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

Michael Gerbush[1], Allison Lewko[2], Adam O'Neill[3], and Brent Waters[4] ⋆

[1] The University of Texas at Austin
mgerbush@cs.utexas.edu
[2] Microsoft Research
allew@microsoft.com
[3] Boston University
amoneill@bu.edu
[4] The 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, and 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.

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., [17,

⋆ 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.

13, 24, 48]) that are both practical and have added structure which has been used to build other primitives ranging from Aggregate Signatures [15, 43] to Oblivious Transfer [25, 32]. Ideally, for such a fundamental cryptographic primitive we would like to have security proofs from straightforward, static complexity assumptions.

Meeting this goal for certain systems is often challenging. For instance, the Camenisch and Lysyanskaya signature scheme [24][5] has been very influential as it is used as the foundation for a wide variety of advanced cryptographic systems, including anonymous credentials [24, 7, 6], group signatures [24, 5], ecash [22], uncloneable functions [21], batch verification [23], and RFID encryption [4]. 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 [44], 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 [9, 10] were also proven in the random oracle model under an 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 [9] relied on an assumption that was shown to be false, and the scheme was insecure [36]. This scheme and proof were corrected in [10]. 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 [48] were used in the lattice setting in [20], techniques from [26] were used in [27], and techniques from [12] 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

[5] 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.

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.

A Dual Form Signature scheme immediately gives a secure signature scheme if we simply set the signing algorithm $\mathrm{Sign} = \mathrm{Sign}_A$. Unforgeability now follows from a hybrid argument. Consider any EUF-CMA [31] 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 [49] 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 [41]. 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 [49, 41]: 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 [41]): 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.

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 [24, 48]. 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 the full version, we provide one more application: a signature scheme using the private key structure in the Lewko-Waters IBE system [41]. The LW system itself can be viewed as a composite order extension of the Boneh-Boyen selectively secure IBE scheme [11], 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 [6] 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.

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 [45] 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 [45] 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 [16, 34, 18, 33, 19, 47, 35, 38, 37, 28, 29, 40, 46]. The general techniques presented in [28, 39] 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. Discussion of additional related works can be found in the full version.

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.

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.

[6] Naor's observation was noted in [14].

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.

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 $\mathcal{V}$. [7] 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.

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.

[7] 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.

***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.

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 *: KeyGen = KeyGen^{DF}, Sign = Sign$_A^{DF}$, Verify = 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.

In the full version, we prove (the argument is rather straightforward):

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

3 Background on Composite Order Bilinear Groups

Composite order bilinear groups were first introduced in [16]. 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 [24]. 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, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. 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$_A$(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

$$\sigma = (g^r R_{2,3}^{r'} R_3,\ (g^r)^y (R_{2,3}^{r'})^{y_e} R'_3,\ (g^r)^{x+mxy} (R_{2,3}^{r'})^{x_e+mx_ey_e} R''_3).$$

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.

Sign$_B$(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$_A$.

Verify(VK, m, σ)**:** 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 forgeries are of the form $\mathcal{V}_I = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_1^*)^z = g_2^{r'}, (\sigma_2^*)^z = g_2^{r' y_e}, (\sigma_3^*)^z = g_2^{r'(x_e + m^* x_e y_e)} \text{ for some } r'\}$, while Type II are of the form$\mathcal{V}_{II} = \{(m^*, \sigma^*) \in \mathcal{V} | (m^*, \sigma^*) \notin \mathcal{V}_I\}$.

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 [41], 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 [8]. The third assumption is new, and in the full version 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 41 *Given a group generator $\mathcal{G}$, we define the following distribution:*

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

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

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

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

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 41 to be:

$$Adv_{\mathcal{A}}^{41}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 1. *We say that $\mathcal{G}$ satisfies Assumption 41 if for any polynomial time algorithm, $\mathcal{A}$, $Adv_{\mathcal{A}}^{41}(\lambda)$ is a negligible function of λ.*

Assumption 42 *Given a group generator $\mathcal{G}$, we define the following distribution:*

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

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

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, X_1X_2, X_3, Y_2Y_3),$$

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

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 42 to be:

$$Adv^{42}_{\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|.$$

Definition 2. *We say that $\mathcal{G}$ satisfies Assumption 42 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{42}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

Assumption 43 *Given a group generator $\mathcal{G}$, we define the following distribution:*

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

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

$$D = (N, \mathbb{G}, \mathbb{G}_T, e, g, g^a, g^rX_2, g^{ra}X_2', g^{ra^2}X_2'', Z_2, X_3),$$

We define the advantage of an algorithm, $\mathcal{A}$, in breaking Assumption 43 to be:

$$Adv^{43}_{\mathcal{A}}(\lambda) := Pr[\mathcal{A}(D) = (g^{r'a^2}R_3, g^{r'}R_3') \text{ and } r' \neq 0 \bmod p_1],$$

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 43 if for any polynomial time algorithm, $\mathcal{A}$, $Adv^{43}_{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

Proof of Security In the full version, we prove that our signature scheme is secure under these assumptions by proving that it satisfies the three properties of a secure dual form signature scheme.

5 BGOY Signatures

Here we give a public key variant of the BGOY signatures and prove existential unforgeability using our dual form framework. In the full version, 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 $N = p_1 p_2 p_3$, 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 [30] 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$(\lambda) \to \text{VK}, \text{SK}$ The key generation algorithm chooses a bilinear group $\mathbb{G}$ of order $N = p_1 p_2 p_3$. 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, SK) → σ 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, SK) → σ 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, σ, VK) → {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 forgeries are of the form $\mathcal{V}_I = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_1^*)^{z_2} = 1, (\sigma_1^*)^{z_3} = 1 \text{ and } (\sigma_5^*)^{z_2} = 1, (\sigma_5^*)^{z_3} = 1\}$, while Type II are of of the form $\mathcal{V}_{II} = \{(m^*, \sigma^*) \in \mathcal{V} | (\sigma_1^*)^{z_2} \neq 1 \text{ or } (\sigma_5^*)^{z_2} \neq 1 \text{ or } (\sigma_1^*)^{z_3} \neq 1 \text{ or } (\sigma_5^*)^{z_3} \neq 1\}$.

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.

We state our complexity assumptions and prove security of this scheme in the full version. Some the assumptions we employ were previously used in [41, 42]. Those that are new are justified in the generic group model.

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. pages 1–20, 2012.
4. 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.
5. 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/`.
6. Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *WPES*, pages 40–46, 2005.
7. 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.
8. Mihir Bellare, Brent Waters, and Scott Yilek. Identity-based encryption secure against selective opening attack. In *TCC*, pages 235–252, 2011.
9. 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.
10. 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/`.
11. 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.
12. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.
13. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology — EUROCRYPT '04*, volume 3027, pages 54–73, 2004.
14. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

15. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT*, pages 416–432, 2003.
16. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–341, 2005.
17. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.
18. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, pages 573–592, 2006.
19. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.
20. Xavier Boyen. Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In *Public Key Cryptography*, pages 499–517, 2010.
21. 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.
22. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT*, pages 302–321, 2005.
23. Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen. Batch verification of short signatures. In *EUROCRYPT*, pages 246–263, 2007.
24. 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.
25. Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.
26. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.
27. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.
28. David M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. To appear in Advances in Cryptology – EUROCRYPT 2010, 2010.
29. 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.
30. Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In *Public Key Cryptography*, pages 257–273, 2006.
31. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2), 1988.
32. Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, pages 265–282, 2007.
33. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *CRYPTO*, pages 97–111, 2006.
34. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.
35. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

36. 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.
37. Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In *Pairing*, pages 75–88, 2008.
38. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.
39. Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.
40. 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.
41. Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–479, 2010.
42. Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.
43. 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.
44. 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.
45. Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *CT-RSA*, pages 376–392, 2011.
46. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
47. 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.
48. Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494, pages 320–329, 2005.
49. Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.