Certificateless Public Key Cryptography*

Sattam S. Al-Riyami and Kenneth G. Paterson†
Information Security Group,
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
s.al-riyami@rhul.ac.uk kenny.paterson@rhul.ac.uk

October 21, 2003

Abstract

This paper introduces the concept of *certificateless public key cryptography* (CL-PKC). In contrast to traditional public key cryptographic systems, CL-PKC does not require the use of certificates to guarantee the authenticity of public keys. It does rely on the use of a trusted third party (TTP) who is in possession of a master key. In these respects, CL-PKC is similar to identity-based public key cryptography (ID-PKC). On the other hand, CL-PKC does not suffer from the key escrow property that seems to be inherent in ID-PKC. Thus CL-PKC can be seen as a model for the use of public key cryptography that is intermediate between traditional certificated PKC and ID-PKC.

We make concrete the concept of CL-PKC by introducing certificateless public key encryption (CL-PKE), signature and key exchange schemes. We also demonstrate how hierarchical CL-PKC can be supported. The schemes are all derived from pairings on elliptic curves. The lack of certificates and the desire to prove the schemes secure in the presence of an adversary who has access to the master key requires the careful development of new security models. For reasons of brevity, the focus in this paper is on the security of CL-PKE. We prove that our CL-PKE scheme is secure in a fully adaptive adversarial model, provided that an underlying problem closely related to the Bilinear Diffie-Hellman Problem is hard.

1 Introduction

The main difficulty today in developing secure systems based on public key cryptography is not the problem of choosing appropriately secure algorithms or implementing those algorithms. Rather, it is the deployment and management of infrastructures to support the authenticity of cryptographic keys: there is a need to provide an assurance to the user about the relationship between a public key and the identity (or authority) of the holder of the corresponding private key. In a traditional Public Key Infrastructure (PKI), this assurance is delivered in the form of certificate, essentially a signature by a Certification Authority (CA) on a public key [1]. The problems of PKI technology are well documented, see for example [22]. Of note are the issues

*This is the full version of a paper with the same title to be presented at Asiacrypt2003.
†This author supported by the Nuffield Foundation, NUF-NAL 02.

associated with certificate management, including revocation, storage and distribution and the computational cost of certificate verification. These are particularly acute in processor or bandwidth-limited environments [13].

Identity-based public key cryptography (ID-PKC), first proposed by Shamir [32], tackles the problem of authenticity of keys in a different way to traditional PKI. In ID-PKC, an entity's public key is derived directly from certain aspects of its identity, for example, an IP address belonging to a network host, or an e-mail address associated with a user. Private keys are generated for entities by a trusted third party called a private key generator (PKG). The first fully practical and secure identity-based public key encryption scheme was presented in [6]. Since then, a rapid development of ID-PKC has taken place. There now exist identity-based key exchange protocols (interactive [31] as well as non-interactive [33]), signature schemes [10, 23, 26], hierarchical schemes [20] and a host of other primitives. It has also been illustrated in [11, 25, 34] how ID-PKC can be used as a tool to enforce what might be termed "cryptographic work-flows", that is, sequences of operations (e.g. authentications) that need to be performed by an entity in order to achieve a certain goal.

The direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them. On the other hand, the dependence on a PKG who uses a system-wide master key to generate private keys inevitably introduces key escrow to ID-PKC systems. For example, the PKG can decrypt any ciphertext in an identity-based public key encryption scheme. Equally problematical, the PKG could forge any entity's signatures in an identity-based signature scheme, so ID-PKC cannot offer true non-repudiation in the way that traditional PKI can. The escrow problem can be solved to a certain extent by the introduction of multiple PKGs and the use of threshold techniques, but this necessarily involves extra communication and infrastructure. Moreover, the compromise of the PKG's master key could be disastrous in an ID-PKC system, and usually more severe than the compromise of a CA's signing key in a traditional PKI. For these reasons, it seems that the use of ID-PKC may be restricted to small, closed groups or to applications with limited security requirements.

1.1 Certificateless Public Key Cryptography

In this paper, we introduce a new paradigm for public key cryptography, which we name certificateless public key cryptography (CL-PKC). Our concept grew out of a search for public key schemes that do not require the use of certificates and yet do not have the built-in key escrow feature of ID-PKC. The solution we propose enjoys both of these properties; in this way, it is a model for the use of public key cryptography that is intermediate between traditional PKI and ID-PKC. Our concept shares some features in common with the self-certificated keys of [21, 27] and with Gentry's recently proposed certificate-based encryption [19].

We demonstrate that our concept of CL-PKC can be made real by specifying certificateless encryption, signature and key exchange schemes. We prove that the encryption scheme is secure in a new and appropriate model, given the hardness of an underlying computational problem. We also demonstrate how certificateless hierarchical schemes can be supported. Our certificateless schemes are all built from bilinear maps on groups. In practice these will be implemented using Weil and Tate pairings on elliptic curves, and the security will rest on a computational problem related to the Bilinear Diffie-Hellman Problem (BDHP).

1.2 Defining CL-PKC

We sketch the defining characteristics of CL-PKC.

A CL-PKC system still makes use of TTP which we name the key generating centre (KGC). By way of contrast to the PKG in ID-PKC, this KGC does *not* have access to entities' private keys. Instead, the KGC supplies an entity A with a *partial private key* D_A which the KGC computes from an identifer $\texttt{ID}_A$ for the entity and a master key. Note that we will often equate A with its identifier $\texttt{ID}_A$. The process of supplying partial private keys should take place confidentially and authentically: the KGC must ensure that the partial private keys are delivered securely to the correct entities. Identifiers can be arbitrary strings.

The entity A then combines its partial private key D_A with some secret information to generate its actual private key S_A. In this way, A's private key is not available to the KGC. The entity A also combines its secret information with the KGC's public parameters to compute its public key P_A. Note that A need not be in possession of S_A before generating P_A: all that is needed to generate both is the same secret information. The system is not identity-based, because the public key is no longer computable from an identity (or identifier) alone.

Entity A's public key might be made available to other entities by transmitting it along with messages (for example, in a signing application) or by placing it in a public directory (this would be more appropriate for an encryption setting). But no further security is applied to the protection of A's public key. In particular, there is no certificate for A's key. To encrypt a message to A or verify a signature from A, entity B makes use of P_A and $\texttt{ID}_A$.

A more formal model for certificateless public key encryption (CL-PKE) will be given in Section 3. Much of this model is also applicable for our other certificateless primitives.

1.3 An Adversarial Model for CL-PKC

Because of the lack of authenticating information for public keys (in the form of a certificate, for example), we must assume that an adversary can replace A's public key by a false key of its choice. This might seem to give the adversary tremendous power and to be disastrous for CL-PKC. However, we will see that an active adversary who attacks in this way gains nothing useful: without the correct private key, whose production requires the partial private key and therefore the cooperation of the KGC, an adversary will not be able to decrypt ciphertexts encrypted under the false public key, produce signatures that verify with the false public key, and so on. (Formally, in the encryption setting, the adversary will not be able to distinguish the encryptions of distinct messages of his choice.)

Of course, we must assume that the KGC does not mount an attack of this type: armed with the partial private key and the ability to replace public keys, the KGC could impersonate any entity in generating a private/public key pair and then making the public key available. Thus we must assume that, while the KGC is in possession of the master key and hence all partial private keys, it is trusted not to replace entities' public keys. However, we assume that the KGC might engage in other adversarial activity, eavesdropping on ciphertexts and making decryption queries, for example. In this way, users invest *roughly* the same level of trust in the KGC as they would in a CA in a traditional PKI – it is rarely made explicit, but such a CA is always assumed not to issue new certificates binding arbitrary public keys and entity combinations of its choice, and especially not for those where it knows the corresponding private key! When compared to ID-PKC, the trust assumptions made of the trusted third party in CL-PKC are much reduced: in ID-PKC, users must trust the PKG not to abuse its knowledge of private keys in performing passive attacks, while in CL-PKC, users need only trust the KGC not to actively propagate false public keys.

The word *roughly* here merits further explanation. In a traditional PKI, if the CA forges certificates, then the CA can be identified as having misbehaved through the existence of two valid certificates for the same identity. This is not the case in our schemes: a new public key could have been created by the legitimate user or by the KGC, and it cannot be easily decided which is the case. The terminology of [21] is useful here: our schemes achieve trust level 2, whereas a traditional PKI reaches trust level 3. However, we can further strengthen security against a malicious KGC in our schemes by allowing entities to choose identifiers $\mathtt{ID}_A$ which bind together their public keys and identities. Now the existence of two different, working public keys for the same identity will identify the KGC as having misbehaved in issuing both corresponding partial private keys. Details of this modification can be found in Section 5.1. With this binding in place, our schemes do reach trust level 3.

In Section 3, we will present an adversarial model for CL-PKE which captures these capabilities in a formal way. The model we present there is a natural generalization of the fully adaptive, multi-user model of [6] to the CL-PKC setting, and involves two distinct types of adversary: one who can replace public keys at will and another who has knowledge of the master key but does not replace public keys. Given our detailed development of this model, the adaptations to existing models that are needed to produce adversarial models for our other certificateless primitives become straightforward.

1.4 Implementation and Applications of CL-PKC

Our presentation of CL-PKC schemes will be at a fairly abstract level, in terms of bilinear maps on groups. However, the concrete realization of these schemes using pairings on elliptic curves is now becoming comparatively routine, after the work of [2, 3, 6, 7, 9, 15, 17, 18] on implementation of pairings and selection of curves with suitable properties. All the schemes we present use a small number of pairing calculations for each cryptographic operation, and some of these can usually be eliminated when repeated operations involving the same identities take place. Public and private keys are small in size: two elliptic curve points for the public key and one for the private key.

The infrastructure needed to support CL-PKC is lightweight when compared to a traditional PKI. This is because, just as with ID-PKC, the need to manage certificates is completely eliminated. This immediately makes CL-PKC attractive for low-bandwidth, low-power situations, for example, mobile security applications, where the need to transmit and check certificates has been identified as a significant limitation [13]. However, it should be pointed out that recently introduced signatures schemes enjoying very short signatures [9] could be used to significantly decrease the size of certificates and create a lightweight PKI. Our CL-PKC signature scheme can also support true non-repudiation, because private keys remain in the sole possession of their legitimate owners.

Revocation of keys in CL-PKC systems can be handled in the same way as in ID-PKC systems. In [6] the idea of appending validity periods to identifiers $\mathtt{ID}_A$ is given as one convenient solution. In the context of CL-PKC, this ensures that any partial private key, and hence any private key, has a limited shelf-life.

As will become apparent, our CL-PKC schemes are actually very closely related to existing pairing-based ID-PKC schemes. One consequence of this is that any infrastructure deployed to support pairing-based ID-PKC (e.g. a PKG) can also be used to support our CL-PKC schemes too: in short, the two types of scheme can peacefully co-exist. In fact, an entity can be granted a private key for a pairing-based ID-PKC scheme and immediately convert it into a private key for our CL-PKC scheme. In this way, an entity who wishes to prevent the PKG exploiting the escrow property of an identity-based system can do so, though at the cost of losing the identity-based nature of its public key.

Although our CL-PKC schemes are no longer identity-based, they do enjoy the property that an entity's private key can be determined after its public key has been generated and used. This is a useful feature. An entity B can encrypt a message for A using A's chosen public key and an identifier $\mathtt{ID}_A$ of B's choice. This identifier should contain A's identity but might also contain a condition that A must demonstrate that it satisfies before the KGC will deliver the corresponding partial private key (which in turn allows A to compute the right private key for decryption). For example, this condition might be that A has a valid driver's licence. The encrypted message then might be A's new insurance document. In this way, B can create a "cryptographic work-flow" that A must carry out before being able to access some information. This kind of application cannot be easily supported using certificate-based systems, because in those systems, the temporal ordering of private key before public key and certificate is fixed. For more applications of work-flows, see [25, 34].

1.5 Related Work

Our work on CL-PKC owes much to the pioneering work of Boneh and Franklin [6, 7] on identity-based public key encryption. In fact, our CL-PKE scheme is derived from the scheme of [6] by making a very simple modification (albeit, one with far-reaching consequences). Our security proofs require significant changes and new ideas to handle our new types of adversary. Likewise, our signature, key exchange and hierarchical schemes also arise by adapting existing ID-PKC schemes.

Another alternative to traditional certificate-based PKI called self-certified keys was introduced by Girault [21] and further developed in [27, 29]. The schemes presented in [21, 27, 29] are structurally somewhat similar to our CL-PKC schemes. In a self-certified scheme, an entity chooses its private key x and corresponding public key y and delivers y to a TTP. The TTP combines y with the identity $\mathtt{ID}$ of that entity to produce a witness w. This witness may just be the TTP's signature on some combination of y and $\mathtt{ID}$ as in [21], part of a signature as in [27], or the result of inverting a trapdoor one-way function based on y and $\mathtt{ID}$ [29]. Given w, $\mathtt{ID}$ and the TTP's public key, any party can extract y, while only the TTP can produce the witness w from y and $\mathtt{ID}$. The schemes offer implicit certification, in that the authenticity of a public key is verified implicitly through the subsequent use of the correct private key.

As in CL-PKC, self-certified keys enable the use of public key cryptography without traditional certificates. However, it can be argued that the witness in a self-certified scheme is really just a lightweight certificate linking $\mathtt{ID}$ and y. Our CL-PKC schemes do not have such witnesses. The self-certified schemes have an advantage over our level 2 CL-PKC schemes in that the communication between an entity and the TTP need not be confidential: there are no partial private keys to be transported to entities. On the other hand, the private key needs to be chosen before the public key can be generated (unlike CL-PKC and ID-PKC schemes), so the elegant applications of CL-PKC to controlling work-flows cannot be realized in self-certified systems. Nor do the self-certified schemes enjoy security proofs. Indeed Saeednia [30] has recently pointed out a basic flaw in the scheme of [21] which allows a cheating TTP to extract an entity's private key; the consequence is that far larger (and less efficient) parameters are needed to create a secure scheme.

Recent work of Gentry [19] exploits pairings to simplify certificate revocation in traditional PKI systems. In Gentry's model, an entity B's private key consists of two components: a first component which that entity chooses for itself and keeps private, and a component which is time-dependent and is issued to B on a regular basis by a CA. Matching the two private key components are two public key components. The first of these is chosen by B's while the second can be computed by A using only some public parameters of the scheme's CA together with the current time value and the assumed value of B's public key. Because of the structure of the CBE scheme, A is then assured that B can only decrypt if he is in possession of both private components. Thus the second private component acts as an *implicit certificate* for relying parties: one that a relying party can be assured is only available to B provided that B's certification has been issued for the current time period by the CA. This approach provides an implicit revocation mechanism for PKIs: notice that there is no need for A to make any status checks on B's public key before encrypting a message for B; rather A's assurance that only B can decrypt comes through trusting the CA to properly update and distribute the second components of private keys.

Gentry's scheme is presented in the context of a traditional PKI model, whereas our work departs from the traditional PKI and ID-PKC models to present a new paradigm for the use of public-key cryptography. However, the two models are conceptually rather similar: both make use of keys that are composed of two parts, one chosen by an entity for itself and the other coming from a trusted authority. In fact, it is possible to modify Gentry's work to divorce it from the setting of a traditional PKI. In the reverse direction, we can modify our scheme to provide CBE functionality by the simple expedient of including expiry information and public keys in identity strings. The details of these modifications are beyond the scope of this paper.

Thus the two models, developed independently[1], are closely related. However, there do remain major differences: our security model assumes an adversary who can extract partial private keys and change public keys even for the challenge identity, whereas Gentry's model, in which the equivalent of partial private keys are already public and bind the public keys to identities, models the adversary slightly differently. Gentry's model requires an adversary to give private keys to the simulator (a rather unusual requirement, though one that can be removed for the specific scheme in [19]). We circumvent this requirement by extending the concept of knowledge extraction to the situation with multiple public keys under the control of the adversary. Moreover, the concrete realizations of the two models are different.

[1]Our work was begun in Autumn 2002, and a version of it presented at a local research seminar in November 2002. The first we became aware of Gentry's work was at the Eurocrypt 2003 conference in May 2003, by which time our work as presented here was substantially complete except for some details in the security proofs.

1.6 Overview of the Paper

Throughout the paper, we focus mostly on certificateless public key encryption for the sake of brevity. Section 2 gives some background definitions for bilinear maps and associated computational problems that we need. In Section 3, we define in detail the notion of a certificateless public key encryption (CL-PKE) scheme, giving a formal model for the capabilities of adversaries and a definition of security. Then in Section 4, we give a concrete CL-PKE scheme and state our result about its security; this result is proved in the appendix. After this, Section 5 sketches certificateless signature, authenticated key exchange and hierarchical schemes. Section 6 concludes the paper.

2 Background Definitions

Throughout the paper, $\mathbb{G}_1$ denotes an additive group of prime order q and $\mathbb{G}_2$ a multiplicative group of the same order. We let P denote a generator of $\mathbb{G}_1$. For us, a pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties:

1. The map e is bilinear: given $Q, W, Z \in \mathbb{G}_1$, we have
$$e(Q, W + Z) = e(Q, W) \cdot e(Q, Z) \text{ and } e(Q + W, Z) = e(Q, Z) \cdot e(W, Z).$$
Consequently, for any $a, b \in \mathbb{Z}_q$, we have
$$e(aQ, bW) = e(Q, W)^{ab} = e(abQ, W) \text{ etc.}$$
2. The map e is non-degenerate: $e(P, P) \neq 1_{\mathbb{G}_2}$.
3. The map e is efficiently computable.

Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. We refer to [2, 3, 6, 7, 9, 15, 17, 18] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security. We note that all our schemes can be adapted to the situation where two different groups are involved on the left-hand side of the pairing map. This increases the range of curves over which our schemes can be realised.

We also introduce here the computational problems that will form the basis of security for our CL-PKC schemes.

Bilinear Diffie-Hellman Problem (BDHP): Let $\mathbb{G}_1$, $\mathbb{G}_2$, P and e be as above. The BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, compute $e(P, P)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if
$$\Pr\left[\mathcal{A}(\langle P, aP, bP, cP \rangle) = e(P, P)^{abc}\right] = \epsilon.$$

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$ and the random bits of $\mathcal{A}$.

Generalized Bilinear Diffie-Hellman Problem (GBDHP): Let $\mathbb{G}_1$, $\mathbb{G}_2$, P and e be as above. The GBDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, output a pair $\langle Q \in \mathbb{G}_1^*, e(P,Q)^{abc} \in \mathbb{G}_2 \rangle$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the GBDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if

$$\Pr\left[\mathcal{A}(\langle P, aP, bP, cP \rangle) = \langle Q, e(P,Q)^{abc} \rangle\right] = \epsilon.$$

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$ and the random bits of $\mathcal{A}$.

Notice that the BDHP is a special case of the GBDHP in which the algorithm outputs the choice $Q = P$. While the GBDHP may appear to be in general easier to solve than the BDHP because the solver gets to choose Q, we know of no polynomial-time algorithm for solving either when the groups $\mathbb{G}_1, \mathbb{G}_2$ and pairing e are appropriately selected. If the solver knows $s \in \mathbb{Z}_q^*$ such that $Q = sP$, then the problems are of course equivalent. The GBDHP is related to generalized versions of the computational Diffie-Hellman problems in $\mathbb{G}_1$ and $\mathbb{G}_2$ in the same way that the BDHP is related to the standard computational Diffie-Hellman problem in those groups [6, 17].

BDH Parameter Generator: As in [6], a randomized algorithm $\mathcal{IG}$ is a BDH parameter generator if $\mathcal{IG}$: (1) takes security parameter $k \geq 1$, (2) runs in polynomial time in k, and (3) outputs the description of groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order q and a pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Formally, the output of the algorithm $\mathcal{IG}(1^k)$ is $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$.

Our security proofs will yield reductions to the BDHP or GBDHP in groups generated by a BDH parameter generator $\mathcal{IG}$. To make statements about the security of our schemes, we will assume that there is no polynomial time algorithm with a non-negligible (in k) advantage in solving the BDHP or GBDHP in groups generated by $\mathcal{IG}$. Our reductions to BDHP and GBDHP can be made concrete.

3 Certificateless Public Key Encryption

In this section we present a formal definition for a certificateless public key encryption (CL-PKE) scheme. We also examine the capabilities which may be possessed by the adversaries against such a scheme and give a security model for CL-PKE.

A CL-PKE scheme is specified by seven randomized algorithms: `Setup`, `Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key`, `Set-Public-Key`, `Encrypt` and `Decrypt`:

`Setup`: This algorithm takes security parameter k and returns the system parameters `params` and `master-key`. The system parameters includes a description of the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Usually, this algorithm is run by the KGC. We assume throughout that `params` are publicly and authentically available, but that only the KGC knows `master-key`.

`Partial-Private-Key-Extract`: This algorithm takes `params`, `master-key` and an identifier for entity A, $\mathtt{ID}_A \in \{0,1\}^*$, as input. It returns a partial private key D_A. Usually this algorithm is run by the KGC and its output is transported to entity A over a confidential and authentic channel.

`Set-Secret-Value`: This algorithm takes as inputs `params` and an entity A's identifier $\mathtt{ID}_A$ as inputs and outputs A's secret value x_A.

Set-Private-Key: This algorithm takes params, an entity A's partial private key D_A and A's secret value x_A as input. The value x_A is used to transform D_A into the (full) private key S_A. The algorithm returns S_A.

Set-Public-Key: This algorithm takes params and entity A's secret value x_A as input and from these constructs the public key P_A for entity A.

Normally both Set-Private-Key and Set-Public-Key are run by an entity A for itself, after running Set-Secret-Value. The same secret value x_A is used in each. Separating them makes it clear that there is no need for a temporal ordering on the generation of public and private keys in our CL-PKE scheme. Usually, A is the only entity in possession of S_A and x_A, and x_A will be chosen at random from a suitable and large set.

Encrypt: This algorithm takes as inputs params, a message $M \in \mathcal{M}$, and the public key P_A and identifier $\mathtt{ID}_A$ of an entity A. It returns either a ciphertext $C \in \mathcal{C}$ or the null symbol $\perp$ indicating an encryption failure. This will always occur in the event that P_A does not have the correct form. In our scheme, this is the only way an encryption failure will occur.

Decrypt: This algorithm takes as inputs params, $C \in \mathcal{C}$, and a private key S_A. It returns a message $M \in \mathcal{M}$ or a message $\perp$ indicating a decryption failure.

Naturally, we insist that output M should result from applying algorithm Decrypt with inputs params, S_A on a ciphertext C generated by using algorithm Encrypt with inputs params, P_A, $\mathtt{ID}_A$ on message M.

3.1 Security model for CL-PKE

Given this formal definition of a CL-PKE scheme, we are now in a position to define adversaries for such a scheme. The standard definition for security for a public key encryption scheme involves indistinguishability of encryptions against a fully-adaptive chosen ciphertext attacker (IND-CCA) [4, 14, 28]. In this definition, there are two parties, the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. The adversary operates in three phases after being presented with a random public key. In Phase 1, $\mathcal{A}$ may make decryption queries on ciphertexts of its choice. In the Challenge Phase, $\mathcal{A}$ chooses two messages M_0, M_1 and is given a challenge ciphertext C^* for one of these two messages M_b by the challenger. In Phase 2, $\mathcal{A}$ may make further decryption queries, but may not ask for the decryption of C^*. The attack ends with $\mathcal{A}$'s guess b' for the bit b. The adversary's advantage is defined to be $\mathrm{Adv}(\mathcal{A}) = 2(\Pr[b' = b] - \frac{1}{2})$.

This model was strengthened for ID-PKC in [6] to handle adversaries who can extract the private keys of arbitrary entities and who choose the identity $\mathtt{ID}_{\mathrm{ch}}$ of the entity on whose public key they are challenged. This extension is appropriate because the compromise of some entities' private keys should not affect the security of an uncompromised entity's encryptions.

Here, we extend the model of [6] to allow adversaries who can extract partial private keys, or private keys, or both, for identities of their choice. Given that our scheme has no certificates, we must further strengthen the model to allow for adversaries who can replace the public key of any entity with a value of their choice. We must also consider carefully how a challenger should respond to key extraction and decryption queries for identities whose public keys have been changed.

Here then is a list of the actions that a general adversary against a CL-PKE scheme may carry out and a discussion of each action.

1. **Extract partial private key of A:** $\mathcal{C}$ responds by running algorithm `Partial-Private-Key-Extract` to generate the partial private key D_A for entity A.

2. **Extract private key for A:** As in [6], we allow our adversary $\mathcal{A}$ to make requests for entities' private keys. If A's public key has not been replaced then $\mathcal{C}$ can respond by running algorithm `Set-Private-Key` to generate the private key S_A for entity A (first running `Set-Secret-Value` for A if necessary). But it is unreasonable to expect $\mathcal{C}$ to be able to respond to such a query if $\mathcal{A}$ has already replaced A's public key. Also as in [6], we insist that $\mathcal{A}$ does not at any point extract the private key for the selected challenge identity $\mathtt{ID}_{\mathrm{ch}}$.

3. **Request public key of A:** Naturally, we assume that public keys are available to $\mathcal{A}$. On receiving a first request for A's public key, $\mathcal{C}$ responds by running algorithm `Set-Public-Key` to generate the public key P_A for entity A (first running `Set-Secret-Value` for A if necessary).

4. **Replace public key of A:** $\mathcal{A}$ can repeatedly replace the public key P_A for any entity A with any value P'_A of its choice. In our concrete CL-PKE schemes, our public keys will have a certain structure that is used to test the validity of public keys before any encryption. We assume here that the adversary's choice P'_A is a valid public key; this assumption can be removed (and our schemes remain secure) at the cost of some additional complexity in our definitions. Note that in our schemes, any entity can easily create public keys that are valid. The current value of an entity's public key is used by $\mathcal{C}$ in any computations (for example, preparing a challenge ciphertext) or responses to $\mathcal{A}$'s requests (for example, replying to a request for the public key). We insist that $\mathcal{A}$ cannot both replace the public key for the challenge identity $\mathtt{ID}_{\mathrm{ch}}$ before the challenge phase *and* extract the partial private key for $\mathtt{ID}_{\mathrm{ch}}$ in some phase – this would enable $\mathcal{A}$ to receive a challenge ciphertext under a public key for which it could compute the private key.

5. **Decryption query for ciphertext C and entity A:** If $\mathcal{A}$ has not replaced the public key of entity A, then $\mathcal{C}$ responds by running the algorithm `Set-Private-Key` to obtain the private key S_A, then running `Decrypt` on ciphertext C and private key S_A and returning the output to $\mathcal{A}$. However, if $\mathcal{A}$ has already replaced the public key of A, then in following this approach, $\mathcal{C}$ will (in general) not decrypt using a private key matching the current public key. So $\mathcal{C}$'s reply to $\mathcal{A}$'s decryption query is likely to be incorrect. Indeed $\mathcal{C}$ most likely will not even know what the private key matching the current public key is! In defining our security model for CL-PKE, we have two options: we could simply accept that these decryptions will be incorrect, or we can insist that $\mathcal{C}$ should somehow properly decrypt ciphertexts even for entities whose public keys have been replaced. The former option could be argued for on grounds of reasonableness: after all, how can $\mathcal{C}$ be expected to provide correct decryptions when $\mathcal{A}$ gets to choose the public key? On the other hand, the latter option results in a more powerful security model, because now decryption queries made under public keys that have been changed will potentially be far more useful to $\mathcal{A}$. For this reason, we adopt the latter option for our model, even though it substantially complicates our proofs of security. (These

decryptions will be handled using special purpose knowledge extractors in our security proofs.) Naturally, as in [6], we prohibit $\mathcal{A}$ from ever making a decryption query on the challenge ciphertext C^* for the combination of identity $\texttt{ID}_{\text{ch}}$ and public key P_{ch} that was used to encrypt M_b. However $\mathcal{A}$ is, for example, allowed to replace the public key for $\texttt{ID}_{\text{ch}}$ with a new value and then request a decryption of C^*, or to change another entity A's public key to P_{ch} (or any other value) and then request the decryption of C^* for entity A.

We also want to consider adversaries who are equipped with `master-key`, in order to model security against an eavesdropping KGC. As discussed in Section 1, we do not allow such an adversary to replace public keys: in this respect, we invest in the KGC the same level of trust as we do in a CA in a traditional PKI. So we will distinguish between two adversary types, with slightly different capabilities:

CL-PKE Type I Adversary: Such an adversary $\mathcal{A}_I$ does not have access to `master-key`. However, $\mathcal{A}_I$ may request public keys and replace public keys with values of its choice, extract partial private and private keys and make decryption queries, all for identities of its choice. As discussed above, we make several natural restrictions on such a Type I adversary:

1. $\mathcal{A}_I$ cannot extract the private key for $\texttt{ID}_{\text{ch}}$ at any point.

2. $\mathcal{A}_I$ cannot request the private key for any identity if the corresponding public key has already been replaced.

3. $\mathcal{A}_I$ cannot both replace the public key for the challenge identity $\texttt{ID}_{\text{ch}}$ before the challenge phase *and* extract the partial private key for $\texttt{ID}_{\text{ch}}$ in some phase.

4. In Phase 2, $\mathcal{A}_I$ cannot make a decryption query on the challenge ciphertext C^* for the combination of identity $\texttt{ID}_{\text{ch}}$ and public key P_{ch} that was used to encrypt M_b.

CL-PKE Type II Adversary: Such an adversary $\mathcal{A}_{II}$ does have access to `master-key`, but may not replace public keys of entities. Adversary $\mathcal{A}_{II}$ can compute partial private keys for itself, given `master-key`. It can also request public keys, make private key extraction queries and decryption queries, both for identities of its choice. The restrictions on this type of adversary are:

1. $\mathcal{A}_{II}$ cannot replace public keys at any point.

2. $\mathcal{A}_{II}$ cannot extract the private key for $\texttt{ID}_{\text{ch}}$ at any point.

3. In Phase 2, $\mathcal{A}_{II}$ cannot make a decryption query on the challenge ciphertext C^* for the combination of identity $\texttt{ID}_{\text{ch}}$ and public key P_{ch} that was used to encrypt M_b.

Chosen ciphertext security for CL-PKE: We say that a CL-PKE scheme is semantically secure against an adaptive chosen ciphertext attack ("IND-CCA secure") if no polynomially bounded adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter k and runs the `Setup` algorithm. It gives $\mathcal{A}$ the resulting system parameters `params`. If $\mathcal{A}$ is of Type I, then the challenger keeps `master-key` to itself, otherwise, it gives `master-key` to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ issues a sequence of requests, each request being either a partial private key extraction, a private key extraction, a request for a public key, a replace public key command or a decryption query for a particular entity. These queries may be asked adaptively, but are subject to the rules on adversary behaviour defined above.

Challenge Phase: Once $\mathcal{A}$ decides that Phase 1 is over it outputs the challenge identity $\mathtt{ID}_{\mathrm{ch}}$ and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$. Again, the adversarial constraints given above apply. In particular, $\mathtt{ID}_{\mathrm{ch}}$ cannot be an identity for which the private key has been extracted. Moreover, if $\mathcal{A}$ is of Type I, then $\mathtt{ID}_{\mathrm{ch}}$ cannot be an identity for which both the public key has been replaced and the partial private key extracted. The challenger now picks a random bit $b \in \{0,1\}$ and computes C^*, the encryption of M_b under the current public key P_{ch} for $\mathtt{ID}_{\mathrm{ch}}$. If the output of the encryption is $\perp$, then $\mathcal{A}$ has immediately lost the game (it has replaced a public key with one not having the correct form). Otherwise, C^* is delivered to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues a second sequence of requests as in Phase 1, again subject to the rules on adversary behaviour above. In particular, no private key extraction on $\mathtt{ID}_{\mathrm{ch}}$ is allowed, and, if $\mathcal{A}$ is of Type I, then the partial private key for $\mathtt{ID}_{\mathrm{ch}}$ cannot be extracted if the corresponding public key was replaced in Phase 1. Moreover, no decryption query can be made on the challenge ciphertext C^* for the combination of identity $\mathtt{ID}_{\mathrm{ch}}$ and public key P_{ch} that was used to encrypt M_b.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}$'s advantage in this game to be $\mathrm{Adv}(\mathcal{A}) := 2(\Pr[b = b'] - \frac{1}{2})$.

4 CL-PKE Schemes from Pairings

In this section, we describe a pair of CL-PKE schemes. Our first scheme, `BasicCL-PKE`, is analogous to the scheme `BasicIdent` of [6], and is included only to serve as a warm-up for our main scheme `FullCL-PKE`. The main scheme is in turn an analogue of the scheme `FullIdent` of [6] and is IND-CCA secure, assuming the hardness of the GBDHP. We prove this in Theorem 1.

4.1 A Basic CL-PKE Scheme

We describe the seven algorithms needed to define `BasicCL-PKE`. We let k be a security parameter given to the `Setup` algorithm and $\mathcal{IG}$ a BDH parameter generator with input k.

`Setup`: This algorithm runs as follows:

1. Run $\mathcal{IG}$ on input k to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a pairing.

2. Choose an arbitrary generator $P \in \mathbb{G}_1$.

3. Select a `master-key` s uniformly at random from $\mathbb{Z}_q^*$ and set $P_0 = sP$.

4. Choose cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$. Here n will be the bit-length of plaintexts.[2]

The system parameters are `params`= $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2 \rangle$. The `master-key` is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

`Partial-Private-Key-Extract`: This algorithm takes as input an identifier $\mathtt{ID}_A \in \{0,1\}^*$, and carries out the following steps to construct the partial private key for entity A with identifier $\mathtt{ID}_A$:

1. Compute $Q_A = H_1(\mathtt{ID}_A) \in \mathbb{G}_1^*$.
2. Output the partial private key $D_A = sQ_A \in \mathbb{G}_1^*$.

The reader will notice that the partial private key of entity A here is identical to that entity's private key in the schemes of [6]. Also notice that A can verify the correctness of the `Partial-Private-Key-Extract` algorithm output by checking $e(D_A, P) = e(Q_A, P_0)$.

`Set-Secret-Value`: This algorithm takes as inputs `params` and an entity A's identifier $\mathtt{ID}_A$ as inputs. It selects $x_A \in \mathbb{Z}_q^*$ at random and outputs x_A as A's secret value.

`Set-Private-Key`: This algorithm takes as inputs `params`, an entity A's partial private key D_A and A's secret value $x_A \in \mathbb{Z}_q^*$. It transforms partial private key D_A to private key S_A by computing $S_A = x_A D_A = x_A s Q_A \in \mathbb{G}_1^*$.

`Set-Public-Key`: This algorithm takes `params` and entity A's secret value $x_A \in \mathbb{Z}_q^*$ as inputs and constructs A's public key as $P_A = \langle X_A, Y_A \rangle$, where $X_A = x_A P$ and $Y_A = x_A P_0 = x_A s P$.

`Encrypt`: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathtt{ID}_A \in \{0,1\}^*$ and public key $P_A = \langle X_A, Y_A \rangle$, perform the following steps:

1. Check that $X_A, Y_A \in \mathbb{G}_1^*$ and that the equality $e(X_A, P_0) = e(Y_A, P)$ holds. If not, output $\perp$ and abort encryption.
2. Compute $Q_A = H_1(\mathtt{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random value $r \in \mathbb{Z}_q^*$.
4. Compute and output the ciphertext:

$$C = \langle rP, M \oplus H_2(e(Q_A, Y_A)^r) \rangle.$$

Notice that this encryption operation is identical to the encryption algorithm in the scheme `BasicIdent` of [6], except for the check on the structure of the public key in step 1 and the use of Y_A in place of $P_0 = P_{pub}$ in step 4.

`Decrypt`: Suppose $C = \langle U, V \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key S_A, compute and output:

$$V \oplus H_2(e(S_A, U)).$$

[2]Note that n needs to grow at least as fast as k in order to obtain security in an OWE model for this scheme. We do not specify n as a function of the group size q or the security parameter k, however, taking $n \approx \log_2 q$ in concrete instantiations would be appropriate.

Notice that if $\langle U = rP, V\rangle$ is the encryption of M for entity A with public key $P_A = \langle X_A, Y_A\rangle$, then we have:

$$\begin{aligned} V \oplus H_2(e(S_A, U)) &= V \oplus H_2(e(x_A s Q_A, rP)) \\ &= V \oplus H_2(e(Q_A, x_A sP)^r) \\ &= V \oplus H_2(e(Q_A, Y_A)^r) \\ &= M, \end{aligned}$$

so that decryption is the inverse of encryption. Again, the similarity to the decryption operation of `BasicIdent` should be apparent.

We have presented this scheme to help the reader understand our `FullCL-PKE` scheme, and so we do not analyse its security in detail. It can be shown that `BasicCL-PKE` is secure in a One-Way Encryption (OWE) model, in which Type I and II adversaries have the same capabilities regarding public and private keys as in our fully adaptive model in Section 3, but where they do not make decryption queries, and where the challenge to the adversary is simply to decrypt a challenge ciphertext. The security relies on the hardness of the GBDHP and assumes H_1 and H_2 are random oracles. In essence, the detailed analysis shows that security against Type II adversaries can be reduced to the difficulty of computing the value $e(Q_A, x_A sP)^r$. Because a Type II adversary has s but not x_A, this is equivalent to the BDHP on input $\langle P, Q_A, U, X_A\rangle$. Likewise, security against a Type I adversary who does not know s but who might replace Y_A by a new value Y'_A can be reduced to the GBDHP on input $\langle P, Q_A = aP, U = rP, P_0 = sP\rangle$, with solution Y'_A, $e(P, Y'_A)^{sra}$. We omit the details.

4.2 A Full CL-PKE Scheme

Now that we have described our basic CL-PKE scheme, we add chosen ciphertext security to it, adapting the Fujisaki-Okamoto padding technique [16]. The algorithms for `FullCL-PKE` are as follows:

`Setup:` Identical to `Setup` for `BasicCL-PKE`, except that we choose two additional cryptographic hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

Now the system parameters are `params`$= \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4\rangle$. The `master-key` and message space $\mathcal{M}$ are the same as in `BasicCL-PKE`. The ciphertext space is now $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$.

`Partial-Private-Key-Extract:` Identical to `BasicCL-PKE`.

`Set-Secret-Value:` Identical to `BasicCL-PKE`.

`Set-Private-Key:` Identical to `BasicCL-PKE`.

`Set-Public-Key:` Identical to `BasicCL-PKE`.

`Encrypt:` To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathtt{ID}_A \in \{0,1\}^*$ and public key $P_A = \langle X_A, Y_A\rangle$, perform the following steps:

1. Check that $X_A, Y_A \in \mathbb{G}_1^*$ and that the equality $e(X_A, P_0) = e(Y_A, P)$ holds. If not, output $\perp$ and abort encryption.

2. Compute $Q_A = H_1(\mathtt{ID}) \in \mathbb{G}_1^*$.

3. Choose a random $\sigma \in \{0,1\}^n$.

4. Set $r = H_3(\sigma, M)$.

5. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_2(e(Q_A, Y_A)^r), M \oplus H_4(\sigma)\rangle.$$

`Decrypt`: Suppose $C = \langle U, V, W\rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key S_A:

1. Compute $V \oplus H_2(e(S_A, U)) = \sigma'$.

2. Compute $W \oplus H_4(\sigma') = M'$.

3. Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output $\perp$ and reject the ciphertext.

4. Output M' as the decryption of C.

When C is a valid encryption of M using P_A and $\mathtt{ID}_A$, it is easy to see that decrypting C will result in an output $M' = M$. We note that W can be replaced by $W = E_{H_4(\sigma)}(M)$ where E denotes a semantically secure symmetric key encryption scheme as in [16] (though our security proofs will require some modifications to handle this case). This concludes the description of `FullCL-PKE`.

4.3 Security of the scheme `FullCL-PKE`

We have the following theorem about the security of `FullCL-PKE`.

Theorem 1 *Let hash functions H_1, H_2, H_3 and H_4 be random oracles. Suppose further that there is no polynomially bounded algorithm that can solve the GBDHP in groups generated by $\mathcal{IG}$ with non-negligible advantage. Then* `FullCL-PKE` *is IND-CCA secure.*

This theorem follows from a sequence of lemmas that are proved in the appendices. It can be made into a concrete security reduction relating the advantage ϵ of a Type I or Type II attacker against `FullCL-PKE` to that of an algorithm to solve GBDHP or BDHP. We omit the rather unaesthetic expressions which result.

The proof strategy for Theorem 1 is in two parts, depending on the type of the adversary.

For a Type II adversary, we first show that the security of `FullCL-PKE` can be reduced to the security of a related (normal) public key encryption scheme `HybridPub` in the usual IND-CCA model. We then use the results of [16] to reduce the security of `HybridPub` to that of a second public key encryption scheme `BasicPub` against OWE adversaries. Finally, we are able to relate the security of `BasicPub` to the hardness of the BDHP.

The proof of Theorem 1 for a Type I attacker is significantly more complicated. Essentially, this is because we have to handle the possibility that a Type I adversary may extract partial private keys as well as replace public keys. The replacement of public keys complicates the handling of decryption queries. We cannot directly apply the results of [16] in this situation; instead we first provide a reduction relating the security of `FullCL-PKE` to that of

HybridPub in an extended IND-CPA model in which adversaries may alter the public key presented by the challenger. This reduction makes use of special-purpose knowledge extraction algorithm to handle decryption queries. Thereafter, we reduce the security to that of BasicPub against similarly extended OWE adversaries. We are then able to relate security to the hardness of the GBDHP. For details, see the appendices.

5 Further CL-PKC Schemes

In this section, we sketch a number of other CL-PKC primitives: a signature scheme based on the identity-based scheme of [23], a key exchange protocol which improves on the security offered by the schemes of [12, 33], and hierarchical and proxy encryption schemes. We begin by outlining an alternative key generation technique which enhances the resilience of our schemes against a cheating KGC and allows for non-repudation of certificateless signatures.

5.1 An Alternative Key Generation Technique

Up to this point, we have assumed that the KGC is trusted to not replace the public keys of users and to only issue one copy of each partial private key, to the correct recipient. This may involve an unacceptable level of trust in the KGC for some users. Our current set up also allows users to create more than one public key for the same partial private key. This can be desirable in some applications, but undesirable in others.

Here we sketch a simple binding technique which ensures that users can only create one public key for which they know the corresponding private key. In our technique, an entity A must first fix its secret value x_A and its public key $P_A = \langle X_A, Y_A \rangle$. We then re-define Q_A to be $Q_A = H_1(\mathtt{ID}_A \| P_A)$ – now Q_A binds A's identifier and public key. The partial private key delivered to entity A is still $D_A = sQ_A$ and the private key created by A is still xsQ_A. However, these are also now bound to A's choice of public key. This binding effectively restricts A to using a single public key, since A can now only compute one private key from D_A.

This technique has a very important additional benefit: it reduces the degree of trust that users need to have in the KGC in our certificateless schemes. In short, the technique raises many of our schemes to trust level 3 in the trust hierarchy of [21], the same level as is enjoyed in a traditional PKI.

In our original scheme, a cheating KGC could replace an entity's public key by one for which it knows the secret value without fear of being identified. We have assumed up to this point that no KGC would engage in such an action, and that users must trust the KGC not to do so. Note that this action is not equivalent to a CA forging a certificate in a traditional PKI: the existence of two valid certificates would surely implicate the CA (though the CA could perhaps revoke the entity's original certificate first).

Now, with our binding technique in place, a KGC who replaces an entity's public key will be implicated in the event of a dispute: the existence of two working public keys for an identity can only result from the existence of two partial private keys binding that identity to two different public keys; only the KGC could have created these two partial private keys. Thus our binding technique makes the KGC's replacement of a public key apparent and equivalent to a CA forging a certificate in a traditional PKI. Theorem 1 still applies for our CL-PKE scheme with this binding in place because of the way in which H_1 is modelled as a random oracle. Notice too that with this binding in place, there is no longer any need to keep partial private keys secret: informally, knowledge of the key $D_A = sQ_A$ does not help an adversary to create the unique private key $S_A = xsQ_A$ that matches the particular public key P_A that is bound to D_A.

The binding technique can be applied to the primitives in this section too. For example, it ensures a stronger form of non-repudiation than is otherwise possible for our certificateless signature scheme in Section 5.2: without the binding, an entity could always repudiate a signature by producing a second working public key and claiming that the KGC had created the signature using the first public key.

Even with this binding in place, the security analysis of our original encryption scheme (in which an adversary can replace public keys) is still important: it models the scenario where an adversary *temporarily* replaces the public key P_A of an entity A with a new value P'_A in an attempt to obtain a ciphertext which he can distinguish, and then resets the public key. In this case, our proof shows that the adversary does not gain any advantage in a distinguishing game unless he has access to the matching partial private key $D'_A = sH_1(\mathtt{ID}_A \| P'_A)$. In turn, this partial private key should not be made available by the KGC. Of course, nothing can prevent a KGC from mounting an attack of this type, but the same applies for the CA in a traditional PKI.

5.2 A Certificateless Signature Scheme

We will describe a certificateless public-key signature (CL-PKS) scheme that is based on a provably secure ID-PKC signature scheme of [23].

In general, a CL-PKS scheme can be specified by seven algorithms: `Setup`, `Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key`, `Set-Public-Key`, `Sign` and `Verify`. These are similar to the algorithms used to define a CL-PKE scheme: `Setup` and `params` are modified to include a description of the signature space $\mathcal{S}$, `Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key` and `Set-Public-Key` are just as before and `Sign` and `Verify` are as follows:

`Sign`: This algorithm takes as inputs `params`, a message $M \in \mathcal{M}$ to be signed and a private key S_A. It outputs a signature $Sig \in \mathcal{S}$.

`Verify`: This algorithm takes as inputs `params`, a message $M \in \mathcal{M}$, the identifier $\mathtt{ID}_A$ and public key P_A of an entity A, and $Sig \in \mathcal{S}$ as the signature to be verified. It outputs `valid`, `invalid` or $\perp$.

Given this general description, we now outline a CL-PKS scheme:

`Setup`: This is identical to `Setup` for our scheme `BasicCL-PKE`, except that now there is only one hash function $H : \{0,1\}^* \times \mathbb{G}_2 \to \mathbb{Z}_q^*$ and `params` is $\langle \mathbb{G}_1, \mathbb{G}_2, n, e, P, P_0, H \rangle$. The signature space is defined as $\mathcal{S} = \mathbb{G}_1 \times \mathbb{Z}_q^*$.

`Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key` and `Set-Public-Key`: Identical to `BasicCL-PKE`.

`Sign`: To sign $M \in \mathcal{M}$ using the private key S_A, perform the following steps:

1. Choose random $a \in \mathbb{Z}_q^*$.
2. Compute $r = e(aP, P) \in \mathbb{G}_2$.
3. Set $v = H(M, r) \in \mathbb{Z}_q^*$.
4. Compute $U = vS_A + aP \in \mathbb{G}_1$.
5. Output as the signature $\langle U, v \rangle$.

`Verify:` To verify a purported signature $\langle U, v \rangle$ on a message $M \in \mathcal{M}$ for identity $\mathtt{ID}_A$ and public key $\langle X_A, Y_A \rangle$:

1. Check that the equality $e(X_A, P_0) = e(Y_A, P)$ holds. If not, output $\perp$ and abort verification.
2. Compute $r = e(U, P) \cdot e(Q_A, -Y_A)^v$.
3. Check if $v = H(M, r)$ holds. If it does, output `valid`, otherwise output `invalid`.

5.3 A Certificateless Authenticated Key Agreement Protocol

A number of identity-based two party key-agreement protocols have been described [12, 33]. All the session keys created in Smart's protocol [33] can trivially be recovered by the TA. The protocol of [33] was later modified by Chen and Kudla [12] to eliminate this escrow capability. However, the TA in the protocol of [12] can still perform a standard man-in-the-middle attack by replacing one ephemeral value with a value of its choice, and can thus impersonate any entity in an undetectable way.

Here we introduce a certificateless key agreement protocol which is only vulnerable to such a man-in-the-middle attack if, in addition to replacing an ephemeral value, a user-specific long-term public key is also replaced. If keys are produced using our binding technique, then such a man-in-the-middle attack mounted by the KGC will leave evidence exposing the KGC's actions.

The initialization for our certificateless key agreement scheme is formally specified using five algorithms: `Setup`, `Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key` and `Set-Public-Key`. These are the same as in `BasicCL-PKE`.

Entities A and B who wish to agree a key first each choose random values $a, b \in \mathbb{Z}_q^*$. Given these initializations, the protocol is as follows:

Protocol messages:
$A \to B$: $T_A = aP$, $\langle X_A, Y_A \rangle$ (1)
$B \to A$: $T_B = bP$, $\langle X_B, Y_B \rangle$ (2)

After the above messages are exchanged, both users check the validity of each other's public keys in the usual way (so A checks $e(X_B, P_0) = e(Y_B, P)$, etc.). Then A computes $K_A = e(Q_B, Y_B)^a \cdot e(S_A, T_B)$ and B computes $e(Q_A, Y_A)^b \cdot e(S_B, T_A)$. It is easy to see that $K = K_A = K_B$ is a key shared between A and B; to ensure forward security, A and B instead use the shared key $H(K \| abP)$ where H is a suitable hash function.

The protocol uses only two passes and is bandwidth-efficient. Bandwidth utilization can be reduced further if the same entities agree many keys: then transmission of only fresh T_A, T_B is needed in each protocol run. Each side computes four pairings; this can be reduced to one pairing each if the same entities agree many keys. The protocol is therefore competitive with those of [12, 33]. Key confirmation can be added with extra protocol passes.

5.4 Hierarchical CL-PKE

In [20], Gentry and Silverberg improved the work of [24] by introducing a totally collusion-resistant, hierarchical, ID-based infrastructure for encryption and signatures. Such an infrastructure spreads the workload of master servers and produces levels which can be used to support short lived keys, for example. However, the hierarchical schemes of [20] still have an undesirable escrow property. Here, we adapt the hierarchical encryption scheme of [20] to our certificateless setting and eliminate the key escrow.

In general, a hierarchical CL-PKE (HCL-PKE) scheme has a root KGC and a hierarchy of entities. Each entity other than the KGC is associated with a level $t \geq 1$ in the hierarchy and with a string `ID-tuple` which identifies that entity's ancestors in the hierarchy. The `ID-tuple` string for an entity at level t with identity $\mathtt{ID}_t$ is $\langle \mathtt{ID}_1, \mathtt{ID}_2, \ldots, \mathtt{ID}_t \rangle$. An HCL-PKE scheme is specified by seven algorithms: `Setup`, `Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key`, `Set-Public-Key`, `Encrypt` and `Decrypt`. Rather than outline the general function of each algorithm, we present a concrete scheme, `BasicHCL-PKE`, whose description should make the general operation of an HCL-PKE scheme clear. The algorithms for `BasicHCL-PKE` are as follows.

`Setup`: This algorithm is identical to `Setup` for `BasicCL-PKE`, except that now the ciphertext space for a level t ciphertext is $\mathcal{C}_t = \mathbb{G}_1^t \times \{0,1\}^n$. The system parameters are `params`= $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2 \rangle$. For ease of presentation, we denote the `master-key` by x_0 instead of s (so we have $P_0 = x_0 P$).

`Partial-Private-Key-Extract`: This algorithm is usually executed by a level $t-1$ entity $\mathtt{ID}_{t-1}$ for a child entity $\mathtt{ID}_t$ at level t. When $t = 1$, this algorithm is executed by the root KGC for $\mathtt{ID}_1$. It takes as input the `ID-tuple` $\langle \mathtt{ID}_1, \mathtt{ID}_2, \ldots, \mathtt{ID}_t \rangle$ and carries out the following steps to construct the partial private key for $\mathtt{ID}_t$:

1. Compute $Q_t = H_1(\mathtt{ID}_1 \| \mathtt{ID}_2 \| \ldots \| \mathtt{ID}_t) \in \mathbb{G}_1^*$.
2. Output $\mathtt{ID}_t$'s partial private key D_t where

$$D_t = D_{t-1} + x_{t-1} Q_t = \sum_{i=1}^{t} x_{i-1} Q_i.$$

The key D_t must be transported to $\mathtt{ID}_t$ over a confidential and authentic channel.

`Set-Secret-value`: This algorithm takes as inputs `params` and level t entity's `ID-tuple` $\langle \mathtt{ID}_1, \mathtt{ID}_2, \ldots, \mathtt{ID}_t \rangle$ as inputs. It selects $x_t \in \mathbb{Z}_q^*$ at random and outputs x_t as $\mathtt{ID}_t$'s secret value.

`Set-Private-Key`: As for `BasicCL-PKE`, except that the private key for $\mathtt{ID}_t$ is denoted by S_t. So $S_t = x_t D_t$.

`Set-Public-Key`: As for `BasicCL-PKE`, except that the public key for $\mathtt{ID}_t$ is denoted by $P_t = \langle X_t, Y_t \rangle$. So $Y_t = x_0 X_t = x_0 x_t P$.

`Encryption`: To encrypt $M \in \mathcal{M}$ for identity $\mathtt{ID}_t$ at level $t \geq 1$ with `ID-tuple` $\langle \mathtt{ID}_1, \mathtt{ID}_2, \ldots, \mathtt{ID}_t \rangle$, perform the following steps:

1. For each $1 \leq i \leq t$, check that the equality $e(X_i, P_0) = e(Y_i, P)$ holds. If any check fails, output $\perp$ and abort encryption.
2. Compute $Q_i = H_1(\mathtt{ID}_1 \| \mathtt{ID}_2 \| \ldots \| \mathtt{ID}_i) \in \mathbb{G}_1^*$ for each $2 \leq i \leq t$.
3. Choose a random $r \in \mathbb{Z}_q^*$.
4. Compute and output the ciphertext:

$$C = \langle U_0, U_2, \ldots, U_t, V \rangle = \langle rP, rQ_2, rQ_3, \ldots, rQ_t, M \oplus H_2(e(Q_1, Y_t)^r) \rangle \in \mathcal{C}_t.$$

Notice that to encrypt a message for a level t entity $\mathtt{ID}_t$, the values Q_i and hence identities $\mathtt{ID}_i$ of all the ancestors of $\mathtt{ID}_t$ are needed.

`Decryption:` Suppose $C = \langle U_0, U_2, \ldots, U_t, V \rangle \in \mathcal{C}_t$ is a `BasicHCL-PKE` ciphertext for a level t entity with `ID-tuple` $\langle \mathtt{ID}_1, \mathtt{ID}_2, \ldots \mathtt{ID}_t \rangle$. Let the public keys of $\mathtt{ID}_i$'s ancestors be $P_i = \langle X_i, Y_i \rangle$ $(1 \leq i < t)$. Then to decrypt the ciphertext C using the private key S_t, compute and output:

$$V \oplus H_2\left(\frac{e(S_t, U_0)}{\prod_{i=2}^{t} e(x_t X_{i-1}, U_i)}\right).$$

Using properties of the bilinear map e, we have:

$$\begin{aligned}
\frac{e(S_t,U_0)}{\prod_{i=2}^{t} e(x_t X_{i-1},U_i)} &= e(x_t \textstyle\sum_{i=1}^{t} x_{i-1} Q_i, rP) \cdot \prod_{i=2}^{t} e(x_t x_{i-1} P, rQ_i)^{-1} \\
&= e(x_t \textstyle\sum_{i=1}^{t} x_{i-1} Q_i, rP) \cdot e(-\sum_{i=2}^{t} x_t x_{i-1} Q_i, rP) \\
&= e(x_t x_0 Q_1, rP) \\
&= e(Q_1, x_t x_0 P)^r \\
&= e(Q_1, Y_t)^r
\end{aligned}$$

so that decryption is the inverse of encryption.

This completes our description of `BasicHCL-PKE`. It is straightforward to adapt this scheme as we did in building `FullCL-PKE` from `BasicCL-PKE`, to obtain a scheme that is secure against fully-adaptive chosen ciphertext attackers. We must assume here that no ancestor $\mathtt{ID}_k$ of our level t entity $\mathtt{ID}_t$ replaces the public key of $\mathtt{ID}_t$. Even with the extra binding step in place, our hierarchical schemes do not offer a true equivalent of trust level 3: although it is then possible to detect that a public key has been replaced by an ancestor, it is not possible to pinpoint exactly which ancestor is responsible. (Moreover, we cannot allow partial private keys to be made public in this setting as this would enable *any* adversary to mount a successful key attack by replacing the public key of $\mathtt{ID}_t$ – finding this attack is left as an exercise for the reader.) We note that an extension of the hybrid PKI/ID-PKC scheme of [11] has stronger security guarantees. However, this approach still requires certification for intermediate entities, and our primary focus is on completely certificate-free infrastructures.

The CL-PKS scheme of Section 5.2 can also be adapted to produce a hierarchical, certificateless signature scheme.

5.5 Proxy Decryption

We demonstrate how our HCL-PKE scheme BasicHCL-PKE supports two kinds of proxy decryption: an entity A with identifier $\mathtt{ID}_t$ at level $t \geq 1$ can efficiently delegate decryption to either a proxy at level $t-1$ (if $t \geq 2$) or a proxy at level $t+1$. This is an important feature because the decryption and encryption costs in our HCL-PKE scheme grow roughly linearly with t, so that an unacceptably high computational burden may be placed on entities located low in the hierarchy.

To prepare a ciphertext $C = \langle U_0, U_2, \ldots, U_t, V \rangle$ for proxy decryption, entity A with identifier $\mathtt{ID}_t$ located at level t transforms C by appending some fixed keying information and a string `proxy` to it to obtain a new ciphertext:

$$C_{\mathtt{proxy}} = \langle C, \langle x_t X_1, x_t X_2, \ldots, x_t X_{t-1} \rangle, \mathtt{proxy} \rangle.$$

Here, the value of `proxy` depends on whether decryption is being delegated to an entity at level $t-1$ or $t+1$. So we have two cases:

Proxy at level $t-1$:

1. A sets $\mathtt{proxy} = \langle x_t U_0 \rangle$ and forwards $C_{\mathtt{proxy}}$ to level $t-1$ entity B with identifier $\mathtt{ID}_{t-1}$.
2. B decrypts $C_{\mathtt{proxy}}$ as follows:

$$V \oplus H_2 \left(\frac{e(D_{t-1} + x_{t-1} Q_t, x_t U_0)}{\prod_{i=2}^{t} e(x_t X_{i-1}, U_i)} \right) = V \oplus H_2 \left(\frac{e(S_t, U_0)}{\prod_{i=2}^{t} e(x_t X_{i-1}, U_i)} \right) = M.$$

Proxy at level $t+1$:

1. A sets $\mathtt{proxy} = \langle x_t U_0, e(x_t Q_{t+1}, x_t U_0) \rangle$ and forwards $C_{\mathtt{proxy}}$ to level $t+1$ entity B with identifier $\mathtt{ID}_{t+1}$.
2. B decrypts $C_{\mathtt{proxy}}$ as follows:

$$V \oplus H_2 \left(\frac{e(D_{t+1}, x_t U_0)}{e(x_t Q_{t+1}, x_t U_0) \cdot \prod_{i=2}^{t} e(x_t X_{i-1}, U_i)} \right) = V \oplus H_2 \left(\frac{e(S_t, U_0)}{\prod_{i=2}^{t} e(x_t X_{i-1}, U_i)} \right) = M.$$

Notice that the proxy capability that A delegates is one-time only: in each of our two cases, to perform decryption, B needs a component $x_t U_0$ that depends both on the ciphertext and on A's secret. Of course, our proxy schemes shield A's secret x_t and private key S_t from all entities, including the proxy. Notice also that the proxy ciphertext in our level $t+1$ proxy scheme contains enough information allowing it to be decrypted by our level $t-1$ proxy. So proxy ciphertexts produced for A's children can also be decrypted by A's parent.

6 Conclusions

In this paper we introduced the concept of *certificateless public key cryptography*, a model for the use of public key cryptography that is intermediate between the identity-based approach and traditional PKI. We showed how our concept can be realized by specifying a certificateless public key encryption (CL-PKE) scheme that is based on bilinear maps. The scheme enjoys short public and private keys. We showed that our CL-PKE scheme is secure in an appropriate model, assuming that the generalized Bilinear Diffie-Hellman Problem (GBDHP) is hard. We also rounded out our treatment by briefly presenting a number of other certificateless primitives: a signature scheme, key agreement protocol, a hierarchical encryption scheme and proxy decryption schemes.

In future work, we intend to develop security models and proofs for these other primitives. We fully expect that certificateless versions of yet more primitives can be devised by adapting existing identity-based schemes. A fruitful area of research may be special-purpose signature schemes [5, 8, 35]. Finally, we anticipate that pairings will give birth to further models for the use of public key cryptography. Our work and the recent work of [19] certainly point in this direction.

Acknowledgement

We would like to thank Alex Dent and Steven Galbraith for their comments on a draft of this paper. We would also like to thank Dan Boneh and Craig Gentry for helpful discussions on the paper.

References

[1] C. Adams and S. Lloyd. *Understanding Public-Key Infrastructure – Concepts, Standards, and Deployment Considerations.* Macmillan Technical Publishing, Indianapolis, USA, 1999.

[2] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *LNCS*, pages 354–368. Springer-Verlag, 2002.

[3] P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in communication networks – SCN'2002*, volume 2576 of *LNCS*, pages 263–273. Springer-Verlag, 2002.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – CRYPTO 98*, volume 1462 of *LNCS*. Springer-Verlag, 1998.

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

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.

[7] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. `http://www.crypto.stanford.edu/~dabo/abstracts/ibe.html`, full version of [6].

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verfiably encrypted

signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer-Verlag, 2003.

[9] D. Boneh, H. Shacham, and B. Lynn. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag, 2001.

[10] J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Y. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *LNCS*, pages 18–30. Springer-Verlag, 2002.

[11] L. Chen, K. Harrison, A. Moss, D. Soldera, and N.P. Smart. Certification of public keys within an identity based system. In A. H. Chan and V. D. Gligor, editors, *Information Security, 5th International Conference, ISC*, volume 2433 of *LNCS*, pages 322–333. Springer-Verlag, 2002.

[12] L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. Cryptology ePrint Archive, Report 2002/184, 2002. `http://eprint.iacr.org/`.

[13] J. Dankers, T. Garefalakis, R. Schaffelhofer, and T. Wright. Public key infrastructure in mobile systems. *IEE Electronics and Commucation Engineering Journal*, 14(5):180–190, 2002.

[14] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.

[15] R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. Cryptology ePrint Archive, Report 2002/094, 2002. `http://eprint.iacr.org/`.

[16] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer-Verlag, 1999.

[17] S.D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Proceedings of AsiaCrypt 2001*, volume 2248 of *LNCS*, pages 495–513. Springer-Verlag, 2001.

[18] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory 5th International Symposium, ANTS-V*, volume 2369 of *LNCS*, pages 324–337. Springer-Verlag, 2002.

[19] C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 272–293. Springer-Verlag, 2003.

[20] C. Gentry and A. Silverberg. Heirarchical ID-based cryptography. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer-Verlag, 2002.

[21] M. Girault. Self-certified public keys. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT'91*, volume 547 of *LNCS*, pages 490–497. Springer-Verlag, 1992.

[22] P. Gutmann. PKI: It's not dead, just resting. *IEEE Computer*, 35(8):41–49, 2002.

[23] F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography 9th Annual International Workshop, SAC 2002*, volume 2595 of *LNCS*, pages 310–324. Springer-Verlag, 2003.

[24] L.R. Knudsen, editor. *Towards Hierarchical Identity-Based Encryption*, volume 2332 of *LNCS*. Springer-Verlag, 2002.

[25] K.G. Paterson. Cryptography from pairings: a snapshot of current research. *Information Security Technical Report*, 7(3):41–54, 2002.

[26] K.G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*,

38(18):1025–1026, 2002.
[27] H. Petersen and P. Horster. Self-certified keys – concepts and applications. In *3rd Int. Conference on Communications and Multimedia Security*. Chapman and Hall, 1997.
[28] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attacks. In *Advances in Cryptology – CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, 1991.
[29] S. Saeednia. Identity-based and self-certified key-exchange protocols. In V. Varadharajan, J. Pieprzyk, and Y. Mu, editors, *Information Security and Privacy, Second Australasian Conference, ACISP*, volume 1270 of *LNCS*, pages 303–313. Springer-Verlag, 1997.
[30] S. Saeednia. A note on Girault's self-certified model. *Information Processing Letters*, 86:323–327, 2003.
[31] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, January 2000.
[32] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1984.
[33] N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.
[34] N.P. Smart. Access control using pairing based cryptography. In *Proceedings CT-RSA 2003*, volume 2612 of *LNCS*, pages 111–121. Springer-Verlag, 2003.
[35] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 533–547. Springer-Verlag, 2002.

Appendix A: Proofs of Security for `FullCL-PKE`

A.1 Two Public Key Encryption Schemes

We define two public key encryption schemes `BasicPub` and `HybridPub`. These will be used as tools in our security proof for `FullCL-PKE`.

`BasicPub`: This scheme is specified by three algorithms: `Key-Generation`, `Encrypt` and `Decrypt`.

`Key-Generation`:

1. Run $\mathcal{IG}$ to generate $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ with the usual properties. Choose a generator $P \in \mathbb{G}_1$.
2. Pick a random $Q \in \mathbb{G}_1^*$, a random $s \in \mathbb{Z}_q^*$ and a random $x \in \mathbb{Z}_q^*$.
3. Set $P_0 = sP$, $X = xP$, $Y = xsP$ and $S = xsQ$.
4. Choose a cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0, 1\}^n$.

The message and ciphertext spaces for `BasicPub` are $\mathcal{M} = \{0, 1\}^n$ and $\mathcal{C} = \mathbb{G}_1 \times \{0, 1\}^n$. The public key is $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2 \rangle$ and the private key is $S = xsQ$.

`Encrypt`: To encrypt $M \in \mathcal{M}$, do the following:

1. Check that the equality $e(X, P_0) = e(Y, P)$ holds. If not, output $\perp$ and abort encryption.

2. Choose a random $r \in \mathbb{Z}_q^*$.

3. Set the ciphertext to be:

$$C = \langle rP, M \oplus H_2(e(Q, Y)^r) \rangle.$$

`Decrypt`: Let $C = \langle U, V \rangle \in \mathcal{C}$ be the ciphertext. To decrypt C using private key S, compute:

$$V \oplus H_2(e(S, U)) = M.$$

It is easy to see that `Decrypt` is the inverse function to `Encrypt`.

`HybridPub`: This scheme is obtained by applying the hybridisation construction of [16] to `BasicPub`. Again, this scheme is specified by three algorithms: `Key-Generation`, `Encrypt` and `Decrypt`.

`Key-Generation`: This algorithm is identical to that for `BasicPub`, except that we choose two additional hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

Now the public key is $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2, H_3, H_4 \rangle$. The private key is still $S = xsQ$, the message space is still $\mathcal{M} = \{0,1\}^n$, but the ciphertext space is now $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$.

`Encrypt`: To encrypt $M \in \mathcal{M}$, perform the following steps:

1. Check that the equality $e(X, P_0) = e(Y, P)$ holds. If not, output $\perp$ and abort encryption.

2. Choose a random $\sigma \in \{0,1\}^n$.

3. Set $r = H_3(\sigma, M)$.

4. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_2(e(Q, Y)^r), M \oplus H_4(\sigma) \rangle.$$

`Decrypt`: To decrypt $C = \langle U, V, W \rangle \in \mathcal{C}$ using private key S, do the following:

1. Compute $V \oplus H_2(e(S, U)) = \sigma'$.

2. Compute $W \oplus H_4(\sigma') = M'$.

3. Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output $\perp$ and reject the ciphertext.

4. Output M' as the decryption of C.

A.2 Adversaries for `BasicPub` and `HybridPub`

Here we define adversaries appropriate to the schemes `BasicPub` and `HybridPub`, remembering that we want to model attackers who can replace public keys, or who may know the value s.

We recall the definition of OWE security for a standard public key encryption scheme from [6]: the adversary $\mathcal{A}$ is given a random public key K_{pub} and a ciphertext C which is the encryption of a random message M under K_{pub}. The adversary's goal is to recover M

and $\mathcal{A}$ is said to have advantage ϵ in attacking the system if $\Pr[\mathcal{A}(K_{pub}, C) = M] = \epsilon$. The adversary is called an *OWE adversary.*

We define $\mathcal{A}_{II}$ to be a Type II OWE adversary against `BasicPub` if $\mathcal{A}_{II}$ is an OWE adversary against `BasicPub` in the sense defined above, but is also in possession of the value s. We define a Type I OWE adversary $\mathcal{A}_I$ against `BasicPub` as follows: $\mathcal{A}_I$ is given a random public key K_{pub} for `BasicPub`; $\mathcal{A}_I$ then decides if it wishes to change the parameters $\langle X, Y\rangle$ in that public key to a valid pair $\langle X', Y'\rangle$ of its choice. The (possibly new) public key K'_{pub} is then used by the challenger to encrypt a random message M to produce a ciphertext C. Note that this ciphertext may be $\perp$, in which case the adversary has failed. The adversary's goal is to recover M; $\mathcal{A}_I$ is said to have advantage ϵ in attacking the system if $Adv(\mathcal{A}_I) := \Pr[\mathcal{A}_I(K'_{pub}, C) = M]$ is equal to ϵ.

Next we define IND-CPA and IND-CCA adversaries for `HybridPub`. A Type II IND-CCA adversary for `HybridPub` is simply the usual IND-CCA adversary against this public key encryption scheme, as defined in [4], except that the adversary is also given the value s. A Type I IND-CCA adversary $\mathcal{A}_I$ for `HybridPub` is a slightly modified version of the usual `IND-CCA` adversary: $\mathcal{A}_I$ may repeatedly replace the public key components $\langle X, Y\rangle$ by a valid pair $\langle X', Y'\rangle$ of its choice. All the challenger's replies to $\mathcal{A}_I$'s decryption queries, as well as the result of encrypting M_b in the Challenge Phase, should be with respect to the current value of the public key. If the challenger's encryption algorithm outputs $\perp$ during the encryption of M_b, then $\mathcal{A}_I$ has failed. As usual, $\mathcal{A}_I$'s task is to output a guess b' for bit b and its advantage $\mathrm{Adv}(\mathcal{A}_I)$ is defined to be $2(\Pr[b' = b] - \frac{1}{2})$. Type I and II IND-CPA adversaries are defined in exactly the same way, except that the adversary is not given any access to the decryption oracle.

A.3: Statements of Lemmas

We present a series of lemmas. Theorem 1 for Type I adversaries follows by combining Lemmas 2, 3, 4, 5 and 8. Similarly, Theorem 1 for Type II adversaries follows by combining Lemmas 6, 7 and 8.

Lemma 2 *Suppose that H_1, H_2, H_3 and H_4 are random oracles and that there exists an IND-CCA Type I adversary $\mathcal{A}_I$ against* `FullCL-PKE`*. Suppose $\mathcal{A}_I$ has advantage ϵ, runs in time t, makes q_i queries to H_i ($1 \le i \le 4$) and makes q_d decryption queries. Then there is an algorithm $\mathcal{B}$ which acts as either a Type I or a Type II adversary against* `HybridPub`*. Moreover, $\mathcal{B}$ either has advantage at least $\epsilon\lambda^{q_d}/4q_1$ when playing as a Type I adversary, or has advantage at least $\epsilon\lambda^{q_d}/4q_1$ when playing as a Type II adversary. $\mathcal{B}$ runs in time $t + O((q_3 + q_4)q_d t')$. Here t' is the running time of the* `BasicCL-PKE` *encryption algorithm and*

$$1 - \lambda \le (q_3 + q_4) \cdot \epsilon_{OWE}(t + O((q_3 + q_4)q_d t', q_2) + \epsilon_{GBDHP}(t + O((q_3 + q_4)q_d t') + 3q^{-1} + 2^{-n+1},$$

where $\epsilon_{OWE}(T, q')$ denotes the highest advantage of any Type I or Type II OWE adversary against `BasicPub` *which operates in time T and makes q' hash queries to H_2, and $\epsilon_{GBDHP}(T)$ denotes the highest advantage of any time T algorithm to solve GBDHP in groups of order q generated by $\mathcal{IG}$.*

Lemma 3 *Suppose that H_3 and H_4 are random oracles. Let $\mathcal{A}_I$ be a Type I IND-CPA adversary against* `HybridPub` *which has advantage ϵ and makes q_4 queries to H_4. Then there* *exists a Type I OWE adversary $\mathcal{A}_I'$ against* `BasicPub` *which runs in time $O(time(\mathcal{A}_I))$ and has advantage at least $\epsilon/2(q_3+q_4)$.*

Lemma 4 *Suppose that H_3 and H_4 are random oracles. Let $\mathcal{A}_I$ be a Type II IND-CPA adversary against* `HybridPub` *which has advantage ϵ and makes q_4 queries to H_4. Then there exists a Type II OWE adversary $\mathcal{A}_I'$ against* `BasicPub` *which runs in time $O(time(\mathcal{A}_{II}))$ and has advantage at least $\epsilon/2(q_3+q_4)$.*

Lemma 5 *Suppose that H_2 is a random oracle. Suppose there exists a Type I OWE adversary $\mathcal{A}_I$ against* `BasicPub` *which makes at most q_2 queries to H_2 and which has advantage ϵ. Then there exists an algorithm $\mathcal{B}$ to solve the GBDHP which runs in time $O(time(\mathcal{A}_I))$ and has advantage at least $(\epsilon - \frac{1}{2^n})/q_2$.*

Lemma 6 *Suppose that H_1 is a random oracle and that there exists an IND-CCA Type II adversary $\mathcal{A}_{II}$ on* `FullCL-PKE` *with advantage ϵ which makes at most q_1 queries to H_1. Then there is an IND-CCA Type II adversary on* `HybridPub` *with advantage at least ϵ/q_1 which runs in time $O(time(\mathcal{A}_{II}))$.*

The following lemma is easily proven using [16, Theorem 14], noting that s can be made available to Type II adversaries against `HybridPub` and `BasicPub` simply by including it in public keys. Doing so converts these adversaries into normal IND-CCA and OWE adversaries against `HybridPub` and `BasicPub` respectively.

Lemma 7 *Suppose that H_3, H_4 are random oracles. Let $\mathcal{A}_{II}$ be a Type II IND-CCA adversary against* `HybridPub` *which has advantage ϵ, makes q_d decryption queries, q_3 queries to H_3 and q_4 queries to H_4. Then there exists a Type II OWE adversary $\mathcal{A}_{II}'$ against* `BasicPub` *with*

$$\begin{array}{rcl} time(\mathcal{A}_{II}') & = & time(\mathcal{A}_{II}) + O(n(q_3+q_4)) \\ Adv(\mathcal{A}_{II}') & \geq & \frac{1}{2(q_3+q_4)}\left((\epsilon+1)(1-q^{-1}-2^{-n})^{q_d}-1\right). \end{array}$$

Here, we have used the fact that `HybridPub` is $1/q$-uniform in the sense of [16, Definition 5].

Lemma 8 *Suppose that H_2 is a random oracle. Suppose there exists a Type II OWE adversary $\mathcal{A}_{II}$ against* `BasicPub` *which makes at most q_2 queries to H_2 and which has advantage ϵ. Then there exists an algorithm $\mathcal{B}$ to solve the BDHP which runs in time $O(time(\mathcal{A}_{II}))$ and has advantage at least $(\epsilon - \frac{1}{2^n})/q_2$.*

A.4: Proofs of Lemmas

Proof of Lemma 2: This proof is complicated. It may be useful to read the proof of Lemma 6 and then return to this proof.

Let $\mathcal{A}_I$ be a Type I IND-CCA adversary against `FullCL-PKE`. Suppose $\mathcal{A}_I$ has advantage ϵ, runs in time t, makes q_i queries to random oracle H_i ($1 \leq i \leq 4$) and makes q_d decryption queries. We show how to construct from $\mathcal{A}_I$ an adversary $\mathcal{B}$ that acts either as a Type I IND-CCA adversary against `HybridPub` or as a Type II IND-CCA adversary against `HybridPub`. We assume that challengers $\mathcal{C}_I, \mathcal{C}_{II}$ for both types of game are available to $\mathcal{B}$.

Adversary $\mathcal{B}$ begins by choosing a random bit c and an index I uniformly at random with $1 \le I \le q_1$. If $c = 0$, then $\mathcal{B}$ chooses to play against $\mathcal{C}_I$ and aborts $\mathcal{C}_{II}$. Here, $\mathcal{B}$ will build a Type I IND-CPA adversary against `HybridPub` and fails against $\mathcal{C}_{II}$. When $c = 1$, $\mathcal{B}$ chooses to play against $\mathcal{C}_{II}$ and aborts $\mathcal{C}_I$. Here, $\mathcal{B}$ will build a Type II IND-CPA adversary against `HybridPub` and fails against $\mathcal{C}_I$. In either case, $\mathcal{C}$ will denote the challenger against which $\mathcal{B}$ plays for the remainder of this proof.

We let $\mathcal{H}$ denote the event that $\mathcal{A}_I$ chooses $\mathtt{ID}_I$ as the challenge identity $\mathtt{ID}_{\text{ch}}$. We let $\mathcal{F}_0$ denote the event that $\mathcal{A}_I$ extracts the partial private key for entity $\mathtt{ID}_I$ and $\mathcal{F}_1$ denote the event that $\mathcal{A}_I$ replaces the public key of entity $\mathtt{ID}_I$ at some point in its attack.

The general strategy of the proof is as follows. If $c = 0$ and the event $\mathcal{F}_0$ occurs, $\mathcal{B}$ will have to abort and will be unsuccessful. If $\mathcal{F}_0$ does not occur, and if the event $\mathcal{H}$ does occur, then $\mathcal{B}$'s success probability will be related to that of $\mathcal{A}_I$. On the other hand, if $c = 1$ and event $\mathcal{F}_1$ occurs, $\mathcal{B}$ will again have to abort and will be unsuccessful. If $\mathcal{F}_1$ does not occur, but $\mathcal{H}$ does occur, then $\mathcal{B}$'s success probability will again be related to that of $\mathcal{A}_I$. Overall, we will show that $\mathcal{B}$'s advantage in its mixed-game strategy is non-negligible if $\mathcal{A}$'s is. It is then easy to see that $\mathcal{B}$ has a non-negligible advantage for at least one of the two game types.

If $c = 0$, then $\mathcal{C}$ is a Type I challenger for `HybridPub` and begins by supplying $\mathcal{B}$ with a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2, H_3, H_4 \rangle$. If $c = 1$, then $\mathcal{C}$ is a Type II challenger and so supplies $\mathcal{B}$ with a public key K_{pub} together with the value s such that $P_0 = sP$.

Then $\mathcal{B}$ simulates the algorithm `Setup` of `FullCL-PKE` for $\mathcal{A}_I$ by supplying $\mathcal{A}_I$ with `params`$= \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4 \rangle$. Here H_1 is a random oracle that will be controlled by $\mathcal{B}$.

Adversary $\mathcal{A}_I$ may make queries of the random oracles H_i, $1 \le i \le 4$, at any time during its attack. These are handled as follows:

H_1 **queries:** $\mathcal{B}$ maintains a list of tuples $\langle \mathtt{ID}_i, Q_i, b_i, x_i, X_i, Y_i \rangle$ which we call the H_1 list. The list is initially empty, and when $\mathcal{A}_I$ queries H_1 on input $\mathtt{ID} \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

1. If `ID` already appears on the H_1 list in a tuple $\langle \mathtt{ID}_i, Q_i, b_i, x_i, X_i, Y_i \rangle$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}) = Q_i \in \mathbb{G}_1^*$.

2. If `ID` does not already appear on the list and `ID` is the I-th distinct H_1 query made by $\mathcal{A}_I$, then $\mathcal{B}$ picks b_I at random from $\mathbb{Z}_q^*$, outputs $H(\mathtt{ID}) = b_I Q$ and adds the entry $\langle \mathtt{ID}, b_I Q, b_I, \perp, X, Y \rangle$ to the H_1 list.

3. Otherwise, when `ID` does not already appear on the list and `ID` is the i-th distinct H_1 query made by $\mathcal{A}_I$ where $i \neq I$, $\mathcal{B}$ picks b_i and x_i at random from $\mathbb{Z}_q^*$, outputs $H(\mathtt{ID}) = b_i P$ and adds $\langle \mathtt{ID}, b_i P, b_i, x_i, x_i P, x_i P_0 \rangle$ to the H_1 list.

Notice that with this specification of H_1, the `FullCL-PKE` partial private key for $\mathtt{ID}_i$ $(i \neq I)$ is equal to $b_i P_0$ while the public key for $\mathtt{ID}_i$ is $\langle x_i P, x_i P_0 \rangle$ and the private key for $\mathtt{ID}_i$ is $x_i b_i P_0$. These can all be computed by $\mathcal{B}$ when $c = 0$. Additionally, when $c = 1$ (so $\mathcal{B}$ has s), $\mathcal{B}$ can compute $s b_I Q$, the partial private key of $\mathtt{ID}_I$.

H_2 queries: Any H_2 queries made by $\mathcal{A}_I$ are passed to $\mathcal{C}$ to answer. We do need to assume in the course of the proof that H_2 is a random oracle.

H_3 and H_4 queries: Adversary $\mathcal{B}$ passes $\mathcal{A}_I$'s H_3 and H_4 queries to $\mathcal{C}$ to answer, but keeps lists $\langle \sigma_j, M_j, H_{3,j}\rangle$ and $\langle \sigma'_i, H_{4,i}\rangle$ of $\mathcal{A}_I$'s distinct queries and $\mathcal{C}$'s replies to them.

Phase 1: After receiving `params` from $\mathcal{B}$, $\mathcal{A}_I$ launches Phase 1 of its attack, by making a series of requests, each of which is either a partial private key extraction for an entity, a private key extraction for an entity, a request for a public key for an entity, a replacement of a public key for an entity or a decryption query for an entity. We assume that $\mathcal{A}_I$ always makes the appropriate H_1 query on the identity `ID` for that entity before making one of these requests. $\mathcal{B}$ replies to these requests as follows:

Partial Private Key Extraction: Suppose the request is on $\mathtt{ID}_i$. There are three cases:

1. If $i \neq I$, then $\mathcal{B}$ replies with b_iP_0.
2. If $i = I$ and $c = 0$, then $\mathcal{B}$ aborts.
3. If $i = I$ and $c = 1$, then $\mathcal{B}$ replies with sb_IQ.

Private Key Extraction: Suppose the request is on $\mathtt{ID}_i$. We can assume that the public key for $\mathtt{ID}_i$ has not been replaced. There are two cases:

1. If $i \neq I$, then $\mathcal{B}$ outputs $x_ib_iP_0$.
2. If $i = I$, then $\mathcal{B}$ aborts.

Request for Public Key: If the request is on $\mathtt{ID}_i$ then $\mathcal{B}$ returns $\langle X_i, Y_i\rangle$ by accessing the H_1 list.

Replace Public Key: Suppose the request is to replace the public key for $\mathtt{ID}_i$ with value $\langle X'_i, Y'_i\rangle$. (We know that this will be a valid public key, i.e. a key satisfying $e(X'_i, P_0) = e(Y'_i, P)$). There are two cases:

1. If $i = I$ and $c = 1$, then $\mathcal{B}$ aborts.
2. Otherwise, $\mathcal{B}$ replaces the current entries X_i, Y_i in the H_1 list with the new entries X'_i, Y'_i. If $i = I$, then $\mathcal{B}$ makes a request to its challenger $\mathcal{C}$ to replace the public key components $\langle X, Y\rangle$ in K_{pub} with new values $\langle X'_I, Y'_I\rangle$.

Decryption Queries: Suppose the request is to decrypt ciphertext $\langle U, V, W\rangle$ for $\mathtt{ID}_\ell$, where (as discussed in Section 3), the private key that should be used is the one corresponding to the current value of the public key for $\mathtt{ID}_i$. Notice that even when $\ell = I$, $\mathcal{B}$ cannot make use of $\mathcal{C}$ to answer the query, because $\mathcal{B}$ is meant to be an IND-CPA adversary. Instead $\mathcal{B}$ makes use of an algorithm $\mathcal{KE}$ to perform all the decryptions. This algorithm, essentially a knowledge extractor in the sense of [4, 16], is not perfect, but as we shall show below, the probability that it decrypts incorrectly is sufficiently low that it can be used in place of a true decryption algorithm making use of private keys. Algorithm $\mathcal{KE}$ is defined as follows:

Algorithm $\mathcal{KE}$: The input to the algorithm is a ciphertext $C = \langle U, V, W\rangle$, an identity $\mathtt{ID}_\ell$ and the current value of the public key $\langle X_\ell, Y_\ell\rangle$. We assume that $\mathcal{KE}$ also has access to the H_3 and H_4 lists. $\mathcal{KE}$ operates as follows:

1. Find all triples $\langle \sigma_j, M_j, H_{3,j} \rangle$ on the H_3 list such that

$$\langle U, V \rangle = \texttt{BasicCL-PKE-Encrypt}_{\texttt{ID}_\ell, \langle X_\ell, Y_\ell \rangle}(\sigma_j; H_{3,j}).$$

   Here, $\texttt{BasicCL-PKE-Encrypt}_{\texttt{ID}_A, \langle X_A, Y_A \rangle}(M; r)$ denotes the BasicCL-PKE encryption of message M for $\texttt{ID}_A$ using public key $\langle X_A, Y_A \rangle$ and random value r. Collect all these triples in a list S_1. If S_1 is empty, output $\perp$ and halt.

2. For each triple $\langle \sigma_j, M_j, H_{3,j} \rangle$ in S_1, find all pairs $\langle \sigma'_i, H_{4,i} \rangle$ in the H_4 list with $\sigma_j = \sigma'_i$. For each such match, place $\langle \sigma_j, M_j, H_{3,j}, H_{4,i} \rangle$ on a list S_2. If S_2 is empty, then output $\perp$ and halt.

3. Check in S_2 for an entry such that $W = M_j \oplus H_{4,i}$. If such an entry exists, then output M_j as the decryption of $\langle U, V, W \rangle$. Otherwise, output $\perp$.

We will show that $\mathcal{KE}$ correctly decrypts with high probability in Lemma 9.

Challenge Phase: At some point, $\mathcal{A}_I$ should decide to end Phase 1 and pick $\texttt{ID}_{\text{ch}}$ and two messages m_0, m_1 on which it wishes to be challenged. We can assume that $\texttt{ID}_{\text{ch}}$ has already been queried of H_1 but that $\mathcal{A}_I$ has not extracted the private key for this identity. Algorithm $\mathcal{B}$ responds as follows. If $\texttt{ID}_{\text{ch}} \neq \texttt{ID}_I$ then $\mathcal{B}$ aborts. Otherwise $\texttt{ID}_{\text{ch}} = \texttt{ID}_I$ and $\mathcal{B}$ gives $\mathcal{C}$ the pair m_0, m_1 as the messages on which it wishes to be challenged. $\mathcal{C}$ responds with the challenge ciphertext $C' = \langle U', V', W' \rangle$, such that C' is the HybridPub encryption of m_b under K_{pub} for a random $b \in \{0, 1\}$. Then $\mathcal{B}$ sets $C^* = \langle b_I^{-1} U', V', W' \rangle$ and delivers C^* to $\mathcal{A}_I$. It is easy to see that C^* is the FullCL-PKE encryption of m_b for identity $\texttt{ID}_I$ under public key $\langle X_I, Y_I \rangle$. We let $\langle X_{\text{ch}}, Y_{\text{ch}} \rangle$ denote the particular value of the public key for identity $\texttt{ID}_{\text{ch}}$ during the challenge phase ($\mathcal{A}_I$ may change this value in Phase 2 of its attack).

Phase 2: $\mathcal{B}$ continues to respond to $\mathcal{A}_I$'s requests in the same way as it did in Phase 1. However the usual restrictions on $\mathcal{A}_I$'s behaviour apply in this phase.

Guess: Eventually, $\mathcal{A}_I$ should make a guess b' for b. Then $\mathcal{B}$ outputs b' as its guess for b. If $\mathcal{A}_I$ has used more than time t, or attempts to make more than q_i queries to random oracle H_i or more than q_d decryption queries, then $\mathcal{B}$ should abort $\mathcal{A}_I$ and output a random guess for bit b (in this case algorithm $\mathcal{KE}$ has failed to perform correctly at some point).

Analysis: Now we analyze the behavior of $\mathcal{B}$ and $\mathcal{A}_I$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation and if all of $\mathcal{B}$'s uses of the algorithm $\mathcal{KE}$ result in correct decryptions, then algorithm $\mathcal{A}_I$'s view is identical to its view in the real attack. Moreover, if this is the case, then $2(\Pr[b = b'] - \frac{1}{2}) \geq \epsilon$. This is not hard to see: $\mathcal{B}$'s responses to all hash queries are uniformly and independently distributed as in the real attack. All responses to $\mathcal{A}_I$'s requests are valid, provided of course that $\mathcal{B}$ does not abort and that $\mathcal{KE}$ performs correctly. Furthermore, the challenge ciphertext C^* is a valid FullCL-PKE encryption of m_b under the current public key for identity $\texttt{ID}_{\text{ch}}$, where $b \in \{0, 1\}$ is random. Thus, by definition of algorithm $\mathcal{A}_I$ we have that $2(\Pr[b = b'] - \frac{1}{2}) \geq \epsilon$.

So we must examine the probability that $\mathcal{B}$ does not abort during the simulation given that the algorithm $\mathcal{KE}$ performs correctly. Examining the simulation, we see that $\mathcal{B}$ can abort for one of four reasons:

0. Because $c = 0$ and the event $\mathcal{F}_0$ occurred during the simulation.

1. Because $c = 1$ and event $\mathcal{F}_1$ occurred during the simulation.

2. Because $\mathcal{A}_I$ made a private key extraction on $\texttt{ID}_I$ at some point.

3. Or because $\mathcal{A}_I$ chose $\texttt{ID}_{\text{ch}} \neq \texttt{ID}_I$.

We name the event $(c = i) \wedge \mathcal{F}_i$ as $\mathcal{H}_i$ for $i = 0, 1$. We also name the last two events here as $\mathcal{F}_2$ and $\mathcal{F}_3$. Of course, $\mathcal{F}_3$ is the same as event $\neg\mathcal{H}$. Now $\mathcal{A}_I$ makes q_1 queries of H_1 and chooses $\texttt{ID}_{\text{ch}}$ from amongst the responses $\texttt{ID}_i$, while $\mathcal{B}$'s choice of I is made uniformly at random from the set of q_1 indices i. So the probability that $\texttt{ID}_{\text{ch}} = \texttt{ID}_I$ is equal to $1/q_1$. Hence $\Pr[\mathcal{H}] = 1/q_1$. Notice too that the event $\neg\mathcal{F}_3$ implies the event $\neg\mathcal{F}_2$ (if $\mathcal{A}_I$ chooses $\texttt{ID}_{\text{ch}} = \texttt{ID}_I$, then no private key extraction on $\texttt{ID}_I$ is allowed). Gathering this information together, we have:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ does not abort}] &= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 \wedge \neg\mathcal{F}_2 \wedge \neg\mathcal{F}_3] \\
&= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\
&= \tfrac{1}{q_1} \cdot \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}].
\end{aligned}$$

Notice now that the events $\mathcal{H}_0$ and $\mathcal{H}_1$ are mutually exclusive (because one involves $c = 0$ and the other $c = 1$). Therefore we have

$$\Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] = 1 - \Pr[\mathcal{H}_0 | \mathcal{H}] - \Pr[\mathcal{H}_1 | \mathcal{H}].$$

Moreover,

$$\begin{aligned}
\Pr[\mathcal{H}_i | \mathcal{H}] &= \Pr[(c = i) \wedge \mathcal{F}_i | \mathcal{H}] \\
&= \Pr[\mathcal{F}_i | (\mathcal{H} \wedge (c = i))] \cdot \Pr[c = i] \\
&= \tfrac{1}{2} \Pr[\mathcal{F}_i | \mathcal{H}]
\end{aligned}$$

where the last equality follows because the event $\mathcal{F}_i | \mathcal{H}$ is independent of the event $c = i$. So we have

$$\Pr[\mathcal{B} \text{ does not abort}] = \frac{1}{q_1} \left(1 - \frac{1}{2} \Pr[\mathcal{F}_0 | \mathcal{H}] - \frac{1}{2} \Pr[\mathcal{F}_1 | \mathcal{H}] \right).$$

Finally, we have that $\Pr[\mathcal{F}_0 \wedge \mathcal{F}_1 | \mathcal{H}] = 0$ because of the rules on adversary behaviour described in Section 3 (an adversary cannot both extract the partial private key and change the public key of the challenge identity). This implies that $\Pr[\mathcal{F}_0 | \mathcal{H}] + \Pr[\mathcal{F}_1 | \mathcal{H}] \leq 1$. Hence we see that

$$\Pr[\mathcal{B} \text{ does not abort}] \geq \frac{1}{2q_1}.$$

Now we examine the probability that algorithm $\mathcal{KE}$ correctly handles all of $\mathcal{A}_I$'s q_d decryption queries. We will show in Lemma 9 below that the probability that $\mathcal{KE}$ correctly replies to individual decryption queries is at least λ, where λ is bounded as in the statement of this lemma.

It is now easy to see that $\mathcal{B}$'s advantage is at least $\frac{\epsilon}{2q_1}\lambda^{q_d}$. It follows that either $\mathcal{B}$'s advantage as a Type I adversary against `HybridPub` or $\mathcal{B}$'s advantage as a Type II adversary against `HybridPub` is at least $\frac{\epsilon}{4q_1}\lambda^{q_d}$. The running time of $\mathcal{B}$ is $\text{time}(\mathcal{A}_I) + q_d \cdot \text{time}(\mathcal{KE}) = t + O((q_3 + q_4)q_d t')$ where t' is the running time of the `BasicCL-PKE` encryption algorithm. This completes the proof of the lemma.

Lemma 9 *In the simulation in the proof of Lemma 2, Algorithm $\mathcal{KE}$ correctly replies to individual decryption queries with probability at least λ where*

$$1 - \lambda \leq (q_3 + q_4) \cdot \epsilon_{OWE}(t + O((q_3 + q_4)q_d t', q_2) + \epsilon_{GBDHP}(t + O((q_3 + q_4)q_d t') + 3q^{-1} + 2^{-n+1}.$$

Here t is the running time of adversary $\mathcal{A}_I$, t' is the running time of the `BasicCL-PKE` *encryption algorithm, $\epsilon_{OWE}(T, q')$ denotes the highest advantage of any Type I or Type II OWE adversary against* `BasicPub` *which operates in time T and makes q' hash queries to H_2, and $\epsilon_{GBDHP}(T)$ denotes the highest advantage of any algorithm to solve GBDHP in time T in groups of order q generated by $\mathcal{IG}$.*

Proof of Lemma 9: Our proof is closely modelled on the proof of [16, Lemma 11], but differs in several key respects: we need to build an algorithm which handles multiple public keys, and the algorithm can be asked to decrypt the challenge ciphertext (but under a different identity/public key combination from the challenge identity).

We recall that queries to $\mathcal{KE}$ come in the form of a ciphertext $C = \langle U, V, W \rangle$, an identity $\mathtt{ID}_\ell$ and the current value of the public key $\langle X_\ell, Y_\ell \rangle$ for that identity. We also assume that $\mathcal{KE}$ has access to the H_3 and H_4 lists as they stand at the point where the decryption query is made. We model the fact that $\mathcal{A}_I$ obtains a challenge ciphertext by considering an additional list of ciphertexts $\mathcal{Y}$ in our proof. This list is empty until the challenge phase and thereafter consists of just the challenge ciphertext $C^* = \langle U^*, V^*, W^* \rangle$. The proof of [16, Lemma 11] generalises this to larger sets $\mathcal{Y}$, but this special case is sufficient for our purposes.

We define a sequence of events:

- `Inv` is the event that there exists some $C' = \langle U', V', W' \rangle \in \mathcal{Y}$ and some $\langle \sigma_j, M_j, H_{3,j} \rangle$ on the H_3 list or some $\langle \sigma'_i, H_{4,i} \rangle$ on the H_4 list such that the `BasicCL-PKE` decryption of $\langle U', V' \rangle$ under the private key corresponding to $\mathtt{ID}_{\text{ch}}, \langle X_{\text{ch}}, Y_{\text{ch}} \rangle$ is equal to σ_j or σ'_i. (For us, `Inv` has zero probability until after a non-abortive challenge phase in $\mathcal{A}_I$'s attack because $\mathcal{Y}$ is empty up to this point.)
- $\mathcal{L}_1$ is the event that S_1 is non-empty.
- $\mathcal{L}_2$ is the event that S_2 is non-empty.
- `Find` is the event that there exists an entry $\langle \sigma_j, M_j, H_{3,j}, H_{4,i} \rangle$ in S_2 such that $W = M_j \oplus H_{4,i}$.
- `Fail` is the event that the output of algorithm $\mathcal{KE}$ is not the decryption of C under the private key corresponding to identity $\mathtt{ID}_\ell$ and public key $\langle X_\ell, Y_\ell \rangle$.

We want to bound the probability of the event `Fail`. To do so, we follow the proof of [16, Lemma 11] and define combined events:

$$\begin{aligned}
\mathtt{1} &= \mathtt{Inv}, \\
\mathtt{00} &= \neg\mathtt{Inv} \wedge \neg\mathcal{L}_1, \\
\mathtt{010} &= \neg\mathtt{Inv} \wedge \mathcal{L}_1 \wedge \neg\mathcal{L}_2, \\
\mathtt{0110} &= \neg\mathtt{Inv} \wedge \mathcal{L}_1 \wedge \mathcal{L}_2 \wedge \neg\mathtt{Find}, \\
\mathtt{0111} &= \neg\mathtt{Inv} \wedge \mathcal{L}_1 \wedge \mathcal{L}_2 \wedge \mathtt{Find}.
\end{aligned}$$

Now we have:

$$\begin{aligned}\Pr[\texttt{Fail}] &= \Pr[\texttt{Fail}|\texttt{1}]\cdot\Pr[\texttt{1}] + \Pr[\texttt{Fail}|\texttt{00}]\cdot\Pr[\texttt{00}] + \Pr[\texttt{Fail}|\texttt{010}]\cdot\Pr[\texttt{010}] \\ &\quad + \Pr[\texttt{Fail}|\texttt{0110}]\cdot\Pr[\texttt{0110}] + \Pr[\texttt{Fail}|\texttt{0111}]\cdot\Pr[\texttt{0111}] \\ &\le \Pr[\texttt{1}] + \Pr[\texttt{Fail}|\texttt{00}] + \Pr[\texttt{Fail}|\texttt{010}] + \Pr[\texttt{Fail}|\texttt{0110}] + \Pr[\texttt{Fail}|\texttt{0111}].\end{aligned}$$

We proceed to bound each of the terms in the above inequality.

Claim: $\Pr[\texttt{1}] \le (q_3+q_4)\cdot\epsilon_{\mathrm{OWE}}(\mathrm{time}(\mathcal{B}), q_2)$. Here $\epsilon_{\mathrm{OWE}}(T, q')$ denotes the highest advantage of any Type I or Type II OWE adversary against `BasicPub` which operates in time T and makes q' hash queries to H_2, while $\mathrm{time}(\mathcal{B})$ denotes the running time of adversary $\mathcal{B}$ in the proof of Lemma 2.

We sketch how to construct an OWE adversary $\mathcal{B}'$ against `BasicPub` by adapting adversary $\mathcal{B}$ in the proof of Lemma 2. Our adversary $\mathcal{B}'$ will have a chance of being successful provided that the event `Inv` occurs in the course of $\mathcal{A}_I$'s attack. When $c=0$, $\mathcal{B}'$ will be of Type I, and when $c=1$, of Type II. The running time of the adversary will be the same as that of $\mathcal{B}$. The existence of this adversary will be used to bound the probability of the event `Inv`.

In fact $\mathcal{B}'$ is almost identical to $\mathcal{B}$. The only differences are that $\mathcal{B}'$ is given by its challenger $\mathcal{C}'$ a `BasicPub` public key $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2\rangle$ (and the value s when $c=1$), that $\mathcal{B}'$ now answers $\mathcal{A}_I$'s H_3 and H_4 queries for itself (keeping lists of all queries made by $\mathcal{A}_I$ and its own replies), and that $\mathcal{B}'$ responds to $\mathcal{A}_I$'s request for a challenge ciphertext with $C^* = \langle b_I^{-1}U', V', W^*\rangle$ where $\langle U', V'\rangle$ is a `BasicPub` challenge ciphertext given to $\mathcal{B}'$ by $\mathcal{C}'$ and W^* is chosen uniformly at random from $\{0,1\}^n$. Notice too that if $\mathcal{A}_I$ changes the key of $\texttt{ID}_I$, then $\mathcal{B}'$ relays the appropriate changes to $\mathcal{C}'$, and that $\mathcal{B}'$ uses the algorithm $\mathcal{KE}$ to handle $\mathcal{A}_I$'s decryption queries (so the responses may be incorrect).

Eventually $\mathcal{A}_I$ outputs a bit b'. If necessary (when $\mathcal{A}_I$ runs for too long or makes too many hash queries), $\mathcal{B}'$ stops $\mathcal{A}_I$. Note that $\mathcal{B}'$ may also be forced to stop because it cannot respond to a particular query from $\mathcal{A}_I$. After stopping for whatever reason, $\mathcal{B}'$ chooses a random value σ from amongst the σ_j on the H_3 list and the σ'_i on the H_4 list and outputs this random choice.

It can be argued that, up to the point where `Inv` occurs in $\mathcal{B}'$'s simulation, the two simulations $\mathcal{B}$ and $\mathcal{B}'$ are indistinguishable to $\mathcal{A}_I$. So the probability that `Inv` occurs in $\mathcal{B}'$'s simulation is exactly the same that it does in $\mathcal{B}$'s. Because of the relationship between the `BasicPub` and `FullCL-PKE` public keys, it can also be seen that if event `Inv` occurs, then $\mathcal{B}'$ has probability $1/(q_3+q_4)$ of outputting the correct `BasicPub` decryption of $\langle U', V'\rangle$. Here we are using the fact that if `Inv` occurs, then $\mathcal{Y}$ is non-empty, so that we must have $\texttt{ID}_{\mathrm{ch}} = \texttt{ID}_I$. So $\mathcal{B}'$'s overall success probability is at least $\Pr[\texttt{Inv}]/(q_3+q_4)$. But this is not greater than the highest success probability of any OWE adversary of Type I or II against `BasicPub` that operates in the same time as $\mathcal{B}'$ and that makes q_2 hash queries. Since the running time of $\mathcal{B}'$ is the same as that of $\mathcal{B}$, the claim follows.

Claim: $\Pr[\texttt{Fail}|\texttt{00}] \le 2/q + 2^{-n} + \epsilon_{\mathrm{GBDHP}}(\mathrm{time}(\mathcal{B}))$. Here $\epsilon_{\mathrm{GBDHP}}(T)$ denotes the highest advantage of any algorithm to solve GBDHP in time T in groups generated by $\mathcal{IG}$.

We analyse the event $\texttt{Fail}|\texttt{00}$ as follows. Here $\mathcal{KE}$ outputs $\perp$ because S_1 is empty, but this is an incorrect decryption. So in fact there exists a message M such that $C = \langle U, V, W\rangle$ encrypts M under $\texttt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$. It is easy to see that, because $\langle U, V\rangle$ is a valid `BasicCL-PKE` ciphertext for $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$, there exist unique $\sigma \in \{0,1\}^n$ and $r \in \mathbb{Z}_q^*$ such that:

$$\langle U, V\rangle = \mathtt{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle}(\sigma; r).$$

Since S_1 is empty, we deduce that H_3 has not been queried on an input containing σ.

We consider two cases: either a valid $C \neq C^*$ has been produced by $\mathcal{A}_I$ from a message M using coins $r = H_3(\sigma, M)$ without σ having been queried of H_3, or in fact $C = C^*$ and this query occurs after the challenge phase. In the former case, it is easy to see that C will be a valid ciphertext with probability at most $1/q$, because a valid ciphertext $C = \langle U, V, W\rangle$ will have $U = rP$ where $r \in \mathbb{Z}_q^*$ is the output of random oracle H_3 on a query not made by $\mathcal{A}_I$.

We consider the latter case, where $C = C^*$ is a valid ciphertext, further. Now $\mathcal{KE}$ can only ever be queried on this ciphertext for a combination of identity and public key $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$ not equal to $\mathtt{ID}_{\mathrm{ch}}, \langle X_{\mathrm{ch}}, Y_{\mathrm{ch}}\rangle$ because of the rules on adversary behaviour. We also know that $\mathtt{ID}_{\mathrm{ch}} = \mathtt{ID}_I$ (because to receive this query, $\mathcal{B}$ must not have aborted at the challenge phase). Suppose then that

$$C^* = \langle r^*P, \sigma^* \oplus H_2(e(b_IQ, Y_{\mathrm{ch}})^{r^*}), m_b \oplus H_4(\sigma^*)\rangle$$

where $r^* = H_3(\sigma^*, m_b)$ and, as usual, $\langle X_{\mathrm{ch}}, Y_{\mathrm{ch}}\rangle$ denotes the value of $\mathtt{ID}_{\mathrm{ch}}$'s public key at the time when the challenge ciphertext was computed. The values σ^*, $H_4(\sigma^*)$ and r^* are unknown to $\mathcal{B}$ and $\mathcal{KE}$ (since $\mathcal{B}$'s challenger produces C^*). Since $C = C^*$, we have $rP = U = U^* = r^*P$ and so $r = r^*$. The probability that $\sigma \neq \sigma^*$ is $1/q$. For suppose that $\sigma \neq \sigma^*$. Then we have $H_3(\sigma, M) = r = r^* = H_3(\sigma^*, m_b)$, giving equal outputs for random oracle H_3 from distinct inputs. The probability of this event is $1/q$. So with probability $1 - 1/q$, we have $\sigma = \sigma^*$. But then we must have

$$H_2(e(Q_\ell, Y_\ell)^r) = H_2(e(b_IQ, Y_{\mathrm{ch}})^{r^*}).$$

If these inputs to H_2 are unequal then we have a collision of the random oracle H_2, an event of probability 2^{-n}. So with probability $1 - 2^{-n}$, we have

$$e(Q_\ell, Y_\ell)^r = e(b_IQ, Y_{\mathrm{ch}})^{r^*}$$

and hence, since $r = r^*$,

$$e(Q_\ell, Y_\ell) = e(b_IQ, Y_{\mathrm{ch}}).$$

Now suppose that $\ell = I$. Then we would have $e(b_IQ, Y_I) = e(b_IQ, Y_{\mathrm{ch}})$ from which we can deduce that $Y_I = Y_{\mathrm{ch}}$. Now because $\langle X_I, Y_I\rangle$ and $\langle X_{\mathrm{ch}}, Y_{\mathrm{ch}}\rangle$ are valid public keys, we know that $Y_I = sX_I$ and $Y_{\mathrm{ch}} = sX_{\mathrm{ch}}$. It follows that $X_I = X_{\mathrm{ch}}$. Thus we have that the combination $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$ is equal to $\mathtt{ID}_{\mathrm{ch}}, \langle X_{\mathrm{ch}}, Y_{\mathrm{ch}}\rangle$. From this contradiction, we deduce that $\ell \neq I$.

Now we can write $Q_\ell = b_\ell P$, $Y_{\mathrm{ch}} = sX_{\mathrm{ch}}$ and:

$$e(P, Y_\ell) = e(Q, sX_{\mathrm{ch}})^{b_\ell^{-1}b_I}, \quad \ell \neq I.$$

Given this analysis, we sketch how to construct an algorithm $\mathcal{B}'$ that, with high probability, solves random instances of GBDHP in groups generated by $\mathcal{IG}$, so long as the event

$\mathtt{Fail}|\mathtt{00}$ occurs. Our algorithm $\mathcal{B}'$ is almost identical to $\mathcal{B}$. Suppose $\mathcal{B}'$ is tasked with solving the BDHP or GBDHP on input $\langle P, fP, gP, hP\rangle$ in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$.

Recall that $\mathcal{B}$ selects a random bit c and then requests a `HybridPub` public key from $\mathcal{C}$. Instead, $\mathcal{B}'$ creates its own public key. When $c = 0$, $\mathcal{B}'$ chooses x at random from $\mathbb{Z}_q^*$ and sets $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0 = fP, X = xP, Y = xP_0, Q = gP, H_2, H_3, H_4\rangle$. When $c = 1$, $\mathcal{B}'$ chooses s at random from $\mathbb{Z}_q^*$ and sets $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0 = sP, X = fP, Y = sfP, Q = gP, H_2, H_3, H_4\rangle$. The only other difference to $\mathcal{B}$'s simulation is that $\mathcal{B}'$ needs to carry out those tasks which $\mathcal{B}$ formerly passed on to $\mathcal{C}$. These are: responding to H_3 and H_4 queries; tracking any changes to K_{pub}; and creating a `HybridPub` challenge ciphertext. These are all dealt with in the obvious way.

It is clear that $\mathcal{B}'$'s simulation is indistinguishable from that of $\mathcal{B}$, so the probability that the event $\mathtt{Fail}|\mathtt{00}$ occurs in $\mathcal{B}'$'s simulation is the same as in $\mathcal{B}$'s. But when this event does occur and $C = C^*$, then our analysis above shows that the equality $e(P, Y_\ell) = e(Q, sX_{\text{ch}})^{b_\ell^{-1} b_I}$ for $\ell \neq I$ holds with probability $(1 - q^{-1})(1 - 2^{-n})$. Here $s = f$ when $c = 0$ and s is the value chosen at random by $\mathcal{B}'$ when $c = 1$. In the case $c = 0$, we see that $\langle X_{\text{ch}}, e(hP, Y_\ell)^{b_\ell b_I^{-1}}\rangle$ is a solution to the GBDHP for input $\langle P, fP, gP, hP\rangle$. When $c = 1$, we know that $\mathcal{A}_I$ has not changed the values $X_{\text{ch}} = X, Y_{\text{ch}} = Y$ at any point in its attack (otherwise $\mathcal{B}'$ would have aborted). Then we have $e(P, Y_\ell) = e(Q, X)^{b_\ell^{-1} b_I s}$ and we see that $\langle P, e(hP, Y_\ell)^{b_\ell b_I^{-1} s^{-1}}\rangle$ is a solution to the GBDHP for input $\langle P, fP, gP, hP\rangle$. (Note that here we actually have a solution to the BDHP.) In either case, $\mathcal{B}'$ can compute the appropriate solution simply by waiting for $\mathcal{A}_I$ to make a decryption query on $C = C^*$. If $\mathcal{A}_I$ does not make such a query in the course of its attack, then $\mathcal{B}'$ outputs a random guess for the solution to the GBDHP. The result is an algorithm that solves GBDHP in groups generated by $\mathcal{IG}$, runs in time bounded by time($\mathcal{B}$) and is almost always successful when $C = C^*$ and the event $\mathtt{Fail}|\mathtt{00}$ occurs. Any such algorithm has success probability at most $\epsilon_{\text{GBDHP}}(\text{time}(\mathcal{B}))$.

A straightforward probability analysis of the events in the above discussion now shows that:

$$\Pr[\mathtt{Fail}|\mathtt{00}] \leq 2/q + 2^{-n} + \epsilon_{\text{GBDHP}}(\text{time}(\mathcal{B})).$$

The claim follows.

Claim: $\Pr[\mathtt{Fail}|\mathtt{010}] = 2^{-n}$**.** In this situation, $\mathcal{KE}$ outputs $\perp$ because S_2 is empty, but this is an incorrect decryption. So in fact there exists a message M such that $C = \langle U, V, W\rangle$ encrypts M under $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$. Now it is easy to see that, because $\langle U, V\rangle$ is a valid `BasicCL-PKE` ciphertext for $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$, there exist unique $\sigma \in \{0,1\}^n$ and $r \in \mathbb{Z}_q^*$ such that:

$$\langle U, V\rangle = \mathtt{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle}(\sigma; r).$$

But S_1 is non-empty, so we also have:

$$\langle U, V\rangle = \mathtt{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle}(\sigma_j; H_{3,j})$$

for some j. This implies that $\sigma = \sigma_j$ and $r = H_{3,j}$. Since S_2 is empty, we can deduce that H_4 has not been queried at σ. Yet we must have $W = M \oplus H_4(\sigma)$ if C is a proper encryption of M. The probability of this event occurring is exactly 2^{-n} and this bounds the probability that $\mathcal{KE}$ incorrectly outputs $\perp$.

Claim: $\Pr[\mathtt{Fail}|\mathtt{0110}] = 1/q$**.** Here $\mathcal{KE}$ outputs $\perp$ because a failure occurs at step 3 , but this is an incorrect decryption. Arguing as in the previous claim, we deduce that there exists a message M such that $C = \langle U, V, W\rangle$ encrypts M under $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$, using unique $\sigma \in \{0,1\}^n$ and $r \in \mathbb{Z}_q^*$. Moreover, there exists a j with $\sigma = \sigma_j$ and $r = H_{3,j}$. Now S_2 is non-empty, so there exists an entry $\langle \sigma_j, M_j, H_{3,j}, H_{4,i}\rangle$ on the S_2 list with $\sigma_i' = \sigma_j = \sigma$.

Now suppose that $\langle \sigma, M\rangle$ has been queried of H_3. Then we would also have an entry $\langle \sigma, M, H_{3,j}, H_{4,i}\rangle$ on the S_2 list. But since C is the encryption of M, we would also have $W = M \oplus H_{4,i}$. Then $\mathcal{KE}$ would output M instead of $\perp$. This contradiction shows that $\langle \sigma, M\rangle$ has not been queried of H_3. Yet we must have $H_3(\sigma, M) = r = H_{3,j}$ if C is a proper encryption of M. The probability of this event occurring is exactly $1/q$ and this bounds the probability that $\mathcal{KE}$ incorrectly outputs $\perp$. Notice that this argument can be used to correct a small flaw in the corresponding part of the proof of [16, Lemma 11].

Claim: $\Pr[\mathtt{Fail}|\mathtt{0111}] = 0$**.** Here, $\mathcal{KE}$ outputs a message M_j whose encryption under the combination $\mathtt{ID}_\ell, \langle X_\ell, Y_\ell\rangle$ yields the ciphertext C with random oracles H_3 and H_4 as defined in $\mathcal{B}$'s simulation. Therefore the decryption of C is M_j, and $\mathcal{KE}$ never fails in this situation. The claim follows.

Gathering together each of these claims, we finally obtain

$$\Pr[\mathtt{Fail}] \le (q_3 + q_4)\cdot \epsilon_{\text{OWE}}(\text{time}(\mathcal{B}), q_2) + \epsilon_{\text{GBDHP}}(\text{time}(\mathcal{B})) + 3q^{-1} + 2^{-n+1}.$$

The running time of $\mathcal{B}$ is $\text{time}(\mathcal{A}_I) + q_d \cdot \text{time}(\mathcal{KE}) = t + O((q_3+q_4)q_d t')$, where t' is the running time of the `BasicCL-PKE` encryption algorithm. This completes the proof of the lemma.

Proof of Lemma 3: Let $\mathcal{A}_I$ be a Type I IND-CPA adversary against `HybridPub` which has advantage ϵ, runs in time t and makes q_4 queries to H_4. We construct a Type I OWE adversary $\mathcal{B}$ against `BasicPub`. Let $\mathcal{C}$ denote a challenger against such a $\mathcal{B}$. $\mathcal{C}$ begins by supplying $\mathcal{B}$ with a public key $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2\rangle$. Algorithm $\mathcal{B}$ creates from this a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2, H_3, H_4\rangle$ for `HybridPub` and delivers it to $\mathcal{A}_I$. Here, H_3 and H_4 will be random oracles under the control of $\mathcal{B}$.

Now $\mathcal{A}_I$ begins Phase 1 of its attack. During this phase, $\mathcal{B}$ records any changes that $\mathcal{A}_I$ makes to the public key components $\langle X, Y\rangle$. $\mathcal{B}$ handles $\mathcal{A}_I$'s H_3 and H_4 queries using lists as follows. To respond to an H_3 query on input $\langle \sigma, M\rangle \in \{0,1\}^n \times \{0,1\}^n$, $\mathcal{B}$ first checks if $\sigma = \sigma_j$ and $M = M_j$ for some index j corresponding to an entry $\langle \sigma_j, M_j, H_{3,j}\rangle$ already on the H_3 list. If it is, then $\mathcal{B}$ responds with $H_{3,j}$. Otherwise, $\mathcal{B}$ chooses H uniformly at random from $\mathbb{Z}_q^*$ and places $\langle \sigma, M, H\rangle$ on the H_3 list. Likewise, to respond to an H_4 query on input $\sigma \in \{0,1\}^n$, $\mathcal{B}$ first checks if $\sigma = \sigma_i'$ for some index i corresponding to an entry $\langle \sigma_i', H_{4,i}\rangle$ already on the H_4 list. If it is, then $\mathcal{B}$ responds with $H_{4,i}$. Otherwise, $\mathcal{B}$ chooses H uniformly at random from $\{0,1\}^n$ and places $\langle \sigma, H\rangle$ on the H_4 list.

Eventually $\mathcal{A}_I$ outputs a pair of messages m_0, m_1 on which it wishes to be challenged. If $\mathcal{A}_I$ has changed the public key components $\langle X, Y\rangle$, then $\mathcal{B}$ passes these changes to $\mathcal{C}$ (recall that $\mathcal{B}$ is of Type I, so is entitled to change these components before receiving its challenge ciphertext). Then $\mathcal{B}$ requests of $\mathcal{C}$ a `BasicPub` ciphertext $C' = \langle U', V'\rangle$. This challenge ciphertext will be the encryption of some message σ^* using the current value of the public key, and it is $\mathcal{B}$'s task to recover σ^*.

Adversary $\mathcal{B}$ now chooses a random $W^* \in \{0,1\}^n$, sets $C^* = \langle U', V', W^*\rangle$ and delivers C^* to $\mathcal{A}_I$ as the `HybridPub` challenge ciphertext.

$\mathcal{A}_I$ now executes Phase 2 of its attack. $\mathcal{B}$ responds to $\mathcal{A}_I$'s H_3 and H_4 queries as before and continues to record changes to public key components made by $\mathcal{A}_I$. After at most q_3

queries to H_3, q_4 queries to H_4 or after at most time t, $\mathcal{A}_I$ should output a bit b'. If it does not, then $\mathcal{B}$ just aborts $\mathcal{A}_I$.

Let q_3' denote the number of H_3 queries and q_4' the number of H_4 queries made in $\mathcal{A}_I$'s attack. Notice that $q_3' \leq q_3$ and $q_4' \leq q_4$. Now $\mathcal{B}$ picks an element uniformly at random from the set $\{\sigma_j : 1 \leq j \leq q_3'\} \cup \{\sigma_i' : 1 \leq i \leq q_4'\}$ and outputs that element as its guess for σ^*.

This completes the description of the algorithm $\mathcal{B}$. Next we evaluate $\mathcal{B}$'s advantage.

Let $\mathcal{H}$ denote the event that $\mathcal{A}_I$ queries either H_3 or H_4 on an input including σ^* and let $\mathcal{R}$ denote the event that $\mathcal{A}_I$ is successful in a real attack, so $\epsilon = 2(\Pr[\mathcal{R}] - 1/2)$.

Now if $\mathcal{H}$ does not occur in a real attack (i.e. an attack against a proper `HybridPub` challenger), then $\mathcal{A}_I$'s probability of success is exactly 1/2. For in a real attack, we have $W^* = m_b \oplus H_4(\sigma^*)$ wherein the uniformly distributed output of random oracle H_4 on input σ^* is used to encrypt m_b. But if $\mathcal{H}$ does not occur, then $\mathcal{A}_I$ has not queried H_4 at this input σ^* and so, in $\mathcal{A}_I$'s view, either of messages m_0, m_1 is equally likely to have been encrypted.

Hence:

$$\begin{aligned} 1/2 + \epsilon/2 &= \Pr[\mathcal{R}] \\ &= \Pr[\mathcal{R}|\mathcal{H}]\Pr[\mathcal{H}] + \Pr[\mathcal{R}|\neg\mathcal{H}]\Pr[\neg\mathcal{H}] \\ &\leq \Pr[\mathcal{H}] + \tfrac{1}{2}. \end{aligned}$$

Hence $\Pr[\mathcal{H}] \geq \epsilon/2$. Notice that $\mathcal{B}$'s simulation is indistinguishable from $\mathcal{A}_I$'s view in a real attack, up to at least the point where $\mathcal{H}$ occurs. Therefore the probability that $\mathcal{H}$ occurs in $\mathcal{B}$'s simulation is exactly the same as in a real attack and so is at least $\epsilon/2$. But if $\mathcal{H}$ occurs, then $\mathcal{B}$'s probability of success is at least $1/(q_3' + q_4') \geq 1/(q_3 + q_4)$, because $\mathcal{B}$ outputs a random choice from the list of H_3 and H_4 queries and at least one of these is equal to σ^* when $\mathcal{H}$ occurs. Hence $\mathcal{B}$'s probability of success is at least $\epsilon/2(q_3 + q_4)$. Notice that the running time of $\mathcal{B}$ is of the same order as that of $\mathcal{A}_I$. This completes the proof.

Proof of Lemma 4: The proof of this lemma is almost identical to that of Lemma 3 and is omitted.

Proof of Lemma 5: Let $\mathcal{A}_I$ be a Type I OWE adversary against `BasicPub` who makes at most q_2 queries to random oracle H_2 and who has advantage ϵ. We show how to construct an algorithm $\mathcal{B}$ which interacts with $\mathcal{A}_I$ to solve the GBDHP.

Suppose $\mathcal{B}$ has as inputs $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and $\langle P, aP, bP, cP\rangle$ (where $a, b, c \in \mathbb{Z}_q^*$ are unknown to $\mathcal{B}$). Let $D = e(P,P)^{abc} \in \mathbb{G}_2$ denote the solution to the BDHP on these inputs. Algorithm $\mathcal{B}$ creates a public key $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2\rangle$ for $\mathcal{A}_I$ by taking $x \in \mathbb{Z}_q^*$ at random, setting $P_0 = aP$, $X = xP$, $Y = xP_0$ and $Q = bP$. $\mathcal{B}$ then gives this key to $\mathcal{A}_I$. $\mathcal{A}_I$ may choose to reset the pair $\langle X, Y\rangle$ to another pair $\langle X', Y'\rangle$. $\mathcal{B}$ then checks if the public key satisfies $e(X', P_0) = e(Y', P)$. If not, then $\mathcal{B}$ outputs $\perp$ and stops the simulation. In this event, $\mathcal{A}_I$ has failed. Otherwise, $\mathcal{B}$ sets $U = cP$, chooses V randomly from $\{0,1\}^n$, and gives $\mathcal{A}_I$ the challenge ciphertext $C = \langle U, V\rangle$.

Notice that if $\perp$ is not output, then the public key must have components $\langle X' = x'P, Y' = x'aP\rangle$ for some x', and the corresponding (unknown) private key is $x'abP$. Then the (unknown) decryption of C is $M = V \oplus H_2(e(P, x'P)^{abc}) = V \oplus H_2(D^{x'})$. Hence a solution $\langle X', D^{x'}\rangle$ to the GBDHP can be derived from examining $\mathcal{A}_I$'s choice of public key and H_2 queries.

To simulate H_2 queries by $\mathcal{A}_I$, $\mathcal{B}$ maintains a list of pairs $\langle Z_j, H_{2,j}\rangle$. To respond to an

H_2 query Z, $\mathcal{B}$ first checks if $Z = Z_j$ for some Z_j already on the list. If it is, then $\mathcal{B}$ responds with $H_{2,j}$. Otherwise, $\mathcal{B}$ chooses H uniformly at random from $\{0,1\}^n$ and places $\langle Z, H \rangle$ on the H_2 list.

Eventually, $\mathcal{A}_I$ will output its guess M' for the decryption of C. Now $\mathcal{B}$ chooses a random pair $\langle Z_j, H_{2,j} \rangle$ from the H_2 list and outputs the pair $\langle X', Z_j \rangle$ as the solution to the GBDHP. (If the list is empty, $\mathcal{B}$ just outputs a random pair from $\mathbb{G}_1 \times \mathbb{G}_2$.)

It is easy to see that $\mathcal{A}_I$'s view in $\mathcal{B}$'s simulation is the same as in a real attack. So $\mathcal{A}_I$'s advantage in this simulation will be ϵ. We let $\mathcal{H}$ be the event that $D^{x'}$ is queried of H_2 during $\mathcal{B}$'s simulation and $\mathcal{R}$ denote the event that $\mathcal{A}_I$'s choice of public key components $\langle X', Y' \rangle$ passes the test (so $\perp$ is not output). We let δ denote the probability that event $\mathcal{H}$ occurs. Now

$$\begin{aligned} \epsilon &= \Pr[(M' = M) \wedge \mathcal{R}] \\ &= \Pr[(M' = M) \wedge \mathcal{R} | \mathcal{H}] \Pr[\mathcal{H}] + \Pr[(M' = M) \wedge \mathcal{R} | \neg\mathcal{H}] \Pr[\neg\mathcal{H}] \\ &\leq \delta + \tfrac{1}{2^n}(1 - \delta) \end{aligned}$$

where we have used the fact that if $\mathcal{H}$ does not occur, then H_2 has not been queried on input $D^{x'}$ so that $\mathcal{A}_I$'s view must be independent of the value of M.

Rearranging, we see that $\delta \geq \epsilon - \frac{1}{2^n}$. Since $\mathcal{B}$'s output is of the form $\langle X', Z_j \rangle$ with Z_j chosen randomly from the H_2 list, we see that $\mathcal{B}$'s success probability is at least δ / q_2. The lemma follows.

Proof of Lemma 6: Let $\mathcal{A}_{II}$ be a Type II IND-CCA adversary against `FullCL-PKE`. Suppose $\mathcal{A}_{II}$ has advantage ϵ and makes q_1 queries to random oracle H_1. We show how to construct from $\mathcal{A}_{II}$ a Type II IND-CCA adversary $\mathcal{B}$ against `HybridPub`. The construction is similar to the one used in the proof of [6, Lemma 4.6].

Let $\mathcal{C}$ denote the challenger against our IND-CCA adversary $\mathcal{B}$ for `HybridPub`. $\mathcal{C}$ begins by supplying $\mathcal{B}$ with a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2, H_3, H_4 \rangle$ and the value s such that $P_0 = sP$. Adversary $\mathcal{B}$ mounts an IND-CCA attack on the key K_{pub} using help from $\mathcal{A}_{II}$ as follows.

First of all $\mathcal{B}$ chooses an index I with $1 \leq I \leq q_1$. Then $\mathcal{B}$ simulates the algorithm `Setup` of `FullCL-PKE` for $\mathcal{A}_{II}$ by supplying $\mathcal{A}_{II}$ with `params`$= \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4 \rangle$ and the value s as `master-key`. Here H_1 is a random oracle controlled by $\mathcal{B}$.

Adversary $\mathcal{A}_{II}$ may make queries of H_1 at any time. These are handled as follows:

H_1 **queries:** $\mathcal{B}$ maintains a list of tuples $\langle \mathtt{ID}_j, Q_j, b_j, x_j \rangle$ which we call the H_1 list. The list is initially empty, and when $\mathcal{A}_{II}$ queries H_1 on input `ID`, $\mathcal{B}$ responds as follows:

1. If `ID` already appears on the H_1 list in a tuple $\langle \mathtt{ID}_i, Q_i, b_i, x_i \rangle$, then $\mathcal{B}$ responds with $H_1(\mathtt{ID}) = Q_i \in \mathbb{G}_1^*$.

2. If `ID` does not already appear on the list and `ID` is the I-th distinct H_1 query made by $\mathcal{A}_{II}$, then $\mathcal{B}$ picks b_I at random from $\mathbb{Z}_q^*$, outputs $H_1(\mathtt{ID}) = b_I Q$ and adds $\langle \mathtt{ID}, b_I Q, b_I, \perp \rangle$ to the H_1 list.

3. Otherwise, when `ID` does not already appear on the list and `ID` is the i-th distinct H_1 query made by $\mathcal{A}_{II}$ where $i \neq I$, $\mathcal{B}$ picks b_i and x_i at random from $\mathbb{Z}_q^*$, outputs $H_1(\mathtt{ID}) = b_i P$ and adds $\langle \mathtt{ID}, b_i P, b_i, x_i \rangle$ to the H_1 list. Notice that the private key for $\mathtt{ID}_i$ is $x_i s b_i P = x_i s Q_i$, which can be computed by $\mathcal{B}$.

Phase 1: Now $\mathcal{A}_{II}$ launches Phase 1 of its attack, by making a series of requests, each of which is either a private key extraction, a request for a public key for a particular entity, or a decryption query. (Recall that a Type II adversary cannot replace public keys and can make partial private key extraction queries for himself given s.) We assume that $\mathcal{A}_{II}$ always makes the appropriate H_1 query on $\mathtt{ID}$ before making one of these requests for that identity. $\mathcal{B}$ replies to these requests as follows:

Private Key Extraction: If the request is on $\mathtt{ID}_I$ then $\mathcal{B}$ aborts. Otherwise, if the request is on $\mathtt{ID}_i$ with $i \neq I$, then $\mathcal{B}$ outputs $x_i s Q_i$.

Request for Public Key: If the request is on $\mathtt{ID}_I$ then $\mathcal{B}$ returns $\langle X, Y\rangle$. Otherwise, if the request is on $\mathtt{ID}_i$ for some i with $i \neq I$, then $\mathcal{B}$ returns $\langle x_i P, x_i s P\rangle$.

Decryption Queries: If the request is to decrypt $\langle U, V, W\rangle$ under the private key for $\mathtt{ID}_I$, then $\mathcal{B}$ relays the decryption query $\langle b_I U, V, W\rangle$ to $\mathcal{C}$. It is easy to see that the FullCL-PKE decryption of $\langle U, V, W\rangle$ under the (unknown) private key for $\mathtt{ID}_I$ is equal to the HybridPub decryption of $\langle b_I U, V, W\rangle$ under the (unknown) private key corresponding to K_{pub}. Hence $\mathcal{C}$'s response to $\mathcal{B}$'s request can be relayed to $\mathcal{A}_{II}$. On the other hand, if the request is to decrypt $\langle U, V, W\rangle$ under the private key for $\mathtt{ID}_i$ $(i \neq I)$, then $\mathcal{B}$ can perform this decryption himself using the private key $x_i s b_i P$ for $\mathtt{ID}_i$.

Challenge Phase: At some point, $\mathcal{A}_{II}$ decides to end Phase 1 and picks $\mathtt{ID}_{\mathrm{ch}}$ and two messages M_0, M_1 on which it wants to be challenged. We can assume that $\mathtt{ID}_{\mathrm{ch}}$ has already been queried of H_1 but that $\mathcal{A}_{II}$ has not extracted the private key for this identity. Algorithm $\mathcal{B}$ responds as follows. If $\mathtt{ID}_{\mathrm{ch}} \neq \mathtt{ID}_I$ then $\mathcal{B}$ aborts. Otherwise $\mathtt{ID}_{\mathrm{ch}} = \mathtt{ID}_I$ and $\mathcal{B}$ gives $\mathcal{C}$ the pair M_0, M_1 as the messages on which it wishes to be challenged. $\mathcal{C}$ responds with the challenge ciphertext $C' = \langle U', V', W'\rangle$, such that C' is the HybridPub encryption of M_b under K_{pub} for a random $b \in \{0,1\}$. Then $\mathcal{B}$ sets $C^* = \langle b_I^{-1} U', V', W'\rangle$ and delivers C^* to $\mathcal{A}_{II}$. It is not hard to see that C^* is the FullCL-PKE encryption of M_b for identity $\mathtt{ID}_I$ (with public key $\langle X, Y\rangle$).

Phase 2: $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. Of course, we now restrict $\mathcal{A}_{II}$ to not make private key extraction requests on $\mathtt{ID}_{\mathrm{ch}}$. If any decryption query relayed to $\mathcal{C}$ is equal to the challenge ciphertext C' then $\mathcal{B}$ aborts.

Guess: Eventually, $\mathcal{A}_{II}$ will make a guess b' for b. $\mathcal{B}$ outputs b' as its guess for b.

Now we analyze the behavior of $\mathcal{B}$ and $\mathcal{A}_{II}$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}_{II}$'s view is identical to its view in the real attack. Moreover, if $\mathcal{B}$ does not abort then $2(\Pr[b = b'] - \frac{1}{2}) \geq \epsilon$.

We justify this claim as follows. $\mathcal{B}$'s responses to H_1 queries are uniformly and independently distributed in $\mathbb{G}_1^*$ as in the real attack. All responses to $\mathcal{A}_{II}$'s requests are valid, provided of course that $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext C^* is a valid FullCL-PKE encryption of M_b where $b \in \{0,1\}$ is random. Thus, by definition of algorithm $\mathcal{A}_{II}$ we have that $2(\Pr[b = b'] - \frac{1}{2}) \geq \epsilon$.

The probability that $\mathcal{B}$ does not abort during the simulation remains to be calculated. Examining the simulation, we see that $\mathcal{B}$ can abort for three reasons: (1) because $\mathcal{A}_{II}$ made a private key extraction on $\mathtt{ID}_I$ at some point, (2) because $\mathcal{A}_{II}$ did not choose $\mathtt{ID}_{\mathrm{ch}} = \mathtt{ID}_I$, or (3) because $\mathcal{B}$ relayed a decryption query on $C' = \langle U', V', W'\rangle$ to C in Phase 2.

Because of the way that $\mathcal{B}$ converts ciphertexts, this last event happens only if $\mathcal{A}_{II}$ queries

$\mathcal{B}$ on the ciphertext $C^* = \langle b_I^{-1}U', V', W' \rangle$ in Phase 2. However, this is exactly $\mathcal{A}_{II}$'s challenge ciphertext on which $\mathcal{A}_{II}$ is forbidden from making a decryption query. So this event never occurs in $\mathcal{B}$'s simulation. We name the remaining events that can cause $\mathcal{B}$ to abort as $\mathcal{E}_1$ and $\mathcal{E}_2$.

Notice that the event $\neg\mathcal{E}_2$ implies the event $\neg\mathcal{E}_1$ (if $\mathcal{A}_{II}$ chooses $\mathtt{ID}_{\mathrm{ch}}$ equal to $\mathtt{ID}_I$, then no private key extraction on $\mathtt{ID}_I$ is allowed). Hence we have

$$\begin{aligned}
\Pr[\mathcal{B} \text{ does not abort}] &= \Pr[\neg\mathcal{E}_1 \wedge \neg\mathcal{E}_2] \\
&= \Pr[\neg\mathcal{E}_2] \cdot \Pr[\neg\mathcal{E}_1 | \neg\mathcal{E}_2] \\
&= \Pr[\neg\mathcal{E}_2] \\
&= 1/q_1
\end{aligned}$$

where the last equality follows from $\mathcal{B}$'s random choice of I being independent of $\mathcal{A}_{II}$'s choice of $\mathtt{ID}_{\mathrm{ch}}$.

Thus we see that $\mathcal{B}$'s advantage is at least ϵ/q_1 and the proof is complete.

Proof of Lemma 8: Let $\mathcal{A}_{II}$ be a Type II OWE adversary against `BasicPub` who makes at most q_2 queries to random oracle H_2 and who has advantage ϵ. We show how to construct an algorithm $\mathcal{B}$ which interacts with $\mathcal{A}_{II}$ to solve the BDHP.

Suppose $\mathcal{B}$ has as inputs $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ and $\langle P, aP, bP, cP \rangle$ (where $a, b, c \in \mathbb{Z}_q^*$ are unknown to $\mathcal{B}$). Let $D = e(P,P)^{abc} \in \mathbb{G}_2$ denote the solution to the BDHP on these inputs. Algorithm $\mathcal{B}$ creates a public key $\langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, X, Y, Q, H_2 \rangle$ for $\mathcal{A}_{II}$ by taking $s \in \mathbb{Z}_q^*$ at random, setting $P_0 = sP$, $X = aP$, $Y = saP$ and $Q = bP$. $\mathcal{B}$ then gives this public key and s to $\mathcal{A}_{II}$. $\mathcal{B}$ now sets $U = cP$, chooses V randomly from $\{0,1\}^n$, and gives $\mathcal{A}_{II}$ the challenge ciphertext $C = \langle U, V \rangle$.

Notice that the (unknown) private key is now $absP$ and the (unknown) decryption of C is $M = V \oplus H_2(D^s)$. Hence the solution D to the BDHP can be derived from examining $\mathcal{A}_{II}$'s H_2 queries.

To simulate H_2 queries by $\mathcal{A}_{II}$, $\mathcal{B}$ acts exactly as in the proof of Lemma 5.

Eventually, $\mathcal{A}_{II}$ will output its guess M' for the decryption of C. Now $\mathcal{B}$ chooses a random pair $\langle Z_j, H_{2,j} \rangle$ from the H_2 list and outputs $Z_j^{s^{-1} \bmod q} \in \mathbb{G}_2$ as the solution to the BDHP. (If the list is empty, $\mathcal{B}$ just outputs a random element of $\mathbb{G}_2$.)

It is easy to see that $\mathcal{A}_{II}$'s view in $\mathcal{B}$'s simulation is the same as in a real attack. So $\mathcal{A}_{II}$'s advantage in this simulation will be ϵ. We let $\mathcal{H}$ be the event that D^s is queried of H_2 during $\mathcal{B}$'s simulation and let δ denote the probability that event $\mathcal{H}$ occurs. Now

$$\begin{aligned}
\epsilon &= \Pr[M' = M] \\
&= \Pr[M' = M | \mathcal{H}] \Pr[\mathcal{H}] + \Pr[M' = M | \neg\mathcal{H}] \Pr[\neg\mathcal{H}] \\
&\leq \delta + \tfrac{1}{2^n}(1 - \delta)
\end{aligned}$$

where we have used the fact that if $\mathcal{H}$ does not occur, then H_2 has not been queried on input D^s, so that $\mathcal{A}_{II}$'s view must be independent of the value of M.

Rearranging, we see that $\delta \geq \epsilon - \frac{1}{2^n}$. Since $\mathcal{B}$'s output is of the form $Z_j^{s^{-1} \bmod q}$ with Z_j chosen randomly from the H_2 list, we see that $\mathcal{B}$'s success probability is at least δ/q_2. The lemma follows.