1^* = g^{\alpha + r(am^* + b)} g_2^{s_{1,2}} g_3^{s_{1,3}},$$

where r and each $s_{i,j}$ are some exponents in $\mathbb{Z}_N$. If $\mathcal{B}$ receives a Type I forgery, then $s_{2,2} = s_{1,2} = 0$.

Then the backdoor verification equation proceeds as follows,

$$\begin{aligned}\frac{e(\sigma_1, C_2)}{e(\sigma_2, C_1)} &= \frac{e(g,g)^{\alpha s} e(g^{am^*+b}, g)^{rs} e(g_2, g_2)^{s_{1,2}t_2}}{e(g^{am^*+b}, g)^{rs} e(g_2, g_2)^{s_{2,2}t_2(am^*+b)}} \\ &= e(g,g)^{\alpha s} e(g_2, g_2)^{t_2(s_{1,2} - s_{2,2}(am^*+b))} \\ &= C_0 e(g_2, g_2)^{t_2(s_{1,2} - s_{2,2}(am^*+b))} \stackrel{?}{=} C_0.\end{aligned}$$

If this equality is false, then $\mathcal{B}$ will output 1. If the equality is true, then $\mathcal{B}$ will flip a coin $b \in \{0, 1\}$ and return b. Notice that if $\mathcal{B}$ tried to create its own signatures using T, if they were verifiable then they would always pass this additional verification test. This means that $\mathcal{B}$ cannot gain any advantage against the Assumption 4.1 challenger without using the output of $\mathcal{A}$.

It must either be the case that $T \in \mathbb{G}_{p_1}$ or $T \in \mathbb{G}_{p_1 p_2}$. If $T \in \mathbb{G}_{p_1}$, then $t_2 = 0$ and the equality will always hold. In this case $\mathcal{B}$ will output 1 with probability 1/2.

If $T \in \mathbb{G}_{p_1 p_2}$, then $t_2 \neq 0$, so $\mathcal{B}$ will pass the verification depending on the values of $s_{2,2}$ and $s_{1,2}$. If $\mathcal{A}$ returns a Type I forgery, then $s_{2,2} = s_{1,2} = 0$ so $s_{1,2} = s_{2,2}(am^* + b)$ and the forgery will always pass both verification tests. In this case $\mathcal{B}$ will output 1 with probability 1/2.

If $T \in \mathbb{G}_{p_1 p_2}$ and $\mathcal{A}$ returns a Type II forgery (i.e. any forgery that is not of Type I), then either $s_{1,2} = s_{2,2}(am^* + b)$ or $s_{1,2} \neq s_{2,2}(am^* + b)$. If $s_{1,2} = s_{2,2}(am^* + b)$, then the forgery will always pass the second verification test and $\mathcal{B}$ will output 1 with probability 1/2. However, in order for an attacker to create a Type II forgery where $s_{1,2} = s_{2,2}(am^* + b)$, the attacker must be able to implicitly determine $am^* + b$ modulo p_2. Of course, m^* will be known to the attacker, but a and b modulo p_2 are not revealed at any point during the query phase, so there is a negligible chance, δ, of an attacker being able to create a Type II forgery that passes the second verification test.

Finally, if $T \in \mathbb{G}_{p_1 p_2}$ and $s_{1,2} \neq s_{2,2}(am^* + b)$ then the test will always fail, so $\mathcal{B}$ will output a 1 with probability 1. Thus, we can calculate the advantage of $\mathcal{B}$ agains the Assumption 4.1 challenger,

$$\begin{aligned}|Pr[\mathcal{B}(D, T_1) = 1] - Pr[\mathcal{B}(D, T_2) = 1]| &= \left| \epsilon \left(\delta \cdot \frac{1}{2} + (1 - \delta) 1 \right) + (1 - \epsilon) \frac{1}{2} - \frac{1}{2} \right| \\ &= \frac{1}{2}\epsilon - \frac{1}{2}\epsilon\delta.\end{aligned}$$

Thus, if ϵ is non-negligible, then $\mathcal{B}$ has non-negligible advantage against the Assumption 4.1 challenger. □

Lemma C.2. If Assumption 4.2 holds, then our signature scheme satisfies dual-oracle invariance.

Proof. Suppose that there exists an attacker, $\mathcal{A}$, that can break dual-oracle invariance with non-negligible advantage. Then we can create an algorithm $\mathcal{B}$ that breaks Assumption 4.2 with non-negligible advantage.

$\mathcal{B}$ receives $\mathbb{G}, g, X_1 X_2, X_3, Y_2 Y_3, T$. $\mathcal{B}$ picks random exponents $a, b, \alpha \in \mathbb{Z}_N$ and sets $PK = (N, u = g^a, g = g, h = g^b, e(g,g)^\alpha = e(g,g)^\alpha)$ and $\text{SK} = (\alpha, X_3, Y_2 Y_3)$. Using these keys, $\mathcal{B}$ will be able to simulate both the Sign_A and the Sign_B algorithms.

If $\mathcal{A}$ requests a signature from the Sign_A oracle for a message m, $\mathcal{B}$ chooses $r, u, v \in \mathbb{Z}_N$ randomly and outputs the signature

$$\sigma = (g^r X_3^u,\ g^\alpha (u^m h)^r X_3^v).$$

Likewise, if $\mathcal{A}$ requests a signature from the Sign_B oracle for a message m, then $\mathcal{B}$ chooses $r, u, v \in \mathbb{Z}_N$ randomly and outputs the signature

$$\sigma = (g^r(Y_2Y_3)^u,\ g^\alpha(u^m h)^r(Y_2Y_3)^v).$$

Finally, $\mathcal{A}$ will query $\mathcal{B}$ on some challenge message, m. $\mathcal{B}$ will choose a random $u \in \mathbb{Z}_N$, and return the challenge signature

$$\sigma = (T,\ g^\alpha T^{am+b} X_3^u).$$

Clearly, if $T \in \mathbb{G}_{p_1p_3}$, then the challenge signature will be a correctly formed signature from the Sign_A algorithm, and if $T \in \mathbb{G}$, then the challenge signature will be a correctly formed signature from the Sign_B algorithm. Note that $am+b$ modulo p_2 will not be correlated with a and b modulo p_1, so the second element will be randomly distributed in the $\mathbb{G}_{p_2}$ subgroup.

Once $\mathcal{A}$ returns the forgery, (σ^*, m^*), $\mathcal{B}$ must first check that $\mathcal{A}$ has not seen a signature for m^* before and that (σ^*, m^*) verifies. If either of these checks fail then $\mathcal{B}$ will guess randomly. If both of these are true, then $\mathcal{B}$ must determine what forgery class (σ^*, m^*) belongs to in order to determine what subgroup T is in. To distinguish between the forgery types, $\mathcal{B}$ must use a backdoor verification test similar to the one used in the proof of Lemma C.1. $\mathcal{B}$ sets

$$C_0 = e(X_1X_2, g)^\alpha,\ C_1 = (X_1X_2)^{am^*+b},\ C_2 = X_1X_2,$$

and checks if

$$\frac{e(\sigma_1, C_2)}{e(\sigma_2, C_1)} = C_0.$$

As we did in the proof of Lemma C.1, we can separate X_1X_2 into its subgroup components,

$$X_1X_2 = g^s g_2^{x_2},$$

where g_2 is some generator for the subgroup $\mathbb{G}_{p_2}$ and s and x_2 are random exponents from $\mathbb{Z}_N^*$. As before, note that since (σ^*, m^*) passes the real verification equation, it will pass this backdoor verification in the $\mathbb{G}_{p_1}$ subgroup. We can also rewrite the forgery elements in terms of their subgroup components,

$$\begin{aligned}\sigma_2^* &= g^r g_2^{s_{2,2}} g_3^{s_{2,3}} \\ \sigma_1^* &= g^{\alpha + r(am^*+b)} g_2^{s_{1,2}} g_3^{s_{1,3}},\end{aligned}$$

where g_3 is a generator for $\mathbb{G}_{p_3}$ and r and each $s_{i,j}$ are exponents in $\mathbb{Z}_N$.

The backdoor verification equation then becomes,

$$\begin{aligned}\frac{e(\sigma_1, C_2)}{e(\sigma_2, C_1)} &= \frac{e(g,g)^{\alpha s} e(g^{am^*+b}, g)^{rs} e(g_2, g_2)^{s_{1,2}x_2}}{e(g^{am^*+b}, g)^{rs} e(g_2, g_2)^{s_{2,2}x_2(am^*+b)}} \\ &= e(g,g)^{\alpha s} e(g_2, g_2)^{x_2(s_{1,2} - s_{2,2}(am^*+b))} \\ &= C_0 e(g_2, g_2)^{x_2(s_{1,2} - s_{2,2}(am^*+b))} \stackrel{?}{=} C_0.\end{aligned}$$

This gives us a result similar to the proof of Lemma C.1. However, in this case we know that x_2 is nonzero, so a forgery will only pass this verification test if $s_{1,2} = s_{2,2}(am^*+b)$. Thus, if the forgery fails the test, then with probability 1 it is a Type II forgery. If the forgery passes the test then it can be either Type I or Type II. If σ^* is a Type I forgery, then $s_{2,2} = s_{1,2} = 0$, and it will pass this backdoor verification test. We claim that a Type II forgery can also pass the additional verification test, but only with negligible probability.

In order to create a Type II forgery where $s_{1,2} = s_{2,2}(am^* + b)$, an attacker must implicitly determine $am^* + b$ modulo p_2. Of course, m^* will be known to the attacker. However, during the query phase, no information about the values of a and b modulo p_2 is revealed. Therefore, if an attacker is to determine the values of a and b modulo p_2, it must be from the challenge signature.

In the challenge signature, a and b are only included in the second signature element. Thus, from the challenge signature, the attacker can derive the single value $am+b$ modulo p_2. However, this single equation has two unknowns (a and b), and it is impossible to determine the unique values of a and b modulo p_2. Moreover, $am+b$ is a pairwise independent function of m modulo p_2. Therefore, the attacker has no better than the negligible probability of achieving the correct value of am^*+b modulo p_2, as long as $m \neq m^*$ modulo p_2. It is possible that m^* is equal to m modulo p_2, but unequal to m modulo N. If this occurs with non-negligible probability, then $\mathcal{B}$ can extract a non-trivial factor of N by computing the g.c.d of N and $m-m^*$. This non-trivial factor can be used to break Assumption 4.2 with non-negligible advantage, as proven in [56]. Thus, we may assume that $m \neq m^*$ modulo p_2, except with negligible probability. Hence, if a forgery passes the additional verification test, then with high probability it is a Type I forgery. □

Lemma C.3. If Assumption C.1 holds, then our signature scheme is B-II Matching.

Proof. Suppose that there exists an attacker, $\mathcal{A}$, that can create a Type I forgery with non-negligible probability ϵ given access to an oracle for the Sign_B algorithm. Then we can create an algorithm $\mathcal{B}$ that breaks Assumption C.1 with non-negligible advantage.

$\mathcal{B}$ first receives $g, g^\alpha X_2, X_3, g^s Y_2, Z_2$. $\mathcal{B}$ then chooses random exponents $a, b, \in \mathbb{Z}_N$ and sets $PK = (u = g^a, g = g, h = g^b, e(g,g)^\alpha = e(g^\alpha X_2, g))$ and $\text{SK} = (\alpha, X_3, Z_2 X_3)$. $\mathcal{B}$ does not know α, but can use $g^\alpha X_2$ to correctly simulate the Sign_B algorithm.

If $\mathcal{A}$ requests the signature for a message m, $\mathcal{B}$ chooses exponents $r, u, v, c, d \in \mathbb{Z}_N$ and outputs the signature

$$\sigma = (g^r Z_2^c X_3^u,\ g^\alpha X_2 (u^m h)^r Z_2^d X_3^v)$$

This is clearly a verifiable signature and it will be randomly distributed in the $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ subgroups.

After the query phase, $\mathcal{A}$ will output some forgery, (m^*, σ^*). First, $\mathcal{B}$ will check that the forgery correctly verifies. If the forgery fails verification, then $\mathcal{B}$ will guess randomly. If the forgery verifies, then $\mathcal{B}$ can use this forgery to determine whether $T = e(g,g)^{\alpha s}$ or if it is randomly selected from $\mathbb{G}_T$. $\mathcal{B}$ will use a backdoor verification test similar to the one used in the proofs of Lemmas C.1 and Lemma C.2. First, $\mathcal{B}$ sets

$$C_0 = T,\ C_1 = (g^s Y_2)^{am^*+b},\ C_2 = g^s Y_2.$$

Since a and b are chosen randomly modulo N, there will be no correlation between the $\mathbb{G}_{p_1}$ and the $\mathbb{G}_{p_2}$ components of C_1. Finally, $\mathcal{B}$ will check that

$$\frac{e(\sigma_1, C_2)}{e(\sigma_2, C_1)} = C_0.$$

We can represent T by

$$T = e(g,g)^{\alpha s r'},$$

where $r' = 1$ if $T = e(g,g)^{\alpha s}$ and $r' \neq 1$ with high probability if T is randomly chosen from $\mathbb{G}_T$. We will use the same representation of σ^* as in the proof of Lemmas C.1 and C.2. If σ^* is a Type I forgery, we know that $s_{2,2}$ and $s_{1,2}$ must be zero.

The backdoor verification equation now becomes:

$$\begin{aligned}\frac{e(\sigma_1, C_2)}{e(\sigma_2, C_1)} &= \frac{e(g,g)^{\alpha s} e(g^{am^*+b}, g)^{rs} e(g_2, Y_2)^{s_{1,2}}}{e(g^{am^*+b}, g)^{rs} e(g_2, Y_2)^{s_{2,2}(am^*+b)}} \\ &= e(g,g)^{\alpha s} e(g_2, Y_2)^{s_{1,2} - s_{2,2}(am^*+b)} \stackrel{?}{=} C_0.\end{aligned}$$

If the equation verifies correctly, then $\mathcal{B}$ will output 1. If the equation fails, then $\mathcal{B}$ will flip a coin $b \in \{0,1\}$ and output b. As before, notice that if $\mathcal{B}$ attempted to create its own signatures using T that they would always pass this verification test. This means that $\mathcal{B}$ cannot gain any advantage over the Assumption C.1 challenger without the output of $\mathcal{A}$.

Suppose that T is selected randomly from $\mathbb{G}_T$, then with all but negligible probability $r' \neq 1$. This means that the additional verification equation will fail with all but with some negligible probability, δ. In this case $\mathcal{B}$ will output 1 with probability 1/2.

On the other hand, consider the case where $T = e(g,g)^{\alpha s}$. With probability ϵ, $\mathcal{A}$ will produce a Type I forgery where it is guaranteed that $s_{2,2} = s_{1,2} = 0$. In this case, since (σ^*, m^*) passes the real verification test, it must pass the backdoor verification test, meaning that $\mathcal{B}$ will output 1 with probability 1.

If $T = e(g,g)^{\alpha s}$ and $\mathcal{A}$ produces a Type II forgery, where $s_{1,2} \neq s_{2,2}(am+b)$, then the verification equation will be false and $\mathcal{B}$ will output 1 with probability 1/2. If $\mathcal{A}$ produces a Type II forgery where $s_{1,2} = s_{2,2}(am+b)$, since (σ^*, m^*) passes the real verification test, the equation will be true and $\mathcal{B}$ will output 1. However, a and b modulo p_2 are not given out at all during the query phase, so the probability of $\mathcal{A}$ achieving this correlation will be some negligible probability, δ'.

Given all these possibilities, we can calculate the advantage of $\mathcal{B}$ as follows,

$$\begin{aligned}
|Pr[\mathcal{B}(D,T_1)=1] - Pr[\mathcal{B}(D,T_2)=1]| &= \left| Pr[\mathcal{B}(D,T_1)=1] - \left(\delta Pr[\mathcal{B}(D,T_1)=1] + (1-\delta)\cdot\frac{1}{2} \right) \right| \\
&= \left| (1-\delta)\left(Pr[\mathcal{B}(D,T_1)=1] - \frac{1}{2} \right) \right| \\
&= (1-\delta)\left(\left(\epsilon + (1-\epsilon)\left(\delta' + (1-\delta')\frac{1}{2} \right) \right) - \frac{1}{2} \right) \\
&= (1-\delta)\left(\frac{1}{2}\epsilon + \frac{1}{2}\delta' - \frac{1}{2}\epsilon\delta' \right),
\end{aligned}$$

which is non-negligible. □