# Secure Blind Decryption

Matthew Green
Information Security Institute
Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218
mgreen@cs.jhu.edu

**Abstract**

In this work we construct public key encryption schemes that admit a protocol for *blindly* decrypting ciphertexts. In a blind decryption protocol, a user with a ciphertext interacts with a secret keyholder such that the user obtains the decryption of the ciphertext and the keyholder learns nothing about what it decrypted. While we are not the first to consider this problem, previous works provided only weak security guarantees against malicious users. We provide, to our knowledge, the first practical blind decryption schemes that are secure under a strong CCA security definition. We prove our construction secure in the standard model under simple, well-studied assumptions in bilinear groups. To motivate the usefulness of this primitive we discuss several applications including privacy-preserving distributed file systems and Oblivious Transfer schemes that admit *public* contribution.

**Keywords:** public-key encryption, privacy-preserving protocols, signatures, bilinear maps.

## 1 Introduction

The past several years have seen a trend towards outsourcing data storage to remote data stores and cloud-based services. While much attention has been paid to securing the data on these services, less has been given to protecting the privacy of users who access it. This is already an acknowledged problem for corporations whose employees access public resources such as patent databases, since their access patterns can leak valuable information about corporate strategy. However, privacy concerns are hardly restricted to these specialized applications. For example, the internal data access patterns (*i.e.,* list of files accessed) of a corporation's executives could be worth millions of dollars to the right person, particularly in advance of a merger or IPO.

To address these concerns, some recent works have proposed tools that allow allow users to transact online without sacrificing their privacy. These tools include (but are not limited to) efficient adaptive oblivious transfer protocols [16, 29, 30, 47], anonymous credential schemes [14, 4], and group signature schemes [17, 7]. One recent application for these tools is to the construction of *oblivious databases* that provide strong access control while preventing the operator from learning which records its users access [21, 13]. Despite this progress, there are still many primitives that we do not know how to implement efficiently using the techniques available to us.

Blind Decryption. In this work we consider one such primitive. A *blind decryption* scheme is a public-key encryption (PKE) scheme that admits an efficient protocol for obliviously decrypting ciphertexts. In this protocol a User who possesses a ciphertext interacts with a Decryptor who holds the necessary secret key. At the conclusion of the protocol, the User obtains the plaintext while the Decryptor learns nothing about what it decrypted. Furthermore, the User should gain no information about any other ciphertext.[1] To formalize

[1]Without this restriction, such a protocol can be achieved trivially, *e.g.,* by having the Decryptor simply give the secret key to the User.

the latter guarantee, we will restrict our investigation to secure blind decryption schemes that retain their security under *adaptive chosen ciphertext attack*.

Blind decryption has many applications to privacy-preserving protocols and systems. For example, blind decryption implies $k$-out-of-$N$ oblivious transfer [12], which is important theoretically as well as practically for its applications to the construction of oblivious databases [16, 21, 13]. Moreover, blind decryption has practical applications to distributed cryptographic filesystems and for supporting rapid deletion [46].

We are not the first to consider the problem of constructing blind decryption schemes. The primitive was originally formalized by Sakurai and Yamane [48] in the mid-1990s, but folklore solutions are thought to have predated that work by more than a decade. Despite an abundance of research in this area, most proposed constructions are insecure under adaptive chosen ciphertext attack [25, 52, 43, 24, 50, 45]. Several protocols have recently been proposed containing "blind decryption-like" techniques (see *e.g.,* the simulatable oblivious transfer protocols of [16, 30, 47, 31, 35]). However, these protocols use symmetric (or at least, non-public) encryption procedures, and it does not seem easy to adapt them to the public-key model.

Of course, blind decryption is an instance of secure multi-party computation (MPC) and can be achieved by applying general techniques (*e.g.,* [53, 28, 36]) to the decryption algorithm of a CCA-secure PKE scheme. However, the protocols yielded by this approach are likely to be quite inefficient, making them impractical for real-world applications.

**Our Contributions.** In this paper we present what is, to our knowledge, the first practical blind decryption scheme that is IND-CCA2-secure in the standard model. We prove our scheme secure under reasonable assumptions in bilinear groups. At the cost of introducing an optional Common Reference String, the protocol can be conducted in a single communication round.

To motivate the usefulness of this new primitive we consider several applications. Chief among these is the construction of privacy-preserving encrypted filesystems (and databases), where a central authority manages the decryption of many ciphertexts without learning users' access patterns. This is important in situations where the access pattern might leak critical information about the information being accessed. Unlike previous attempts to solve this problem [16, 21, 13], our encryption algorithm is *public*, *i.e.,* users can encrypt new messages offline without assistance from a trusted party. By combining blind decryption with the new oblivious access control techiques of [21, 13] (which use anonymous credentials to enforce complex access control policies) we can achieve strong proactive access control without sacrificing privacy.

Of potential theoretical interest, blind decryption can be used as a building block in constructing adaptive $k$-out-of-$N$ Oblivious Transfer protocols [16, 30, 47, 31, 35, 39]. In fact, it is possible to achieve a multi-party primitive that is more flexible than traditional OT, in that *any* party can commit messages to the message database (rather than just the Sender). We refer to this enhanced primitive as Oblivious Transfer with Public Contribution (OTPC). We discuss these applications in Section 5.

## 1.1 Related Work

The first blind decryption protocol is generally attributed Chaum [20], who proposed a technique for blinding an RSA ciphertext in order to obtain its decryption $c^d \ mod \ N$. Since traditional RSA ciphertexts are malleable and hence vulnerable to chosen ciphertext attack, this approach does not lead to a secure blind decryption scheme. Furthermore, standard encryption padding techniques [5] do not seem helpful.

Subsequent works [48, 25, 52] adapted Chaum's approach to other CPA-secure cryptosystems such as Elgamal. These constructions were employed within various protocols, including a 1-out-of-$N$ Oblivious Transfer scheme due to Dodis *et al.* [25]. Unfortunately, since the cryptosystems underlying these protocols are not CCA-secure, security analyses were frequently conducted in weak security models with honest-but-curious adversaries.[2] Mambo, Sakurai and Okamoto [43] proposed to address chosen ciphertext attacks by signing the ciphertexts to prevent an adversary from mauling them. Their *transformable signature* could be

[2]For example, Dodis *et al.* [25] analyzed their 1-out-of-$N$ oblivious transfer construction in the honest-but-curious model. However, the authors informally proposed to deal with malicious adversaries by having the OT Receiver prove in zero-knowledge that each ciphertext to be decrypted belongs to an honestly-generated set published by the Sender. Such proofs are not efficient: using traditional techniques, this requires $O(N)$ effort [49, 22]. More fundamentally, this approach does not extend to more complex protocols, since it assumes that there is only one (trusted) party performing all encryption.

blinded in tandem with the ciphertext. The trouble with this approach and other related approaches [16, 30, 31, 35, 47] is that the encryption scheme is no longer a PKE, since encryption now requires a knowledge of a secret signing key (furthermore, these transformable signatures were successfully cryptanalyzed [24]). Schnorr and Jakobsson [50] proposed a scheme secure under the weaker *one-more decryption attack* and used this to construct a PIR protocol. Unfortunately, their protocol is secure only for *random* messages, and furthermore cannot be extended to construct stronger primitives such as simulatable OT [16].

Recently, Green and Hohenberger [29] proposed a technique for blindly extracting decryption keys in an Identity-Based Encryption scheme. Ogata and Le Trieu [45] subsequently used this tool to implement a weak blind decryption scheme (by encrypting ciphertexts under a random identity, then blindly extracting the appropriate secret key). This approach is efficient, however the ciphertexts are malleable and thus vulnerable to adaptive chosen ciphertext attack.

## 1.2 Intuition

To develop a blind decryption scheme we would ideally begin with an existing CCA-secure public-key encryption scheme, then add to it an efficient two-party protocol for computing the decryption algorithm. Indeed, the literature provides us with many candidate PKE constructions that can be so adapted if we are willing to accept the costs associated with general multi-party computation techniques [53, 28, 36].

However, in this work we are interested in protocols that are both secure and *practical*. This rules out inefficient gate-by-gate decryption protocols, limiting us to a relatively small collection of techniques that can be used to build efficient protocols. This toolbox includes primitives such as homomorphic commitment schemes, which we might combine with zero knowledge proofs for statements involving algebraic relations among cyclic group elements, *e.g.,* [49, 32]. While these techniques have been deployed successfully to construct other privacy-preserving protocols, there are strict limitations on what they can accomplish.

To illustrate this point, let us review several of the most popular encryption techniques in the literature. Random oracle paradigms such as OAEP [5] and Fujisaki-Okamoto [27] seem fundamentally difficult to adapt, since these approaches require the decryptor to evaluate an ideal hash function on a partially-decrypted value *prior* to outputting a result. Even the more efficient standard-model CCA-secure paradigms such as Cramer-Shoup [23] and recent bilinear constructions (*e.g.,* [8, 10, 37]) require components that we cannot efficiently adapt. For example, when implemented in a group $\mathbb{G}$ of order $p$, the Cramer-Shoup scheme assumes a collision-resistant mapping $H : \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_p$. We know of no efficient two-party technique for evaluating such a function.[3]

*Our approach.* Rather than adapt an existing scheme, we set out to design a new one. Our approach is based on the TBE-to-PKE paradigm proposed independently by Canetti, Halevi and Katz [19] and MacKenzie, Reiter and Yang [42]. This technique converts a Tag-Based Encryption (TBE) scheme into a CCA-secure PKE with the assistance of a strongly unforgeable one-time signature (OTS). In this generic transform, encryption is conducted by first generating a keypair $(vk, sk)$ for the OTS, encrypting the message using the TBE with $vk$ as the tag, then signing the resulting ciphertext with $sk$. Intuitively the presence of the signature (which is verified at decryption time) prevents an adversary from mauling the ciphertext.

To blindly decrypt such a ciphertext, we propose the following approach: the User first commits to the ciphertext and $vk$ using a homomorphic commitment or encryption scheme. She then efficiently proves knowledge of the associated signature for these committed values. If this proof verifies, the Decryptor may then apply the TBE decryption algorithm to the (homomorphically) committed ciphertext, secure in the knowledge that the commitment contains an appropriately-distributed value. Finally, the result can be opened by the User.

For this protocol to be efficient, we must choose our underlying primitives with care. Specifically, we must ensure that (1) the OTS verification key maps to the tag-space of the TBE, (2) and the TBE ciphertext maps to the message space of the OTS. Of course, the easiest way to achieve these goals is to use an OTS that directly signs the TBE ciphertext space, with a TBE whose tag-space includes the OTS verification

[3] Conceivably it might be possible to develop such hash functions for various popular constructions of $\mathbb{G}$, and we consider this an interesting open question. However this approach might be quite inflexible.

keyspace. These primitives must admit efficient protocols for the operations we will conduct with them. Finally, we would like to avoid relying on complex or novel complexity assumptions in order to achieve these goals.

Our proposed construction is based on a variant of Cramer-Shoup that was adapted by Shacham [51] for security in bilinear groups. We first modify Shacham's construction into a TBE with the following ciphertext structure. Let $\alpha \in \mathbb{Z}_p^*$ be an arbitrary ciphertext tag and $m \in \mathbb{G}$ a message to be encrypted. Given a public key $g, g_1, g_2, g_3, h_1, h_2, c_1, c_2, d_1, d_2 \in \mathbb{G}$ an encryptor selects random elements $r_1, r_2 \in \mathbb{Z}_p^*$ and outputs the ciphertext:

$$(u_1, u_2, u_3, e, v, vk) \quad = \quad (g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2}, m \cdot h_1^{r_1} h_2^{r_2}, (c_1 d_1^{\alpha})^{r_1} \cdot (c_2 d_2^{\alpha})^{r_2}, g^{\alpha})$$

An important feature of this construction is that the decryptor does *not* need to know the tag value $\alpha$.[4] Therefore, in constructing our PKE we can "dual-purpose" $\alpha$ as both the ciphertext tag *and* as the secret key of a one-time signature (OTS) scheme. Specifically, our encryption process will select a random $\alpha$, encrypt the message using the TBE with $\alpha$ as the tag, and finally sign the ciphertext elements $(u_1, u_2, u_3, e, v)$ with $\alpha$. The resulting ciphertext consists of $(u_1, u_2, u_3, e, v, vk)$ along with the signature.

The remaining challenge is therefore to construct an efficient OTS that can sign multiple bilinear group elements, yet admits an efficient proof-of-knowledge for a signature on committed elements. To address this we propose a new multi-block one-time "$F$-signature" that we believe may be of independent interest.[5] Interestingly, our signing algorithm does not actually operate on elements of $\mathbb{G}$, but rather signs message vectors of the form $(m_1, \ldots, m_n) \in \mathbb{Z}_p^{*n}$ (for some arbitrary vector length $n$). Once a message is signed, however, the signature can be *verified* given the tuple $(g_1^{m_1}, \ldots, g_n^{m_n}) \in \mathbb{G}^n$ for some $g_1, \ldots, g_n$, rather than the original message vector. Strictly speaking, this construction does not meet our requirements—an encryptor may not be able to calculate the discrete logarithm of the ciphertext elements $(u_1, u_2, u_3, e, v)$, especially when the message $m$ is adversarially-chosen. Our key insight is to show that encryptors can produce an identically distributed "workalike" signature even when these discrete logarithms are not known. We prove that, in the context of our encryption scheme, no adversary can forge these workalike signatures. Our signature construction is presented in Section 2.4.

# 2 Technical Preliminaries

## 2.1 Bilinear Groups and Cryptographic Assumptions

Let $\lambda$ be a security parameter. We define $\mathsf{BMsetup}$ as an algorithm that, on input $1^\lambda$, outputs the parameters for a bilinear mapping as $\gamma = (p, \mathbb{G}, \mathbb{G}_T, e, g \in \mathbb{G})$, where $g$ generates $\mathbb{G}$, the groups $\mathbb{G}, \mathbb{G}_T$ each have prime order $p$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. For $\langle g \rangle = \langle h \rangle = \mathbb{G}$ the efficiently-computable mapping $e$ must be both *non-degenerate* ($\langle e(g, h) \rangle = \mathbb{G}_T$) and *bilinear* (for $a, b \in \mathbb{Z}_p^*$, $e(g^a, h^b) = e(g, h)^{ab}$).

**The Decision Linear Assumption (DLIN)** [7]. Let $\mathbb{G}$ be a group of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$, the following probability is 1/2 plus an amount negligible in $\lambda$:

$$\begin{aligned} \Pr[f, g, h, z_0 \xleftarrow{R} \mathbb{G}; a, b \xleftarrow{R} \mathbb{Z}_p^*; z_1 \leftarrow h^{a+b}; d \xleftarrow{R} \{0, 1\}; \\ d' \leftarrow \mathcal{A}(f, g, h, f^a, g^b, z_d) : d = d']. \end{aligned}$$

**The Flexible Diffie-Hellman Assumption (FDH)** [38, 31]. Let $\mathbb{G}$ be a group of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$, the following probability is negligible in $\lambda$:

$$\Pr[g, g^a, g^b; a, b \xleftarrow{R} \mathbb{Z}_p^*; (w, w') \leftarrow \mathcal{A}(g, g^a, g^b) : w \neq 1 \ \wedge \ w' = w^{ab}].$$

[4] This differs from many other candidate TBE and IBE schemes, *e.g.,* Boneh and Boyen's IBE [6] and Kiltz's TBE [37] where the tag/identity is an element of $\mathbb{Z}_p^*$ and *must* be provided at decryption time (or in the case of IBE, when a secret key is extracted). This requirement stems from the nature of those schemes' security proofs.

[5] $F$-signature is a contraction of $F$-unforgeable signature, which is a concept proposed by Belinkiy *et al.* [4], and later developed by Green and Hohenberger [31]. In this paradigm, the signing algorithm operates on a message $m$, but there exists a signature verification algorithm that can operate given only $F(m)$ for some one-way function $F$.

This assumption was previously described as the 2-out-of-3 CDH assumption by Kunz-Jacques and Pointcheval [38]. We adopt the name Flexible Diffie-Hellman for consistency with recent work [41, 26, 31]. To instill confidence in this assumption, Green and Hohenberger [31] showed that a solver for the Flexible Diffie-Hellman problem implies a solver for a related decisional problem, the Decisional 3-Party Diffie-Hellman assumption (3DDH) which has been used several times in the literature [40, 9, 33, 31]. An adversary solves the 3DDH problem if, given $(g, g^a, g^b, g^c, z)$ for some random $a, b, c \in \mathbb{Z}_p^*$, it outputs 1 if $z = g^{abc}$ and 0 otherwise.

## 2.2 Proofs of Knowledge

We use several standard results for proving statements about the satisfiability of one or more pairing-product equations. For variables $\{\mathcal{X}\}_{1...n} \in \mathbb{G}$ and constants $\{\mathcal{A}\}_{1...n} \in \mathbb{G}, a_{i,j} \in \mathbb{Z}_p^*$, and $t_T \in \mathbb{G}_T$, these equations have the form:

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \prod_{i=1}^{n} \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{i,j}} = t_T$$

The proof-of-knowledge protocols in this work can be instantiated using one of two approaches. The first approach is to use the interactive zero-knowledge proof technique of Schnorr [49], with extensions due to *e.g.,* [22, 15, 18, 11, 2, 16]. Note that this may require that the proofs be executed sequentially (indeed, this requirement is explicit in our security definitions). For details, see the work of Adida *et al.* [2], which provides a taxonomy of interactive proof techniques for pairing-based statements.

Alternatively, the proofs can be instantiated using the Groth-Sahai proof system [32] which permits efficient non-interactive proofs of the satisfiability of multiple pairing product equations. In the general case these proofs are witness indistinguishable. However a subset of special cases (including where $t_T = 1$) may be conducted in zero-knowledge.[6] The Groth-Sahai system can be instantiated under the Decision Linear assumption in the Common Reference String model. An important limitation of the Groth-Sahai proof system has to do with the knowledge extractor used to show proof soundness: specifically, the extractor can only extract elements of the bilinear image group $\mathbb{G}$ (not $\mathbb{Z}_p^*$ or $\mathbb{G}_T$). We have designed our constructions with this restriction in mind. We refer the reader to [32] for further details.

We refer the reader to the cited works for formal security definitions of ZK and WI proof systems. In our security analysis we will assume some generic instantiation $\Pi_{ZK}$ that is secure under the Decision Linear assumption in $\mathbb{G}$. Either of the techniques mentioned above can satisfy this requirement. When referring to WI and ZK proofs we will use the notation of Camenisch and Stadler [17]. For instance, $\mathsf{WIPoK}\{(g, h) : e(g, h) = T \wedge e(g, v) = 1\}$ denotes a witness indistinguishable proof of knowledge of elements $g$ and $h$ that satisfy both $e(g, h) = T$ and $e(g, v) = 1$. All values not in enclosed in ()'s are assumed to be known to the verifier.

## 2.3 Linear Encryption

Our blind decryption protocol employs a multiplicatively homomorphic scheme that encrypts elements of $\mathbb{G}$. We instantiate this scheme with the Linear Encryption scheme of Boneh, Boyen and Shacham [7] which is semantically secure under the Decision Linear assumption. The scheme consists of the following algorithms:

- $\mathsf{LE.KG}$. On input a group description $\gamma$, pick $h \xleftarrow{R} \mathbb{G}$, $x, y \xleftarrow{R} \mathbb{Z}_p^*$, $f = h^{1/x}, g = h^{1/y}$ and output $pk = (f, g, h)$, $sk = (x, y)$.
- $\mathsf{LE.Enc}$. On input $pk$ and a message $m \in \mathbb{G}$, select $a, b \xleftarrow{R} \mathbb{Z}_p^*$ and output $(c_1, c_2, c_3) = (f^a, g^b, mh^{a+b})$.
- $\mathsf{LE.Dec}$. On input $sk, (c_1, c_2, c_3)$, output $m' = c_3/(c_1^x c_2^y)$.

[6] In many cases it is easy to re-write pairing products equation as a composition of multiple distinct equations having $t_T = 1$ (see [32]). Although we do not explicitly perform this translation in our protocols, we note that it can be applied to all of the ZKPoKs used in our constructions.

The homomorphic operation is simple pairwise multiplication, and exponentiation by a scalar $z$ can be performed as $c_1^z, c_2^z, c_3^z$. To re-randomize a ciphertext one multiplies it by $\mathsf{LE.Enc}(pk, 1)$.

Our protocols require an efficient ZK proof-of-knowledge of the plaintext $m$ underlying a ciphertext $C$, which we denote by $\mathsf{ZKPoK}\{(m) : C \in \mathsf{LE.Enc}(pk, m)\}$. This can be expressed via the following pairing product equation:

$$\mathsf{ZKPoK}\{(h^a, h^b) : e(h^a, f) = e(c_1, h) \ \wedge\ e(h^b, g) = e(c_2, h)\}$$

Knowledge of $m$ is implicit in this proof, since $m = c_3/(h^a h^b) = c_3/(c_1^x c_2^y)$.[7]

## 2.4 A One-Time $F$-Signature on Multiblock Messages

Our constructions require a strongly unforgeable one-time $F$-signature scheme that signs messages of the form $(m_1, \ldots, m_N) \in \mathbb{Z}_p^{*n}$ (for arbitrary values of $n$), but can *verify* signatures given only a function of the messages, specifically, $(g_1^{m_1}, \ldots, g_n^{m_n}) \in \mathbb{G}^n$ for fixed $g_1, \ldots, g_n \in \mathbb{G}$. Note that $g_1, \ldots, g_n$ need not be distinct.

To construct $\mathsf{FS}$, we adapt a weakly-unforgeable signature due to Green and Hohenberger [31] to admit multi-block messages, while simplifying the scheme into a one-time signature. The latter modification has the incidental effect of strengthening the signature to be strongly unforgeable. Let us now describe $\mathsf{FS}$:

$\mathsf{FS.KG}$. On input group parameters $\gamma$, a vector length $n$, select $g, g_1, \ldots, g_n, v, d, u_1, \ldots, u_n \xleftarrow{R} \mathbb{G}$ and $a \xleftarrow{R} \mathbb{Z}_p^*$. Output $vk = (\gamma, g, g^a, v, d, g_1, \ldots, g_n, u_1, \ldots, u_n, n)$ and $sk = (vk, a)$.

$\mathsf{FS.Sign}$. Given $sk$ and a message vector $(m_1, \ldots, m_n) \in \mathbb{Z}_p^{*n}$, first select $r \xleftarrow{R} \mathbb{Z}_p^*$ and output the signature $\sigma = ((\prod_{i=1}^n u_i^{m_i} \cdot v^r\ d)^a, g_1^{am_1}, \ldots, g_n^{am_n}, u_1^{m_1}, \ldots, u_n^{m_n}, r)$.

$\mathsf{FS.Verify}$. Given $pk, (g_1^{m_1}, \ldots, g_n^{m_n})$, parse $\sigma = (\sigma_1, e_1, \ldots, e_n, f_1, \ldots, f_n, r)$, output 1 if the following holds:

$$e(\sigma_1, g) = e(\prod_{i=1}^n f_i \cdot v^r d, g^a) \ \wedge\ \{e(g_i^{m_i}, g^a) = e(e_i, g) \ \wedge\ e(g_i^{m_i}, u_i) = e(g_i, f_i)\}_{i \in [1,n]}$$

Note that verification is a pairing product equation. Thus we can efficiently prove knowledge of a signature using the techniques described in Section 2.2.[8] We denote such a proof by *e.g.,* $\mathsf{WIPoK}\{(\sigma) : \mathsf{Verify}(vk, (g^{m_1}, \ldots, g^{m_n}), \sigma) = 1\}$. Note that $vk$ or the messages may reside within a commitment.

In Appendix A we provide definitions of security, and prove that the scheme $(\mathsf{FS.KG}, \mathsf{FS.Sign}, \mathsf{FS.Verify})$ is strongly unforgeable under the Flexible Diffie-Hellman assumption.

**Workalike signatures.** Our blind decryption constructions make use of the "workalike" algorithms $(\mathsf{WAKG}, \mathsf{WASign})$. While the outputs of these algorithms are identically distributed those of $\mathsf{KG}$ and $\mathsf{Sign}$, the $\mathsf{WASign}$ algorithm operates on messages of the form $(h_1, \ldots, h_n) \in \mathbb{G}^n$. We stress that $(\mathsf{WAKG}, \mathsf{WASign}, \mathsf{Verify})$ is not a secure signature scheme on arbitrary group elements, but can be used securely under the special conditions of our constructions..

$\mathsf{FS.WAKG}$. Select $x_1, \ldots, x_n \xleftarrow{R} \mathbb{Z}_p^*$ and set $(u_1, \ldots, u_n) = (g_1^{x_1}, \ldots, g_n^{x_n})$, with the remaining elements as in $\mathsf{KG}$. Set $sk = (vk, a, x_1, \ldots, x_n)$.

$\mathsf{FS.WASign}$. Given a message vector $(h_1, \ldots, h_n) \in \mathbb{G}^n$, first select $r \xleftarrow{R} \mathbb{Z}_p^*$ and output the signature $\sigma = ((\prod_{i=1}^n h_i^{x_i} \cdot v^r d)^a, h_1^a, \ldots, h_n^a, h_1^{x_1}, \ldots, h_n^{x_n}, r)$.

[7] If using the Groth-Sahai proof system, this proof must be expanded to *e.g.,* $\mathsf{ZKPoK}\{(h^a, h^b, h') : e(h^a, f)e(c_1^{-1}, h') = 1 \ \wedge\ e(h^b, g)e(c_2^{-1}, h') = 1 \ \wedge\ e(h', h) = e(h, h)\}$.

[8] This proof can be conducted natively using Schnorr-type techniques. Unfortunately, the equivalent proof in the Groth-Sahai system may be more complicated, since that system's knowledge extractor cannot extract the element $r \in \mathbb{Z}_p^*$ on which the signature's security depends. Fortunately, the prover can instead prove knowledge of $(h_1, h_2) = (g^{ar}, g^r)$ derived from $r$, and use the following revised verification check: $e(\sigma_1, g) = e(\prod_{i=1}^n f_i \cdot d, g^a)e(h_1, v) \ \wedge\ e(h_1, g) = e(h_2, g^a) \ \wedge\ \{e(g^{m_i}, g^a) = e(e_i, g) \ \wedge\ e(g^{m_i}, u_i) = e(f_i, g)\}_{i \in [1,n]}$.

## 3 Definitions

**Notation:** Let $\mathcal{M}$ be the message space and $\mathcal{C}$ be the ciphertext space. We write $P(\mathcal{A}(a), \mathcal{B}(b)) \rightarrow (c, d)$ to indicate the protocol $P$ is between parties $\mathcal{A}$ and $\mathcal{B}$, where $a$ is $\mathcal{A}$'s input, $c$ is $\mathcal{A}$'s output, $b$ is $\mathcal{B}$'s input and $d$ is $\mathcal{B}$'s output. We will define $\nu(\cdot)$ as a negligible function.

**Definition 3.1 (Blind Decryption Scheme)** A public-key blind decryption scheme consists of a tuple of algorithms $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and a protocol $\mathsf{BlindDec}$.

$\mathsf{KG}(1^\lambda)$. On input a security parameter $\lambda$, the key generation algorithm $\mathsf{KG}$ outputs a public key $pk$ and a secret key $sk$.

$\mathsf{Enc}(pk, m)$. On input a public key $pk$ and a message $m$, $\mathsf{Enc}$ outputs a ciphertext $C$.

$\mathsf{Dec}(pk, sk, C)$. On input $pk, sk$ and a ciphertext $C$, $\mathsf{Dec}$ outputs a message $m$ or the error symbol $\perp$.

The two-party protocol $\mathsf{BlindDec}$ is conducted between a user $\mathcal{U}$ and a decryptor $\mathcal{D}$:

$\mathsf{BlindDec}(\{\mathcal{U}(pk, C)\}, \{\mathcal{D}(pk, sk)\}) \rightarrow (m, \mathsf{nothing})$. On input $pk$ and a ciphertext $C$, an honest user $\mathcal{U}$ outputs the decryption $m$ or the error symbol $\perp$. The decryptor $\mathcal{D}$ outputs $\mathsf{nothing}$ or an error message.

We now present the standard definition of adaptive chosen ciphertext security for public key encryption.

**Definition 3.2** ($\mathsf{IND\text{-}CCA2}$) A public key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is $\mathsf{IND\text{-}CCA2}$ secure if every *p.p.t.* adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has advantage $\leq \nu(\lambda)$ in the following experiment.

$$
\begin{array}{l}
\mathsf{IND\text{-}CCA2}(\Pi, \mathcal{A}, \lambda) \\
\quad (pk, sk) \leftarrow \mathsf{KG}(1^\lambda) \\
\quad (m_0, m_1, z) \leftarrow \mathcal{A}_1^{\mathcal{O}_{dec}(pk, sk, \cdot)}(pk) \text{ s.t. } m_0, m_1 \in \mathcal{M} \\
\quad b \leftarrow \{0, 1\}; c^* \leftarrow \mathsf{Enc}(pk, m_b) \\
\quad b' \leftarrow \mathcal{A}_2^{\mathcal{O}'_{dec}(pk, sk, \cdot)}(c^*, z) \\
\quad \text{Output } b'
\end{array}
$$

Where $\mathcal{O}_{dec}$ is an oracle that, on input a ciphertext $c$, returns $\mathsf{Dec}(pk, sk, c)$ and $\mathcal{O}'_{dec}$ operates identically but returns $\perp$ whenever $c = c^*$. We define $\mathcal{A}$'s advantage in the above game by:

$$|\mathbf{Pr}\,[\, b = b'\,] - 1/2|$$

**Additional security properties.** A secure blind decryption scheme must possess the additional properties of *leak-freeness* and *blindness*. Intuitively, leak-freeness [29] ensures that an adversarial User gains no more information from the blind decryption protocol than she would from access to a standard decryption oracle. Blindness prevents a malicious Decryptor from learning *which* ciphertext a User is attempting to decrypt, even when the Decryptor can induce failures in the protocol. Let us now formally state these properties.

**Definition 3.3 (Leak-Freeness [29])** A protocol $\mathsf{BlindDec}$ associated with a PKE scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is *leak free* if for all *p.p.t.* adversaries $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ such that for every value $\lambda$, no *p.p.t.* distinguisher $D$ can distinguish the output of Game Real from Game Ideal with non-negligible advantage:

**Game Real:** Run $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$ and publish $pk$. As many times as $D$ wants, $\mathcal{A}$ chooses a ciphertext $C$ and atomically executes the $\mathsf{BlindDec}$ protocol with $\mathcal{D}$:
$\mathsf{BlindDec}(\{\mathcal{U}(pk, C)\}, \{\mathcal{D}(pk, sk)\})$. $\mathcal{A}$'s output (which is the output of the game) includes the list of ciphertexts and decrypted plaintexts.

**Game Ideal:** A trusted party runs $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$ and publishes $pk$. As many times as $D$ wants, $\mathcal{S}$ chooses a ciphertext $C$ and queries the trusted party to obtain the output of $\mathsf{Dec}(pk, sk, C)$, if $C \in \mathcal{C}$ and $\perp$ otherwise. $\mathcal{S}$'s output (which is the output of the game) includes the list of ciphertexts and decrypted plaintexts.

In the games above, $\mathsf{BlindDec}$ and $\mathsf{Dec}$ are treated as atomic operations. Hence $D$ and $\mathcal{A}$ (or $\mathcal{S}$) may communicate at any time except during the execution of those protocols. Additionally, while we do not explicitly specify that auxiliary information is given to the parties, this information must be provided in order to achieve a sequential composition property.

**Definition 3.4 (Ciphertext Blindness)** Let $\mathcal{O}_{\mathcal{U}}(pk, C)$ be an oracle that, on input a public key and ciphertext, initiates the User's portion of the $\mathsf{BlindDec}$ protocol, interacting with an adversary. A protocol $\mathsf{BlindDec}(\mathcal{U}(\cdot,\cdot)\ \mathcal{A}(\cdot,\cdot))$ is $\mathsf{Blind}$ secure if every *p.p.t.* adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ has advantage $\leq \nu(\lambda)$ in the following game.

$$\begin{array}{l} \mathsf{Blind}(\mathsf{BlindDec}, \mathcal{A}, \lambda) \\ \quad (pk, C_0, C_1, z) \leftarrow \mathcal{A}_1(1^\lambda) \\ \quad b \leftarrow \{0,1\}; b' \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathcal{U}}(pk, C_b), \mathcal{O}_{\mathcal{U}}(pk, C_{b-1})}(z) \end{array}$$

We define $\mathcal{A}$'s advantage in the above game as: $|\Pr[b' = b] - 1/2|$. Note that a stronger notion of blindness is *selective-failure* blindness, which was proposed by Camenisch *et al.* [16]. While our constructions do not natively achieve this definition, in section 4.2 we discuss techniques for achieving this stronger definition.

We thus arrive at the following definition.

**Definition 3.5 (CCA2-secure Blind Decryption)** A blind decryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{BlindDec})$ is IND-CCA2-secure if and only if: (1) ($\mathsf{KG}$, $\mathsf{Enc}$, $\mathsf{Dec}$) is IND-CCA2-secure, (2) $\mathsf{BlindDec}$ is leak free, and (3) $\mathsf{BlindDec}$ posses the property of ciphertext blindness.

# 4 Constructions

In this section we present a new blind decryption scheme that is secure under the Decision Linear and Flexible Diffie-Hellman assumptions in bilinear groups.

## 4.1 An Efficient Blind Decryption Scheme

We now present our blind decryption scheme $\mathsf{BCS}$, and prove its security under the Decision Linear and Flexible Diffie-Hellman assumptions. $\mathsf{BCS}$ is based on a variant of Cramer-Shoup that was proposed by Shacham [51], with significant extensions to permit blind decryption.

**The core algorithms.** Figure 4.1 details the algorithms $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$, which are responsible for key generation, encryption and decryption respectively. $\mathsf{BCS}$ encrypts elements of $\mathbb{G}$, which may necessitate an encoding scheme from other message spaces (see *e.g.,* [3]). Ciphertexts consist of 24 elements of $\mathbb{G}$ plus two element of $\mathbb{Z}_p^*$. While at first glance these ciphertexts may seem large, note that the scheme can be instantiated in asymmetric bilinear settings such as the MNT group of elliptic curves [44], where group elements can be represented in as little as 170 bits at the 80-bit security level. In this setting we are able to achieve a relatively ciphertext size of approximately 5100 bits. While this is large compared to RSA, a 640-byte per file overhead is quite reasonable for many practical applications.

Also note that in our description the $\mathsf{KG}$ algorithm samples a unique set of bilinear group parameters $\gamma$ for each key; however, it is perfectly acceptable for many keyholders to share the same group parameters.

*Correctness.* By substitution it is relatively easy to show that the public key encryption scheme $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is correct, *i.e.,* that $\mathsf{Dec}(pk, sk, \mathsf{Enc}(pk, m)) = m$ with probability 1 for valid inputs $pk, m$. We leave the details for Appendix B.

$\mathsf{BCS.KG}(1^\lambda)$

1. Sample $\gamma = (p, \mathbb{G}, \mathbb{G}_T, \hat{e}, g \in \mathbb{G}) \leftarrow \mathsf{BMsetup}(1^\lambda)$.
2. Choose $g, g_1, g_2, g_3, v', d', u'_1, \ldots, u'_5 \xleftarrow{R} \mathbb{G}$, and $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3 \xleftarrow{R} \mathbb{Z}_p^*$.
3. Compute:

$$\begin{array}{lll} c_1 \leftarrow g_1^{x_1} g_3^{x_3} & d_1 \leftarrow g_1^{y_1} g_3^{y_3} & h_1 \leftarrow g_1^{z_1} g_3^{z_3} \\ c_2 \leftarrow g_2^{x_2} g_3^{x_3} & d_2 \leftarrow g_2^{y_2} g_3^{y_3} & h_2 \leftarrow g_2^{z_2} g_3^{z_3} \end{array}$$

4. Output $pk = (\gamma, g, g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, v', d', u'_1, \ldots, u'_5)$, $sk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$.

$\mathsf{BCS.Enc}(pk, m \in \mathbb{G})$.

1. Select $\alpha, r_1, r_2, c, \psi \xleftarrow{R} \mathbb{Z}_p^*$. Construct a $\mathsf{FS}$ keypair $(vk_1, sk_1)$ and a second "workalike" keypair $(vk_2, sk_2)$ as follows:

$$\begin{array}{rl} vk_1 \leftarrow (\gamma, g, g^\alpha, v', d', g_1, g_2, g_3, g, g, u'_1, \ldots, u'_5, 5) & vk_2 = (\gamma, g, g^\psi, v', d', g, g^c, 1) \\ sk_1 \leftarrow (vk_1, \alpha) & sk_2 = (vk_2, \psi, c) \end{array}$$

   Next, compute the ciphertext as:

$$u_1 \leftarrow g_1^{r_1} \quad u_2 \leftarrow g_2^{r_2} \quad u_3 \leftarrow g_3^{r_1+r_2} \quad e \leftarrow m \cdot h_1^{r_1} h_2^{r_2} \quad v \leftarrow (c_1 d_1^\alpha)^{r_1} \cdot (c_2 d_2^\alpha)^{r_2}$$
$$vk \leftarrow g^\alpha \quad e_1 \leftarrow u_1^\alpha \quad e_2 \leftarrow u_2^\alpha \quad e_3 \leftarrow u_3^\alpha \quad f_1 \leftarrow g^c \quad f_2 \leftarrow g^\psi$$
$$\sigma_1 \leftarrow \mathsf{FS.Sign}\,(sk_1, (r1, r2, r1 + r2, c, \psi)) \quad \sigma_2 \leftarrow \mathsf{FS.WASign}\,(sk_2, e)$$

2. Output $C = (u_1, u_2, u_3, e, v, vk, e_1, e_2, e_3, f_1, f_2, \sigma_1, \sigma_2)$.

$\mathsf{BCS.Dec}(pk, sk, C)$.

1. Parse $sk$ and $C$ as above. Assemble $vk_1 \leftarrow (\gamma, g, vk, v', d', g_1, g_2, g_3, g, g, u'_1, \ldots, u'_5, 5)$ and $vk_2 \leftarrow (\gamma, g, f_2, v', d', g, f_1, 1)$. Now, verify the relations:

$$\begin{array}{r} \{\hat{e}(vk, u_i) = \hat{e}(e_i, g)\}_{i \in [1,3]} \wedge \\ \mathsf{FS.Verify}\,(vk_1, (u_1, u_2, u_3, f_1, f_2), \sigma_1) = 1 \ \wedge\ \mathsf{FS.Verify}\,(vk_2, (e), \sigma_2) = 1\} \end{array} \tag{1}$$

2. If this check fails, output $\perp$. Otherwise, parse $sk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$ and select $z \xleftarrow{R} \mathbb{Z}_p^*$. Compute the decryption $m'$ as:

$$m' = e \cdot \frac{(u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} e_2^{y_2} \cdot u_3^{x_3} e_3^{y_3})^z}{u_1^{z_1} u_2^{z_2} u_3^{z_3} \cdot v^z} \tag{2}$$

Figure 1: Key generation, encryption and decryption algorithms for the BCS scheme.

**The Blind Decryption Protocol.** The blind decryption protocol BlindDec with respect to BCS is shown in Figure 2. The protocol requires a multiplicatively homomorphic IND-CPA-secure encryption scheme, which we instantiate using the Linear Encryption scheme (LE) of Boneh *et al.* [7].[9]

The protocol employs the homomorphic property of LE to construct a a two-party implementation of the Dec algorithm, with ZKPoKs used to ensure that both the User and Decryptor's contributions are correctly formed. Note that for security reasons it is critical that the Decryptor *re-randomize* the ciphertext that it sends back to the User in its portion of the protocol. In the LE scheme this can be accomplished by multiplying a ciphertext with a fresh encryption of the identity element.

### 4.1.1 Security

Let $\Pi_{ZK}$ be a zero-knowledge (and, implicitly, witness indistinguishable) proof system secure under the Decision Linear assumption (possibly in the Common Reference String model). In the following theorems we will show that if the Decision Linear and Flexible Diffie-Hellman assumptions hold in $\mathbb{G}$ then

[9] In asymmetric bilinear groups where the Decisional Diffie-Hellman problem is hard, this can easily be replaced with Elgamal encryption, resulting in a significant efficiency improvement.

**$\mathcal{U}(pk, C)$** | **$\mathcal{D}(pk, sk)$**

1. Parse $C$ as $(u_1, u_2, u_3, e, v, vk, e_1, e_2, e_3, f_1, f_2, \sigma_1, \sigma_2)$,
   and parse $pk = (\gamma, g, g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, v', d', u'_1, \ldots, u'_5)$.
   Verify that $C$ satisfies equation (1) of the Dec algorithm. If not, abort and output $\perp$.
2. Generate $(pk_{\mathcal{U}}, sk_{\mathcal{U}}) \leftarrow \mathsf{LE.KG}(\gamma)$ and select $\bar{z} \xleftarrow{R} \mathbb{Z}_p^*$. Compute:
   $\mathbf{c}_1 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, u_1^{\bar{z}})$, $\mathbf{c}_2 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, u_2^{\bar{z}})$, $\mathbf{c}_3 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, u_3^{\bar{z}})$,
   $\mathbf{c}_4 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, e_1^{\bar{z}})$, $\mathbf{c}_5 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, e_2^{\bar{z}})$, $\mathbf{c}_6 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, e_3^{\bar{z}})$,
   $\mathbf{c}_7 \leftarrow \mathsf{LE.Enc}(pk_{\mathcal{U}}, v^{\bar{z}})$ and set $vk_1 \leftarrow (\gamma, g, vk, v', d', g_1, g_2, g_3, g, g, u'_1, \ldots, u'_5, 5)$,
   $vk_2 \leftarrow (\gamma, g, f_2, v', d', g, f_1, 1)$
3. Send $pk_{\mathcal{U}}, \mathbf{c}_1, \ldots, \mathbf{c}_7$ to $\mathcal{D}$ and conduct the following proof of knowledge with $\mathcal{D}$:
   $\mathsf{WIPoK}\{(u_1, u_2, u_3, v, vk, e_1, e_2, e_3, f_1, f_2, \sigma_1, \sigma_2, vk_1, vk_2, \bar{z})$:
   $\mathbf{c}_1 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, u_1^{\bar{z}}) \wedge \mathbf{c}_2 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, u_2^{\bar{z}}) \wedge \mathbf{c}_3 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, u_3^{\bar{z}}) \wedge$
   $\mathbf{c}_4 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, e_1^{\bar{z}}) \wedge \mathbf{c}_5 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, e_2^{\bar{z}}) \wedge \mathbf{c}_6 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, e_3^{\bar{z}}) \wedge$
   $\mathbf{c}_7 = \mathsf{LE.Enc}(pk_{\mathcal{U}}, v^{\bar{z}}) \;\wedge\; \hat{e}(vk, u_i) = \hat{e}(e_i, g)\}_{i \in [1,3]} \wedge$
   $\mathsf{FS.Verify}\,(vk_1, (u_1, u_2, u_3, f_1, f_2), \sigma_1) = 1 \;\wedge\; \mathsf{FS.Verify}\,(vk_2, (e), \sigma_2) = 1\}$

   4. If the proof does not verify, abort.
   5. Compute $\mathbf{c}' = \mathsf{LE.Enc}(pk_{\mathcal{U}}, 1)$ and $\bar{z}' \xleftarrow{R} \mathbb{Z}_p^*$.
   6. Using the homomorphic property of LE, compute:
      $$\mathbf{c}'' \leftarrow \frac{(\mathbf{c}_1^{x_1}\mathbf{c}_4^{y_1} \cdot \mathbf{c}_2^{x_2}\mathbf{c}_5^{y_2} \cdot \mathbf{c}_3^{x_3}\mathbf{c}_6^{y_3})^{\bar{z}'}}{\mathbf{c}_1^{z_1}\mathbf{c}_2^{z_2}\mathbf{c}_3^{z_3} \cdot \mathbf{c}_7^{\bar{z}'}} \cdot \mathbf{c}'.$$
   7. Return $\mathbf{c}''$ and conduct the following proof:
      $ZKPoK\{(x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3, \bar{z}', \mathbf{c}')$ :
      $\mathbf{c}' = \mathsf{LE.Enc}(pk, 1) \wedge$
      $$\mathbf{c}'' = \frac{(\mathbf{c}_1^{x_1}\mathbf{c}_4^{y_1} \cdot \mathbf{c}_2^{x_2}\mathbf{c}_5^{y_2} \cdot \mathbf{c}_3^{x_3}\mathbf{c}_6^{y_3})^{\bar{z}'}}{\mathbf{c}_1^{z_1}\mathbf{c}_2^{z_2}\mathbf{c}_3^{z_3} \cdot \mathbf{c}_7^{\bar{z}'}} \cdot \mathbf{c}'\}$$

8. If the proof does not verify, abort and return $\perp$.
9. Compute $m' = e \cdot (\mathsf{LE.Dec}(sk, \mathbf{c}''))^{1/\bar{z}}$.

Output $m'$. | Output nothing.

Figure 2: The Blind Decryption protocol $\mathsf{BlindDec}(\mathcal{U}(pk, C), \mathcal{D}(pk, sk)) \rightarrow (m', \mathsf{nothing})$. For compactness of notation we represent the homomorphic operation on two LE ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ using simple multiplicative notation $(\mathbf{c}_1\mathbf{c}_2)$, and exponentiation by a scalar $z$ as $\mathbf{c}_1^z$.

BCS = (KG, Enc, Dec, BlindDec) implemented with $\Pi_{ZK}$ is a secure blind decryption scheme in the sense of Definition 3.5. This requires three separate arguments: (1) that the algorithms (KG, Enc, Dec) comprise an IND-CCA2-secure encryption scheme, (2) that the BlindDec protocol is leak-free, and (3) that BlindDec is selective-failure blind.

**Theorem 4.1** *If the Decision Linear and Flexible Diffie-Hellman assumptions hold in $\mathbb{G}$ and $\Pi_{ZK}$ is secure under the Decision Linear assumption in the standard model (resp. CRS model), then* (BCS.KG, BCS.Enc, BCS.Dec) *comprise an* IND-CCA2*-secure public-key encryption scheme secure in the standard model (resp. CRS model).*

We present a proof of Theorem 4.1 in Appendix C.1. Here we will summarize the intuition behind the proof, which employs techniques from the Cramer-Shoup variant due to Shacham [51]. Our simulator knows the scheme's secret key, and can use it to answer decryption queries. The exceptions to this rule are certain queries related to the challenge ciphertext. Specifically, we must be careful with queries that are $(a)$ "malformed", *i.e.,* the queried value $v \neq u_1^{x_1}e_1^{y_1} \cdot u_2^{x_2}e_2^{y_2} \cdot u_3^{x_3}e_3^{y_3}$, or that $(b)$ embed the value $vk^*$ from the challenge ciphertext.

Note that equation (2) of the Dec algorithm ensures that malformed ciphertexts decrypt to a random element of $\mathbb{G}$, so the first case is easily dealt with in our simulation. The adversary cannot maul the ciphertext due to the presence of the checksum $v$. Thus it remains to consider well-formed ciphertexts with $vk = vk^*$. We argue that the challenge ciphertext itself is the only ciphertext that will pass all of our checks.

Intuitively our simulation accomplishes this by setting $vk = g^{\alpha^*}$ as the public key of a strongly unforgeable OTS which is secure under the Flexible Diffie-Hellman assumption. In principle we use this key to sign the challenge ciphertext components $(u_1^*, u_2^*, u_3^*, e^*)$, which produces all of the remaining components of the ciphertext. When the adversary submits a decryption query with $vk = vk^*$ we can be assured that the query is identical to the challenge ciphertext, as any other result would require the adversary to forge the OTS.

It remains to argue that our OTS is indeed unforgeable. This is non-trivial, since the OTS is based on an $F$-signature where signing operates on messages of the form $m_1, \ldots, m_n \in \mathbb{Z}_p^*$, but verification can be conducted given $g_1^{m_1}, \ldots, g_n^{m_n} \in \mathbb{G}$. In principle our simulation can select the elements $u_1^*, u_2^*, u_3^*$ such the simulator knows their discrete logarithm base $g$. Unfortunately, even this is not sufficient, since our simulator does not always know the discrete logarithm of the value $e^*$ which is based on a message value chosen by the adversary. The core intuition of our proof is to give two separate simulations: in one the signing key $\alpha^*$ is known and we can *simulate* the signature, producing a correctly-distributed (but not unforgeable) signature over arbitrary group elements. In the second simulation the signing key is unknown: the simulator chooses $(u_1^*, u_2^*, u_3^*, e^*)$ at random such that it knows the relevant discrete logarithm of each value. Although the resulting ciphertext does not encrypt either $m_0$ *or* $m_1$, we are able to show that under the Decision Linear assumption no adversary will be able to detect this condition.

**Theorem 4.2** *If the Decision Linear assumption holds in $\mathbb{G}$ and $\Pi_{ZK}$ is secure under the Decision Linear assumption, then the* BCS *protocol* BlindDec *is leak-free.*

We present a proof sketch of Theorem 4.2 in Appendix C.2. Intuitively this proof is quite simple: we show that for any real-world adversary $\mathcal{A}$ we can construct an ideal-world adversary $\mathcal{S}$ that, whenever $\mathcal{A}$ initiates the BlindDec protocol, operates as follows: (1) $\mathcal{S}$ uses the extractor for the PoK system to obtain $\mathcal{A}$'s requested ciphertext, (2) queries this result to the trusted decryption oracle, (3) re-blinds and returns the correctly formulated result to the adversary, simulating the necessary ZK proofs. We show that under the Decision Linear assumption no *p.p.t.* distinguisher can differentiate the output of $\mathcal{S}$ playing the Ideal-World game from the output of $\mathcal{A}$ in the Real-World game except with negligible probability.

**Theorem 4.3** *If the Decision Linear assumption holds in $\mathbb{G}$ and $\Pi_{ZK}$ is secure under the Decision Linear assumption, then the* BCS *protocol* BlindDec *satisfies the property of ciphertext blindness.*

We sketch a proof of Theorem 4.3 in Appendix C.3.

## 4.2 Extensions

**Tag-Based Encryption.** Tag-Based Encryption (TBE) allows encryptors to apply a tag (label) to each ciphertext. This tag is used during the decryption process. The BCS construction is in fact natively based on a TBE scheme, but this functionality is lost in the final construction. We now show that with some minor extensions to the KG, Enc, Dec algorithms (and BlindDec) it is possible to retain the scheme's TBE functionality.

Our modified KG algorithm selects additional *sk* elements $x_1'$, $x_2'$, $x_3'$, $y_1'$, $y_2'$, $y_3'$ and computes new *pk* elements $c_1' = g_1^{x_1'} g_3^{x_3'}$, $c_2' = g_2^{x_2'} g_3^{x_3'}$, $d_1' = g_1^{y_1'} g_3^{y_3'}$, $d_2' = g_2^{y_2'} g_3^{y_3'}$. When an encryptor calls the tag-based encryption algorithm $\mathsf{Enc}(pk, m, \bar{t})$ with $\bar{t} \in \mathbb{Z}_p^*$ as the tag value, Enc computes an additional check value $\bar{v} = (c_1' d_1'^{\bar{t}})^{r_1} \cdot (c_2' d_2'^{\bar{t}})^{r_2}$ and adds $\bar{v}, \bar{t}$ to the existing BCS ciphertext. To compute $\mathsf{Dec}(pk, sk, C, \bar{t}')$, the modified decryption algorithm selects $z' \xleftarrow{R} \mathbb{Z}_p^*$ and computes $m'$ as:

$$e \cdot \frac{(u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} e_2^{y_2} \cdot u_3^{x_3} e_3^{y_3})^z (u_1^{x_1' + \bar{t}' y_1'} \cdot u_2^{x_2' + \bar{t}' y_2'} \cdot u_3^{x_3' + \bar{t}' y_3'})^{z'}}{u_1^{z_1} u_2^{z_2} u_3^{z_3} \cdot v^z v'^{z'}}$$

We omit the revised BlindDec protocol and security proofs, but simply observe that an adversary's probability of "forging" a correct tag $\bar{v}$ is roughly the same as their probability of forging the original check $v$. By the arguments presented in the proof of Theorem 4.1 this probability is negligible. We refer the reader to the work of MacKenzie *et al.* [42] for additional intuition and formal TBE security definitions.

**Selective-failure blindness.** Camenisch *et al.* [16] propose a stronger definition of blindness (for signature schemes) that they refer to as "selective-failure" blindness. Intuitively, this definition captures the notion that an adversarial Decryptor might attempt to induce failures in the protocol (*e.g.,* by generating malformed ciphertexts) in order to deprive the User of privacy. Unfortunately our protocols do not natively achieve this definition because the Decryptor can create ciphertexts with an improperly formed check value $v$. Unfortunately, due to the nature of our scheme this check cannot be verified independently by the user. One potential solution to this problem is to add to each ciphertext a non-interactive proof that $v$ is correctly formed. Such a proof could be constructed using the Fiat-Shamir heuristic in the random oracle model, or using the Groth-Sahai system in the Common Reference String model. Note that this approach would not require any changes to the blind decryption protocol.

# 5 Applications

Blind decryption has applications to a number of privacy-preserving protocols. Several applications have already been proposed in the literature, *e.g.,* [46, 25]. Below we will propose two specific applications motivated by our construction.

**Privacy-preserving Distributed Filesystems.** Many organizations are responding to the difficulty of securing data in a distributed network, where storage locations can include semi-trusted file servers, desktop computers and mobile devices. An increasingly popular approach is to employ cryptographic access control to restrict and monitor file access in these environments. In this approach (*e.g.,* [1]), access control is performed by encrypting files at rest; authorized users contact a centralized server in order to decrypt them when necessary.

Once concern when centralizing access control is the high value of the query pattern. Specifically, knowing *which* content a user is accessing may by itself leak confidential information. For example, the pattern of file accesses by executives during a corporate merger can have enormous financial value if placed in the right hands. While it is desirable to centralize access control, it can therefore be important to restrict this centralized party from learning which information is being managed. While these goals seems contradictory, Coull *et al.* [21] and Camenisch *et al.* [13] recently showed how to construct sophisticated access control mechanisms using anonymous credentials. In these protocols a server provides strong, and even *history-dependent* access control without ever learning user's access pattern.

Our blind decryption protocols are amenable to integration with these access control techniques. In particular, by extending $\mathsf{BCS}$ to include *encryption tags* as in Section 4.2, data can be explicitly categorized and policies can be defined around these categories.

**Oblivious Transfer with Public Contribution.** In an adaptively-secure $k$-out-of-$N$ Oblivious Transfer protocol ($\mathsf{OT}^N_{k\times 1}$) a Receiver obtains up to $k$ items from a Sender's $N$-item database, without revealing to the Sender *which* messages were transferred. There has been much recent interest in $\mathsf{OT}^N_{k\times 1}$ [16, 29, 30, 35, 47, 21, 13], as it is particularly well suited for constructing privacy-preserving databases in which the user's query pattern is cryptographically protected (this is critical in *e.g.,* patent and medical databases).

For practical reasons, there are situations in which it is desirable to distribute the authorship of records, particularly when database updates are performed offline. Unfortunately, existing $\mathsf{OT}^N_{k\times 1}$ protocols seem fundamentally incapable of supporting message contributions by third parties without the explicit cooperation of the Sender.

Our blind decryption constructions admit new $\mathsf{OT}^N_{k\times 1}$ protocols. While this is interesting in and of itself, these protocols can be extended to permit *public contribution.* Intuitively, the contributors achieve this by simply encrypting their messages using the $\mathsf{Enc}$ algorithm under the Sender's public key and sending the resulting ciphertexts directly to the Receiver. The Receiver can then obtain up to $k$ decryptions by running $\mathsf{BlindDec}$ with the Sender. Proving this intuitive protocol secure under a strong simulation-based definition [16, 29] requires some additional components that are easily achieved using the techniques available to us. We leave such a construction for the full version of this work.

## References


[1] Eruces Tricryption. `http://www.eruces.com/index.php`.

[2] Ben Adida, Susan Hohenberger, and Ronald L. Rivest. Ad-hoc group signatures from hijacked keypairs. In *DIMACS Workshop on Theft in E-Commerce (preliminary version)*, April 2005.

[3] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID tags via insubvertible encryption. In *CCS '05*, pages 92–101. ACM Press, 2005.

[4] Mira Belenkiy, Melissa Chase, Markulf Kolweiss, and Anna Lysyanskaya. Non-interactive anonymous credentials. In *TCC '08*, volume 4948 of LNCS, pages 356–374, 2008.

[5] Mihir Bellare and Philip Rogaway. Optimal asymmetric encryption padding — how to encrypt with rsa. In *EUROCRYPT '94*, pages 92–111, 1994.

[6] Dan Boneh and Xavier Boyen. Efficient selective-ID secure Identity-Based Encryption without random oracles. In *EUROCRYPT '04*, volume 3027 of LNCS, pages 223–238, 2004.

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO '04*, volume 3152 of LNCS, pages 45–55, 2004.

[8] Dan Boneh and Jonathan Katz. Improved efficiency for cca-secure cryptosystems built using identity based encryption. In *CT-RSA '05*, volume 3376 of LNCS. Springer, 2005.

[9] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Advances in Cryptology – EUROCRYPT '06*, volume 4004 of LNCS, pages 573–592, 2006.

[10] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM Conference on Computer and Communications Security*, pages 320–329. ACM Press, 2005.

[11] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT '97*, volume 1233 of LNCS, pages 318–333, 1997.

[12] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *CRYPTO '86*, volume 263 of LNCS, pages 234–238, 1986.

[13] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In *CCS '09*, pages 131–140, New York, NY, USA, 2009. ACM.

[14] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO '04*, volume 3152 of LNCS, pages 56–72. Springer, 2004.

[15] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number $n$ is the product of two safe primes. In *EUROCRYPT '99*, volume 1592 of LNCS, pages 107–122, 1999.

[16] Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT '07*, volume 4515 of LNCS, pages 573–590, 2007.

[17] Jan Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, volume 1296 of LNCS, pages 410–424, 1997.

[18] Jan Leonhard Camenisch. *Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem*. PhD thesis, ETH Zürich, 1998.

[19] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT '03*, volume 3027 of LNCS, pages 255–271, 2003.

[20] David Chaum. Blind signatures for untraceable payments. In *CRYPTO '82*, pages 199–203. Plenum Press, 1982.

[21] Scott Coull, Matthew Green, and Susan Hohenberger. Controlling access to an oblivious database using stateful anonymous credentials. In *Public Key Cryptography*, volume 5443 of LNCS, pages 501–520, 2009.

[22] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, volume 839 of LNCS, pages 174–187, 1994.

[23] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, pages 13–25, London, UK, 1998. Springer.

[24] Ivan Damgård, Masahiro Mambo, and Eiji Okamoto. Further study on the transformability of digital signatures and the blind decryption. In *SCIS '97-33B*, 1997.

[25] Yevgeniy Dodis, Shai Halevi, and Tal Rabin. A cryptographic solution to a game theoretic problem. In *CRYPTO '00*, volume 1880 of LNCS, pages 112–130. Springer, 2000.

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

[27] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99*, volume 1666 of LNCS, pages 537–554. Springer, 1999.

[28] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC '87*, pages 218–229, 1987.

[29] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT '07*, volume 4833 of LNCS, pages 265–282, 2007.

[30] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. In Josef Pieprzyk, editor, *ASIACRYPT '08*, volume 5350 of LNCS, pages 179–197. Springer, 2008. Full version available at `http://eprint.iacr.org/2008/163`.

[31] Matthew Green and Susan Hohenberger. Practical adaptive oblivious transfer from a simple assumption. In *TCC '11 (To appear)*. Springer, 2011. Available at `http://eprint.iacr.org/2010/109`.

[32] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT '08*, volume 4965 of LNCS, pages 415–432. Springer, 2008.

[33] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In *TCC '07*, volume 4392 of LNCS, pages 233–252, 2007.

[34] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In *Advances in Cryptology – EUROCRYPT '09*, 2009.

[35] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *TCC '09*, volume 5444 of LNCS, pages 577–594, 2009.

[36] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC '88*, pages 20–31, 1988.

[37] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC '06*, volume 3876 of LNCS, pages 581–600. Springer, 2006.

[38] Sébastien Kunz-Jacques and David Pointcheval. About the security of MTI/C0 and MQV. In *SCN '06*, volume 4116 of LNCS, pages 156–172. Springer, 2006.

[39] Kaoru Kurosawa and Ryo Nojima. Simple adaptive oblivious transfer without random oracle. In *ASIACRYPT '09*, volume 5912 of LNCS, pages 334–346, 2009.

[40] Fabien Laguillaumie, Pascal Paillier, and Damien Vergnau. Universally convertible directed signature. In *ASIACRYPT '05*, volume 3788 of LNCS, pages 682–701, 2005.

[41] Benoît Libert and Damien Vergnaud. Multi-use unidirectional proxy re-signatures. In *CCS '08*, pages 511–520, New York, NY, USA, 2008. ACM.

[42] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In *TCC '04*, volume 2951 of LNCS, pages 171–190. Springer, 2004.

[43] Masahiro Mambo, Kouichi Sakurai, and Eiji Okamoto. How to utilize the transformability of digital signatures for solving the oracle problem. In *ASIACRYPT '96*, pages 322–333, London, UK, 1996. Springer-Verlag.

[44] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, E84-A(5):1234–1243, 2001.

[45] Wakaha Ogata and Phong Le Trieu. Blind HIBE and its application to blind decryption. In *SCIS '08*, pages 4D1–2, 2008.

[46] Perlman Radia. The ephemerizer: Making data disappear. *Journal of Information System Security*, 1(1):51–68, 2005.

[47] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In *Pairing 2009*, volume 5671 of LNCS, pages 231–247, 2009.

[48] Kouichi Sakurai and Yoshinori Yamane. Blind decoding, blind undeniable signatures, and their applications to privacy protection. In *Proceedings of the First International Workshop on Information Hiding*, pages 257–264, London, UK, 1996. Springer-Verlag.

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

[50] Claus-Peter Schnorr and Markus Jakobsson. Security of Signed ElGamal Encryption. In *ASIACRYPT '00*, volume 1976 of LNCS, pages 72–89, 2000.

[51] Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[52] Vanessa Teague. Selecting correlated random actions. In *Financial Cryptography '04*, volume 3110 of LNCS, pages 181–195. Springer, 2004.

[53] Andrew Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.

# A Security of our Multi-Block $F$-signature

In this section we address the security of our one-time $F$-signature, which was presented in Section 2.4. We will first informally describe the security game. Let $\gamma$ be a set of bilinear group parameters, and $n$ a fixed message-length that is polynomial in $|\gamma|$. An adversary is given $pk$ and allowed to request one signature $\sigma$ on an $n$-block message $(m_1, \ldots, m_n)$ of the adversary's choosing. We say that the scheme is strongly $F$-unforgeable if no p.p.t. adversary has more than a negligible probability of outputting $(g_1^{m'_1}, \ldots, g_n^{m'_n}, \sigma')$ where $\mathsf{Verify}(pk, g_1^{m'_1}, \ldots, g_n^{m'_n}, \sigma') = 1$ and $(g_1^{m'_1}, \ldots, g_n^{m'_n}) \neq (g_1^{m_1}, \ldots, g_n^{m_n})$ or $\sigma' \neq \sigma$.

**Theorem A.1** *Let $n = \text{poly}(|\gamma|)$ for some polynomial function $\text{poly}(\cdot)$. The scheme $\mathsf{FS} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Verify})$ is $F$-unforgeable if the Flexible Diffie-Hellman assumption holds in $\mathbb{G}$.*

*Proof.* The proof is similar to the Type II case of the proof presented by Hohenberger and Waters in [34]. Given an adversary $\mathcal{A}$ that forges $\mathsf{FS}$ with non-negligible probability, we construct a solver $\mathcal{B}$ for Flexible Diffie-Hellman that also succeeds with non-negligible probability.

**Simulation Setup.** $\mathcal{B}$ takes as input a Flexible-Diffie Hellman tuple $(\gamma, g, g^a, g^b)$ and selects a message index $i^* \xleftarrow{R} [1, n]$. It next chooses $y_v, y_d, x_v, x_d \xleftarrow{R} \mathbb{Z}_p^*$ and sets $v = g^{bx_v} g^{y_v}$ and $d = g^{-bx_d} g^{y_d}$. It sets $u_{i^*} = g^b$ and for all $i \neq i^*$, selects $x_{u_i} \xleftarrow{R} \mathbb{Z}_p^*$ and sets $u_i = g^{x_{u_i}}$. For $i = 1$ to $n$ it selects $x_{g_i} \xleftarrow{R} \mathbb{Z}_p^*$ and sets $g_i \leftarrow g^{x_{g_i}}$. Finally, $\mathcal{B}$ outputs $pk = (g, g^a, v, d, g_1, \ldots, g_n, u_1, \ldots, u_n, n)$ to $\mathcal{A}$. When $\mathcal{A}$ queries on $m_1, \ldots, m_n$, $\mathcal{B}$ sets $r = (x_d - m_{i^*})/x_v$. It outputs the signature $\sigma$ as:

$$\sigma_1 = \prod_{i=1, i\neq i^*}^{n} g^{ax_{u_i} m_i} \cdot (g^a)^{y_v r + y_d}, \quad g^{ax_{g_1} m_1}, \quad \ldots, \quad g^{ax_{g_n} m_n}, \quad u_1^{m_1}, \quad \ldots, \quad u_n^{m_n}, \quad r$$

Note that the signature above is correctly distributed.

**Forgery.** When $\mathcal{A}$ outputs a valid forgery $(g_1^{m'_1}, \ldots, g_n^{m'_n}, \sigma')$, $\mathcal{B}$ parses $\sigma'$ as $(\sigma'_1, e'_1, \ldots, e'_n, f'_1, \ldots, f'_n, r')$ and computes $g^{m'_{i^*}} = (g_{i^*}^{m'_{i^*}})^{1/x_{g_{i^*}}}$. If either of the following conditions are true, $\mathcal{B}$ aborts: (1) $g_{i^*}^{m'_{i^*}}$ is equal to $g_{i^*}^{m_{i^*}}$ and simultaneously $r = r'$ or (2) $g^{m'_{i^*}} (g^{r'})^{x_v} g^{-x_d} = 1$.

*Abort probability.* Note that since $i^*$ is random and independent of the adversary's view, the probability of abort case (1) is at most $\frac{n-1}{n}$ since the forgery's message vector must be different in at least one position from the adversary's query, or alternatively the value $r'$ must be different than the $r$ provided in $\sigma$. Note that $e_1, \ldots, e_n, f_1, \ldots, f_n$ are uniquely determined by $m'_1, \ldots, m'_n$ (and their structure is checked by $\mathsf{Verify}$), thus the only way $\sigma'$ can differ from $\sigma$ is when $(m_1, \ldots, m_n) \neq (m'_1, \ldots, m'_n)$ or when $r \neq r'$.

We claim that condition (2) occurs with probability at most $1/p$. To show this (as in [34]) we observe that the values $x_v$ and $x_d$ are hidden by blinding factors $y_v$ and $y_d$, respectively. The adversary could hypothesize that $m_{i^*} + x_v r - x_d = 0$, however, there are $p$ possible $(x_v, x_d)$ pairs that satisfy this equation and each of them are equally likely. Information-theoretically, the adversary can output a pair $(g^{m'_{i^*}}, r')$ satisfying $g^{m'_{i^*}} (g^{r'})^{x_v} g^{-x_d} = 1$ with probability at most $1/p$.

**Solution.** Thus, if $\mathcal{A}$ forges with probability $\epsilon$ then with probability at least $\frac{\epsilon p}{n(p-1)}$ $\mathcal{B}$ will output a solution to the Flexible Diffie-Hellman problem $(w, w^{ab})$ as:

$$w = g^{m'_{i^*}} (g^{r'})^{x_v} g^{-x_d}, \quad w^{ab} = \frac{\sigma'_1}{\prod_{i=1, i\neq i^*}^{n} e_i'^{x_{u_i}} \cdot g^{ar' y_v} (g^a)^{y_d}}$$

Note that this works, since if $\sigma'$ passes the verification checks then we can re-write $\sigma_1$ as:

$$\begin{aligned} \sigma'_1 &= \left( \prod_{i=1}^{n} u_i^{m'_i} \cdot v^{r'} d \right)^a = \left( \prod_{i=1, i\neq i^*}^{n} g^{x_{u_i} m'_i} \cdot (g^b)^{m'_{i^*}} (g^{bx_b} g^{y_v})^{r'} (g^{-bx_d} g^{y_d}) \right)^a \\ &= \prod_{i=1, i\neq i^*}^{n} g^{ax_{u_i} m'_i} \cdot g^{a(y_v r' + y_d)} \cdot g^{ab(m'_{i^*} + x_v r' - x_d)} \end{aligned}$$

We conclude by addressing three issues that were raised earlier in this work:

1. *Compatiblilty with the Groth-Sahai proof system.* The security proofs for our blind decryption protocol use a knowledge extractor to obtain $\sigma$ from $\mathsf{WIPoK}\{(\sigma) : \mathsf{Verify}(pk, m, \sigma) = 1\}$. Unfortunately, the Groth-Sahai proof system does not permit the extraction of elements of $\mathbb{Z}_p^*$ and thus cannot obtain the component $r$. To work around this issue, we can have the prover first translate $r$ into a pair of values $(h_1, h_2) = (g^{ar}, g^r)$ and commit to these, while adding the additional condition $\hat{e}(h_1, g) = \hat{e}(h_2, g^a)$ to the verification equation. By observation it is easy to see that a pair $(g^{ar'}, g^{r'})$ can stand in for $r'$ in the reduction above, and this modified signature is also unforgeable under the Flexible Diffie-Hellman assumption.
2. *Simplification for single-block messages.* For the special case of single-block messages (where $n = 1$) we will completely omit the value $e_1$ from the signature, since this value is not required in signature verification or in the proof.
3. *Non-distinct $g_i$ values.* The $\mathsf{KG}$ algorithm selects $g_1, \dots, g_n$ randomly from $\mathbb{G}$. In fact, one or more of these values can be the same, provided that they have a random distribution. The modifications to the security proof above are straightforward, as long as the reduction knows any (possibly non-distinct) values $x_{g_1}, \dots, x_{g_n}$ such that $(g_1, \dots, g_n) = (g^{x_{g_1}}, \dots, g^{x_{g_n}})$.
4. *Security of "workalike" signatures.* In section 2.4 we presented workalike algorithms that can "sign" messages of the form $(g_1, \dots, g_n) \in \mathbb{G}^n$. It does not seem easy to adapt the security proof above to admit this type of signing. Where these algorithms are used in our blind decryption schemes we address their security explicitly in the corresponding proofs.

# B Correctness of $\mathsf{BCS}$

To show that the scheme is correct, we will first describe the following alternative decryption procedure.

1. First, verify the relations described in equation (1) of the $\mathsf{Dec}$ algorithm, outputting $\perp$ if the relations are not satisfied.
2. Next, verify that $u_1^{x_1} e^{y_1} \cdot u_2^{x_2} e^{y_2} \cdot u_3^{x_3} e^{y_3} = v$.
3. If this comparison is satisfied, output $m' = \frac{e}{u_1^{z_1} u_2^{z_2} u_3^{z_3}}$. Otherwise, output a random element of $\mathbb{G}$.

To demonstrate why this alternative decryption is equivalent to the $\mathsf{Dec}$ algorithm, observe that we can re-write equation (2) of the $\mathsf{Dec}$ algorithm as follows:

$$m' = \frac{e}{u_1^{z_1} u_2^{z_2} u_3^{z_3}} \cdot \left( \frac{u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} e_2^{y_2} \cdot u_3^{x_3} e_3^{y_3}}{v} \right)^z$$

When $u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} e_2^{y_2} \cdot u_3^{x_3} e_3^{y_3} \neq v$ then for $z \in_R \mathbb{Z}_p^*$ the equation $(\frac{u_1^{x_1} e^{y_1} \cdot u_2^{x_2} e^{y_2} \cdot u_3^{x_3} e^{y_3}}{v})^z$ evaluates to a random element of $\mathbb{G}$, and thus $m'$ is also random. When the equality holds, we obtain $m' = \frac{e}{(u_1^{z_1} u_2^{z_2} u_3^{z_3})}$.

It remains to show that this alternative decryption is correct. By substitution we can see that an honestly-generated ciphertext will always satisfy the check of equation (1). Similarly, it will satisfy $u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} \sigma_2^{y_2} \cdot u_3^{x_3} \sigma_3^{y_3} \stackrel{?}{=} v$:

$$\begin{aligned} & u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} e_2^{y_2} \cdot u_3^{x_3} e_3^{y_3} \\ = {} & (g_1^{r_1 x_1} g_1^{r_1 \alpha y_1}) \cdot (g_2^{r_2 x_2} g_2^{r_2 \alpha y_2}) \cdot (g_3^{(r_1+r_2)x_3} g_3^{(r_1+r_2)\alpha y_3}) \\ = {} & g_1^{r_1 x_1} g_3^{r_1 x_3} g_1^{r_1 \alpha y_1} g_3^{r_1 \alpha y_3} \cdot g_2^{r_2 x_2} g_3^{r_2 x_3} g_2^{r_2 \alpha y_2} g_3^{r_2 \alpha y_3} \\ = {} & (c_1 d_1^{\alpha})^{r_1} \cdot (c_2 d_2^{\alpha})^{r_2}) = v \end{aligned}$$

Finally, when all of the previous checks succeed, the output $m$ is correctly distributed:

$$\frac{e}{(u_1^{z_1} u_2^{z_2} u_3^{z_3})} = \frac{m \cdot h_1^{r_1} h_2^{r_2}}{(u_1^{z_1} u_2^{z_2} u_3^{z_3})} = \frac{m \cdot (g_1^{z_1} g_3^{z_3})^{r_1} (g_2^{z_2} g_3^{z_3})^{r_2}}{g_1^{r_1 z_1} g_2^{r_2 z_2} g_3^{(r_1+r_2) z_3}} = m$$

# C Proofs of Security

## C.1 Proof of Theorem 4.1

*Proof.* The BCS scheme is based on the scheme of Shacham [51], and the first twelve paragraphs of this proof directly quote that proof [51, §3]. To avoid unnecessary detail we will refer the reader to the above proof for some of the analysis of abort probabilities. Let $\mathcal{A}$ be an adversary with non-negligible advantage in the IND-CCA2 security game. We show how to construct a solver for the Decision Linear or Flexible Diffie-Hellman problem using $\mathcal{A}$.

In the first case, let $\mathcal{B}$ be an algorithm that, given a Decision Linear instance $(\gamma, g_1, g_2, g_3, u_1, u_2, u_3)$, outputs 1 if $log_{g_3} u_3 = log_{g_1} u_1 + log_{g_2} u_2$ and 0 otherwise. As in the real KG algorithm, $\mathcal{B}$ chooses secrets $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3, x_{u_1}, \ldots, x_{u_5}$, sets $(u_1, \ldots, u_5) = (g^{x_{u_1}}, \ldots, g^{x_{u_5}})$ and using $\gamma, g_1, g_2, g_3$ constructs the remainder of $pk$ normally.

$\mathcal{B}$ answers $\mathcal{A}$'s decryption queries as in BCS.Dec using its knowledge of the secret key. When $\mathcal{A}$ submits the messages $m_0, m_1$ on which it wishes to be challenged, $\mathcal{B}$ chooses $b \stackrel{R}{\leftarrow} \{0,1\}$ and selects $\alpha, c, \psi \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$. It constructs the following workalike signing keypairs:

$$vk_1 \leftarrow (\gamma, g, g^\alpha, v', d', g_1, g_2, g_3, g, g, u'_1, \ldots, u'_5, 5) \qquad vk_2 = (\gamma, g, g^\psi, v', d', g, g^c, 1)$$
$$sk_1 \leftarrow (vk_1, \alpha, x_{u_1}, \ldots, x_{u_5}) \qquad sk_2 = (vk_2, \psi, c)$$

And calculates the remaining ciphertext elements as:

$$e \leftarrow m_b \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3} \qquad v \leftarrow u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3} \qquad vk \leftarrow g^\alpha$$
$$e_1 \leftarrow u_1^\alpha \qquad e_2 \leftarrow u_2^\alpha \qquad e_3 \leftarrow u_3^\alpha \qquad f_1 \leftarrow g^c \qquad f_2 \leftarrow g^\psi$$
$$\sigma_1 \leftarrow \mathsf{FS.WASign}(sk_1, (u_1, u_2, u_3, g^c, g^\psi)) \qquad \sigma_2 \leftarrow \mathsf{FS.WASign}(sk_2, (e))$$

$\mathcal{B}$ supplies to $\mathcal{A}$ the challenge ciphertext ciphertext $C^* = (u_1, u_2, u_3, e, v, vk, e_1, e_2, e_3, f_1, f_2, \sigma_1, \sigma_2)$, and responds to $\mathcal{A}$'s further decryption queries as before. Finally, $\mathcal{A}$ outputs its guess $b'$. If $b = b'$, $\mathcal{B}$ outputs 1, otherwise it outputs 0.

If $\mathcal{A}$ has a different advantage in guessing the bit $b$ when $\mathcal{B}$ is run with a Linear tuple $(g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2})$ and when $\mathcal{B}$ is run with a random tuple $(g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, \eta \in_R \mathbb{G})$, we obtain a distinguisher for the Linear problem. In the remainder of this sketch, we establish that in the first case $\mathcal{A}$'s advantage is nonnegligible, whereas in the second case $\mathcal{A}$'s advantage is negligible.

Note that in both cases, the public key and decryption queries are distributed as in the real protocol. However, when $\mathcal{B}$ is run with a Linear tuple, we can show that challenge ciphertext is correctly distributed. Specifically, $u_1, u_2, u_3$ are correctly formed, and all remaining elements are formed as in the real protocol. Though they are computed using alternative formulae, it is easy to verify that $e, v$ are also correctly distributed.

When $\mathcal{B}$ is run with a random tuple, the bit $b$ remains independent of $\mathcal{A}$'s view except with negligible probability. Let "$log(\cdot)$" stand for "$log_{g_1}(\cdot)$" and define $t_2 = log(g_2)$ and $t_3 = log(g_3)$. Consider the three elements $(z_1, z_2, z_3)$ of the secret key. The public key values $h_1$ and $h_2$ constrain these to line on the line at the intersection of the planes defined by $log(h_1) = z_1 + t_2 z_3$ and $log(h_2) = t_2 z_2 + t_3 z_3$. A decryption query for a valid ciphertext whose first three components form a valid Linear tuple $(u'_1, u'_2, u'_3) = (g_1^{r'_1}, g_2^{r'_2}, g_3^{r'_1+r'_2})$ will allow the adversary to obtain $((u'_1)^{z_1}(u'_2)^{z_2}(u'_3)^{z_3})$, but in this case we have

$$log\,((u'_1)^{z_1}(u'_2)^{z_2}(u'_3)^{z_3}) = (r'_1)(z_1 + t_3 z_3) + (r'_2)(t_2 z_2 + t_3 z_3),$$

which is linearly dependent on values already known to the adversary. This analysis does not hold if the decryption oracle accepts a ciphertext whose first three elements do not form a linear tuple. Below we show that the decryption oracle accepts such invalid ciphertexts only with negligible probability.

Now consider the challenge ciphertext $C^* = (u_1,\ u_2,\ u_3,\ e,\ v,\ vk,\ \ldots)$. Let $u_1 = g_1^{r_1}, u_2 = g_2^{r_2}$ and $u_3 = g_3^{r_3}$, with $r_3 \neq r_1 + r_2$ (except with negligible probability). The message in $e$ is blinded by $u_1^{z_1} u_2^{z_2} u_3^{z_3}$, which has the discrete logarithm:

$$log(u_1^{z_1} u_2^{z_2} u_3^{z_3}) = r_1 z_1 + r_2 t_2 z_2 + r_3 t_3 z_3 = (r_1)(z_1 + t_3 z_3) + (\Delta r)(t_3 z_3),$$

where $\Delta r = r_3 - r_1 - r_2 \neq 0$. To an adversary who has received decryption queries only for valid ciphertexts, this value is independent of its view. This means that $M_b$ is independent of the adversary's view even given $e$.

It remains only to show that, given that through query $i$ the decryption oracle has not accepted an invalid ciphertext, the probability that it accepts an invalid one at query $i+1$ is negligible. We restrict our consideration to decryption queries made by $\mathcal{A}$ after it has seen the challenge ciphertext, since the challenge ciphertext gives the adversary strictly more information about the values $(x_1, x_2, x_3, y_1, y_2, y_3)$ by which $\mathsf{BCS.Dec}$ checks ciphertext validity. Let $C'_i = (u'_1,\ u'_2,\ u'_3,\ e',\ v',\ vk',\ e'_1,\ e'_2,\ e'_3,\ f'_1,\ f'_2,\ \sigma'_1,\ \sigma'_2)$ be the $i^{th}$ query, which satisfied the pairing-based checks of the $\mathsf{Dec}$ algorithm. We consider the following three cases:

1. $(u_1, u_2, u_3, e, vk) = (u'_1, u'_1, u'_3, e', vk')$, but $v \neq v'$. In this case $\mathcal{B}$ can simply return a random element of $\mathbb{G}$, since $v$ as calculated in generating $C^*$ is the only correct checksum value for $(u_1, u_2, u_3, e)$. The $\mathsf{Dec}$ algorithm will always return a random element when $v \neq v'$ since equation (2) embeds the term $\left(\frac{u_1^{x_1} e_1^{y_1} \cdot u_2^{x_2} e_2^{y_2} \cdot u_3^{x_3} e_3^{y_3}}{v'}\right)^z = (\frac{v}{v'})^z$ for $z \in_R \mathbb{Z}_p^*$.

2. $(u_1, u_2, u_3, e) \neq (u'_1, u'_1, u'_3, e')$ and $vk \neq vk'$. In this case $\mathcal{B}$ will also return a random element of $\mathbb{G}$, since the correct checksum value $(u_1'^{x_1} e_1'^{y_1} \cdot u_2'^{x_2} e_2'^{y_2} \cdot u_3'^{x_3} e_3'^{y_3})$ is independent of $\mathcal{A}$'s view, and thus with all but negligible probability the submitted value $v'$ will not pass the decryption check. We omit the full argument here, but refer the reader to [51, §3] for the details.

3. $C'_i \neq C^*$ and $vk = vk'$. We refer to this case as $\mathsf{Event\_Forge}$. Proving that this event occurs with at most negligible probability represents a new component of the proof. In Lemma C.1 below we show that an adversary who presents such a ciphertext implies a forger for the $\mathsf{FS}$ signature scheme presented in Section 2.4. Thus, if the Flexible Diffie-Hellman assumption holds in $\mathbb{G}$ this condition must occur with at most negligible probability.

Based on the above, $\mathcal{A}$ has only a negligible probability of obtaining the decryption of an invalid ciphertext (or detecting an invalid response) after a polynomial number of decryption queries. This concludes our main proof. We now turn our attention to the following Lemma.

**Lemma C.1** *If the Flexible Diffie-Hellman and Decision Linear assumptions hold in $\mathbb{G}$, then for all p.p.t. adversaries $\mathcal{A}$,* $\mathsf{Event\_Forge}$ *will occur with probability negligible in $\lambda$.*

*Proof.* Let $\mathcal{A}$ be a CCA adversary who, having received a challenge ciphertext of the form $C^* = (u_1, u_2, u_3, e, vk, \ldots)$, where $u_1, u_2, u_3 \in_R \mathbb{G}$, issues a decryption query on $C' = (u'_1, u'_2, u'_3, e', vk', \ldots)$ where $vk' = vk$ and yet $C' \neq C^*$.[10] If $\mathcal{A}$ makes such a decryption query with non-negligible probability, we show how to construct an algorithm that uses $\mathcal{A}$ to solve the Flexible Diffie-Hellman problem or Decision Linear problem with non-negligible advantage.

Our primary strategy in this proof is to show that $\mathcal{A}$ can be used to construct a forger for the strongly unforgeable multi-block $F$-signature $\mathsf{FS}$ described in Section 2.4. By Theorem A.1 a forger for this scheme implies a solver for the Flexible Diffie-Hellman problem with related advantage. We refer the reader to the proof of Theorem A.1 for complete details.

A wrinkle in our simulation is that the values it gives to the adversary $\mathcal{A}$ will not necessarily be identically distributed to the real protocol. To complete our proof we show via a separate Lemma, below, that under

[10]Note that it does not matter if $\mathcal{A}$ makes this query before or after receiving $C^*$, both cases are acceptable for this proof.

the Decision Linear assumption, all p.p.t. adversaries have at most a negligible advantage in distinguishing the simulation from the real protocol.

We will now describe the forger $\mathcal{B}'$, which runs $\mathcal{A}$ internally and interacts with a challenger playing the strong $F$-unforgeability game (see Section 2.4) instantiated with the one-time signature $\mathsf{FS}$.

Simulation Setup. $\mathcal{B}'$ flips a coin and with probability 1/2 chooses one of two strategies below. In each strategy $\mathcal{B}'$ will play the $\mathsf{IND\text{-}CCA2}$-game with $\mathcal{A}$, and will query the $\mathsf{FS}$ signing oracle to obtain the component $\sigma_1$ or $\sigma_2$ for the challenge ciphertext. In both strategies, $\mathcal{B}'$ will select group parameters $\gamma$, derive the $\mathsf{BCS}$ secret key $sk = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$ and compute the public parameters for the encryption scheme.

**Strategy 1.** $\mathcal{B}'$ fixes signature parameters $\gamma, n = 5$ and requests a $\mathsf{FS}$ verification key of the form $vk_1 = (g, g^\alpha, v', d', g_1, g_2, g_3, g, g, u'_1, \ldots, u'_5, 5)$. Note that the use of the non-distinct base elements $(g_1, g_2, g_3, g, g)$ is a deviation from the scheme as it was defined in Section 2.4. However, in Appendix A we noted that the signature retains its security when using parameters of this form. $\mathcal{B}'$ computes $(c_1, c_2, d_1, d_2, h_1, h_2)$ as in the normal key generation algorithm and outputs $pk = (\gamma,\ g,\ g_1,\ g_2,\ g_3,\ c_1,\ c_2,\ d_1,\ d_2,\ h_1,\ h_2,\ v',\ d',\ u'_1,\ \ldots,\ u'_5)$.

$\mathcal{B}'$ can now answer decryption queries using its knowledge of the secret key. When $\mathcal{A}$ eventually submits challenge messages $m_0, m_1$, $\mathcal{B}'$ selects $b \xleftarrow{R} \{0,1\}$, $r_1, r_2, r_3, c, \psi \xleftarrow{R} \mathbb{Z}_p^*$, computes $(u_1, u_2, u_3) = (g_1^{r_1}, g_2^{r_2}, g_3^{r_3})$ and uses its knowledge of the secret key to compute $(e, v) = (m_b g_1^{r_1 z_1} g_2^{r_2 z_2} g_3^{r_3 z_3},\ u_1^{x_1} e_1^{y_1} u_2^{x_2} e_2^{y_2} u_3^{x_3} e_3^{y_3})$. It next queries the $\mathsf{FS}$ challenger on the message vector $(r_1, r_2, r_3, c, \psi)$ to obtain the signature $\sigma_1$, and parses $\sigma_1$ to obtain $(e_1, e_2, e_3)$. Finally it constructs the workalike signing keypair $vk_2 = (\gamma, g, g^\psi, v', d', g_1, g^c, 1)$ and $sk_2 = (vk_2, \psi, c)$ and calculates $\sigma_2 \leftarrow \mathsf{WASign}(sk_2, (e))$. It outputs the challenge ciphertext as:

$$C^* = (u_1, u_2, u_3, e, v, vk = g^\alpha, e_1, e_2, e_3, g^c, g^\psi, \sigma_1, \sigma_2)$$

Note that the ciphertext above is correctly distributed, and embeds the $\mathsf{FS}$ signature $\sigma_1$ that was produced by the $\mathsf{FS}$ signing oracle. With overwhelming probability $(g_1, g_2, g_3, u_1, u_2, u_3)$ do *not* form a Linear tuple, but this is expected and in line with the conditions of $\mathsf{Event\_Forge}$. Ultimately $\mathcal{A}$ submits a valid query $C' \neq C^*$ that embeds both $v$ and $vk$ as specified by the conditions of $\mathsf{Event\_Forge}$:

$$C' = (u'_1, u'_2, u'_3, e', v, vk, e'_1, e'_2, e'_3, f'_1, f'_2, \sigma'_1, \sigma'_2)$$

If $C'$ differs from $C^*$ in any of the components $(u'_1, u'_2, u'_3, e'_1, e'_2, e'_3, f'_1, f'_2, \sigma'_1)$ then $\mathcal{B}'$ will output a forgery consisting of the message vector $(u'_1, u'_2, u'_3, f'_1, f'_2)$ and signature $\sigma'_1$. Otherwise (if $C'$ differs only in the components $e'$ or $\sigma'_2$) it will abort. To see why the above forgery is valid, first observe that when equation 1 of the $\mathsf{Dec}$ algorithm is satisfied, then $(e'_1, e'_2, e'_3) = (u_1'^\alpha, u_2'^\alpha, u_3'^\alpha)$, for $\alpha = log_g(vk)$. Thus if $(e'_1, e'_2, e'_3)$ differ from their counterparts in $C^*$, then $(u'_1, u'_2, u'_3)$ must differ as well. Similarly $vk$ and $v$ are identical in both ciphertexts. Thus whenever $C' \neq C^*$ and $\mathcal{B}'$ does not abort, at least one of the values $(u'_1, u'_2, u'_3, f'_1, f'_2, \sigma'_1)$ must differ from its counterpart in $C^*$. This satisfies the conditions for a forgery.

**Strategy 2.** $\mathcal{B}'$ fixes signature parameters $\gamma, n = 1$ and obtains a $\mathsf{FS}$ verification key which it parses as $vk_2 = (g, f_2, v', d', g, f_1, 1)$ (as above, this key uses a variant of the normal $\mathsf{FS}$ key generation algorithm). It selects $\rho_1, \rho_2, \rho_3, x_{u_1}, \ldots, x_{u_5} \xleftarrow{R} \mathbb{Z}_p^*$, sets $(g_1, g_2, g_3) = (g^{\rho_1}, g^{\rho_2}, g^{\rho_3})$, $(u'_1, \ldots, u'_5) = (g_1^{x_{u_1}}, g_2^{x_{u_2}}, g_3^{x_{u_3}}, g^{x_{u_4}}, g^{x_{u_5}})$, and calculates the rest of the public and secret keys as in $\mathsf{BCS.KG}$. Next it hands $\mathcal{A}$ the public key $pk = (\gamma,\ g,\ g_1,\ g_2,\ g_3,\ c_1,\ c_2,\ d_1,\ d_2,\ h_1,\ h_2,\ v',\ d',\ u'_1,\ \ldots,\ u'_5)$.

$\mathcal{B}'$ answers decryption queries using the secret key. When $\mathcal{A}$ eventually submits challenge messages $m_0, m_1$, $\mathcal{B}'$ selects $b \xleftarrow{R} \{0,1\}$, $\alpha, x, r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_p^*$ and queries the $\mathsf{FS}$ challenger on the single-message vector $(x + (\rho_1 r_1 z_1) + (\rho_2 r_2 z_2) + (\rho_3 r_3 z_3))$ to obtain the signature $\sigma_2$. It computes $(u_1, u_2, u_3) = (g_1^{r_1}, g_2^{r_2}, g_3^{r_3})$, $(e_1, e_2, e_3) = (u_1^\alpha, u_2^\alpha, u_3^\alpha)$, $e = g^x g^{r_1 z_1} g^{r_2 z_2} g^{r_3 z_3}$ and $v = u_1^{x_1} e_1^{y_1} u_2^{x_2} e_2^{y_2} u_3^{x_3} e_3^{y_3}$. Finally it assembles a "workalike" signature key $vk_1 \leftarrow (\gamma, g, g^\alpha, v', d', g_1, g_2, g_3, g, g, u'_1, \ldots, u'_5, 5)$ and $sk_1 = (vk_1, \alpha, x_{u_1}, \ldots, x_{u_5})$ which it uses to compute $\sigma_1 \leftarrow \mathsf{WASign}(sk_1, (u_1, u_2, u_3, f_1, f_2))$. It outputs the challenge ciphertext:

$$C^* = (u_1, u_2, u_3, e, v, vk = g^\alpha, e_1, e_2, e_3, f_1, f_2, \sigma_1, \sigma_2)$$

Note that the ciphertext above will pass the verification checks of $\mathsf{Dec}$ equation (1) but is incorrectly distributed, since $e$ encrypts a random message $g^x$ rather than $m_0$ or $m_1$. With the exception of this difference, and its reflection in the structure of the signature $\sigma_2$, the ciphertext is otherwise formulated correctly. Ultimately $\mathcal{A}$ submits a valid query $C'$ of the form:

$$C' = (u_1', u_2', u_3', e', v, vk, e_1', e_2', e_3', f_1', f_2', \sigma_1', \sigma_2')$$

where $v$ and $vk$ are as in the challenge ciphertext and yet $C' \neq C^*$. If $C'$ differs from $C^*$ in the elements $(e', \sigma_2')$ then $\mathcal{B}'$ will output the message $e'$ and signature $\sigma_2'$ as a forgery for the $\mathsf{FS}$ signature scheme. Otherwise it will abort.

**Abort probabilities.** Whenever $\mathcal{A}$ satisfies the conditions of $\mathsf{Event\_Forge}$ (specifically, $C' \neq C^*$ and both ciphertexts embed $vk, v$) then at least *one* of the strategies above is capable of producing a forgery for $\mathsf{FS}$. Unfortunately, there is no guarantee that $\mathcal{B}'$ will have chosen the right strategy, and thus the reduction may have to abort. Clearly if the distribution of Strategy 1 and Strategy 2 were identical from $\mathcal{A}$'s point of view (they are not), then we could still ensure that $\mathcal{B}'$ would output a forgery for $\mathsf{FS}$ with probability at least $\mathbf{Pr}[\mathsf{Event\_Forge}]/2$, since the choice of strategy would be independent of $\mathcal{A}$'s view. By Theorem A.1 we could use this fact to show that under the Flexible Diffie-Hellman assumption $\mathbf{Pr}[\mathsf{Event\_Forge}] \leq \nu(\lambda)$.

Unfortunately our analysis is not complete, since the distributions of Strategy 1 and 2 are *not* identical, specifically, they differ in the structure of the challenge ciphertext $C^*$. Thus it is possible that $\mathcal{A}$ might distinguish the two strategies and thus construct its output $C'$ to induce abort with probabilities non-negligibly higher than 1/2. In the following Lemma C.2 we claim that under the Decision Linear assumption, $\mathcal{A}$ *cannot* succeed at inducing abort with probability greater than $1/2 + \nu'(\lambda)$. This gives us the bound $\mathbf{Pr}[\mathsf{Event\_Forge}]/(2 + \nu'(\lambda)) \leq \nu(\lambda)$ and thus concludes the proof of Lemma C.1.

**Lemma C.2 (Indistinguishability of Simulations)** *If the Decisional Linear assumption holds in $\mathbb{G}$, then all p.p.t. adversaries $\mathcal{A}$ induce $\mathcal{B}'$ to abort with probability at most $1/2 + \nu'(\lambda)$, for some negligible function $\nu'(\cdot)$.*

*Proof sketch.* Let us assume that some p.p.t. $\mathcal{A}$ induces our simulation $\mathcal{B}'$ to abort with probability $1/2 + \epsilon$. We will first argue that under the Decision Linear assumption, no p.p.t. adversary can distinguish the distributions of Strategy 1 from Strategy 2 (from $\mathcal{A}$'s point of view) with greater than negligible advantage, provided that the adversary *does not receive the decryption of any ciphertext that would satisfy the conditions of* $\mathsf{Event\_Forge}$. Having shown this, we will then bound $\epsilon$ to be at most negligible in the security parameter $\lambda$, concluding our proof.

(Note that the above restriction on $\mathcal{A}$'s decryption queries is entirely consistent with the operation of $\mathcal{B}'$. Recall that $\mathcal{B}'$ terminates immediately after receiving the *first* decryption query $C'$ that satisfies the conditions of $\mathsf{Event\_Forge}$. Thus $\mathcal{A}$ will never receive the decryption of such a query.)

Observe that from $\mathcal{A}$'s point of view, the only difference between the distribution of Strategy 1 and Strategy 2 is in the structure of $C^*$. In both cases $(g_1, g_2, g_3, u_1, u_2, u_3)$ do *not* form a Linear tuple, except with negligible probability. However, in each Strategy the value $e$ is constructed differently, as is $\sigma_2$ which is based on $e$. To show that these Strategies are indistinguishable we must describe the following intermediate hybrid games. In every case, the adversary plays the $\mathsf{IND\text{-}CCA2}$ game with the challenger:

**Game 0.** The interaction of Strategy 1. $(u_1, u_2, u_3)$ are random, and $e = m_b \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}$.
**Game 1.** An interaction with the real protocol, where $e = m_b \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}$.
**Game 2.** An interaction with the real protocol, where $e = h \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}$ for some $h \in_R \mathbb{G}$.
**Game 3.** The interaction of Strategy 2. $(u_1, u_2, u_3)$ are random, and $e = h \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}$ for some $h \in_R \mathbb{G}$.

We have already addressed the indistinguishability of **Game 0** and **Game 1** in the $\mathsf{IND\text{-}CCA2}$ reduction at the top of this section: **Game 0** is the distribution produced by the reduction when run with a random tuple, while **Game 1** is the distribution when the reduction is run with a Linear tuple. Since the adversary does not obtain the decryption of any ciphertext satisfying $\mathsf{Event\_Forge}$, we can fix $\mathbf{Pr}[\mathsf{Event\_Forge}] = 0$ (*i.e.,* we do not need to consider Lemma C.1) and thus under the Decision Linear assumption these distributions are distinguishable with at most negligible advantage. Although we do will provide it explicitly, a very similar argument can be made for the case of **Game 2** and **Game 3**, *i.e.,* a distinguisher that differentiates these distributions can be used to construct a solver for Decision Linear.

Finally, imagine that $\mathcal{A}$ differentiates **Game 1** and **Game 2** with non-negligible advantage. Clearly we can use $\mathcal{A}$ to construct an IND-CCA2 adversary against the scheme as follows. At the challenge phase, this adversary would select $b \xleftarrow{R} \{0,1\}, h \xleftarrow{R} \mathbb{G}$ and challenge on the values $(m_b, h)$, returning $C^*$ to $\mathcal{A}$. Clearly if the IND-CCA2 challenger encrypts $m_b$ the distribution is identical to **Game 1**, and otherwise it is identical to **Game 2**. Thus if $\mathcal{A}$ distinguishes the two distributions with non-negligible advantage (as evidenced by its final query $C'$) we obtain a distinguisher for the IND-CCA2 security of the scheme, and hence a solver for Decision Linear. By summation over all hybrids, we see that all p.p.t. adversaries must distinguish Strategy 1 from Strategy 2 with at most negligible advantage.

It remains to show that if no p.p.t. adversary can distinguish Strategy 1 from Strategy 2, then $\epsilon$ must be negligible. As we noted above, were the distributions of Strategy 1 and Strategy 2 identical, then $\mathcal{B}'$ would abort with probability at most $1/2$, *i.e.,* $\epsilon = 0$. However, if $\mathcal{A}$ induces abort with some non-negligible $\epsilon$, we will obtain a distinguisher that differentiates Strategy 1 and Strategy 2 with non-negligible advantage, which is clearly a contribution. This distinguisher would simply present $\mathcal{A}$ with one of the chosen distributions; when $\mathcal{A}$ submits a query $C'$ that would induce an abort in Strategy 1, it outputs 0; if the query would induce an abort in Strategy 2, it outputs 1, and in all other cases it outputs a random guess. Clearly if $\epsilon$ is non-negligible then the output of our distinguisher will equal 1 with non-negligibly different probability depending on which Strategy it is provided with, and hence we obtain a distinguisher. Thus under the Decision Linear problem in $\mathbb{G}$, we can bound the value $\epsilon$ to be at most $\nu'(\lambda)$ for some negligible function $\nu'$. This completes the proof of Lemma C.2.

## C.2 Proof of Theorem 4.2

*Proof sketch.* For all real-world adversaries $\mathcal{A}$ we show how to construct an ideal-world adversary $\mathcal{S}$ such that under the Decision Linear assumption no p.p.t. $D$ can distinguish the output of the Ideal and Real experiments except with negligible probability.

$\mathcal{S}$ runs $\mathcal{A}$ internally, handing it $pk$ output by the trusted party. Whenever $\mathcal{A}$ initiates the User's portion of the BlindDec protocol, $\mathcal{S}$ employs the knowledge extractor for $\mathcal{A}$'s WIPoK to obtain the components of the ciphertext being decrypted.[11] Assuming $\mathcal{A}$'s PoK is valid, $\mathcal{S}$ submits this ciphertext to the trusted party and receives a decryption $m'$ in response. It returns $\mathbf{c}'' = \mathsf{LE.Enc}(pk_{\mathcal{U}}, m')$ to $\mathcal{A}$ and simulates the Decryptor's ZKPoK. If $\Pi_{ZK}$ is secure under the Decision Linear assumption, then we can bound the combined probability that the extractor fails and that $\mathcal{A}$ distinguishes the simulated proof to at most a negligible value $\nu(\lambda)$. Thus it remains only to observe that when $\mathcal{A}$'s WIPoK validates, then distribution of $\mathbf{c}'$ is computationally indistinguishable from the distribution expected from a correct run of the BlindDec protocol.

## C.3 Proof of Theorem 4.3

*Proof sketch.* Let $\mathcal{A}$ be an adversary that wins the Blind game with non-negligible advantage $\epsilon$. We show that $\mathcal{A}$ implies a solver $\mathcal{B}$ for the Decision Linear problem. $\mathcal{B}$ operates as follows. First, $\mathcal{A}$ outputs $(pk, C_0, C_1)$. $\mathcal{B}$ selects $b \in_R \{0,1\}$ and interacts with $\mathcal{B}$ by running $\mathcal{U}(pk, C_b)$ and $\mathcal{U}(pk, C_{b-1})$. Note that LE is IND-CPA-secure under the Decision Linear assumption and we assume that WIPoK is WI under the same. Clearly the two transcripts received by $\mathcal{A}$ thus far are computationally indistinguishable, and if not we can construct a distinguisher for the Decision Linear problem.

[11] Observe that the entire ciphertext consists of elements of the group $\mathbb{G}$. This is particularly important when using Groth-Sahai NIZKs, as that proof system does not possess a knowledge extractor for $\mathbb{Z}_p^*$.