A study of Pair Encodings: Predicate Encryption in prime order groups

Shashank Agrawal * Melissa Chase †

Abstract

Pair encodings and predicate encodings, recently introduced by Attrapadung (Eurocrypt 2014) and Wee (TCC 2014) respectively, greatly simplify the process of designing and analyzing predicate and attribute-based encryption schemes. However, they are still somewhat limited in that they are restricted to composite order groups, and the information theoretic properties are not sufficient to argue about many of the schemes. Here we focus on pair encodings, as the more general of the two. We first study the structure of these objects, then propose a new relaxed but still information theoretic security property. Next we show a generic construction for predicate encryption in prime order groups from our new property; it results in either selective or full security depending on the encoding, and gives security under SXDH or DLIN. Finally, we demonstrate the range of our new property by using it to design the first selectively secure CP-ABE scheme with constant size ciphertexts.

Keywords. Predicate Encryption, Attribute-based Encryption, Pair Encoding Schemes, Dual System technique, Short Ciphertexts

1 Introduction

In traditional public key encryption systems, a message is encrypted under a particular public key, with the guarantee that it can only be decrypted by the party holding the corresponding secret key. Attribute based encryption (ABE), introduced in [SW05] instead allows us to use attributes to determine who has the power to decrypt. In these systems, there is a single entity which publishes system parameters and distributes the appropriate decryption keys to various parties. In key-policy ABE (KP-ABE) [GPSW06], a message is encrypted under a set of attributes describing that message, and each decryption key is associated with a policy describing which ciphertexts it can decrypt. Conversely, in ciphertext-policy ABE (CP-ABE) [BSW07] each user is a given a decryption key that depends on his attributes, and ciphertexts are encrypted with policies describing what users can decrypt them. ABE has been proposed for a variety of applications, from social network privacy to pay-per-view broadcasting to health record access-control to cloud security (see e.g. [PTMW06, TBEM08, BBS+09, APG+11, SRGS12].)

Recently there has been a lot of progress in terms of both security and functionality. Using the dual system framework introduced by Waters [Wat09], several works [LOS+10, LW12] have designed ABE schemes that satisfy the natural security definition, avoiding the restrictions of selective

*University of Illinois Urbana-Champaign. Email: sagrawl2@illinois.edu.
†Microsoft Research. Email: melissac@microsoft.com.

security. [1] Other works consider extra features like short ciphertexts whose length is independent of the size of the associated attribute set and policy [ALdP11, YAHK14], or "unbounded" schemes that place no bounds on the space of possible attributes or the number of attributes that can be tied to a ciphertext or key [LW11, OT12, RW13]. Predicate encryption [BSW11] generalizes the concept to require only that the ciphertext and key are associated with values x, y, and decryption succeeds iff some predicate $P(x, y)$ holds. Note that in this work we assume that x and y are revealed by the ciphertext and key respectively; we do not consider attribute-hiding [BW07, KSW08] or predicate-hiding [SSW09, BRS13].

As these schemes have progressed, however, constructions and proofs have become increasingly complex. Many of the proposed schemes require composite order pairings, in which the order of the pairing groups is a product of two or more primes; since these schemes require that factoring the group order is hard, this in practice means that these groups must be at least an order of magnitude larger than prime order groups of comparable security level, and according to [Gui13] composite order pairing computations are at least 2 orders of magnitude slower. This has prompted efforts to design schemes in prime order groups [OT10, OT12, Fre10, Lew12, HHH+14], but many of these schemes still have fairly high cost as compared to their selectively secure counterparts, and designing and analyzing security of such schemes can be quite challenging.

Two very recent works, by Wee [Wee14] and Attrapadung [Att14] make significant progress in simplifying the design and analysis of new constructions. These works introduce simple new objects, called predicate encodings and pair encodings respectively in the two works, which can be used to construct ABE and other predicate encryption schemes. Essentially, they consider one decryption key and one ciphertext, and focus on what happens in the exponent space. Both formalisms introduce simple information theoretic properties on these objects and show that if these properties are met, they can be extended into fully secure ABE/predicate encryption schemes. The major advantage of this approach is that instead of having to design and prove security of a complex scheme, now all one has to do is design and analyze an appropriate encoding, which is a much simpler task. This vastly simplifies the design of new schemes, and in fact, both works resulted in both new constructions, and more efficient variants of previously known schemes.

Currently these works have two primary limitations. First, they both result in ABE schemes that rely on composite order pairings, which as explained above is very undesirable from an efficiency standpoint. The second drawback is that the strict information theoretic properties they require from the underlying objects mean that there are many constructions that they cannot capture in their model. [Att14] addresses this by introducing a computational security notion, which allows several more interesting constructions to be captured in the framework. However, this security notion is much harder to analyze - it involves not only the encodings in the exponent space, but also elements in the composite order group in which it is embedded, and the proofs that the encodings satisfy this notion are not only computational (rather than information theoretic) but are based on much stronger assumptions.

Still these encodings seem extremely promising as a way to simplify the design and analysis of predicate encryption schemes. In our work we further study these objects, with the aim of understanding them better and beginning to address these limitations. In particular we focus on the pair encodings from [Att14], as they seem able to capture more constructions.

Our Contributions. First, we study the ***structure of pair encodings***. Attrapadung's pair encodings have only limited structural requirements. This means that he is able to capture many existing con-

[1] The original construction of Sahai and Waters [SW05], and much of the following work, considers what is referred to as the selective security model, in which the adversary must commit to the attributes/policy used in the challenge ciphertext before requesting any decryption keys.

structions in his framework, although as mentioned above, in many cases the information theoretic security property he defines does not hold for these schemes. A better understanding of the natural structure of these schemes may help to design new schemes, by providing better intuition for what is important and simply by limiting the search space.

Here we consider two structural properties. First we assume a simple property that describes where the parameters appear in the key and ciphertext. This seems to reflect some basic structure, as all the pair encodings in [Att14] have this property. Looking ahead, this property allows us to instantiate these schemes efficiently in prime order groups. We then show that this implies a second, seemingly unrelated property involving the use of random variables in the key and ciphertexts. We can the use this second property to simplify our security definitions and analyses.

Using this understanding, we propose a ***relaxation of the information theoretic security property*** proposed in [Att14]. This property essentially allows us to consider the scheme at smaller granularity than an an entire key or ciphertext. It is still information theoretic, and it does not depend on the group in which it will be used; this means it is still easy to analyze whether a given encoding satisfies this property. We consider two flavors of this property and show that the stronger of the two is implied by the security properties in [Att14]. However, we will see that our new property is indeed a relaxation in that it allows us to consider encodings that did not satisfy the original property. Thus, we make a first step towards addressing the limitations of the strict information theoretic property of previous work.

Next we present a ***generic construction of predicate encryption*** from pair encodings. Here we make use of the dual system groups introduced by [CW13]; although we must modify their properties slightly, we show that their instantiations are still sufficient.[2] We show that pair encodings which satisfy the stronger flavor of our new property result in fully secure predicate encryption schemes, while pair encodings which satisfy the weaker flavor result in schemes which can still be shown to be selectively secure. While full security is preferable, we will see that this second result allows us to design schemes in areas in which even selectively secure constructions are hard to construct.

This approach has two advantages. First, this means that we can transform any pair encoding scheme which satisfies the information theoretic security properties in [Att14] into a fully secure ABE or predicate encryption scheme ***in a prime order group*** based only on the SXDH or DLIN assumption. The result then is schemes which are of practical efficiency, with strong security guarantees based on mild assumptions. Moreover, the advantage of this approach is that while proof of our generic construction is fairly involved, analyzing a given pair encoding scheme to verify the necessary property is still quite straightforward.

Finally, to demonstrate how our relaxed security property allows us to consider additional functionalities, we present a new pair encoding for ***CP-ABE with constant-size ciphertext***. When used in our generic construction, this results in a CP-ABE with constant size-ciphertext which is selectively secure and can be instantiated under either SXDH or DLIN. To the best of our knowledge, prior to our work there were no known schemes for constant-size CP-ABE, not even selectively secure under very strong assumptions.[3] This shows then that our new techniques allow us to consider a strictly greater range of schemes; we hope that they will continue to prove useful and lead to other interesting constructions.

Other related work. In a very recent work, Chen, Gay, and Wee [CGW15] use a somewhat different approach to go from pair encodings to prime order predicate encryption schemes. They consider

[2] Since we use these groups in a black box way, any improvement in the underlying instantiation will translate directly into an improvement in our generic construction. In particular we believe that the simplified new dual system groups proposed in [CGW15] satisfy our modified definitions as well, so they could be used to simplify our construction.

[3] Here we discount threshold access policies because when only threshold policies are considered, CP-ABE and KP-ABE are equivalent.

strict restrictions on the structure of pair encodings, which are not satisfied by most of the encodings which had previously been proposed. However, they show that all of the previous encodings which satisfy the information theoretic property from [Att14] have counterparts which satisfy these stricter requirements. They then show that given pair encodings which satisfy these strict requirements and a natural information theoretic security property, they can construct fully secure predicate encryption schemes. This results in the most efficient known constructions for a number of problems. As mentioned above, our generic construction can be applied directly to the original [Att14] pair encodings; this will yield similar constructions, with slightly different tradeoffs (generally smaller parameters but slower decryption). Interestingly, our relaxed security property is designed to leverage exactly the kind of structure they prohibit, so perhaps it suggests another way forward for predicates that cannot be addressed under their model.

2 Preliminaries

Notation. We formally define probabilistic polynomial time (PPT) algorithms, negligible functions and different types of indistinguishability in Appendix A. We use $\cong, \equiv$ and $\approx$ to denote statistical, perfect and computational indistinguishability respectively. Security parameter is denoted by λ.

In Appendix A, we also discuss our notation for vectors, matrices, dot products and quantities related to them. We also describe what the symbols $\leftarrow$ and $\leftarrow_R$ mean (in short, the former denotes running an algorithm to obtain an output, while the latter denotes sampling uniformly from a set). A short discussion on *bilinear pairings*, *exponent* of a group, and *homomorphism* is also available.

Predicate family. We consider a predicate family $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ (for some constant c) where $P_\kappa : \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$ maps a ciphertext attribute $x \in \mathcal{X}_\kappa$ and a key attribute $y \in \mathcal{Y}_\kappa$ to a binary value. We assume that the first entry of κ is a number $N \in \mathbb{N}$, which specifies the size of some domain, and rest of the entries would be collectively referred to as par, i.e., $\kappa := (N, \mathsf{par})$

Predicate Encryption. We briefly discuss predicate encryption schemes, their correctness and security; a formal treatment can be found in Appendix A.1. An encryption scheme for a predicate family P_κ over a message space consists of four PPT algorithms Setup, Encrypt, KeyGen, and Decrypt, which carry the usual meaning. Setup takes λ and par as inputs, and outputs master public and secret keys. (The output of Setup defines a number $N \in \mathbb{N}$, perhaps implicitly, and κ is set to (N, par).) Encrypt is used to encrypt a message according to an $x \in \mathcal{X}_\kappa$, and KeyGen produces a key for a $y \in \mathcal{Y}_\kappa$ (given the master secret key). If $P_\kappa(x, y) = 1$, then Decrypt recovers m.

Informally, a scheme is secure if no PPT adversary can distinguish between encryptions of two messages under an x of its choice (except with negligible probability), given keys for any number of y such that $P_\kappa(x, y) = 0$. If the adversary is allowed to request keys any time, we get *full* or *adaptive* security. On the other hand, if requests can only be made after declaring x, then we get *selective* security. Once again, please refer to Appendix A.1 for a formal definition.

3 Pair encoding schemes

The notion of pair encoding schemes (PES) was introduced by Attrapadung [Att14]. Our definition of this scheme is slightly different from the one given by [Att14] in that we place a restriction on the structure. Though the latter definition is more general, we believe that our formulation mirrors the concrete design of such schemes more closely. In particular, all the constructions of pair encoding schemes given in [Att14] fit into our framework without any changes.

A pair encoding scheme for a predicate family $P_\kappa : \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \to \{0,1\}$ indexed by $\kappa = (N, \mathsf{par})$ consists of four polynomial-time deterministic algorithms which satisfy a correctness condition as defined below.

- $\mathsf{Param}(\mathsf{par}) \to n$. The Param algorithm takes the parameters par as input, and outputs a positive integer $n \in \mathbb{N}$ which specifies the number of common variables shared by the following two algorithms. Let $\mathbf{b} := (b_1, b_2, \ldots, b_n)$ denote the common variables.

- $\mathsf{EncC}(x, N) \to (\mathbf{c} := (c_1, c_2, \ldots, c_{w_1}); w_2)$. The EncC algorithm takes an $N \in \mathbb{N}$ and an $x \in \mathcal{X}_{(N,\mathsf{par})}$ as input, and outputs a sequence of w_1 polynomials $c_1, c_2, \ldots, c_{w_1}$ with coefficients in $\mathbb{Z}_N$ and a $w_2 \in \mathbb{N}$. Every polynomial c_ℓ is a linear combination of monomials of the form s, s_i, sb_j, s_ib_j in variables $s, s_1, s_2, \ldots, s_{w_2}$ and $b_1, \ldots, b_n$. More formally, for $\ell \in [1, w_1]$,

$$c_\ell \quad := \quad \zeta_\ell s \quad + \quad \sum_{i \in [1,w_2]} \eta_{\ell,i} s_i \quad + \quad \sum_{j \in [1,n]} \theta_{\ell,j} s b_j \quad + \quad \sum_{i \in [1,w_2], j \in [1,n]} \vartheta_{\ell,i,j} s_i b_j,$$

  where $\zeta_\ell, \eta_{\ell,i}, \theta_{\ell,j}, \vartheta_{\ell,i,j} \in \mathbb{Z}_N$ are constants which define c_ℓ.

- $\mathsf{EncK}(y, N) \to (\mathbf{k} := (k_1, k_2, \ldots, k_{m_1}); m_2)$. The EncK algorithm takes an $N \in \mathbb{N}$ and a $y \in \mathcal{Y}_{(N,\mathsf{par})}$ as input, and outputs a sequence of m_1 polynomials $k_1, k_2, \ldots, k_{m_1}$ with coefficients in $\mathbb{Z}_N$ and an $m_2 \in \mathbb{N}$. Every polynomial k_t is a linear combination of monomials of the form $\alpha, r_{i'}, r_{i'}b_j$ in variables $\alpha, r_1, r_2, \ldots, r_{m_2}$ and $b_1, \ldots, b_n$. More formally, for $t \in [1, m_1]$,

$$k_t \quad := \quad \tau_t \alpha \quad + \quad \sum_{i' \in [1,m_2]} \upsilon_{t,i'} r_{i'} \quad + \quad \sum_{i' \in [1,m_2], j \in [1,n]} \phi_{t,i',j} r_{i'} b_j,$$

  where $\tau_t, \upsilon_{t,i'}, \phi_{t,i',j} \in \mathbb{Z}_N$ are constants which define k_t.

- $\mathsf{Pair}(x, y, N) \to \mathbf{E}$. The EncC algorithm takes an $N \in \mathbb{N}$, an $x \in \mathcal{X}_{(N,\mathsf{par})}$ and a $y \in \mathcal{Y}_{(N,\mathsf{par})}$ as input, and outputs a matrix $\mathbf{E} \in \mathbb{Z}_N^{m_1 \times w_1}$.

Correctness: A pair encoding scheme is correct if for every $\kappa = (N, \mathsf{par})$, $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$ the following holds symbolically

$$\mathbf{kEc} = \sum_{\substack{t \in [1,m_1], \\ \ell \in [1,w_1]}} E_{t,\ell} k_t c_\ell = \alpha s.$$

Structural restrictions. We impose an additional restriction on the form of $\mathbf{E}$. Essentially this says that if k_t has a monomial of the form $r_{i'}b_{j'}$ and a c_ℓ has a monomial of the form sb_j or s_ib_j then $E_{t,\ell}$ must be 0. One can easily verify that *every* pair encoding scheme given in [Att14] (as well as the new one we propose) satisfies this.

Moreover, we can show that given this restriction, we can assume that the set of polynomials output by EncC and EncK have a fairly restricted structure. In simple words, if a polynomial contains the monomial sb_j (or s_ib_j, $r_{i'}b_j$), then there must exist a polynomial which only contains the monomial s (resp. s_i, $r_{i'}$). More precisely, we show that for any pair encoding which satisfies the restriction on E, there is a corresponding one in which EncC and EncK have this structure, and this correspondance preserves all of the security properties defined in [Att14].

For formal statements see Appendix C. For the rest of this work then, we will assume w.l.o.g. that all pair encodings satisfy both of these properties.

3.1 Security

Attrapadung provided two security notions for pair encoding schemes: perfect and computational. As discussed in Section 1, in this paper, we focus on perfect security, which is the information theoretic property, and for which we propose a relaxation. First, we restate here the original security definition given by Attrapadung (which is also referred to as *perfectly master-key hiding* in his paper).

Definition 3.1 (Perfect security [Att14])**.** *A pair encoding scheme* $(\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$ *for a predicate family* P_κ *is perfectly secure if for every* $\kappa = (N, \mathsf{par})$*,* $x \in \mathcal{X}_\kappa$ *and* $y \in \mathcal{Y}_\kappa$ *such that* $P_\kappa(x, y) = 0$*,*

$$\{\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}(0, \mathbf{r}, \mathbf{b})\} \equiv \{\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}(\alpha, \mathbf{r}, \mathbf{b})\}, \tag{1}$$

where $\mathbf{s} \leftarrow_R \mathbb{Z}_N^{w_2+1}$*,* $\mathbf{b} \leftarrow_R \mathbb{Z}_N^n$*,* $\mathbf{r} \leftarrow_R \mathbb{Z}_N^{m_2}$ *and* $\alpha \leftarrow_R \mathbb{Z}_N$*.*

We propose a new relaxed notion of perfect security that allows more flexibility in the design of pair encoding schemes. Very roughly, this property will allow us to add noise gradually to the parameters used in the key, as long as this noise is not detectable given the relevant part of the key and the ciphertext. The goal is to eventually add sufficient noise to completely hide the master secret.

Towards this, we define a new *randomized* sampling algorithm for pair encoding schemes. While the algorithms above are used in the generic construction, the Samp algorithm described below will be used in the security proof.

- $\mathsf{Samp}(d, x, y, N) \to (\mathbf{b}_d := (b_{d,1}, b_{d,2}, \ldots, b_{d,n}))$. This algorithm takes a $d \in [1, m_2]$, an $N \in \mathbb{N}$, an $x \in \mathcal{X}_{(N,\mathsf{par})}$ and a $y \in \mathcal{Y}_{(N,\mathsf{par})}$ as input, and outputs a sequence of n numbers in $\mathbb{Z}_N$. We require that the probability of this algorithm producing $(u \cdot b_{d,1}, u \cdot b_{d,2}, \ldots, u \cdot b_{d,n})$ as output is equal to the probability that it produces $(b_{d,1}, b_{d,2}, \ldots, b_{d,n})$ as output, for any $u \in \mathbb{Z}_N^*$.

Jumping ahead, the dependence of Samp on its inputs will play a crucial role in the proof of security of our generic construction. We will see that if Samp doesn't depend on x, then we can prove our construction to be *fully* secure. But in case it does, we can only prove *selective* security.

Recall that EncK on input y and N produces a sequence of polynomials $\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b})$ with coefficients in $\mathbb{Z}_N$, where every polynomial is a linear combination of monomials of the form $\alpha, r_{i'}, r_{i'} b_j$ in variables $\alpha, r_1, r_2, \ldots, r_{m_2}$ and $b_1, \ldots, b_n$. In the following we use $\mathbf{k}_d(\alpha, r_d, \mathbf{b})$, for $d \in [1, m_2]$, to denote the polynomials in $\mathbf{k}$ obtained by setting all the variables in $\{r_1, r_2, \ldots, r_{m_2}\}$ except r_d to 0. We are now ready to define our new notion of perfect security.

Definition 3.2 (Relaxed perfect security)**.** *A pair encoding scheme* $(\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$ *for a predicate family* P_κ *is relaxed perfectly secure if there exists an algorithm* Samp *(as defined above) such that for every* par*,* $x \in \mathcal{X}_\kappa$ *and* $y \in \mathcal{Y}_\kappa$ *such that* $P_\kappa(x, y) = 0$*, and every* $d \in [1, m_2]$*:*

$$\{\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}_d(0, r_d, \mathbf{b})\} \quad \cong \quad \{\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}_d(0, r_d, \mathbf{b} + \mathbf{b}_d)\}, \tag{2}$$

where $\mathbf{s} \leftarrow_R \mathbb{Z}_N^{w_2+1}$*,* $\mathbf{b} \leftarrow_R \mathbb{Z}_N^n$*,* $r_d \leftarrow_R \mathbb{Z}_N$*,* $\mathbf{b}_d \leftarrow \mathsf{Samp}(d, x, y, N)$*. Furthermore,*

$$\left\{\mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}_d(0, r_d, \mathbf{b} + \mathbf{b}_d)\right\} \cong \left\{\mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}_d(\alpha, r_d, \mathbf{b} + \mathbf{b}_d)\right\}, \tag{3}$$

where $\mathbf{s} \leftarrow_R \mathbb{Z}_N^{w_2+1}$*,* $\mathbf{b} \leftarrow_R \mathbb{Z}_N^n$*,* $r_1, r_2, \ldots, r_{m_2} \leftarrow_R \mathbb{Z}_N$*,* $\alpha \leftarrow_R \mathbb{Z}_N$*,* $\mathbf{b}_d \leftarrow \mathsf{Samp}(d, x, y, N)$ *for* $d \in [1, m_2]$*, and* $\cong$ *denotes statistical indistinguishability. We say* Γ *satisfies strong relaxed perfect hiding if* Samp *does not depend on* x*.*

In Appendix D we show that any pair encoding scheme perfectly secure under the original definition is also secure under the stronger flavor of the relaxed definition.

4 Dual System Groups

A dual system group (DSG) [CW14] is parameterized by a security parameter λ and a number n. It consists of six PPT algorithms as described below.

4.1 Syntax

- $\mathsf{SampP}(1^\lambda, 1^n)$: On input 1^λ and n, SampP outputs public parameters PP and secret parameters SP, which have the following properties:
  - PP contains a triple of groups $(\mathbb{G}, \mathbb{H}, \mathbb{G}_T)$ and a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, a homomorphism μ from $\mathbb{H}$ to $\mathbb{G}_T$, along with some additional parameters used by SampG, SampH. Given PP, we know the exponent of group $\mathbb{H}$ and how to sample uniformly from it. Let $N = \exp(\mathbb{H})$ (see Appendix A). We require that N is a product of distinct primes of $\Theta(\lambda)$ bits.
  - SP contains $\tilde{h} \in \mathbb{H}$ (where $\tilde{h} \neq 1_{\mathbb{H}}$) along with additional parameters used by $\overline{\mathsf{SampG}}$ and $\overline{\mathsf{SampH}}$.
- SampGT takes an element in the image of μ and outputs another element from $\mathbb{G}_T$.
- SampG and SampH take PP as input and output a vector of $n+1$ elements from $\mathbb{G}$ and $\mathbb{H}$ respectively.
- $\overline{\mathsf{SampG}}$ and $\overline{\mathsf{SampH}}$ take both PP and SP as inputs and output a vector of $n+1$ elements from $\mathbb{G}$ and $\mathbb{H}$ respectively.

4.2 Properties

Let the first element of SampG be denoted by SampG_0. Analogously, SampH_0, $\overline{\mathsf{SampG}}_0$ and $\overline{\mathsf{SampH}}_0$ can be defined. A dual system group is *correct* if it satisfies the following three properties:

Projective: For all PP, $h \in \mathbb{H}$ and coin tosses σ, $\mathsf{SampGT}(\mu(h); \sigma) = e(\mathsf{SampG}_0(\text{PP}; \sigma), h)$.

Associative: If $(g_0, g_1, \ldots, g_n)$ and $(h_0, h_1, \ldots, h_n)$ are samples from SampG and SampH respectively, then for all $i \in [1, n]$, $e(g_0, h_i) = e(g_i, h_0)$.

$\mathbb{H}$-subgroup[4]: The output distribution of SampH is the uniform distribution over a subgroup of $\mathbb{H}^{n+1}$.

For *security* we require the following three properties to hold:

Orthogonality: $\tilde{h} \in \mathsf{Kernel}(\mu)$, i.e., $\mu(\tilde{h}) = 1_{\mathbb{G}_T}$.

Non-degeneracy: The following should hold for every PP and SP (recall that $\cong$ denotes statistical indistinguishability):

1. $\overline{\mathsf{SampH}}_0(\text{PP}, \text{SP}) \cong \tilde{h}^\delta$, where $\delta \leftarrow_R \mathbb{Z}_N$.
2. $\exists\, g \in \mathbb{G}$ such that $\overline{\mathsf{SampG}}_0(\text{PP}, \text{SP}) \cong g^\alpha$, where $\alpha \leftarrow_R \mathbb{Z}_N$.
3. For all $\hat{g}_0 \leftarrow \overline{\mathsf{SampG}}_0(\text{PP}, \text{SP})$, $e(\hat{g}_0, \tilde{h})^\beta$ is uniformly distributed over $\mathbb{G}_T$, where $\beta \leftarrow_R \mathbb{Z}_N$.

[4] This property is required to construct encryption schemes with key delegation like HIBE. We do not use this property in our constructions.

Remark 1. In [CW14], the non-degeneracy property is defined in a slightly different way. First, they require that for all $\hat{h}_0 \leftarrow \overline{\mathsf{SampH}}_0(\mathrm{PP}, \mathrm{SP})$, $\tilde{h}$ lies in the group generated by $\hat{h}_0$, instead of the first point above. And secondly, they do not have any constraint on the output of $\overline{\mathsf{SampG}}_0(\mathrm{PP}, \mathrm{SP})$ like in the second point above.

Indistinguishability. For two (positive) polynomials $\mathsf{poly}_1(\cdot)$ and $\mathsf{poly}_2(\cdot)$, define $\mathbf{G}, \mathbf{H}, \hat{\mathbf{G}}, \hat{\mathbf{H}}, \hat{\mathbf{G}}', \hat{\mathbf{H}}'$ as follows:

$$(\mathrm{PP}, \mathrm{SP}) \leftarrow \mathsf{SampP}(1^\lambda, 1^n); \quad \gamma_1, \gamma_2, \ldots, \gamma_n \leftarrow_R \mathbb{Z}_N;$$

$$\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{\mathsf{poly}_1(\lambda)} \leftarrow \mathsf{SampG}(\mathrm{PP}); \mathbf{G} := (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{\mathsf{poly}_1(\lambda)});$$

$$\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{\mathsf{poly}_2(\lambda)} \leftarrow \mathsf{SampH}(\mathrm{PP}); \mathbf{H} := (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{\mathsf{poly}_2(\lambda)});$$

$$\forall i \in [1, \mathsf{poly}_1(\lambda)], \quad \hat{\mathbf{g}}_i := (\hat{g}_{i,0}, \ldots) \leftarrow \overline{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP}); \quad \hat{\mathbf{g}}'_i := (1, \hat{g}_{i,0}^{\gamma_1}, \hat{g}_{i,0}^{\gamma_2}, \ldots, \hat{g}_{i,0}^{\gamma_n})$$

$$\forall j \in [1, \mathsf{poly}_2(\lambda)], \quad \hat{\mathbf{h}}_j := (\hat{h}_{j,0}, \ldots) \leftarrow \overline{\mathsf{SampH}}(\mathrm{PP}, \mathrm{SP}); \quad \hat{\mathbf{h}}'_j := (1, \hat{h}_{j,0}^{\gamma_1}, \hat{h}_{j,0}^{\gamma_2}, \ldots, \hat{h}_{j,0}^{\gamma_n})$$

$$\hat{\mathbf{G}} := (\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_{\mathsf{poly}_1(\lambda)}); \hat{\mathbf{H}} := (\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_{\mathsf{poly}_2(\lambda)});$$

$$\hat{\mathbf{G}}' := (\hat{\mathbf{g}}'_1, \hat{\mathbf{g}}'_2, \ldots, \hat{\mathbf{g}}'_{\mathsf{poly}_1(\lambda)}); \hat{\mathbf{H}}' := (\hat{\mathbf{h}}'_1, \hat{\mathbf{h}}'_2, \ldots, \hat{\mathbf{h}}'_{\mathsf{poly}_2(\lambda)}).$$

We call a dual system group *Left Subgroup Indistinguishable* (LSI), *Right Subgroup Indistinguishable* (RSI) and *Parameter hiding* (PH) if for all polynomials $\mathsf{poly}_1(\cdot)$ and $\mathsf{poly}_2(\cdot)$,

$$\{\mathrm{PP}, \mathbf{G}\} \approx \{\mathrm{PP}, \mathbf{G} \cdot \hat{\mathbf{G}}\}, \tag{4}$$

$$\{\mathrm{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}}, \mathbf{H}\} \approx \{\mathrm{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}}, \mathbf{H} \cdot \hat{\mathbf{H}}\}, \text{and} \tag{5}$$

$$\{\mathrm{PP}, \tilde{h}, \hat{\mathbf{G}}, \hat{\mathbf{H}}\} \equiv \{\mathrm{PP}, \tilde{h}, \hat{\mathbf{G}} \cdot \hat{\mathbf{G}}', \hat{\mathbf{H}} \cdot \hat{\mathbf{H}}'\} \tag{6}$$

hold respectively. Observe that the two distributions in (4) and (5) are computationally indistinguishable, while the two distributions in (6) are identical.

Instantiations of DSG. The three indistinguishability properties defined above are generalizations of the corresponding ones in Chen and Wee. In Appendix B, we show that the two instantiations of DSG – in composite-order groups under the subgroup decision assumption and in prime-order groups under the decisional linear assumption (d-LIN) – given by them satisfy our generalized indistinguishability properties as well as our new definition of non-degeneracy.

Remark 2. We remark that in the prime-order instantiation of dual system groups under the d-LIN assumption given by [CW14], an element from groups $\mathbb{G}$ or $\mathbb{H}$ is represented by $d+1$ elements from a source prime-order group (an element from $\mathbb{G}_T$ is mapped to just one element of a target prime-order group). Now, suppose we have an encryption scheme in dual system groups where the ciphertext/key consists of elements from $\mathbb{G}$ or $\mathbb{H}$ (and possibly an element from $\mathbb{G}_T$). Then, a concrete instantiation in prime-order groups would only double the size of ciphertext/key, if we make the SXDH assumption (special case of d-LIN with $d = 1$), and only triple it if we make the DLIN assumption (special case of d-LIN with $d = 2$).

5 Generic Construction

In this section, we show how to construct a predicate encryption scheme $\Pi_P = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ for any predicate family $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ for which we have a pair encoding scheme $\Gamma_P = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$, using dual system groups. Recall that κ specifies a number $N \in \mathbb{N}$ and some additional parameters par.

- $\mathsf{Setup}(1^\lambda, \mathsf{par})$: First run $\mathsf{Param}(\mathsf{par})$ to obtain n, then run $\mathsf{SampP}(1^\lambda, n)$ to obtain PP and SP. Output

$$\text{MSK} \leftarrow_R \mathbb{H} \qquad \text{MPK} := (\text{PP}, \mu(\text{MSK})).$$

  Recall that given PP, we know the exponent of group $\mathbb{H}$ and can sample uniformly from it. Set $N = \exp(\mathbb{H})$ and $\kappa = (N, \mathsf{par})$.

- $\mathsf{Encrypt}(\text{MPK}, x, m)$: On input an $x \in \mathcal{X}_\kappa$ and an $m \in \mathbb{G}_T$, run $\mathsf{EncC}(x, N)$ to obtain a sequence of w_1 polynomials $(c_1, c_2, \ldots, c_{w_1})$ and a $w_2 \in \mathbb{N}$. Draw $w_2 + 1$ samples from SampG:

$$(g_{0,0}, \ldots, g_{0,n}) \leftarrow \mathsf{SampG}(\text{PP}; \sigma)$$

$$(g_{1,0}, \ldots, g_{1,n}) \leftarrow \mathsf{SampG}(\text{PP}), \ldots, (g_{w_2,0}, \ldots, g_{w_2,n}) \leftarrow \mathsf{SampG}(\text{PP}),$$

  where σ denotes the coin tosses used in drawing the first sample from SampG.

  Output $\text{CT} := (\text{CT}_1, \ldots, \text{CT}_{w_1}, \text{CT}_{w_1+1})$ as the encryption of m under x where

$$\text{CT}_\ell \quad := \quad g_{0,0}^{\zeta_\ell} \quad \cdot \quad \prod_{i \in [1,w_2]} g_{i,0}^{\eta_{\ell,i}} \quad \cdot \quad \prod_{j \in [1,n]} g_{0,j}^{\theta_{\ell,j}} \quad \cdot \quad \prod_{i \in [1,w_2], j \in [1,n]} g_{i,j}^{\vartheta_{\ell,i,j}}$$

  for $\ell \in [1, w_1]$ and $\text{CT}_{w_1+1} := m \cdot \mathsf{SampGT}(\mu(\text{MSK}); \sigma)$.

- $\mathsf{KeyGen}(\text{MPK}, \text{MSK}, y)$: On input a $y \in \mathcal{Y}_\kappa$, run $\mathsf{EncK}(y, N)$ to obtain a sequence of m_1 polynomials $(k_1, k_2, \ldots, k_{m_1})$ and an $m_2 \in \mathbb{N}$. Draw m_2 samples from SampH:

$$(h_{1,0}, \ldots, h_{1,n}) \leftarrow \mathsf{SampH}(\text{PP}), \ldots, (h_{m_2,0}, \ldots, h_{m_2,n}) \leftarrow \mathsf{SampH}(\text{PP}).$$

  Output the key as $\text{SK} := (\text{SK}_1, \text{SK}_2, \ldots, \text{SK}_{m_1})$ where for $t \in [1, m_1]$

$$\text{SK}_t \quad := \quad \text{MSK}^{\tau_t} \quad \cdot \quad \prod_{i' \in [1,m_2]} h_{i',0}^{\upsilon_{t,i'}} \quad \cdot \quad \prod_{i' \in [1,m_2], j \in [1,n]} h_{i',j}^{\phi_{t,i',j}}.$$

- $\mathsf{Decrypt}(\text{MPK}, \text{SK}_y, \text{CT}_x)$: On input $\text{SK}_y := (\text{SK}_1, \text{SK}_2, \ldots, \text{SK}_{m_1})$ and $\text{CT}_x := (\text{CT}_1, \text{CT}_{w_1+1})$, run $\mathsf{Pair}(x, y, N)$ to obtain an $m_1 \times w_1$ matrix $\mathbf{E}$ with entries in $\mathbb{Z}_N$. Output

$$\text{CT}_{w_1+1} \cdot \left[\prod_{t \in [1,m_1], \ell \in [1,w_1]} e(\text{CT}_\ell, \text{SK}_t^{E_{t,\ell}}) \right]^{-1}.$$

Correctness (Sketch). We know that if $P_\kappa(x, y) = 1$, then $\sum_{t \in [1,m_1], \ell \in [1,w_1]} E_{t,\ell} k_t c_\ell = \alpha s$. Consider two polynomials k_t and c_ℓ. When these polynomials are multiplied together, no two monomials – one from k_t and one from c_ℓ – combine to give the same monomial in the product polynomial $k_t c_\ell$, except when

- s is multiplied with $r_{i'} b_j$ and $s b_j$ is multiplied with $r_{i'}$, or
- s_i is multiplied with $r_{i'} b_j$ and $s_i b_j$ is multiplied with $r_{i'}$,

because of the restriction on the form of $\mathbf{E}$. Now, recall that s is mapped to $g_{0,0}$, $r_{i'}b_j$ is mapped to $h_{i',j}$, sb_j is mapped to $g_{0,j}$ and $r_{i'}$ is mapped to $h_{i',0}$. By the associativity property of dual system groups, we know that $e(g_{0,0}, h_{i',j}) = e(g_{0,j}, h_{i',0})$. Further, we mapped s_i to $g_{i,0}$ and s_ib_j to $g_{i,j}$, and associativity guarantees that $e(g_{i,0}, h_{i',j}) = e(g_{i,j}, h_{i',0})$. Therefore, from the observations above, it follows that

$$\prod_{t\in[1,m_1],\ell\in[1,w_1]} e(\mathrm{CT}_\ell, \mathrm{SK}_t^{E_{t,\ell}}) = e(g_{0,0}, \mathrm{MSK}).$$

Finally, by projective property we know that $e(g_{0,0}, \mathrm{MSK}) = \mathsf{SampGT}(\mu(\mathrm{MSK}); \sigma)$.

Remark 3 (Preserving size)**.** Observe that the output of Encrypt consists of w_1+1 elements, w_1 from $\mathbb{G}$ and 1 from $\mathbb{G}_T$ – only one more than the number of polynomials output by EncC. Further, any key has the same number of elements from $\mathbb{H}$ as the number of polynomials output by EncK. Hence, in particular, if w_1 (resp. m_1) is a constant then ciphertexts (resp. keys) are also of constant size, in terms of dual system group elements. Further, if we instantiate dual system groups in prime-order groups under SXDH or DLIN assumption, then the ciphertexts (resp. keys) would still be of constant size (see Remark 2.)

6 Generic proof

In this section, we show that the encryption scheme Π_P constructed for a predicate family $P = \{P_\kappa\}_{\kappa\in\mathbb{N}^c}$ in the previous section is secure using the properties of dual system groups and relaxed perfect security of pair encoding schemes. More formally, we prove the following theorem.

Theorem 6.1. *For any predicate family $P = \{P_\kappa\}_{\kappa\in\mathbb{N}^c}$, if $\Gamma_P = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$ is a **relaxed perfectly secure** pair encoding scheme, then the encryption scheme $\Pi_P = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ constructed in Section 5 (using Γ_P) is **selectively** secure. Furthermore, if the algorithm Samp does not depend on input x, then Π_P is **fully** secure (see Definition A.1).*

Using Lemma D.1, a corollary of the above theorem is that:

Corollary 6.2. *For any predicate family $P = \{P_\kappa\}_{\kappa\in\mathbb{N}^c}$, if $\Gamma_P = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair}, \mathsf{Samp})$ is a **perfectly secure** pair encoding scheme, then the encryption scheme $\Pi_P = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ constructed in Section 5 (using Γ_P) is **fully** secure.*

Recall that dual system groups can be instantiated in prime-order groups under the d-LIN assumption. Together with the above corollary, this gives a useful and interesting result:

Corollary 6.3. *Every perfectly secure pair encoding scheme proposed by Attrapadung [Att14] has a fully secure predicate encryption scheme in prime order groups under the d-LIN assumption.*

The rest of this section is devoted to the proof of Theorem 6.1. We first define auxiliary algorithms for encryption and key generation.

- $\overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; (\mathbf{g}'_0, \mathbf{g}'_1, \ldots, \mathbf{g}'_{w_2}), \mathrm{MSK})$: This algorithm is same as Encrypt except that it uses $\mathbf{g}'_i \in \mathbb{G}^{n+1}$ instead of the samples $\mathbf{g}_i$ from SampG for $i \in [0, w_2]$, and sets $\mathrm{CT}_{w_1+1} := m \cdot e(g'_{0,0}, \mathrm{MSK})$, where $g'_{0,0}$ if the first element of the vector $\mathbf{g}'_0$.
- $\overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}'_0, \mathbf{h}'_1, \ldots, \mathbf{h}'_{m_2}))$: This algorithm is same as KeyGen except that it uses $\mathbf{h}'_i$ instead of the samples $\mathbf{h}_i$ from SampH for $i \in [1, m_2]$.

Type of key	*Inputs to* $\overline{\mathsf{KeyGen}}$ *(besides* PP *and* y*)*
Normal	$\mathrm{MSK}; (\mathbf{h}_1, \ldots, \mathbf{h}_{m_2})$
ρ-Intermediate-1	$\mathrm{MSK}; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})$
ρ-Intermediate-2	$\mathrm{MSK}; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \boxed{\mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}}, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})$
ρ-Intermediate-3	$\mathrm{MSK}; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \boxed{\mathbf{h}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}}, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})$
Pseudo-normal noisy	$\mathrm{MSK}; (\mathbf{h}_1 \cdot \hat{\mathbf{h}}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2} \cdot \hat{\mathbf{h}}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}})$
Pseudo-SF noisy	$\boxed{\overline{\mathrm{MSK}}}; (\mathbf{h}_1 \cdot \hat{\mathbf{h}}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2} \cdot \hat{\mathbf{h}}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}})$
SF noisy	$\overline{\mathrm{MSK}}; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}})$
ρ-SF-intermediate-1	$\overline{\mathrm{MSK}}; (\mathbf{h}_1, \ldots, \mathbf{h}_{\rho-1}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, \mathbf{h}_{\rho+1} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}})$
ρ-SF-intermediate-2	$\overline{\mathrm{MSK}}; (\mathbf{h}_1, \ldots, \mathbf{h}_{\rho-1}, \boxed{\mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho}, \mathbf{h}_{\rho+1} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}})$
ρ-SF-intermediate-3	$\overline{\mathrm{MSK}}; (\mathbf{h}_1, \ldots, \mathbf{h}_{\rho-1}, \boxed{\mathbf{h}_\rho}, \mathbf{h}_{\rho+1} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}})$
SF	$\overline{\mathrm{MSK}}; (\mathbf{h}_1, \ldots, \mathbf{h}_{m_2})$

Table 1: Various types of keys

Using the algorithms described above, we define alternate forms for the ciphertext and master secret key:

- *Semi-functional master secret key* is defined to be $\overline{\mathrm{MSK}} := \mathrm{MSK} \cdot \tilde{h}^{\beta}$ where $\beta \leftarrow_R \mathbb{Z}_N$.
- *Semi-functional ciphertext* is given by $\overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G} \cdot \hat{\mathbf{G}}, \mathrm{MSK})$ where $\mathbf{G} := (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{w_2})$, $\hat{\mathbf{G}} := (\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{w_2})$, $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_{w_2} \leftarrow \mathsf{SampG}(\mathrm{PP})$, and $\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_{w_2} \leftarrow \overline{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$. Observe that $\overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G}, \mathrm{MSK})$ is identically distributed to $\mathsf{Encrypt}(\mathrm{MPK}, x, m)$ – the normal ciphertext – by the projective property of dual system groups.

Table 1 defines various forms of keys for $\rho \in [1, m_2]$ and the inputs that need to be passed to $\overline{\mathsf{KeyGen}}$ (besides PP and y) in order to generate them. In the table, $h_1, \ldots, h_{m_2} \leftarrow \mathsf{SampH}(\mathrm{PP})$, $\hat{h}_1, \ldots, \hat{h}_{m_2} \leftarrow \overline{\mathsf{SampH}}(\mathrm{PP}, \mathrm{SP})$, and $\mathbf{z}_d := (1, z_{d,1}, \ldots, z_{d,n})$, where $(z_{d,1}, \ldots, z_{d,n}) \leftarrow \mathsf{Samp}(d, x, y, N)$ for all $d \in [1, m_2]$. For convenience in the following, we define a slightly modified form of Samp, called $\overline{\mathsf{Samp}}$, which just prepends 1 to the output of Samp. Intermediate-3 and SF-intermediate-3 keys are also defined for $\rho = 0$. Note that 0-Intermediate-3 is distributed identically to a normal key and 0-SF-intermediate-3 is distributed identically to a SF noisy key. Since we have many forms of keys, (where appropriate) we use a box to highlight the part of a key which is different from the previous key.

Proof structure: The novelty in our proof is that instead of working at the level of a key, we work at the level of samples that form the key. Let ξ denote the number of queries made by the adversary, and let y_φ denote the φth query for $\varphi \in [1, \xi]$. Further, let $m_{2,\varphi}$ be the second output of $\mathsf{EncK}(y_\varphi, N)$. We define the following hybrids for $\varphi \in [1, \xi]$ and $\rho \in [1, m_{2,\varphi}]$ (fix any $b \in \{0, 1\}$).

- Hyb_0: This is the real security game $\mathsf{Expt}^{(b)}_{\mathcal{A}, \Pi_P}(\lambda, \mathsf{par})$ described in Appendix A.1.
- Hyb_1: This game is same as the above except that the ciphertext is semi-functional.

Indistinguishability	*Properties needed*	*Proof*
$\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1$	left subgroup indistinguishability	Lemma E.1
$\mathsf{Hyb}_{2,\varphi,3,\rho-1} \approx \mathsf{Hyb}_{2,\varphi,1,\rho}$	right subgroup indistinguishability	Lemma E.2
$\mathsf{Hyb}_{2,\varphi,1,\rho} \cong \mathsf{Hyb}_{2,\varphi,2,\rho}$	non-degeneracy, parameter-hiding, RPS (2)	Lemma E.3
$\mathsf{Hyb}_{2,\varphi,2,\rho} \approx \mathsf{Hyb}_{2,\varphi,3,\rho}$	right subgroup indistinguishability	similar to Lemma E.2
$\mathsf{Hyb}_{2,\varphi,3,m_{2,\varphi}} \approx \mathsf{Hyb}_{2,\varphi,4}$	right subgroup indistinguishability	similar to Lemma E.2
$\mathsf{Hyb}_{2,\varphi,4} \cong \mathsf{Hyb}_{2,\varphi,5}$	non-degeneracy, parameter-hiding, RPS (3)	Lemma E.4
$\mathsf{Hyb}_{2,\varphi,5} \approx \mathsf{Hyb}_{2,\varphi,6}$	right subgroup indistinguishability	similar to Lemma E.2
$\mathsf{Hyb}_{2,\varphi,9,\rho-1} \approx \mathsf{Hyb}_{2,\varphi,7,\rho}$	right subgroup indistinguishability	similar to Lemma E.2
$\mathsf{Hyb}_{2,\varphi,7,\rho} \cong \mathsf{Hyb}_{2,\varphi,8,\rho}$	non-degeneracy, parameter-hiding, RPS (2)	similar to Lemma E.3
$\mathsf{Hyb}_{2,\varphi,8,\rho} \approx \mathsf{Hyb}_{2,\varphi,9,\rho}$	right subgroup indistinguishability	similar to Lemma E.2
$\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}} \cong \mathsf{Hyb}_3$	projective, orthogonality, non-degeneracy	Lemma E.5

Table 2: An outline of the proof structure. After proving a lemma for a certain pair of hybrids, we discuss how the proof can be modified to show indistinguishability of other related pairs. In the above, RPS is a shorthand for relaxed perfect security.

- $\mathsf{Hyb}_{2,\varphi,i,\rho}$ for $i \in \{1,2,3\}$: This game is same as the above except that the first $\varphi - 1$ keys are semi-functional, φth key is of the form ρ-intermediate-i, and rest of the keys are normal.
- $\mathsf{Hyb}_{2,\varphi,4}$: This game is same as the above except that the φth key is Pseudo-normal noisy.
- $\mathsf{Hyb}_{2,\varphi,5}$: This game is same as the above except that the φth key is Pseudo-SF noisy.
- $\mathsf{Hyb}_{2,\varphi,6}$: This game is same as the above except that the φth key is SF noisy.
- $\mathsf{Hyb}_{2,\varphi,i,\rho}$ for $i \in \{7,8,9\}$: This game is same as the above except that the φth key is of the form ρ-SF-intermediate-$(i-6)$.
- Hyb_3: This game is same as $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}}$ except that the ciphertext is a semi-functional encryption of a random message in $\mathbb{G}_T$.

Our goal is to show that Hyb_0 and Hyb_3 are computationally indistinguishable from each other, irrespective of the bit b used by Chl in the security game $\mathsf{Expt}^{(b)}_{\mathcal{A},\Pi_P}(\lambda, \mathsf{par})$. Since Chl encrypts a random message in Hyb_3, there would be no way for a PPT adversary to tell whether m_0 or m_1 was encrypted. This would imply that Π_P is a secure encryption scheme.

Our proof proceeds as follows. We first show that Hyb_0 and Hyb_1 are computationally indistinguishable due to the left subgroup indistinguishability (LSI) property of dual system groups in Lemma E.1; this takes the ciphertext from normal to semi-functional space (the form of the ciphertext doesn't change after this step). After that, we take the keys one by one from normal to semi-functional space by going through a series of hybrids. We show that $\mathsf{Hyb}_{2,1,3,0}$ (or, equivalently, Hyb_1) is computationally indistinguishable from $\mathsf{Hyb}_{2,1,9,m_{2,1}}$ by following the steps shown in Table 2 for $\varphi = 1$; this makes the first key semi-functional while keeping the rest of the keys unchanged. Then, we show that $\mathsf{Hyb}_{2,2,3,0}$ (or, equivalently, $\mathsf{Hyb}_{2,1,9,m_{2,1}}$) is computationally indistinguishable from $\mathsf{Hyb}_{2,1,9,m_{2,2}}$ by once again following the steps shown in Table 2, but now for $\varphi = 2$; as a result, the second key also moves into the semi-functional space. We continue in the same fashion till all the keys are in the

semi-functional space, i.e., we are in the hybrid $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}}$. The last step of the proof is to show that $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}}$ and Hyb_3 are statistically close to each other, which we do in Lemma E.5.

We formally state and prove Lemma E.1, E.2, E.3, E.4 and E.5 in Appendix E. Though Lemma E.2 shows that $\mathsf{Hyb}_{2,\varphi,3,\rho-1}$ and $\mathsf{Hyb}_{2,\varphi,1,\rho}$ are computationally indistinguishable, we discuss immediately afterwards how the proof can be modified to show indistinguishability of other related pairs of hybrids. We do the same with other lemmas too.

Remark 4 (Full vs. selective security.)**.** In transitioning from $\mathsf{Hyb}_{2,\varphi,1,\rho}$ to $\mathsf{Hyb}_{2,\varphi,2,\rho}$ in Lemma E.3, we add randomness using the algorithm $\overline{\mathsf{Samp}}$ to the ρ-th sample of the φ-th key. Observe that if $\overline{\mathsf{Samp}}$ depends on input x, then this transition can only take place if x is known *before* any key queries are issued. Therefore, in this case, we can prove selective security. On the other hand, if $\overline{\mathsf{Samp}}$ does not depend on x, then we get full security (like in the case of all perfectly secure pair encoding schemes of [Att14]).

Remark 5 (Tighter reduction)**.** Recall from the proof of Lemma D.1 that for any perfectly secure pair encoding scheme, we can define a dummy sampling algorithm that always outputs a vector of 0s. When this is the case, the security proof can be considerably simplified: we could directly go from Hyb_1 to $\mathsf{Hyb}_{2,\phi,4}$ and also from $\mathsf{Hyb}_{2,\phi,5}$ to $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}}$ using right subgroup indistinguishability.

7 Ciphertext-Policy ABE

In this section, we design a relaxed perfectly secure pair encoding scheme for Ciphertext-Policy Attribute Based Encryption (CP-ABE). The access policy is represented by a linear secret sharing (LSS) scheme $(\mathbf{A}, \pi)$, where $\mathbf{A}$ is a matrix of size $n_1 \times n_2$ with entries in $\mathbb{Z}_N$ and π is a mapping from $[1, n_1]$ to a universe of attributes $\mathcal{U}$. Let $S \subseteq \mathcal{U}$ be a set of attributes. Let $\Upsilon = \{i \mid i \in [1, n_1], \pi(i) \in S\}$ be the set of rows in $\mathbf{A}$ associated with S. We say that the LSS scheme $(\mathbf{A}, \pi)$ accepts S if $\mathbf{e} = (1, 0, \ldots, 0)$ lies in the span of rows associated with S (otherwise the scheme rejects S). In other words, if S is acceptable, there exists constants $\varepsilon_1, \ldots, \varepsilon_{n_1} \in \mathbb{Z}_N$ such that $\sum_{i \in \Upsilon} \varepsilon_i \mathbf{a_i} = \mathbf{e}$. (This set of constants can be easily computed given S.) An interesting property of LSS schemes that will be useful to us later in the proofs is that if (A, π) rejects S, then there must exist a vector $\mathbf{w} = (w_1, \ldots, w_{n_2})$ such that $\mathbf{w} \cdot \mathbf{a}_i$ for all $i \in \Upsilon$ but $\mathbf{w} \cdot \mathbf{e} = 1$. This, in particular, implies that $w_1 = 1$. (See [Bei11], Claim 2, for a proof of this and other properties below about secret sharing schemes.)

Let $\mathbf{a_i}$ denote the ith row of $\mathbf{A}$ for $i \in [1, n_1]$. In order to share a secret $s \in \mathbb{Z}_N$, one picks $v_2, v_3, \ldots, v_{n_1} \leftarrow_R \mathbb{Z}_N$, and outputs $\mathbf{a_i} \cdot \mathbf{v}$ as the ith share for $i \in [1, n_1]$, where $\mathbf{v} = (s, v_2, v_3, \ldots, v_{n_1})$. This way of sharing a secret leads to two useful properties:

- **Correctness**: For every S accepted by $(\mathbf{A}, \pi)$, every secret $s \in \mathbb{Z}_N$ and any $v_2, v_3, \ldots, v_{n_1} \in \mathbb{Z}_N$, $\sum_{i \in \Upsilon} \varepsilon_i (\mathbf{a_i} \cdot \mathbf{v}) = \mathbf{v} \cdot \sum_{i \in \Upsilon} \varepsilon_i \mathbf{a_i} = s$.
- **Privacy**: For every S rejected by $(\mathbf{A}, \pi)$, the distribution of $\{\mathbf{a_i} \cdot \mathbf{v}\}_{i \in \Upsilon}$ is independent of the secret s being shared.

The predicate family for CP-ABE is indexed by $\kappa = (N, n_1, n_2, \mathcal{U}, T)$. $\mathcal{X}_\kappa$ is the set of all LSS schemes where the matrix is of size $n_1 \times n_2$ with entries in $\mathbb{Z}_N$ and the mapping is from $[1, n_1]$ to $\mathcal{U}$. $\mathcal{Y}_\kappa$ is given by the set $\{S \mid S \subseteq \mathcal{U}, |S| \leq T\}$. For all $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$, $P_\kappa(x, y) = 1$ if and only if x accepts y. It is clear from the definition of predicate family that there is a bound on the size of matrices and the number of attributes associated with a key. But there are no other restrictions: the size of attribute universe $\mathcal{U}$ could be arbitrary and π need not be injective.

We are now ready to design a relaxed perfectly secure pair encoding scheme $\Phi_{\mathsf{cp\text{-}abe}} = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$ for the CP-ABE predicate family.

7.1 Pair Encoding Scheme

- $\mathsf{Param}(\mathsf{par}) \to n_1(n_2 + T + 1)$. Let $\mathbf{b} = (\{b_{i,j}\}_{i\in[1,n_1],j\in[1,n_2]}, \{b'_{i,t}\}_{i\in[1,n_1],t\in[0,T]})$.
- $\mathsf{EncC}((A,\pi), N) \to \mathbf{c}(\mathbf{s}, \mathbf{b}) := (c_1, c_2)$ where

$$c_1 = s \qquad c_2 = s\left(\sum_{\substack{i\in[1,n_1]\\ j\in[1,n_2]}} a_{i,j}b_{i,j} + \sum_{\substack{i\in[1,n_1]\\ t\in[0,T]}} \pi(i)^t b'_{i,t}\right),$$

  and $\mathbf{s} = (s)$, and $a_{i,j}$ denotes the entry in the ith row and jth column of A.

- $\mathsf{EncK}(S, N) \to \mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}) := (\{k_{1,i}, k_{2,i,j}\ k_{3,i,\ell,j}, k_{4,i,y}\ k_{5,i,\ell,t}\}_{\ i,\ell\in[1,n_1],i\neq\ell,j\in[1,n_2],y\in S,t\in[0,T]})$ where

$$k_{1,i} = r_i \quad k_{2,i,j} = r_i b_{i,j} - v_j \qquad k_{3,i,\ell,j} = r_i b_{\ell,j}$$
$$k_{4,i,y} = r_i \sum_{t\in[0,T]} y^t b'_{i,t} \qquad k_{5,i,\ell,t} = r_i b'_{\ell,t}$$

  and $\mathbf{r} = (r_1, r_2, \ldots, r_{n_1}, v_2, \ldots, v_{n_2})$ and $v_1 = \alpha$.

We define Pair algorithm and prove correctness in Appendix F. Here we informally discuss how we can recover αs by combining the polynomials generated by EncC and EncK, with an intent to provide some intuition about the scheme. We can think of $v_2, v_3, \ldots, v_{n_1}$ as the randomness picked in order to share $v_1 = \alpha$ according to the scheme (A, π). Hence, if we find $\mathbf{a}_i \cdot \mathbf{v}$ for all $i \in \Upsilon$, we can recover α (ignore s for now). One could start out by multiplying $a_{i,j}$ by $k_{2,i,j}$ and summing over j, for an $i \in \Upsilon$. This does give $\sum_j a_{i,j}v_j$ but also produces an extra term $r_i \sum_j a_{i,j}b_{i,j}$ (ignore r_i for now). We could try to get rid of this term by using c_2 but the product $a_{i,j}b_{i,j}$ there is also summed over i (since we want EncC to produce a constant number of polynomials, we are forced to pack as much into one polynomial as possible). Fortunately, we have the polynomials $k_{3,i,\ell,j}$ for $\ell \neq i$. We can multiply these by $a_{\ell,j}$ and remove the unwanted $a_{i,j}b_{i,j}$ terms. But we are not done yet: we must also remove the term $\sum_{i,t} \pi(i)^t b'_{i,t}$ left in the mix because we used c_2. If $\pi(i) \in S$, then this is easy: use $k_{4,i,\pi(i)}$ to remove $\sum_t \pi(i)^t b'_{i,t}$, and $k_{5,i,\ell,t} \cdot \pi(\ell)^t$ to remove the rest. However, if $\pi(i) \notin S$, there is no way to do this. Indeed, one can show that in this case c_2 is uniformly distributed.

7.2 Relaxed Perfect Security

We now prove that the pair encoding scheme $\Phi_{\mathsf{cp\text{-}abe}}$ designed above is relaxed perfectly secure (Definition 3.2). Towards this, we first define a sampling algorithm Samp as follows. On input an $i \in [1, n_1]$, $(A, \pi) \in \mathcal{X}_\kappa$, $S \in \mathcal{Y}_\kappa$ and N, Samp checks whether $\pi(i) \notin S$. If yes, it picks elements $\hat{b}_{i,1}, \hat{b}_{i,2}, \ldots, \hat{b}_{i,n_2}$ independently and uniformly from $\mathbb{Z}_N$; otherwise it picks them uniformly but with the constraint that $\sum_{j\in[1,n_2]} a_{i,j}\hat{b}_{i,j} = 0$. Samp outputs

$$\widehat{\mathbf{b}}_i := (\underbrace{0, \ldots, \ldots, \ldots, 0}_{(i-1)n_2}, \hat{b}_{i,1}, \hat{b}_{i,2}, \ldots, \hat{b}_{i,n_2}, \underbrace{0, \ldots, \ldots, \ldots, \ldots, 0}_{(n_1-i)n_2 + n_1(T+1)}). \tag{7}$$

We consider only those $N \in \mathbb{N}$ which are a product of distinct primes of $\Theta(\lambda)$ bits. This is sufficient for our purposes because the Setup algorithm of the generic construction in Section 5 outputs N of exactly this form. We first want to show that for $i \in [1, n_1]$ Equation (2) holds, i.e.,

$$\{\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}_i(0, r_i, \mathbf{b})\} \quad \equiv \quad \{\mathbf{c}(\mathbf{s}, \mathbf{b}), \mathbf{k}_i(0, r_i, \mathbf{b} + \widehat{\mathbf{b}}_i)\}, \tag{8}$$

where $\mathbf{s} \leftarrow_R \mathbb{Z}_N^1$, $\mathbf{b} \leftarrow_R \mathbb{Z}_N^n$, $r_i \leftarrow_R \mathbb{Z}_N$, $\widehat{\mathbf{b}}_i \leftarrow \mathsf{Samp}(i, (A, \pi), S, N)$. This equation trivially holds for $i \in [n_1 + 1, n_1 + n_2 - 1]$ irrespective of what Samp outputs because $\mathbf{k}_i$ does not contain any term with an element of $\mathbf{b}$ in it. (That is why we don't care about defining Samp's behavior on such inputs.)

Let us refer to the left and right distributions in Equation (8) above as Δ_L and Δ_R respectively. Fix an arbitrary $i^* \in [1, n_1]$. By the definition of k_{i^*}, we know that in these two distributions only those components of the key survive which have subscript i^*. Further, in the components $k_{2,i^*,1}, \ldots, k_{2,i^*,n_2}$, the variables $v_2, \ldots, v_{n_2}$ are all set to 0. It is also clear from Equation (7) that the added randomness $\widehat{\mathbf{b}}_{i^*}$ affects only $k_{2,i^*,1}, \ldots, k_{2,i^*,n_2}$ components. For $i \in [1, n_1]$ and $j \in [1, n_2]$, let $\delta_{i,j} := b_{i,j}$ if $i \neq i^*$ and $\delta_{i^*,j} := b_{i^*,j} + \hat{b}_{i^*,j}$ otherwise. Since $b_{i,j}$ are uniformly and independently distributed, so are $\delta_{i,j}$. Now the second component of ciphertext, c_2, can be rewritten as

$$s \left(\sum_{\substack{i \in [1,n_1], i \neq i^* \\ j \in [1,n_2]}} a_{i,j} \delta_{i,j} - \sum_{j \in [1,n_2]} a_{i^*,j} \hat{b}_{i^*,j} + \sum_{t \in [0,T]} \pi(i^*)^t b'_{i^*,t} + \sum_{\substack{i \in [1,n_1], i \neq i^* \\ t \in [0,T]}} \pi(i)^t b'_{i,t} \right).$$

Observe that the only difference between Δ_L and Δ_R is that in the latter case there is an additional term $\mathsf{rand} := \sum_{j \in [1,n_2]} a_{i^*,j} \hat{b}_{i^*,j}$ in the c_2 component of the ciphertext. If $\pi(i^*) \in S$, then this term is 0. On the other hand when $\pi(i^*) \notin S$, we show that $\sum_{t \in [0,T]} \pi(i^*)^t b'_{i^*,t}$ is an independent uniform random variable over $\mathbb{Z}_N$, and therefore, the additional term rand does not matter. Towards this, consider the polynomial $f(x) = b'_{i^*,T} \cdot x^T + b'_{i^*,T-1} \cdot x^{T-1} + \ldots + b'_{i^*,0}$. Since $b'_{i^*,T}, \ldots, b'_{i^*,0}$ are chosen at random, any $T + 1$ distinct points on $f(x)$ are uniformly distributed over $\mathbb{Z}_N^{T+1}$. The only components of the key which depend on $b'_{i^*,T}, \ldots, b'_{i^*,0}$ are $\{k_{4,i^*,y}\}_{y \in S}$, which could also be rewritten as $\{r_{i^*} f(y)\}_{y \in S}$. There could be at most T such components because $|S| \leq T$. Therefore, $\sum_{t \in [0,T]} \pi(i^*)^t b'_{i^*,t} = F(\pi(i^*))$ is independently and uniformly distributed.

The second and last step in proving relaxed perfect security is to show that when (A, π) does not accept S, Equation (3) holds, i.e.,

$$\left\{ \mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{i \in [1, n_1 + n_2 - 1]} \mathbf{k}_i(0, r_i, \mathbf{b} + \widehat{\mathbf{b}}_i) \right\} \equiv \left\{ \mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{i \in [1, n_1 + n_2 - 1]} \mathbf{k}_i(\alpha, r_i, \mathbf{b} + \widehat{\mathbf{b}}_i) \right\}, \tag{9}$$

where $\mathbf{s} \leftarrow_R \mathbb{Z}_N^1$, $\mathbf{b} \leftarrow_R \mathbb{Z}_N^n$, $\mathbf{r} \leftarrow_R \mathbb{Z}_N^{n_1 + n_2 - 1}$, $\alpha \leftarrow_R \mathbb{Z}_N$, and $\widehat{\mathbf{b}}_i \leftarrow \mathsf{Samp}(i, (A, \pi), S, N)$ for $i \in [1, n_1 + n_2 - 1]$. Let us denote the left and right distributions in Equation (9) above by Γ_L and Γ_R respectively. The second component of the key in these two distributions is given by

$$k_{2,i,j} = r_i b_{i,j} + r_i \hat{b}_{i,j} - v_j$$

for $i \in [1, n_1]$ and $j \in [1, n_2]$. The only difference between the distributions is in the components $k_{2,1,1}, \ldots, k_{2,n_1,1}$. In the case of Γ_L, $v_1 = \alpha = 0$, while in the case of Γ_R, it is chosen independently and uniformly from $\mathbb{Z}_N$.

Let us focus on the distribution Γ_L. We claim that if we replace the variables $\hat{b}_{i,j}$ by $\hat{b}_{i,j} + r_i^{-1} w_j \alpha$, where $\alpha \leftarrow_R \mathbb{Z}_N$, then Γ_L is not affected. (With high probability $r_i \in \mathbb{Z}_N^*$, so r_i^{-1} exists. Also, recall

that $\mathbf{w} = (w_1, \ldots, w_{n_2})$ is orthogonal to all the rows associated with S and $\mathbf{w} \cdot \mathbf{e} = 1$; see the discussion at the beginning of this section.) If $\pi(i) \notin S$, we know that $\hat{b}_{i,1}, \hat{b}_{i,2}, \ldots, \hat{b}_{i,n_2}$ are independently and uniformly distributed. Hence adding $r_i^{-1} w_j \alpha$ has no effect on their joint distribution. On the other hand when $\pi(i) \in S$, $\hat{b}_{i,1}, \hat{b}_{i,2}, \ldots, \hat{b}_{i,n_2}$ are uniformly chosen but they satisfy the constraint $\sum_{j \in [1,n_2]} a_{i,j} \hat{b}_{i,j} = 0$. Now, when $r_i^{-1} w_j \alpha$ is added,

$$\sum_{j \in [1,n_2]} a_{i,j} (\hat{b}_{i,j} + r_i^{-1} w_j \alpha) \quad = \quad \sum_{j \in [1,n_2]} a_{i,j} \hat{b}_{i,j} \quad + \quad r_i^{-1} \alpha \sum_{j \in [1,n_2]} a_{i,j} w_j \quad = \quad 0$$

because $\mathbf{w}$ is orthogonal to every $\mathbf{a}_i$ such that $\pi(i) \in S$. Hence, the variables $\hat{b}_{i,1}, \hat{b}_{i,2}, \ldots, \hat{b}_{i,n_2}$ still satisfy the constraint they did before.

The final step in the proof is to replace the variables $w_1\alpha, v_2 + w_2\alpha, \ldots, v_{n_2} + w_{n_2}\alpha$ by $\alpha, v_2, \ldots, v_{n_2}$. This does not affect Γ_L because the latter set of variables are picked independently and uniformly from $\mathbb{Z}_N$ (and $w_1 = 1$). But now Γ_L is exactly the distribution Γ_R.

7.3 Instantiation: Constant-size ciphertext

We briefly comment about instantiating the pair encoding scheme $\Phi_{\mathsf{cp\text{-}abe}} = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$. Using the generic method in Section 5, one can construct a predicate encryption $\Pi_{\mathsf{cp\text{-}abe}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ scheme for CP-ABE using $\Phi_{\mathsf{cp\text{-}abe}}$. Since EncC outputs only two polynomials, $\mathsf{Encrypt}$ outputs only two elements from $\mathbb{G}$ (and one element from $\mathbb{G}_T$). Now, from Remark 2, it follows that one can design a concrete scheme for CP-ABE in prime-order groups where the ciphertext contains only 4 group elements under the SXDH assumption, and only 6 elements under the DLIN assumption (plus an additional element from the target group). Furthermore, only a constant number of pairing operations would be required to decrypt a ciphertext.

References

[CW14] Jie Chen and Hoeteck Wee. Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014. http://eprint.iacr.org/2014/265.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Canetti and Garay [CG13], pages 435–460.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, February 2007.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, May 2005.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, August 2012.

[LW11] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, May 2011.

[CG13] Ran Canetti and Juan A. Garay, editors. *CRYPTO 2013, Part II*, volume 8043 of *LNCS*. Springer, August 2013.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, December 2012.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, August 2010.

[RW13] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 463–474. ACM Press, November 2013.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, February 2014.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, April 2015.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, March 2009.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, April 2008.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, March 2011.

[JWV06] Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors. *ACM CCS 06*. ACM Press, October / November 2006.

[BRS13] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In Canetti and Garay [CG13], pages 461–478.

[Lew12] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, April 2012.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Gilbert [Gil10], pages 62–91.

[BBS+09] Randolph Baden, Adam Bender, Neil Spring, Bobby Bhattacharjee, and Daniel Starin. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM 2009 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 135–146, 2009.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In YeowMeng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer Berlin Heidelberg, 2011.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, August 2009.

[HHH+14] Gottfried Herold, Julia Hesse, Dennis Hofheinz, Carla Ràfols, and Andy Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 261–279. Springer, August 2014.

[Gil10] Henri Gilbert, editor. *EUROCRYPT 2010*, volume 6110 of *LNCS*. Springer, May 2010.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Gilbert [Gil10], pages 44–61.

[APG+11] Joseph A. Akinyele, Matthew W. Pagano, Matthew D. Green, Christoph U. Lehmann, Zachary N. J. Peterson, and Aviel D. Rubin. Securing electronic medical records using attribute-based encryption on mobile devices. In *SPSM'11, ACM Workshop Security and Privacy in Smartphones and Mobile Devices 2011*, pages 75–86, 2011.

[Gui13] Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 357–372. Springer, June 2013.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, May 2014.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Juels et al. [JWV06], pages 89–98. Available as Cryptology ePrint Archive Report 2006/309.

[SRGS12] Nuno Santos, Rodrigo Rodrigues, Krishna P. Gummadi, and Stefan Saroiu. Policy-sealed data: A new abstraction for building trusted cloud services. In *USENIX Security Symposium 2012*, pages 175–188, 2012.

[YAHK14] Shota Yamada, Nuttapong Attrapadung, Goichiro Hanaoka, and Noboru Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 275–292. Springer, March 2014.

[TBEM08] Patrick Traynor, Kevin R. B. Butler, William Enck, and Patrick McDaniel. Realizing massive-scale conditional access systems through attribute-based cryptosystems. In *NDSS 2008*. The Internet Society, February 2008.

[PTMW06] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In Juels et al. [JWV06], pages 99–112.

[ALdP11] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, March 2011.

A Preliminaries

An algorithm is probabilistic polynomial time (PPT) if its running time is upper-bounded by some polynomial in the size of its input, and it is allowed to use random coin tosses during its execution. A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible if for every positive polynomial $\mathsf{poly}(\cdot)$ and all sufficiently large n, $f(n) \leq 1/\mathsf{poly}(n)$. (A polynomial is positive if every input is mapped to a positive real number.)

We denote the security parameter by λ. We use $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. The statistical distance between two distributions D and D^* which take values from the set Ω is given by

$$\Delta(D, D^*) := \frac{1}{2} \sum_{\alpha \in \Omega} |\mathsf{Pr}[D_1 = \alpha] - \mathsf{Pr}[D^* = \alpha]|.$$

Two families of distributions $\mathbb{D} := \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathbb{D}^* := \{\mathcal{D}^*_\lambda\}_{\lambda \in \mathbb{N}}$ indexed by λ are statistically indistinguishable if the function $\Delta_{\mathbb{D},\mathbb{D}^*}(\lambda) = \Delta(\mathcal{D}_\lambda, \mathcal{D}^*_\lambda)$ is negligible. These distributions are perfectly indistinguishable if $\Delta_{\mathbb{D},\mathbb{D}^*}(\lambda)$ is always 0. Furthermore, we say that $\mathbb{D}$ is computational indistinguishable from $\mathbb{D}^*$ if for every PPT algorithm $\mathcal{A}$:

$$|\mathsf{Pr}[\mathcal{A}(\mathcal{D}_\lambda) = 1] - \mathsf{Pr}[\mathcal{A}(\mathcal{D}^*_\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

We use $\cong, \equiv$ and $\approx$ to denote statistical, perfect and computational indistinguishability respectively.

We normally use lower case letters in bold to denote vectors; but if a vector's elements are themselves vectors, we use upper case. For two vectors $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$, we use $\mathbf{u} \cdot \mathbf{v}$ to denote the entry-wise product, i.e., $(u_1 v_1, \ldots, u_n v_n)$. The same notation extends to vectors of vectors, i.e., if $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_m)$ and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_m)$, then $\mathbf{U} \cdot \mathbf{V} = (\mathbf{u}_1 \cdot \mathbf{v}_1, \ldots, \mathbf{u}_m \cdot \mathbf{u}_m)$. $g^{\mathbf{u}}$ should be interpreted as the vector $(g^{u_1}, \ldots, g^{u_n})$. $g^{\mathbf{A}}$, where $\mathbf{A}$ is a matrix, should be interpreted in an analogous way.

We use $\mathbf{u}_1, \ldots, \mathbf{u}_m \leftarrow \mathsf{SampAlg}(\cdot)$ to denote that the algorithm SampAlg is run m times with independent coin tosses to generate samples $\mathbf{u}_1, \ldots, \mathbf{u}_m$. Since the output of this algorithm is a vector, we also use $(u_1, \ldots, u_n) \leftarrow \mathsf{SampAlg}(\cdot)$ to denote that a single sample with co-ordinates $u_1, \ldots, u_n$ is drawn from SampAlg (this should not be confused with the previous notation). Finally, $a \leftarrow_R S$ denotes drawing an element a uniformly at random from the set S.

Bilinear Pairings: Let $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ be three multiplicative groups. A pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ is bilinear if for all $g \in \mathbb{G}, h \in \mathbb{H}$ and $a, b \in \mathbb{Z}$, $e(g^a, h^b) = e(g, h)^{ab}$. This pairing is non-degenerate if whenever $e(g, h) = 1_{\mathbb{G}_T}$, then either $g = 1_{\mathbb{G}}$ or $h = 1_{\mathbb{H}}$ (where $1_{\mathbb{G}}$, for instance, denotes the identity element of $\mathbb{G}$.) We will only be interested in bilinear pairings that are efficiently computable.

The order of an element g of a group G is the smallest positive integer a such that $g^a = 1_G$. The exponent of a group is defined as the least common multiple of the orders of all elements of the group.

One can show that if a non-degenerate bilinear pairing $e : \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ can be defined over three groups $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$, then they all have the same exponent. We use $\exp(G)$ to denote the exponent of a group G.

Homomorphism: A homomorphism from a group G to a group H is a function $\psi : G \to H$ such that for all $g_1, g_2 \in G$, $\psi(g_1 \cdot g_2) = \psi(g_1) \cdot \psi(g_2)$. We define two sets with respect to a homomorphism: $\mathsf{Image}(\psi) = \{\psi(g) \mid g \in G\}$ and $\mathsf{Kernel}(\psi) = \{g \in G \mid \psi(g) = 1_H\}$.

A.1 Predicate Encryption (PE)

An encryption scheme for a predicate family $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of four PPT algorithms which satisfy a correctness condition defined below.

- $\mathsf{Setup}(1^\lambda, \mathsf{par}) \to (\text{MPK}, \text{MSK})$. The Setup algorithm takes as input the unary representation of the security parameter λ and some additional parameters par. It outputs a master public key MPK and a master secret key MSK. The output of Setup defines a number $N \in \mathbb{N}$ (perhaps implicitly), and κ is set to (N, par).
- $\mathsf{Encrypt}(\text{MPK}, x, m) \to \text{CT}$. The encryption algorithm takes public parameters MPK, an $x \in \mathcal{X}_\kappa$ and an $m \in \mathcal{M}_\lambda$ as inputs, and outputs a ciphertext CT.
- $\mathsf{KeyGen}(\text{MPK}, \text{MSK}, y) \to \text{SK}$. The key generation algorithm takes as input the public parameters MPK, the master secret key MSK and a $y \in \mathcal{Y}_\kappa$, and outputs a secret key SK.
- $\mathsf{Decrypt}(\text{MPK}, \text{SK}, \text{CT}) \to m'$. The decryption algorithm takes as input the public parameters MPK, a secret key SK and a ciphertext CT, and outputs a message $m' \in \mathcal{M}_\lambda$.

Correctness: For all par, MPK and MSK output by $\mathsf{Setup}(1^\lambda, \mathsf{par})$, $m \in \mathcal{M}_\lambda$, $x \in \mathcal{X}_\kappa$ and $y \in \mathcal{Y}_\kappa$ such that $P_\kappa(x, y) = 1$, if:

$$\text{CT} \leftarrow \mathsf{Encrypt}(\text{MPK}, x, m) \quad \text{SK} \leftarrow \mathsf{KeyGen}(\text{MPK}, \text{MSK}, y),$$

then

$$\Pr[\mathsf{Decrypt}(\text{MPK}, \text{CT}, \text{SK}) \neq m] \leq \mathsf{negl}(\lambda),$$

where the probability is over the random coin tosses of Encrypt, KeyGen and Decrypt.

Security: Let Π be an encryption scheme for a predicate family $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$. Consider the following experiment $\mathsf{Expt}^{(b)}_{\mathcal{A},\Pi}(\lambda, \mathsf{par})$ between an adversary $\mathcal{A}$ and a challenger Chl for $b \in \{0, 1\}$ when both are given input 1^λ and par:

1. Setup: Chl runs $\mathsf{Setup}(1^\lambda, \mathsf{par})$ to obtain MPK and MSK. It gives MPK to $\mathcal{A}$.
2. Query: $\mathcal{A}$ issues a key query by sending $y \in \mathcal{Y}_\kappa$ to Chl, and obtains $\text{SK} \leftarrow \mathsf{KeyGen}(\text{MPK}, \text{MSK}, y)$ in response. This step can be repeated any number of times $\mathcal{A}$ desires.
3. Challenge: $\mathcal{A}$ sends two messages $m_0, m_1 \in \mathcal{M}_\lambda$ and an $x \in \mathcal{X}_\kappa$ to Chl, and gets $\text{CT} \leftarrow \mathsf{Encrypt}(\text{MPK}, x, m_b)$ as the challenge ciphertext.
4. Query: This step is identical to step 2.

At the end of the experiment, $\mathcal{A}$ outputs a bit b' which is defined to be the output of the experiment. We call an adversary admissible if for every $y \in \mathcal{Y}_\kappa$ queried in steps 2 and 4, $P_\kappa(x, y) = 0$. This prevents $\mathcal{A}$ from succeeding in the experiment simply by decrypting CT.

Definition A.1. *An encryption scheme* Π *is adaptively or fully secure for a predicate family* $P = \{P_\kappa\}_{\kappa \in \mathbb{N}^c}$ *if for every* PPT *admissible adversary* $\mathcal{A}$ *and every* par*,*

$$|\mathsf{Pr}[\mathsf{Expt}^{(0)}_{\mathcal{A},\Pi}(\lambda, \mathsf{par}) = 1] - \mathsf{Pr}[\mathsf{Expt}^{(1)}_{\mathcal{A},\Pi}(\lambda, \mathsf{par}) = 1]| \leq \mathsf{negl}(\lambda),$$

where the probabilities are taken over the coin tosses of $\mathcal{A}$ *and* Chl*. On the other hand,* Π *is selectively secure if the above condition is satisfied w.r.t. to a modified experiment where* $\mathcal{A}$ *provides* $x \in \mathcal{X}_\kappa$ *to* Chl *right after the setup phase (instead of the challenge phase), i.e., before it starts querying.*

B Dual System Groups

Chen and Wee instantiate dual system groups under the subgroup decision assumption in composite-order groups as well as the decisional linear assumption (d-LIN) in prime-order groups. We show that both these instantiations satisfy the generalized indistinguishability properties and the new non-degeneracy property we defined (first two conditions) in Section 4.2. (For the rest of the properties, the proofs given in [CW14] carry over.)

Remark 6 (Sampling algorithms)**.** In the two concrete constructions of dual system groups discussed below, the running time of the four sampling algorithms (SampG, SampH, $\overline{\mathsf{SampG}}$, $\overline{\mathsf{SampH}}$) depends linearly on the number of elements we require from a sample. This could significantly improve the efficiency of encryption schemes built on top of dual system groups. For example, if we need only the first and third elements from a sample of SampG (which consists of $n + 1$ elements), then we could just pass 1 and 3 to SampG (after modifying its definition suitably) and get the required elements, saving a considerable amount of time.

B.1 Composite-order construction

A composite-order bilinear group generator $\mathcal{G}$ takes the security parameter λ as input and outputs $(N, G_N, G_T, g_1, g_2, g_3, e)$. G_N and G_T are two multiplicative cyclic groups of order $N = p_1 p_2 p_3$, where p_1, p_2 and p_3 are three distinct primes of $\Theta(\lambda)$ bits each. e is an efficiently computable non-degenerate bilinear map which maps two elements of G_N to an element of G_T. g_1, g_2 and g_3 denote the generators of G_{p_1}, G_{p_2} and G_{p_3} respectively, where for every divisor n of N, we use G_n to denote the subgroup of G_N of order n. We require that the following two subgroup decision assumptions hold with respect to $\mathcal{G}$.

Definition B.1 (Assumption 1)**.** *Consider the following distribution:*

$$(N, G_N, G_T, g_1, g_2, g_3, e) \leftarrow \mathcal{G}(1^\lambda);$$

$$h_{123} \leftarrow_R G_N;$$

$$D := ((N, G_N, G_T, e); g_1, g_3, h_{123});$$

$$T_0 \leftarrow_R G_{p_1}, T_1 \leftarrow_R G_{p_1 p_2}.$$

We assume that for any PPT *algorithm* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{SD1}}_{\mathcal{A}}(\lambda) := |\mathsf{Pr}[\mathcal{A}(D, T_0) = 1] - \mathsf{Pr}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in λ*.*

Definition B.2 (Assumption 2)**.** *Consider the following distribution:*

$$(N, G_N, G_T, g_1, g_2, g_3, e) \leftarrow \mathcal{G}(1^\lambda);$$

$$h_{123} \leftarrow_R G_N, h_{23} \leftarrow_R G_{p_2p_3}, g_{12} \leftarrow_R G_{p_1p_2};$$
$$D := ((N, G_N, G_T, e); g_1, g_3, h_{123}, h_{23}, g_{12});$$
$$T_0 \leftarrow_R G_{p_1p_3}, T_1 \leftarrow_R G_N.$$

We assume that for any PPT *algorithm* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\mathsf{SD2}}_{\mathcal{A}}(\lambda) := |\mathsf{Pr}[\mathcal{A}(D, T_0) = 1] - \mathsf{Pr}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in λ*.*

We show that the construction given in Section 5.2 of [CW14] satisfies non-degeneracy, LSI, RSI and parameter-hiding properties.

Non-degeneracy: The statistical distance between $\tilde{h}^\delta$ and $(g^2 \cdot g^3)^{\hat{r}}$, where $\delta \leftarrow_R \mathbb{Z}_N$ and $\hat{r} \leftarrow_R \mathbb{Z}_N^*$, is at most $1/p_2 + 1/p_3$, which is negligible in λ. Similarly, if we set $g := g_2$, then the statistical distance between g^μ and $g_2^{\hat{s}}$ for $\mu \in \mathbb{Z}_N$ and $\hat{s} \in \mathbb{Z}_N^*$, is at most $1/p_2$.

Lemma B.3 (SD1 to LSI)**.** *For any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{LSI}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{SD1}}_{\mathcal{B}}(\lambda) + 2/p_1 + 2/p_2 + 1/p_3.$$

Proof. The adversary $\mathcal{B}$ gets as input $((N, G_N, G_T, e); g_1, g_3, h_{123}, T)$, where T is chosen uniformly at random from G_{p_1} or $G_{p_1p_2}$. Using this input, $\mathcal{B}$ simulates the public parameters as follows. It picks $\mathbf{w} \leftarrow_R \mathbb{Z}_N^n$ and gives

$$\text{PP} := ((N, G_N, G_T, e); g_1, g_1^{\mathbf{w}}, g_3, h_{123})$$

to $\mathcal{A}$. Note that PP is properly distributed if h_{123} is a generator of G_N, which happens with probability at least $1 - 1/p_1 - 1/p_2 - 1/p_3$.

Consider any (positive) polynomial $\mathsf{poly}(x)$, and let $\ell := \mathsf{poly}(\lambda)$. $\mathcal{B}$ picks ℓ numbers $u_1, u_2, \ldots, u_\ell \leftarrow_R \mathbb{Z}_N$ such that $u_i \mod p_2 \neq 0$ for $i \in [1, \ell]$, and gives

$$\mathbf{G}' = ((T^{u_1}, T^{u_1\mathbf{w}}), \ldots, (T^{u_\ell}, T^{u_\ell\mathbf{w}}))$$

as the challenge to $\mathcal{A}$. If $T \leftarrow G_{p_1}$, then $\mathbf{G}'$ is identically distributed to $\mathbf{G}$ when T is a generator of G_{p_1}, which happens with probability $1 - 1/p_1$. On the other hand when $T \leftarrow G_{p_1p_2}$, then $\mathbf{G}'$ is identically distributed to $\mathbf{G} \cdot \hat{\mathbf{G}}$ when T is a generator of $G_{p_1p_2}$, which happens with probability at least $1 - 1/p_1 - 1/p_2$. □

Lemma B.4 (SD2 to RSI)**.** *For any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{RSI}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{SD2}}_{\mathcal{B}}(\lambda) + 3/p_1 + 4/p_2 + 3/p_3.$$

Proof. The adversary $\mathcal{B}$ gets as input $((N, G_N, G_T, e); g_1, g_3, h_{123}, h_{23}, g_{12}, T)$, where T is chosen uniformly at random from $G_{p_1p_3}$ or from the whole group. Let poly_1 and poly_2 be two (positive) polynomials. Let ℓ and m denote $\mathsf{poly}_1(\lambda)$ and $\mathsf{poly}_2(\lambda)$ respectively. $\mathcal{B}$ picks $\mathbf{w} \leftarrow_R \mathbb{Z}_N^n$ and ℓ numbers $u_1, u_2, \ldots, u_\ell \leftarrow_R \mathbb{Z}_N$ such that $u_i \mod p_2 \neq 0$ for $i \in [1, \ell]$, and gives

$$\text{PP} := ((N, G_N, G_T, e); g_1, g_1^{\mathbf{w}}, g_3, h_{123})$$

$$\tilde{h} := h_{23} \quad \text{and} \quad \mathbf{G} \cdot \hat{\mathbf{G}} := ((g_{12}^{u_1}, g_{12}^{u_1 \mathbf{w}}), \ldots, (g_{12}^{u_\ell}, g_{12}^{u_\ell \mathbf{w}})$$

to $\mathcal{A}$. In order for $(\text{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}})$ to be properly distributed, we need h_{123}, h_{23} and g_{12} to be generators of G_N, $G_{p_2 p_3}$ and $G_{p_1 p_2}$ respectively, which happens with probability at least $1 - 2/p_1 - 3/p_2 - 2/p_3$.

Now to simulate the challenge, $\mathcal{B}$ picks m numbers $v_1, v_2, \ldots, v_m \leftarrow_R \mathbb{Z}_N$ such that $v_j \mod p_2 \neq 0$ for $j \in [1, m]$ and vectors $\mathbf{X}'_1, \mathbf{X}'_2, \ldots, \mathbf{X}'_m \leftarrow_R G_{p_3}^n$ (using g_3), and outputs

$$\mathbf{H}' = ((T^{v_1}, T^{v_1 \mathbf{w}} \cdot \mathbf{X}'_1), \ldots, (T^{v_m}, T^{v_m \mathbf{w}} \cdot \mathbf{X}'_m)).$$

If $T \leftarrow G_{p_1 p_3}$, then $\mathbf{H}'$ is identically distributed to $\mathbf{H}$, except when T is not a generator of $G_{p_1 p_3}$, which happens with probability at most $1/p_1 + 1/p_3$. On the other hand when $T \leftarrow G_N$, then $\mathbf{H}'$ is identically distributed to $\mathbf{H} \cdot \hat{\mathbf{H}}$, except when T is not a generator G_N, which happens with probability at most $1/p_1 + 1/p_2 + 1/p_3$. □

Lemma B.5 (Parameter-hiding)**.** *For any polynomials* $\mathsf{poly}_1(x)$ *and* $\mathsf{poly}_2(x)$*, the following distributions are identical:*

$$\{\text{PP}, \tilde{h}, ((g_2^{\hat{s}_1}, g_2^{\hat{s}_1 \mathbf{w}}), \ldots, (g_2^{\hat{s}_\ell}, g_2^{\hat{s}_\ell \mathbf{w}})), ((g_2^{\hat{r}_1} \cdot g_3^{\hat{r}_1}, g_2^{\hat{r}_1 \mathbf{w}} \cdot \mathbf{X}_1), \ldots, (g_2^{\hat{r}_m} \cdot g_3^{\hat{r}_m}, g_2^{\hat{r}_m \mathbf{w}} \cdot \mathbf{X}_m))\}$$

and

$$\{\text{PP}, \tilde{h}, ((g_2^{\hat{s}_1}, g_2^{\hat{s}_1 (\mathbf{w} + \mathbf{w}')}), \ldots, (g_2^{\hat{s}_\ell}, g_2^{\hat{s}_\ell (\mathbf{w} + \mathbf{w}')})), \\ ((g_2^{\hat{r}_1} \cdot g_3^{\hat{r}_1}, g_2^{\hat{r}_1 (\mathbf{w} + \mathbf{w}')} \cdot \mathbf{X}_1), \ldots, (g_2^{\hat{r}_m} \cdot g_3^{\hat{r}_m}, g_2^{\hat{r}_m (\mathbf{w} + \mathbf{w}')} \cdot \mathbf{X}_m))\},$$

where

$$\ell := \mathsf{poly}_1(\lambda) \quad \textit{and} \quad m := \mathsf{poly}_2(\lambda);$$
$$(\text{PP}, \text{SP}) \leftarrow \mathsf{SampP}(1^\lambda, 1^n);$$
$$\mathbf{w}, \mathbf{w}' \leftarrow_R \mathbb{Z}_N^n;$$
$$\hat{s}_1, \ldots, \hat{s}_\ell \leftarrow_R \mathbb{Z}_N^*;$$
$$\hat{r}_1, \ldots, \hat{r}_m \leftarrow_R \mathbb{Z}_N^*;$$
$$\mathbf{X}_1, \ldots, \mathbf{X}_m \leftarrow_R G_{p_3}^n.$$

Proof. Note that $\mathbf{w}$ appears in the public parameters PP in the form $g_1^{\mathbf{w}}$. Hence, $\mathbf{w} \mod p_2$ is a uniformly random number in $\mathbb{Z}_{p_2}$ given PP (by Chinese remainder theorem), and the lemma follows. □

B.2 Prime-order construction

A prime-order bilinear group generator $\mathcal{G}$ takes the security parameter λ as input and outputs $(p, G_1, G_2, G_T, g_1, g_2, e)$. G_1, G_2 and G_T are three multiplicative groups of order p, where p is a prime of $\Theta(\lambda)$ bits. e is an efficiently computable non-degenerate bilinear map which maps an element of G_1 and an element of G_2 to an element of G_T. g_1 and g_2 are generators of G_1 and G_2 respectively.

We first define the following generalization of d-LIN assumption with respect to $\mathcal{G}$, whose security follows tightly from d-LIN itself.

Definition B.6 (gen-d-LIN Assumption)**.** *Let* $\mathsf{poly}(x)$ *be a (positive) polynomial in* x*. Given a group generator* $\mathcal{G}$*, we define the following distribution:*

$$(p, G_1, G_2, G_T, g_1, g_2, e) \leftarrow \mathcal{G}(1^\lambda);$$

$$m := \mathsf{poly}(\lambda);$$

$$s_{1,1}, \ldots, s_{1,d}, \ldots, s_{m,1}, \ldots, s_{m,d} \leftarrow_R \mathbb{Z}_p;$$

$$a_1, a_2, \ldots, a_{d+1}, s_{1,d+1}, \ldots, s_{m,d+1} \leftarrow_R \mathbb{Z}_p^*;$$

$$D := ((p, G_1, G_2, G_T, e); g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}}, g_1^{a_1 s_{1,1}}, \ldots, g_1^{a_d s_{1,d}}, \ldots, g_1^{a_1 s_{m,1}}, \ldots, g_1^{a_d s_{m,d}});$$

$$T_0 := (g_1^{a_{d+1}(s_{1,1}+\ldots+s_{1,d})}, \ldots, g_1^{a_{d+1}(s_{m,1}+\ldots+s_{m,d})});$$

$$T_1 := (g_1^{a_{d+1}(s_{1,1}+\ldots+s_{1,d})+s_{1,d+1}}, \ldots, g_1^{a_{d+1}(s_{m,1}+\ldots+s_{m,d+1})+s_{m,d+1}}).$$

We assume that for any polynomial p *and any* PPT *algorithm* $\mathcal{A}$*,*

$$\mathsf{Adv}_{\mathcal{A}}^{gen\text{-}d\text{-}\mathsf{LIN}}(\lambda) := |\mathsf{Pr}[\mathcal{A}(D, T_0) = 1] - \mathsf{Pr}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in λ*.*

The above assumption is defined with respect to the first group G_1 output by $\mathcal{G}$. We also assume that this assumption holds with respect to the second group G_2. We now show how the d-LIN assumption (which is a special case of the above with $m = 1$) can be reduced gen-d-LIN.

Lemma B.7 (d-LIN to gen-d-LIN)**.** *If the* d-LIN *assumption holds for a group generator* $\mathcal{G}$*, then the* gen-d-LIN *assumption stated in Definition B.6 also holds in* $\mathcal{G}$*.*

Proof. Consider any (positive) polynomial $\mathsf{poly}(x)$, and let $m := \mathsf{poly}(\lambda)$. Let $\mathcal{A}$ be a PPT algorithm that gets a non-negligible advantage in the gen-d-LIN security game w.r.t. to the polynomial p. We construct a PPT algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to break the d-LIN assumption as follows. $\mathcal{B}$ obtains

$$((p, G_1, G_2, G_T, e); g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}}, g_1^{a_1 r_1}, \ldots, g_1^{a_d r_d}, T := g_1^{a_{d+1}(r_1+\ldots+r_d)+r_{d+1}})$$

as input, where r_{d+1} is either 0 or uniformly chosen from $\mathbb{Z}_p^*$. It picks

$$s_{1,1}, \ldots, s_{1,d}, \ldots, s_{m,1}, \ldots, s_{m,d} \leftarrow_R \mathbb{Z}_p,$$

$$a_1, a_2, \ldots, a_{d+1}, s_{1,d+1}, \ldots, s_{m,d+1} \leftarrow_R \mathbb{Z}_p^*,$$

and computes

$$[g_1^{a_1 r_1} \cdot (g_1^{a_1})^{s_{i,1}}]^{s_{i,d+1}}, \ldots, [g_1^{a_1 r_d} \cdot (g_1^{a_1})^{s_{i,d}}]^{s_{i,d+1}},$$

$$\begin{aligned} T_i' &:= \left[T \cdot (g_1^{a_{d+1}})^{s_{i,1}} \cdot (g_1^{a_{d+1}})^{s_{i,2}} \ldots (g_1^{a_{d+1}})^{s_{i,d}}\right]^{s_{i,d+1}} \\ &= g_1^{a_{d+1}\{s_{i,d+1}(r_1+s_{i,1})+\ldots+s_{i,d+1}(r_d+s_{i,d})\}+r_{d+1}s_{i,d+1}}, \end{aligned}$$

for every $i \in [1, m]$. $\mathcal{B}$ then gives $((p, G_1, G_2, G_T, e); g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}})$ along with the group elements computed above as challenge to $\mathcal{A}$. It is easy to see that the challenge has the right distribution. □

We are now ready to show how the prime order construction satisfies the security properties we desire.

Non-degeneracy: We know that $\tilde{h}$ is defined to be $g_2^{\mathbf{f}^*}$ and $\overline{\mathsf{SampH}}_0$ is distributed as $g_2^{\hat{r}\mathbf{f}^*}$ for $\hat{r} \leftarrow_R \mathbb{Z}_p^*$. Hence, the statistical distance between $\tilde{h}^\delta$, when $\delta \leftarrow_R \mathbb{Z}_p$, and $\overline{\mathsf{SampH}}_0$ can be at most $1/p$. Similarly, if we set $g := g_1^{\mathbf{f}}$, we can show that the second condition is satisfied.

Lemma B.8 (gen-d-LIN to LSI)**.** *For any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT *adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{LSI}}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{gen\text{-}d\text{-}\mathsf{LIN}}_{\mathcal{B}}(\lambda).$$

Proof. Consider any (positive) polynomial $\mathsf{poly}(x)$, and let $m := \mathsf{poly}(\lambda)$. We first write $(\mathrm{PP}, \mathbf{g}_1, \ldots, \mathbf{g}_m, \mathbf{g}_1 \cdot \hat{\mathbf{g}}_1, \ldots, \mathbf{g}_m \cdot \hat{\mathbf{g}}_m)$ in terms of the prime order construction.

$$\mathrm{PP} := ((p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, e); g_1^{\rho_L(\mathbf{B})}, g_1^{\rho_L(\mathbf{BA_1})}, \ldots, g_1^{\rho_L(\mathbf{BA_n})}, g_2^{\rho_L(\mathbf{B}^*\mathbf{R})}, g_1^{\rho_L(\mathbf{B}^*\mathbf{A_1^T R})}, \ldots, g_1^{\rho_L(\mathbf{B}^*\mathbf{A_n^T R})}),$$

$$\forall i \in [1,m] \qquad \mathbf{g}_i := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_i\\0\end{pmatrix}}, g_1^{\mathbf{BA_1}\begin{pmatrix}\mathbf{s}_i\\0\end{pmatrix}}, \ldots, g_1^{\mathbf{BA_n}\begin{pmatrix}\mathbf{s}_i\\0\end{pmatrix}}\right)$$

$$\forall i \in [1,m] \qquad \mathbf{g}_i \cdot \hat{\mathbf{g}}_i := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_i\\\hat{s}_i\end{pmatrix}}, g_1^{\mathbf{BA_1}\begin{pmatrix}\mathbf{s}_i\\\hat{s}_i\end{pmatrix}}, \ldots, g_1^{\mathbf{BA_n}\begin{pmatrix}\mathbf{s}_i\\\hat{s}_i\end{pmatrix}}\right)$$

In the above, $\mathbf{s}_1, \ldots, \mathbf{s}_m \leftarrow_R \mathbb{Z}_p^d$ and $\hat{s}_1, \ldots, \hat{s}_m \leftarrow_R \mathbb{Z}_p^*$.

$\mathcal{B}$ obtains as input

$$((p, G_1, G_2, G_T, e); g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}}, g_1^{a_1 s_{1,1}}, \ldots, g_1^{a_d s_{1,d}}, \ldots, g_1^{a_1 s_{m,1}}, \ldots, g_1^{a_d s_{m,d}}$$
$$g_1^{a_{d+1}(s_{1,1}+\ldots+s_{1,d})+s_{1,d+1}}, \ldots, g_1^{a_{d+1}(s_{m,1}+\ldots+s_{m,d+1})+s_{m,d+1}}),$$

where $s_{1,d+1}, \ldots, s_{m,d+1}$ are all 0 or uniformly chosen from $\mathbb{Z}_p^*$.

To begin with, $\mathcal{B}$ implicitly sets

$$\mathbf{s}_i := (s_{i,1}, s_{i,2}, \ldots, s_{i,d}),$$

$$\hat{s}_i := s_{i,d+1},$$

for all $i \in [1,m]$. It programs $\mathbf{B}, \mathbf{B}^*, \mathbf{A}_1, \ldots, \mathbf{A}_n, \mathbf{R}$ and simulates the public parameters PP in the same manner as the proof of Lemma 10 in [CW14]. This involves defining

$$\mathbf{W} := \begin{pmatrix} a_1 & & & & \\ & a_2 & & & \\ & & \ddots & & \\ & & & a_d & \\ a_{d+1} & a_{d+1} & \ldots & a_{d+1} & 1 \end{pmatrix},$$

and sampling $\tilde{\mathbf{B}} \leftarrow_R \mathrm{GL}_{d+1}(\mathbb{Z}_p)$ along with $\tilde{\mathbf{A}}_1, \ldots, \tilde{\mathbf{A}}_n \leftarrow_R \mathbb{Z}_p^{(d+1)\times(d+1)}$.

Now, observe that

$$\mathbf{W}\begin{pmatrix}\mathbf{s}_i\\ \hat{s}_i\end{pmatrix} = \begin{pmatrix} a_1 s_{i,1} \\ \vdots \\ a_d s_{i,d} \\ a_{d+1}(s_{i,1}+\ldots+s_{i,d}) + s_{i,d+1}\end{pmatrix},$$

and hence $\mathcal{B}$ can compute

$$g_1^{\mathbf{W}\begin{pmatrix}\mathbf{s}_i\\ \hat{s}_i\end{pmatrix}}$$

for all $i \in [1,m]$ using its input. Lastly, it outputs the challenge as

$$g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_i\\ \hat{s}_i\end{pmatrix}} = g_1^{\tilde{\mathbf{B}}\mathbf{W}\begin{pmatrix}\mathbf{s}_i\\ \hat{s}_i\end{pmatrix}} \quad \text{and} \quad g_1^{\mathbf{B}\mathbf{A_j}\begin{pmatrix}\mathbf{s}_i\\ \hat{s}_i\end{pmatrix}} = g_1^{\tilde{\mathbf{B}}\tilde{\mathbf{A}}_\mathbf{j}\mathbf{W}\begin{pmatrix}\mathbf{s}_i\\ \hat{s}_i\end{pmatrix}}$$

for all $j \in [1,n]$ and $i \in [1,m]$. If $s_{1,d+1},\ldots,s_{m,d+1}$ are all 0, implying that $\hat{s}_1,\ldots,\hat{s}_m$ are 0 as well, then the view of $\mathcal{A}$ is identically distributed to $(\mathsf{PP}, \mathbf{g}_1,\ldots,\mathbf{g}_m)$, otherwise the view is distributed according to $(\mathsf{PP}, \mathbf{g}_1\cdot\hat{\mathbf{g}}_1,\ldots,\mathbf{g}_m\cdot\hat{\mathbf{g}}_m)$. □

Lemma B.9 (gen-d-LIN to RSI)**.** *For any* PPT *adversary $\mathcal{A}$, there exist a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{RSI}}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{gen\text{-}d\text{-}\mathsf{LIN}}_{\mathcal{B}}(\lambda) + \mathsf{poly}(\lambda)/p,$$

where $\mathsf{poly}(\lambda)$ *is independent of* $\mathsf{Adv}^{gen\text{-}d\text{-}\mathsf{LIN}}_{\mathcal{B}}(\lambda)$.

Proof. Consider two (positive) polynomials $\mathsf{poly}_1(x)$ and $\mathsf{poly}_2(x)$. Define $\ell := \mathsf{poly}_1(\lambda)$ and $m := \mathsf{poly}_2(\lambda)$. We first write $(\mathsf{PP}, \tilde{h}, \mathbf{g}_1\cdot\hat{\mathbf{g}}_1,\ldots,\mathbf{g}_\ell\cdot\hat{\mathbf{g}}_\ell, \mathbf{h}_1,\ldots,\mathbf{h}_m, \mathbf{h}_1\cdot\hat{\mathbf{h}}_1,\ldots,\mathbf{h}_m\cdot\hat{\mathbf{h}}_m)$ in terms of the prime order construction.

$$\mathsf{PP} := ((p,\mathbb{G},\mathbb{H},\mathbb{G}_T,e); g_1^{\rho_L(\mathbf{B})}, g_1^{\rho_L(\mathbf{BA_1})},\ldots,g_1^{\rho_L(\mathbf{BA_n})}, g_2^{\rho_L(\mathbf{B^*R})}, g_1^{\rho_L(\mathbf{B^*A_1^TR})},\ldots,g_1^{\rho_L(\mathbf{B^*A_n^TR})}),$$

$$\tilde{h} := g_2^{\rho_R(\mathbf{B^*R})},$$

$$\forall j\in[1,\ell] \qquad \mathbf{g}_j\cdot\hat{\mathbf{g}}_j := \left(g_1^{\mathbf{B}\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix}}, g_1^{\mathbf{BA_1}\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix}},\ldots,g_1^{\mathbf{BA_n}\begin{pmatrix}\mathbf{s}_j\\ \hat{s}_j\end{pmatrix}}\right),$$

$$\forall i\in[1,m] \qquad \mathbf{h}_i := \left(g_2^{\mathbf{B^*R}\begin{pmatrix}\mathbf{r}_i\\ 0\end{pmatrix}}, g_2^{\mathbf{B^*A_1^TR}\begin{pmatrix}\mathbf{r}_i\\ 0\end{pmatrix}},\ldots,g_2^{\mathbf{B^*A_n^TR}\begin{pmatrix}\mathbf{r}_i\\ 0\end{pmatrix}}\right),$$

$$\forall i\in[1,m] \qquad \mathbf{h}_i\cdot\hat{\mathbf{h}}_i := \left(g_2^{\mathbf{B^*R}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}}, g_2^{\mathbf{B^*A_1^TR}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}},\ldots,g_2^{\mathbf{B^*A_n^TR}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}}\right).$$

In the above, $\mathbf{s}_1, \ldots, \mathbf{s}_\ell, \mathbf{r}_1, \ldots, \mathbf{r}_m \leftarrow_R \mathbb{Z}_p^d$ and $\hat{s}_1, \ldots, \hat{s}_\ell, \hat{r}_1, \ldots, \hat{r}_m \leftarrow_R \mathbb{Z}_p^*$.

$\mathcal{B}$ obtains as input

$$((p, G_1, G_2, G_T, e); g_1, g_2, g_1^{a_1}, \ldots, g_1^{a_d}, g_1^{a_{d+1}}, g_1^{a_1 r_{1,1}}, \ldots, g_1^{a_d r_{1,d}}, \ldots, g_1^{a_1 r_{m,1}}, \ldots, g_1^{a_d r_{m,d}}$$
$$g_1^{a_{d+1}(r_{1,1}+\ldots+r_{1,d})+r_{1,d+1}}, \ldots, g_1^{a_{d+1}(r_{m,1}+\ldots+r_{m,d+1})+r_{m,d+1}}),$$

where $r_{1,d+1}, \ldots, r_{m,d+1}$ are all 0 or uniformly chosen from $\mathbb{Z}_p^*$.

To begin with, $\mathcal{B}$ picks $\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_d \leftarrow_R \mathbb{Z}_p^*$ and implicitly sets

$$\mathbf{R} := \begin{pmatrix} a_1\tilde{r}_1 & & & & \\ & a_2\tilde{r}_2 & & & \\ & & \ddots & & \\ & & & a_d\tilde{r}_d & \\ & & & & 1 \end{pmatrix},$$

$$\mathbf{r}_i := (\tilde{r}_1^{-1} r_{i,1}, \tilde{r}_2^{-1} r_{i,2}, \ldots, \tilde{r}_d^{-1} r_{i,d}),$$
$$\hat{r}_i := r_{i,d+1},$$

for all $i \in [1, m]$. It programs $\mathbf{B}, \mathbf{B}^*, \mathbf{A}_1, \ldots, \mathbf{A}_n$ and simulates $\mathrm{PP}, \tilde{h}$ along with $\mathbf{g}_1 \cdot \hat{\mathbf{g}}_1, \ldots, \mathbf{g}_\ell \cdot \hat{\mathbf{g}}_\ell$ (there is an error of ℓ/p in simulating this) in the same manner as the proof of Lemma 11 in [CW14]. This involves defining

$$\mathbf{W}^* := \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ a_1^{-1}a_{d+1} & a_2^{-1}a_{d+1} & \ldots & a_d^{-1}a_{d+1} & 1 \end{pmatrix},$$

sampling $\tilde{\mathbf{B}} \leftarrow_R \mathrm{GL}_{d+1}(\mathbb{Z}_p)$ along with $\tilde{\mathbf{A}}_1, \ldots, \tilde{\mathbf{A}}_n \leftarrow_R \mathbb{Z}_p^{(d+1)\times(d+1)}$, and setting $\tilde{\mathbf{B}}^* := (\tilde{\mathbf{B}}^{-1})^T$.

Now, observe that

$$\mathbf{W}^*\mathbf{R}\begin{pmatrix} \mathbf{r}_i \\ \hat{r}_i \end{pmatrix} = \begin{pmatrix} a_1 r_{i,1} \\ \vdots \\ a_d r_{i,d} \\ a_{d+1}(r_{i,1} + \ldots + r_{i,d}) + r_{i,d+1} \end{pmatrix},$$

and hence $\mathcal{B}$ can compute

$$g_2^{\mathbf{W}^*\mathbf{R}\begin{pmatrix} \mathbf{r}_i \\ \hat{r}_i \end{pmatrix}}$$

for all $i \in [1, m]$ using its input. Lastly, it outputs the challenge as

$$g_2^{\mathbf{B}^*\mathbf{R}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}} = g_2^{\tilde{\mathbf{B}}^*\mathbf{W}^*\mathbf{R}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}} \quad \text{and} \quad g_2^{\mathbf{B}^*\mathbf{A}_j^{\mathrm{T}}\mathbf{R}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}} = g_2^{\tilde{\mathbf{B}}^*\tilde{\mathbf{A}}_j^{\mathrm{T}}\mathbf{W}^*\mathbf{R}\begin{pmatrix}\mathbf{r}_i\\ \hat{r}_i\end{pmatrix}}$$

for all $j \in [1, n]$ and $i \in [1, m]$. If $r_{1,d+1}, \ldots, r_{m,d+1}$ are all 0, implying that $\hat{r}_1, \ldots, \hat{r}_m$ are 0 as well, then the view of $\mathcal{A}$ is identically distributed to $(\mathrm{PP}, \tilde{h}, \mathbf{g}_1 \cdot \hat{\mathbf{g}}_1, \ldots, \mathbf{g}_\ell \cdot \hat{\mathbf{g}}_\ell, \mathbf{h}_1, \ldots, \mathbf{h}_m)$, otherwise the view is distributed according to $(\mathrm{PP}, \tilde{h}, \mathbf{g}_1 \cdot \hat{\mathbf{g}}_1, \ldots, \mathbf{g}_\ell \cdot \hat{\mathbf{g}}_\ell, \mathbf{h}_1 \cdot \hat{\mathbf{h}}_1, \ldots, \mathbf{h}_m \cdot \hat{\mathbf{h}}_m)$. □

Lemma B.10 (Parameter-hiding)**.** *For any (positive) polynomials* $\mathsf{poly}_1(x)$ *and* $\mathsf{poly}_2(x)$*, the following distributions are identical:*

$$\{\mathrm{PP}, g_2^{\mathbf{f}^*}, (g_1^{\hat{s}_1\mathbf{f}}, g_1^{\hat{s}_1\mathbf{f}_1}, \ldots, g_1^{\hat{s}_1\mathbf{f}_n}), \ldots, (g_1^{\hat{s}_\ell\mathbf{f}}, g_1^{\hat{s}_\ell\mathbf{f}_1}, \ldots, g_1^{\hat{s}_\ell\mathbf{f}_n}) \\ (g_2^{\hat{r}_1\mathbf{f}^*}, g_2^{\hat{r}_1\mathbf{f}_1^*}, \ldots, g_2^{\hat{r}_1\mathbf{f}_n^*}), \ldots, (g_2^{\hat{r}_m\mathbf{f}^*}, g_2^{\hat{r}_m\mathbf{f}_1^*}, \ldots, g_2^{\hat{r}_m\mathbf{f}_n^*})\}$$

and

$$\{\mathrm{PP}, g_2^{\mathbf{f}^*}, (g_1^{\hat{s}_1\mathbf{f}}, g_1^{\hat{s}_1(\mathbf{f}_1+\gamma_1\mathbf{f})}, \ldots, g_1^{\hat{s}_1(\mathbf{f}_n+\gamma_n\mathbf{f})}), \ldots, (g_1^{\hat{s}_\ell\mathbf{f}}, g_1^{\hat{s}_\ell(\mathbf{f}_1+\gamma_1\mathbf{f})}, \ldots, g_1^{\hat{s}_\ell(\mathbf{f}_n+\gamma_n\mathbf{f})}) \\ (g_2^{\hat{r}_1\mathbf{f}^*}, g_2^{\hat{r}_1(\mathbf{f}_1^*+\gamma_1\mathbf{f}^*)}, \ldots, g_2^{\hat{r}_1(\mathbf{f}_n^*+\gamma_n\mathbf{f}^*)}), \ldots, (g_2^{\hat{r}_m\mathbf{f}^*}, g_2^{\hat{r}_m(\mathbf{f}_1^*+\gamma_1\mathbf{f}^*)}, \ldots, g_2^{\hat{r}_m(\mathbf{f}_n^*+\gamma_n\mathbf{f}^*)})\}$$

where

$$\ell := \mathsf{poly}_1(\lambda) \quad \textit{and} \quad m := \mathsf{poly}_2(\lambda);$$
$$(\mathrm{PP}, \mathrm{SP}) \leftarrow \mathsf{SampP}(1^\lambda, 1^n);$$
$$\hat{s}_1, \ldots, \hat{s}_\ell, \hat{r}_1, \ldots, \hat{r}_m \leftarrow_R \mathbb{Z}_p^*;$$
$$\gamma_1, \ldots, \gamma_n \leftarrow_R \mathbb{Z}_p.$$

Proof. Our proof closely follows the one given for Lemma 12 in [CW14]. In a manner similar to their's, we could define

$$\mathbf{A}_i' := \mathbf{A}_i + \gamma_i \mathbf{V}$$

for $i \in [1, n]$, where $\mathbf{V}$ is a matrix which is 0 everywhere except the bottom right entry which is 1. We run SampP with $(\mathbf{A}_1', \ldots, \mathbf{A}_n')$ instead of $(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ to generate $(\mathrm{PP}, \mathrm{SP})$ and

$$\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_\ell \leftarrow \overline{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP});$$
$$\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_m \leftarrow \overline{\mathsf{SampH}}(\mathrm{PP}, \mathrm{SP}).$$

Since all the samples above share the same $\gamma_1, \ldots, \gamma_n$, one can easily verify that rest of the proof in [CW14] goes through for the present generalized case as well. □

C Pair encoding schemes

Recall that we consider polynomials of the form:

$$k_t \quad := \quad \tau_t \alpha \quad + \quad \sum_{i' \in [1,m_2]} \upsilon_{t,i'} r_{i'} \quad + \quad \sum_{i' \in [1,m_2], j \in [1,n]} \phi_{t,i',j} r_{i'} b_j,$$

$$c_\ell \quad := \quad \zeta_\ell s \quad + \quad \sum_{i \in [1,w_2]} \eta_{\ell,i} s_i \quad + \quad \sum_{j \in [1,n]} \theta_{\ell,j} s b_j \quad + \quad \sum_{i \in [1,w_2], j \in [1,n]} \vartheta_{\ell,i,j} s_i b_j,$$

Here we consider two properties. The first property says that in decryption, an $s_i b_j$ term is never paired with an $r_{i'} b_{j'}$ term. This property holds for all known pair encodings. The second property says that if $s_i b_j$ appears as a term in one of the polynomials in $\mathbf{c}$, then the polynomial s_i also appears in $\mathbf{c}$, and similarly for s and for $r_{i'}$. We will show that w.l.o.g. we can assume that if the first property hold, then the second holds as well, in that we can always construct an equivalent scheme for which it does hold. This structure allows us to simplify our relaxed perfect security property for pair encodings.

Definition C.1. *We say that a pair encoding is* dual-system-group-compatible (DSG-compatible) *if for all x, y, N and for all outputs of $\mathsf{EncC}(x, N)$, $\mathsf{EncK}(y, N)$, $\mathsf{Pair}(x, y, N)$, for all $t \in [1, m_1], i' \in [1, m_2], j \in [1, n], \ell \in [1, w_1]$ and $i \in [1, w_2]$, if $\phi_{t,i',j} \neq 0$ and $\theta_{\ell,j} \neq 0$ or $\vartheta_{\ell,i,j} \neq 0$, then $E_{t,\ell} = 0$.*

Definition C.2. *We say that a pair encoding* has specified variables *if for all x, y, N and for all outputs of $\mathsf{EncC}(x, N)$ and $\mathsf{EncK}(y, N)$ it holds that*

- *if $\theta_{\ell,j} \neq 0$ then $s \in \mathbf{c}$;*
- *if $\vartheta_{\ell,i,j} \neq 0$, then $s_i \in \mathbf{c}$; and,*
- *if $\phi_{t,i',j} \neq 0$, then $r_{i'} \in \mathbf{k}$;*

for all $\ell \in [1, w_1]$, $j \in [1, n]$, $i \in [1, w_2]$, $i' \in [1, m_2]$ and $t \in [1, m_1]$.

Lemma C.3. *We show that for any pair encoding which is DSG-compatible, we can construct an equivalent encoding which also has specified variables. Here by equivalent we mean that the number of polynomials produced by $\mathsf{EncK}, \mathsf{EncC}$ is increased by at most 1, and that perfect security and the computation security and co-security properties defined by [Att14] are preserved.*

Proof. Let $\Gamma_P = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$ be a DSG-compatible pair encoding scheme, and suppose that it does not have specified variables. We will describe a process which iteratively removes tuples (ℓ, j), (ℓ, i, j), or (t, i', j) for which the property does not hold.

For the new EncC', we first run EncC to generate polynomials $\mathbf{c}$, and then we modify them as follows:

First, for any ℓ, i, j tuple in the above set, suppose that there exists a polynomial $c_{\ell'}$ which contains s_i, and which does not contain any $s b_j$ or $s_{i'} b_j$ terms. In this case we will create a variable s' and replace $c_{\ell'} = \zeta_{\ell'} s + \sum_{i'} \eta_{\ell',i'} s_{i'}$ with the polynomial $c'_\ell = \eta_{\ell',i} s'$. Then for every other occurrence of s_i in the encoding, we will replace it with $s' - (\zeta_{\ell'} s + \sum_{i' \neq i} \eta_{\ell',i'} s_{i'})$. We adjust the coefficients produced by Pair accordingly. Finally, we can rename s' back to s_i. (Note that the result will still be DSG-compatible.) Correctness is clearly preserved. To see that the distributions of $\mathbf{k}, \mathbf{c}$ are preserved, note that rather than choose $s_i \leftarrow_R Z_N$ and then compute c_ℓ, it is equivalent to choose the value for c_ℓ at random, and then solve for s_i, as long as $\eta_{\ell',i} \neq 0$. Thus, perfect security is preserved. Similarly this preserves the selective or co-selective security properties defined in [Att14].

We can repeat this above process until there is no ℓ, i, j satisfying the second condition in Definition C.2 and for which this process applies. Note that each time we create a polynomial consisting of a single s_i. (And there can be at most one such polynomial per s_i, because we will only apply the process for i, ℓ, j in tuples in the above set, i.e. i such that there is no polynomial s_i in $\mathbf{c}$.) Since the number of s_i variables does not increase, this process will take at most w_2 steps.

Now, if there are remaining ℓ, i, j tuples, it must be the case that any $c_{\ell'}$ which contains s_i also contains either an $sb_{j'}$ term or an $s_{i'}b_{j'}$ term. In this case, we claim that we can replace $s_i b_j$ in all polynomials with an additional variable s'. To see why this works note that DSG-compatability requires that for every y, any c'_ℓ containing $s_{i'}b_{j'}$ is never paired with $r_{i''}b_j$ for any i''. This means that for every y, any c'_ℓ containing s_i is never paired with $r_{i''}b_j$ for any i''. And by correctness we know that the coefficients of $s_i b_j \cdot r_{i''}$ and $s_i \cdot r_{i''} b_j$ must sum to 0. Thus, we conclude that the coefficients of $s_i b_j \cdot r_{i''}$ sum to 0 for all i''. This means that correctness is still satisfied after the replacement because the $s' \cdot r_{i''}$ terms will all vanish in the reconstruction. (Note that this transformation does increase the total number of variables, however it does not change the number of polynomials in $\mathbf{c}, \mathbf{k}$.) To see that perfect security is preserved, note that replacing sb_j with a new variable has essentially the same result on the distributions in Definition 3.1 as adding a uniform random variable to both sides: if the original distributions were indistinguishable, the resulting ones will be as well. Similarly, for Attrapadung's selective and co-selective security, it can only decrease the adversary's advantage.

We repeat the above process until there are no ℓ, i, j tuples for which the second condition applies.

For the new EncK', we first run EncK to generate polynomials $\mathbf{k}$, and then we modify them by applying a similar process, with one exception. Before we begin the first step, if α occurs in more than one polynomial, we cancel α from all but one of the polynomials. (E.g. if α appears in with coefficient τ_1 in k_1 and τ_2 in k_2, then we replace k_2 with $\tau_1 k_2 - \tau_2 k_1$.) Again, we adjust the coefficients produced by Pair accordingly. (For our example, we would change the pairing coefficient $E_{1,\ell}$ with $E_{1,\ell} + \tau_2/\tau_1 E_{2,\ell}$ and $E_{2,\ell}$ with $E_{2,\ell}/\tau_1$ for all ℓ.) Assume w.l.o.g. that after this process, α only appears in k_1. Now, we apply a similar process to that above with the exception that we never apply the first step to k_1 (i.e., we never replace k_1 with a new random k'.) This results in a pair encoding scheme which has no t, i', j tuples for which the property in Definition C.2 does not hold.

Finally, we note that in the resulting encoding EncC, for any x there must be a polynomial containing only the term s. This is because α only appears in one polynomial, and we must be able to reconstruct αs. This means there are no ℓ, j tuples for which the property in Definition C.2 does not hold. Thus, this encoding has specified variables.

□

D Perfect vs Relaxed

Lemma D.1. *Let $\Gamma = (\mathsf{Param}, \mathsf{EncC}, \mathsf{EncK}, \mathsf{Pair})$ be a pair encoding scheme. If Γ is prefectly secure (Definition 3.1), then Γ is also relaxed perfectly secure (Definition 3.2). Moreover, we can define a Samp algorithm for Γ that does not depend on the input x.*

Proof. For any pair encoding scheme Γ, define Samp to output a vector of zeroes on any input. With this definition, (2) is trivially satisfied for every $d \in [1, m_2]$, and the two distributions in (3) reduce to

$$\left\{ \mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}_d(0, r_d, \mathbf{b}) \right\} \quad \text{and} \quad \left\{ \mathbf{c}(\mathbf{s}, \mathbf{b}), \sum_{d \in [1, m_2]} \mathbf{k}_d(\alpha, r_d, \mathbf{b}) \right\}. \tag{10}$$

Since Γ is perfectly secure, we know that if $\mathbf{s} \leftarrow_R \mathbb{Z}_N^{w_2+1}$, $\mathbf{b} \leftarrow_R \mathbb{Z}_N^n$, $\mathbf{r} \leftarrow_R \mathbb{Z}_N^{m_2}$ and $\alpha \leftarrow_R \mathbb{Z}_N$,

then

$$\{\mathbf{c}(\mathbf{s},\mathbf{b}), \mathbf{k}(0,\mathbf{r},\mathbf{b})\} \equiv \{\mathbf{c}(\mathbf{s},\mathbf{b}), \mathbf{k}(\alpha,\mathbf{r},\mathbf{b})\}.$$

Since $m_2 \in \mathbb{Z}_N^*$, we can replace $\mathbf{k}(\alpha,\mathbf{r},\mathbf{b})$ with $\mathbf{k}(m_2\alpha,\mathbf{r},\mathbf{b})$ in the above without changing the joint distribution. Now, observe that $\mathbf{k}(0,\mathbf{r},\mathbf{b}) = \sum_{d\in[1,m_2]} \mathbf{k}_d(0, r_d, \mathbf{b})$ and $\mathbf{k}(m_2\alpha,\mathbf{r},\mathbf{b}) = \sum_{d\in[1,m_2]} \mathbf{k}_d(\alpha, r_d, \mathbf{b})$ symbolically. Therefore, the two distributions in (10) are identical. □

E Proof of security

We use $\mathsf{Adv}_{\mathcal{A}}^{p-q}(\lambda)$ to denote the advantage of an adversary $\mathcal{A}$ in distinguishing Hyb_p from Hyb_q when the security parameter is λ.

Lemma E.1. *For any* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathcal{A}}^{0-1}(\lambda) \le \mathsf{Adv}_{\mathcal{B}}^{\mathsf{LSI}}(\lambda).$$

Proof. $\mathcal{B}$ gets as input $(\mathrm{PP}, \mathbf{G}')$ where $\mathbf{G}'$ is either $\mathbf{G}$ or $\mathbf{G}\cdot\hat{\mathbf{G}}$. While $\mathbf{G}$ is an ordered set of w_2+1 samples from SampG, $\hat{\mathbf{G}}$ is an ordered set of the same size with samples from $\overline{\mathsf{SampG}}$ (recall that LSI property holds for every polynomial, and in particular, for w_2+1). $\mathcal{B}$ first picks $\mathrm{MSK} \leftarrow_R \mathbb{H}$ and outputs $(\mathrm{PP}, \mu(\mathrm{MSK}))$ as the master public key. When $\mathcal{A}$ sends a challenge x^* and two messages m_0, m_1, $\mathcal{B}$ responds with $\overline{\mathsf{Encrypt}}(\mathrm{PP}, x^*, m_b; \mathbf{G}', \mathrm{MSK})$ as the ciphertext, where b is uniformly chosen bit. Further, when $\mathcal{A}$ issues a key query y (either before or after the challenge), $\mathcal{B}$ responds with $\overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1, \ldots, \mathbf{h}_{m_2}))$ by sampling $\mathbf{h}_1, \ldots, \mathbf{h}_{m_2}$ from SampH.

When $\mathbf{G}' = \mathbf{G}$, then the view of $\mathcal{A}$ is identically distributed as in Hyb_0. On the other hand, when $\mathbf{G}' = \mathbf{G}\cdot\hat{\mathbf{G}}$, it is easy to see that view of $\mathcal{A}$ is identical to Hyb_1. □

Lemma E.2. *For any* PPT *adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}_{\mathcal{A}}^{(2,\varphi,3,\rho-1)-(2,\varphi,1,\rho)}(\lambda) \le \mathsf{Adv}_{\mathcal{B}}^{\mathsf{RSI}}(\lambda),$$

for every $\varphi \in [1,\xi]$ and $\rho \in [1, m_{2,\varphi}]$.

Proof. $\mathcal{B}$ gets as input $(\mathrm{PP}, \tilde{h}, \mathbf{G}\cdot\hat{\mathbf{G}}, \mathbf{h}')$ where $\mathbf{h}'$ is either $\mathbf{h}$ or $\mathbf{h}\cdot\hat{\mathbf{h}}$ (special case of RSI with $\mathsf{poly}_2(x) = 1$). $\mathcal{B}$ first picks $\mathrm{MSK} \leftarrow_R \mathbb{H}$ and outputs $(\mathrm{PP}, \mu(\mathrm{MSK}))$ as the master public key. When $\mathcal{A}$ sends a challenge x^* and two messages m_0, m_1, $\mathcal{B}$ responds with $\overline{\mathsf{Encrypt}}(\mathrm{PP}, x^*, m_b; \mathbf{G}\cdot\hat{\mathbf{G}}, \mathrm{MSK})$ as the ciphertext, where b is uniformly chosen bit.

$\mathcal{B}$ picks a $\beta \leftarrow_R \mathbb{Z}_N$ and sets $\overline{\mathrm{MSK}} := \mathrm{MSK}\cdot(\tilde{h})^\beta$. When $\mathcal{A}$ issues ςth key query y_ς, it responds with

$$\mathrm{SK}_{y_\varsigma} := \begin{cases} \overline{\mathsf{KeyGen}}(\mathrm{PP}, \overline{\mathrm{MSK}}, y_\varsigma; (\mathbf{h}_1^{(\varsigma)}, \ldots, \mathbf{h}_{m_{2,\varsigma}}^{(\varsigma)})) & \text{if } \varsigma < \varphi \\ \overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y_\varsigma; (\mathbf{h}_1^{(\varsigma)}\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1}^{(\varsigma)}\cdot\tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}', \mathbf{h}_{\rho+1}^{(\varsigma)}, \ldots, \mathbf{h}_{m_{2,\varsigma}}^{(\varsigma)})) & \text{if } \varsigma = \varphi \\ \overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y_\varsigma; (\mathbf{h}_1^{(\varsigma)}, \ldots, \mathbf{h}_{m_{2,\varsigma}}^{(\varsigma)})) & \text{otherwise,} \end{cases}$$

where for every $\varsigma \in [1,\xi]$ and $i \in [1, m_{2,\varsigma}]$ (except when $\varsigma = \varphi$ and $i = \rho$), $\mathbf{h}_i^{(\varsigma)} \leftarrow \mathsf{SampH}(\mathrm{PP})$, and for every $j \in [1, \rho-1]$, $\mathbf{z}_j \leftarrow \overline{\mathsf{Samp}}(j, x, y, N)$. It is easy to see that when $\mathbf{h}' = \mathbf{h}$, then the view of $\mathcal{A}$ is identically distributed to $\mathsf{Hyb}_{2,\varphi,3,\rho-1}$, and when $\mathbf{h}' = \mathbf{h}\cdot\hat{\mathbf{h}}$, then it is identically distributed to $\mathsf{Hyb}_{2,\varphi,1,\rho}$. □

We now see how the above proof can be adapted to show indistinguishability between other pairs of hybrids. Below, we only describe the changes that need to be made; other details can be easily worked out.

- $\mathsf{Hyb}_{2,\varphi,2,\rho} \approx \mathsf{Hyb}_{2,\varphi,3,\rho}$: In order to generate the φth key, use

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y_\varphi; (\mathbf{h}_1^{(\varphi)} \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1}^{(\varphi)} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}' \cdot \tilde{h}^{\mathbf{z}_\rho}, \mathbf{h}_{\rho+1}^{(\varphi)}, \ldots, \mathbf{h}_{m_{2,\varphi}}^{(\varphi)})),$$

  where $\mathbf{z}_\rho \leftarrow \overline{\mathsf{Samp}}(\rho, x, y, N)$.

- $\mathsf{Hyb}_{2,\varphi,3,m_{2,\varphi}} \approx \mathsf{Hyb}_{2,\varphi,4}$: Assume that $\mathcal{B}$ gets $(\mathrm{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}}, \mathbf{H}')$ as input where $\mathbf{H}' := (\mathbf{h}'_1, \ldots, \mathbf{h}'_{m_{2,\varphi}})$ is a vector of $m_2 + 1$ samples (instead of just 1). In order to generate the φth key, it uses

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y_\varphi; (\mathbf{h}'_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}'_{m_{2,\varphi}} \cdot \tilde{h}^{\mathbf{z}_{m_{2,\varphi}}})),$$

  where $\mathbf{z}_j \leftarrow \overline{\mathsf{Samp}}(j, x, y, N)$ for all $j \in [1, m_{2,\varphi}]$.

- $\mathsf{Hyb}_{2,\varphi,5} \approx \mathsf{Hyb}_{2,\varphi,6}$: Once again assume that $\mathcal{B}$ gets $(\mathrm{PP}, \tilde{h}, \mathbf{G} \cdot \hat{\mathbf{G}}, \mathbf{H}')$ as input where $\mathbf{H}' := (\mathbf{h}'_1, \ldots, \mathbf{h}'_{m_{2,\varphi}})$ is a vector of $m_2 + 1$ samples. In order to generate the φth key, it uses

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \overline{\mathrm{MSK}}, y_\varphi; (\mathbf{h}'_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}'_{m_{2,\varphi}} \cdot \tilde{h}^{\mathbf{z}_{m_{2,\varphi}}})),$$

  where $\mathbf{z}_j \leftarrow \overline{\mathsf{Samp}}(j, x, y, N)$ for $j \in [1, m_{2,\varphi}]$.

- $\mathsf{Hyb}_{2,\varphi,9,\rho-1} \approx \mathsf{Hyb}_{2,\varphi,7,\rho}$: In order to generate the φth key, use

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \overline{\mathrm{MSK}}, y_\varphi; (\mathbf{h}_1^{(\varphi)}, \ldots, \mathbf{h}_{\rho-1}^{(\varphi)}, \mathbf{h}' \cdot \tilde{h}^{\mathbf{z}_\rho}, \mathbf{h}_{\rho+1}^{(\varphi)} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_{2,\varphi}}^{(\varphi)} \cdot \tilde{h}^{\mathbf{z}_{m_{2,\varphi}}})),$$

  where $\mathbf{z}_j \leftarrow \overline{\mathsf{Samp}}(j, x, y, N)$ for $j \in [\rho, m_{2,\varphi}]$.

- $\mathsf{Hyb}_{2,\varphi,8,\rho} \approx \mathsf{Hyb}_{2,\varphi,9,\rho}$: In order to generate the φth key, use

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \overline{\mathrm{MSK}}, y_\varphi; (\mathbf{h}_1^{(\varphi)}, \ldots, \mathbf{h}_{\rho-1}^{(\varphi)}, \mathbf{h}', \mathbf{h}_{\rho+1}^{(\varphi)} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_{2,\varphi}}^{(\varphi)} \cdot \tilde{h}^{\mathbf{z}_{m_{2,\varphi}}})),$$

  where $\mathbf{z}_j \leftarrow \overline{\mathsf{Samp}}(j, x, y, N)$ for $j \in [\rho + 1, m_{2,\varphi}]$.

Lemma E.3. *For every $\varphi \in [1, \xi]$ and $\rho \in [1, m_{2,\varphi}]$, $\mathsf{Hyb}_{2,\varphi,1,\rho} \cong \mathsf{Hyb}_{2,\varphi,2,\rho}$.*

Proof. Given PP, MSK and $\tilde{h}$, one can generate MPK and every key except φth (because in order to generate this key, we need to be able to sample from $\hat{\mathbb{H}}$, which means we need secret parameters SP). Hence, it suffices to show that the following two distributions are statistically close (for clarity, we omit φ in the following):

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G} \cdot \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2}))\},$$

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G} \cdot \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2}))\}.$$

But observe that:

$$\overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G} \cdot \hat{\mathbf{G}}, \mathrm{MSK}) = \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G}, \mathrm{MSK}) \cdot \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}}, \mathrm{MSK}),$$

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})) = \overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})) \cdot \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho, 1, \ldots, 1)),$$

$$\overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})) = \overline{\mathsf{KeyGen}}(\mathrm{PP}, \mathrm{MSK}, y; (\mathbf{h}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{\rho-1} \cdot \tilde{h}^{\mathbf{z}_{\rho-1}}, \mathbf{h}_\rho, \mathbf{h}_{\rho+1}, \ldots, \mathbf{h}_{m_2})) \cdot \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, 1, \ldots, 1)),$$

because of the way Encrypt and KeyGen are defined and bilinearity of e (see the construction in Section 5). The first component on the right hand side of each of the above equations can be generated given PP, MSK and $\tilde{h}$. Hence, we only need to focus on the second components, i.e., it is enough to show that the following two distributions are statistically close:

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho, 1, \ldots, 1))\}, \tag{11}$$

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, 1, \ldots, 1))\}. \tag{12}$$

Let us focus on the first distribution between the two above. By the parameter-hiding property of dual system groups we know that $\{\mathrm{PP}, \tilde{h}, \hat{\mathbf{G}}, \hat{\mathbf{h}}_\rho\}$ and $\{\mathrm{PP}, \tilde{h}, \hat{\mathbf{G}} \cdot \hat{\mathbf{G}}', \hat{\mathbf{h}}_\rho \cdot \hat{\mathbf{h}}'_\rho\}$ are identically distributed. Hence (11) is identically distributed to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}} \cdot \hat{\mathbf{G}}', \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho \cdot \hat{\mathbf{h}}'_\rho, 1, \ldots, 1))\}. \tag{13}$$

Let $\hat{\mathrm{CT}} := (\hat{\mathrm{CT}}_1, \ldots, \hat{\mathrm{CT}}_{w_1+1})$ and $\hat{\mathrm{SK}} := (\hat{\mathrm{SK}}_1, \ldots, \hat{\mathrm{SK}}_{m_1})$ denote the output of $\overline{\mathsf{Encrypt}}$ and $\overline{\mathsf{KeyGen}}$ respectively. We know that for $\ell \in [1, w_1]$,

$$\hat{\mathrm{CT}}_\ell = \hat{g}_{0,0}^{\zeta_\ell} \cdot \prod_{i \in [1,w_2]} \hat{g}_{i,0}^{\eta_{\ell,i}} \cdot \prod_{j \in [1,n]} (\hat{g}_{0,j} \cdot \hat{g}_{0,0}^{\gamma_j})^{\theta_{\ell,j}} \cdot \prod_{i \in [1,w_2], j \in [1,n]} (\hat{g}_{i,j} \cdot \hat{g}_{i,0}^{\gamma_j})^{\vartheta_{\ell,i,j}},$$

where $(\hat{g}_{i,0}, \ldots, \hat{g}_{i,n}) \leftarrow \overline{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$ for $i \in [1, w_2 + 1]$ and $\gamma_1, \ldots, \gamma_n \leftarrow_R \mathbb{Z}_N$. Using the non-degeneracy property of dual system groups, we can write $\hat{g}_{0,0}$ and $\hat{g}_{i,0}$ as g^δ and g^{δ_i} respectively, for $i \in [1, w_2]$, where $\delta, \delta_1, \ldots, \delta_{w_2} \leftarrow_R \mathbb{Z}_N$. Then we consider $\hat{g}_{0,j}$ (and $\hat{g}_{i,j}$) for $j = 1, \ldots, n$ to be values sampled from $\overline{\mathsf{SampG}}$ conditioned on the value of $\hat{g}_{0,0}$ (resp. $\hat{g}_{i,0}$). (These values may not be efficiently sampleable.) Therefore, we have

$$\hat{\mathrm{CT}}_\ell = g^{\zeta_\ell \delta + \sum_{i \in [1,w_2]} \eta_{\ell,i} \delta_i + \sum_{j \in [1,n]} \theta_{\ell,j} \delta \gamma_j + \sum_{i \in [1,w_2], j \in [1,n]} \vartheta_{\ell,i,j} \delta_i \gamma_j} \cdot \prod_{j \in [1,n]} \hat{g}_{0,j}^{\theta_{\ell,j}} \cdot \prod_{i \in [1,w_2], j \in [1,n]} \hat{g}_{i,j}^{\vartheta_{\ell,i,j}} \tag{14}$$

Shifting our focus to the key, we know that its tth component is given by

$$\hat{\mathrm{SK}}_t = \hat{h}_{\rho,0}^{\upsilon_{t,\rho}} \cdot \prod_{j \in [1,n]} (\hat{h}_{\rho,j} \cdot \hat{h}_{\rho,0}^{\gamma_j})^{\phi_{t,\rho,j}},$$

for $t \in [1, m_1]$, where $(\hat{h}_{\rho,0}, \ldots, \hat{h}_{\rho,n}) \leftarrow \overline{\mathsf{SampH}}(\mathrm{PP}, \mathrm{SP})$. Using non-degeneracy once again, we can write $\hat{h}_{\rho,0}$ as $\tilde{h}^\omega$ for an $\omega \leftarrow_R \mathbb{Z}_N$, and consider $\hat{h}_{\rho,j}$ for $j = 1, \ldots, n$ to be sampled from $\overline{\mathsf{SampH}}$ conditioned on the value of $\hat{h}_{\rho,0}$. Hence,

$$\hat{\mathrm{SK}}_t = \tilde{h}^{\upsilon_{t,\rho}\omega + \sum_{j \in [1,n]} \phi_{t,\rho,j} \omega \gamma_j} \cdot \prod_{j \in [1,n]} \hat{h}_{\rho,j}^{\phi_{t,\rho,j}}. \tag{15}$$

Now, observe the superscripts of g and $\tilde{h}$ in (14) and (15) respectively (over $\ell \in [1, w_1]$ and $t \in [1, m_1]$). We know that $\delta, \delta_1, \ldots, \delta_{w_2}$, $\gamma_1, \ldots, \gamma_n$ and ω are randomly chosen from $\mathbb{Z}_N$. Hence, we can use the first property (2) of relaxed perfect security associated with pair encoding schemes to add noise to the ρ-th sample used in the key. But the problem is that in any sample drawn from $\overline{\mathsf{SampG}}$ and $\overline{\mathsf{SampH}}$, elements of the sample may depend on each other. In particular $\hat{g}_{0,j}$ may reveal some information about δ, and similarly for $\hat{g}_{i,j}$ and for $\hat{h}_{\rho,j}$, so we must ensure that (2) applies even given this information. Note, however, that $\delta, \delta_1, \ldots, \delta_{w_2}$ and ω are an *explicit* part of the distribution in (2) (see the discussion on *structural restrictions* after the definition of pair encoding schemes, and Lemma C.3 in Appendix C for more details). Therefore, given a sample from either of the distributions in (2), one can compute the first element of the samples from $\overline{\mathsf{SampG}}$ and $\overline{\mathsf{SampH}}$, and then draw rest of the elements conditioned on the first ones. (If δ, for instance, is not *explicit*, then we know that $\theta_{\ell,j} = 0$ for all $\ell \in [1, w_1]$ and $j \in [1, n]$. Hence, we don't need to worry about $\hat{g}_{0,j}$ revealing some information about δ. We can argue about $\delta_1, \ldots, \delta_{w_2}$ or ω not being explicit in a similar way.)

In a nutshell, we can apply (2) to conclude that the distribution

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, (\hat{\mathrm{CT}}_1, \ldots, \hat{\mathrm{CT}}_{w_1+1}), (\hat{\mathrm{SK}}_1, \ldots, \hat{\mathrm{SK}}_{m_1})\}$$

is statistically close to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, (\hat{\mathrm{CT}}_1, \ldots, \hat{\mathrm{CT}}_{w_1+1}), (\tilde{\mathrm{SK}}_1, \ldots, \tilde{\mathrm{SK}}_{m_1})\},$$

where

$$\begin{aligned} \tilde{\mathrm{SK}}_t &:= \tilde{h}^{\upsilon_{t,\rho}\omega + \sum_{j\in[1,n]} \phi_{t,\rho,j}\omega(\gamma_j + z_j)} \cdot \prod_{j\in[1,n]} \hat{h}_{\rho,j}^{\phi_{t,\rho,j}} \\ &= \tilde{h}^{\upsilon_{t,\rho}\omega + \sum_{j\in[1,n]} \phi_{t,\rho,j}\omega\gamma_j} \cdot \prod_{j\in[1,n]} (\hat{h}_{\rho,j} \cdot \tilde{h}^{\omega z_j})^{\phi_{t,\rho,j}}, \end{aligned}$$

for $t \in [1, m_1]$, and $\mathbf{z}_\rho = (z_1, \ldots, z_n) \leftarrow \mathsf{Samp}(\rho, x, y, N)$. Observe that the only difference between $\hat{\mathrm{SK}}_t$ and $\tilde{\mathrm{SK}}_t$ is that an extra $\tilde{h}^{\omega z_j}$ is multiplied with $\hat{h}_{\rho,j}$ in the latter case. Hence, the key $(\tilde{\mathrm{SK}}_1, \ldots, \tilde{\mathrm{SK}}_{m_1})$ can be generated by giving $\hat{\mathbf{h}}_\rho \cdot \hat{\mathbf{h}}'_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}$ as the ρ-th sample to $\overline{\mathsf{KeyGen}}$ ($\mathbf{z}_\rho$ has the same distribution as $\omega \cdot \mathbf{z}_\rho$ since $\omega \in \mathbb{Z}_N^*$ with high probability). Therefore, (13) is statistically close to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}} \cdot \hat{\mathbf{G}}', \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho \cdot \hat{\mathbf{h}}'_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, 1, \ldots, 1))\}.$$

Using parameter-hiding once again, we can show that the above distribution is identical to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, 1, y; (1, \ldots, 1, \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, 1, \ldots, 1))\},$$

which completes the proof. □

The above proof can be easily adapted to show that $\mathsf{Hyb}_{2,\varphi,7,\rho} \cong \mathsf{Hyb}_{2,\varphi,8,\rho}$. In this case, we need to show that

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G} \cdot \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, \overline{\mathrm{MSK}}, y; (\mathbf{h}_1, \ldots, \mathbf{h}_{\rho-1}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho \cdot \tilde{h}^{\mathbf{z}_\rho}, \mathbf{h}_{\rho+1} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}}))\},$$

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, m; \mathbf{G} \cdot \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, \overline{\mathrm{MSK}}, y; (\mathbf{h}_1, \ldots, \mathbf{h}_{\rho-1}, \mathbf{h}_\rho \cdot \hat{\mathbf{h}}_\rho, \mathbf{h}_{\rho+1} \cdot \tilde{h}^{\mathbf{z}_{\rho+1}}, \ldots, \mathbf{h}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}}))\}.$$

Observe that the only difference now is that we have $\overline{\text{MSK}}$ instead of MSK, and noise is present in the samples $\rho+1,\ldots,n$ instead of $1,\ldots,\rho-1$. So, we can split $\overline{\mathsf{Encrypt}}$ and $\overline{\mathsf{KeyGen}}$ in a way similar to the above proof, and once again it suffices to show that exactly the distributions in (11) and (12) are indistinguishable.

Lemma E.4. *For every* $\varphi \in [1,\xi]$, $\mathsf{Hyb}_{2,\varphi,4} \cong \mathsf{Hyb}_{2,\varphi,5}$.

Proof. This proof proceeds in a manner similar to the proof of Lemma E.3. To begin with, we observe as before that given PP, MSK and $\tilde{h}$, one can generate MPK and every key except φth. Hence, it suffices to show that the following two distributions are statistically close (for clarity, we omit φ in the following):

$$\{\text{PP}, \text{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\text{PP}, x, m; \mathbf{G}\cdot\hat{\mathbf{G}}, \text{MSK}), \overline{\mathsf{KeyGen}}(\text{PP}, \text{MSK}, y; (\mathbf{h}_1\cdot\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2}\cdot\hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\},$$

$$\{\text{PP}, \text{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\text{PP}, x, m; \mathbf{G}\cdot\hat{\mathbf{G}}, \text{MSK}), \overline{\mathsf{KeyGen}}(\text{PP}, \overline{\text{MSK}}, y; (\mathbf{h}_1\cdot\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2}\cdot\hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\}.$$

Note that the only difference between the two distributions is in the form of the master secret key: in the first case, we have a normal master key, while in the second case, we have a semi-functional one. Also,

$$\overline{\mathsf{Encrypt}}(\text{PP}, x, m; \mathbf{G}\cdot\hat{\mathbf{G}}, \text{MSK}) = \overline{\mathsf{Encrypt}}(\text{PP}, x, m; \mathbf{G}, \text{MSK}) \cdot \overline{\mathsf{Encrypt}}(\text{PP}, x, 1; \hat{\mathbf{G}}, \text{MSK}),$$

$$\begin{aligned}&\overline{\mathsf{KeyGen}}(\text{PP}, \text{MSK}, y; (\mathbf{h}_1\cdot\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2}\cdot\hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\\ &\quad= \overline{\mathsf{KeyGen}}(\text{PP}, \text{MSK}, y; (\mathbf{h}_1, \ldots, \mathbf{h}_{m_2})) \cdot \overline{\mathsf{KeyGen}}(\text{PP}, 1, y; (\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}})),\end{aligned}$$

$$\begin{aligned}&\overline{\mathsf{KeyGen}}(\text{PP}, \overline{\text{MSK}}, y; (\mathbf{h}_1\cdot\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \mathbf{h}_{m_2}\cdot\hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\\ &\quad= \overline{\mathsf{KeyGen}}(\text{PP}, \text{MSK}, y; (\mathbf{h}_1, \ldots, \mathbf{h}_{m_2})) \cdot \overline{\mathsf{KeyGen}}(\text{PP}, \tilde{h}^{\beta}, y; (\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}})),\end{aligned}$$

where $\beta \leftarrow_R \mathbb{Z}_N$. The first component on the right hand side of each of the above equations can be generated given PP, MSK and $\tilde{h}$. Hence, it is enough to show that the following two distributions are statistically close:

$$\{\text{PP}, \text{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\text{PP}, x, 1; \hat{\mathbf{G}}, \text{MSK}), \overline{\mathsf{KeyGen}}(\text{PP}, 1, y; (\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\}, \tag{16}$$

$$\{\text{PP}, \text{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\text{PP}, x, 1; \hat{\mathbf{G}}, \text{MSK}), \overline{\mathsf{KeyGen}}(\text{PP}, \tilde{h}^{\beta}, y; (\hat{\mathbf{h}}_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\}. \tag{17}$$

Let us focus on the first distribution between the two above. By the parameter-hiding property of dual system groups, it is identically distributed to

$$\{\text{PP}, \text{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\text{PP}, x, 1; \hat{\mathbf{G}}\cdot\hat{\mathbf{G}}', \text{MSK}), \overline{\mathsf{KeyGen}}(\text{PP}, 1, y; (\hat{\mathbf{h}}_1\cdot\hat{\mathbf{h}}'_1\cdot\tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2}\cdot\hat{\mathbf{h}}'_{m_2}\cdot\tilde{h}^{\mathbf{z}_{m_2}}))\}. \tag{18}$$

Let $\hat{\text{CT}} := (\hat{\text{CT}}_1, \ldots, \hat{\text{CT}}_{w_1+1})$ and $\hat{\text{SK}} := (\hat{\text{SK}}_1, \ldots, \hat{\text{SK}}_{m_1})$ denote the output of $\overline{\mathsf{Encrypt}}$ and $\overline{\mathsf{KeyGen}}$ respectively. We know that for $\ell \in [1, w_1]$,

$$\hat{\text{CT}}_\ell \;=\; \hat{g}_{0,0}^{\zeta_\ell} \;\cdot\; \prod_{i\in[1,w_2]} \hat{g}_{i,0}^{\eta_{\ell,i}} \;\cdot\; \prod_{j\in[1,n]} (\hat{g}_{0,j}\cdot\hat{g}_{0,0}^{\gamma_j})^{\theta_{\ell,j}} \;\cdot\; \prod_{i\in[1,w_2],j\in[1,n]} (\hat{g}_{i,j}\cdot\hat{g}_{i,0}^{\gamma_j})^{\vartheta_{\ell,i,j}},$$

where $(\hat{g}_{i,0}, \ldots, \hat{g}_{i,n}) \leftarrow \overline{\mathsf{SampG}}(\mathrm{PP}, \mathrm{SP})$ for $i \in [1, w_2 + 1]$ and $\gamma_1, \ldots, \gamma_n \leftarrow_R \mathbb{Z}_N$. Using non-degeneracy property of dual system groups, we can write $\hat{g}_{0,0}$ and $\hat{g}_{i,0}$ as g^{δ} and g^{δ_i} respectively, for $i \in [1, w_2]$, where $\delta, \delta_1, \ldots, \delta_{w_2} \leftarrow_R \mathbb{Z}_N$. Therefore, we have

$$\hat{\mathrm{CT}}_\ell = g^{\zeta_\ell \delta + \sum_{i \in [1,w_2]} \eta_{\ell,i} \delta_i + \sum_{j \in [1,n]} \theta_{\ell,j} \delta \gamma_j + \sum_{i \in [1,w_2], j \in [1,n]} \vartheta_{\ell,i,j} \delta_i \gamma_j} \cdot \prod_{j \in [1,n]} \hat{g}_{0,j}^{\theta_{\ell,j}} \cdot \prod_{i \in [1,w_2], j \in [1,n]} \hat{g}_{i,j}^{\vartheta_{\ell,i,j}} \quad (19)$$

Shifting our focus to the key, we know that its tth component is given by

$$\hat{\mathrm{SK}}_t = \prod_{i' \in [1,m_2]} \hat{h}_{i',0}^{\upsilon_{t,i'}} \cdot \prod_{i' \in [1,m_2], j \in [1,n]} (\hat{h}_{i',j} \cdot \hat{h}_{i',0}^{\gamma_j} \cdot \tilde{h}^{z_{i',j}})^{\phi_{t,i',j}},$$

for $t \in [1, m_1]$, where $(\hat{h}_{i',0}, \ldots, \hat{h}_{i',n}) \leftarrow \overline{\mathsf{SampH}}(\mathrm{PP}, \mathrm{SP})$ and $(z_{i',1}, \ldots, z_{i',n}) \leftarrow \mathsf{Samp}(i', x, y, N)$ for $i' \in [1, m_2]$. Using non-degeneracy once again, we can write $\hat{h}_{i',0}$ as $\tilde{h}^{\omega_{i'}}$ for an $\omega_{i'} \leftarrow_R \mathbb{Z}_N$. Hence,

$$\begin{aligned} \hat{\mathrm{SK}}_t &= \tilde{h}^{\sum_{i' \in [1,m_2]} [\upsilon_{t,i'} \omega_{i'} + \sum_{j \in [1,n]} (\phi_{t,i',j} \omega_{i'} \gamma_j + \phi_{t,i',j} z_{i',j})]} \cdot \prod_{i' \in [1,m_2], j \in [1,n]} \hat{h}_{i',j}^{\phi_{t,i',j}} \\ &= \tilde{h}^{\sum_{i' \in [1,m_2]} [\upsilon_{t,i'} \omega_{i'} + \sum_{j \in [1,n]} (\phi_{t,i',j} \omega_{i'} (\gamma_j + z_{i',j}))]} \cdot \prod_{i' \in [1,m_2], j \in [1,n]} \hat{h}_{i',j}^{\phi_{t,i',j}}, \quad (20) \end{aligned}$$

since the distribution of $(z_{i',1}, \ldots, z_{i',n})$ is statistically close to $(\omega_{i'} z_{i',1}, \ldots, \omega_{i'} z_{i',n})$ (with high probability $\omega_{i'} \in \mathbb{Z}_N^*$) for all $i' \in [1, m_2]$.

Now, observe the superscripts of g and $\tilde{h}$ in (19) and (20) respectively (over $\ell \in [1, w_1]$ and $t \in [1, m_1]$). We know that $\delta, \delta_1, \ldots, \delta_{w_2}$, $\gamma_1, \ldots, \gamma_n$ and $\omega_1, \ldots, \omega_{m_2}$ are randomly chosen from $\mathbb{Z}_N$. Hence, we can use the second property (3) of relaxed perfect security associated with pair encoding schemes to add noise to the master secret key. (The dependencies between the elements of the samples drawn from $\overline{\mathsf{SampG}}$ and $\overline{\mathsf{SampH}}$ can be handled as in the previous proof.) Therefore, we have that the distribution

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, (\hat{\mathrm{CT}}_1, \ldots, \hat{\mathrm{CT}}_{w_1+1}), (\hat{\mathrm{SK}}_1, \ldots, \hat{\mathrm{SK}}_{m_1})\}$$

is statistically close to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, (\hat{\mathrm{CT}}_1, \ldots, \hat{\mathrm{CT}}_{w_1+1}), (\tilde{\mathrm{SK}}_1, \ldots, \tilde{\mathrm{SK}}_{m_1})\},$$

where

$$\tilde{\mathrm{SK}}_t := \tilde{h}^{\tau_t \beta + \sum_{i' \in [1,m_2]} [\upsilon_{t,i'} \omega_{i'} + \sum_{j \in [1,n]} (\phi_{t,i',j} \omega_{i'} (\gamma_j + z_{i',j}))]} \cdot \prod_{i' \in [1,m_2], j \in [1,n]} \hat{h}_{i',j}^{\phi_{t,i',j}},$$

for $t \in [1, m_1]$, and $\beta \leftarrow_R \mathbb{Z}_N$. Observe that the only difference between $\hat{\mathrm{SK}}_t$ and $\tilde{\mathrm{SK}}_t$ is that an extra $\tau_t \beta$ is begin added to the exponent of $\tilde{h}$ in the latter case. Hence, the key $(\tilde{\mathrm{SK}}_1, \ldots, \tilde{\mathrm{SK}}_{m_1})$ can be generated by providing $\tilde{h}^{\beta}$ as master secret key to $\overline{\mathsf{KeyGen}}$. Therefore, (18) is statistically close to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}} \cdot \hat{\mathbf{G}}', \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, \tilde{h}^{\beta}, y; (\hat{\mathbf{h}}_1 \cdot \hat{\mathbf{h}}_1' \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2} \cdot \hat{\mathbf{h}}_{m_2}' \cdot \tilde{h}^{\mathbf{z}_{m_2}}))\}.$$

Using parameter-hiding once again, we can show that the above distribution is identical to

$$\{\mathrm{PP}, \mathrm{MSK}, \tilde{h}, \overline{\mathsf{Encrypt}}(\mathrm{PP}, x, 1; \hat{\mathbf{G}}, \mathrm{MSK}), \overline{\mathsf{KeyGen}}(\mathrm{PP}, \tilde{h}^{\beta}, y; (\hat{\mathbf{h}}_1 \cdot \tilde{h}^{\mathbf{z}_1}, \ldots, \hat{\mathbf{h}}_{m_2} \cdot \tilde{h}^{\mathbf{z}_{m_2}}))\},$$

which completes the proof. □

Lemma E.5. $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}} \cong \mathsf{Hyb}_3$.

Proof. The only difference between the hybrids $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}}$ and Hyb_3 is that in the former case the message m_b is encrypted, while in the latter case a random message is encrypted; all the keys as well as the ciphertext in both the cases are in the semi-functional space. The following line of argument is very similar to the one in [CW14] for the corresponding lemma.

We can assume that MSK and $\overline{\text{MSK}}$ are sampled as follows: first pick $\overline{\text{MSK}} \leftarrow_R \mathbb{H}$ and then set $\text{MSK} := \overline{\text{MSK}} \cdot \tilde{h}^{\beta}$, where $\beta \leftarrow_R \mathbb{Z}_N$. Observe that

$$\mu(\text{MSK}) = \mu(\overline{\text{MSK}} \cdot \tilde{h}^{\beta}) = \mu(\overline{\text{MSK}}) \cdot \mu(\tilde{h})^{\beta} = \mu(\overline{\text{MSK}}) \tag{21}$$

due to the linearity of μ and the orthogonality property ($\mu(\tilde{h}) = 1$). Further, for any public parameters PP and coin tosses σ,

$$e(\mathsf{SampG}_0(\text{PP}; \sigma), \tilde{h}) = \mathsf{SampGT}(\mu(\tilde{h}); \sigma) = \mathsf{SampGT}(1; \sigma) = e(\mathsf{SampG}_0(\text{PP}; \sigma), 1) = 1 \tag{22}$$

due to the projective and orthogonality properties.

We now show that the view of any adversary $\mathcal{A}$ in both the hybrids can be simulated given PP and $\mathbf{G} \cdot \hat{\mathbf{G}}$ only. First pick $\overline{\text{MSK}}$ and MSK as described above. Output $(\text{PP}, \mu(\overline{\text{MSK}}))$ as the master public key MPK; using (21), this is identically distributed to $(\text{PP}, \mu(\text{MSK}))$. When $\mathcal{A}$ issues a key query y, respond with $\overline{\mathsf{KeyGen}}(\text{PP}, \overline{\text{MSK}}, y; (\mathbf{h}_1, \ldots, \mathbf{h}_{m_2}))$, where $\mathbf{h}_1, \ldots, \mathbf{h}_{m_2} \leftarrow \mathsf{SampH}(\text{PP})$. When $\mathcal{A}$ sends a pair of messages (m_0, m_1) and an x, where $m_0, m_1 \in \mathbb{G}_T$, output $\text{CT} := \overline{\mathsf{Encrypt}}(\text{PP}, x, m_b; \mathbf{G} \cdot \hat{\mathbf{G}}, \text{MSK})$. It is clear that the view of $\mathcal{A}$ in this experiment is identically distributed to its view in $\mathsf{Hyb}_{2,\xi,9,m_{2,\xi}}$. In order to prove that this view is also identically distributed to the view in Hyb_3, we only need to show that CT is the encryption of a random message.

We know that CT has $w_1 + 1$ components. The first w_1 components depend on PP, x and $\mathbf{G} \cdot \hat{\mathbf{G}}$, while the last one, $\text{CT}_{w_1+1} := m_b \cdot e(g_{0,0} \cdot \hat{g}_{0,0}, \text{MSK})$, depends on m_b and MSK (see Section 5). Now,

$$\begin{aligned}
e(g_{0,0} \cdot \hat{g}_{0,0}, \text{MSK}) &= e(g_{0,0} \cdot \hat{g}_{0,0}, \overline{\text{MSK}} \cdot \tilde{h}^{\beta}) \\
&= e(g_{0,0} \cdot \hat{g}_{0,0}, \overline{\text{MSK}}) \cdot e(\hat{g}_{0,0}, \tilde{h}^{\beta}) \cdot e(g_{0,0}, \tilde{h}^{\beta}) \\
&= e(g_{0,0} \cdot \hat{g}_{0,0}, \overline{\text{MSK}}) \cdot e(\hat{g}_{0,0}, \tilde{h}^{\beta}) && \text{(due to (22))} \\
&= e(g_{0,0} \cdot \hat{g}_{0,0}, \overline{\text{MSK}}) \cdot e(\hat{g}_{0,0}, \tilde{h})^{\beta}.
\end{aligned}$$

Observe that MPK, the keys and other parts of the ciphertext do not depend on β, which is chosen uniformly from $\mathbb{Z}_N$. Therefore, $e(\hat{g}_{0,0}, \tilde{h})^{\beta}$ is uniformly distributed over $\mathbb{G}_T$ from the non-degeneracy property. This implies that CT is identically distributed to the encryption of a random message. □

F Ciphertext-Policy ABE

Correctness: If (A, π) accepts S, then we know that there exists constants $\varepsilon_1, \ldots, \varepsilon_{n_1} \in \mathbb{Z}_N$ such that $\sum_{i \in \Upsilon} \varepsilon_i \sum_{j \in [1,n_2]} a_{i,j} v_j = v_1 = \alpha$, where $\Upsilon = \{i \mid i \in [1, n_1], \pi(i) \in S\}$. Below we show how to combine the polynomials generated by EncC and EncK using $\{\varepsilon_i\}_{i \in \Upsilon}$ in order to recover αs (this

implicitly defines the output of $\mathsf{Pair}((A,\pi),S,N)$):

$$c_2\left(\sum_{i\in\Upsilon}\varepsilon_i k_{1,i}\right) - c_1\left[\sum_{i\in\Upsilon}\varepsilon_i\left(\sum_{j\in[1,n_2]}a_{i,j}k_{2,i,j} \quad + \quad \sum_{\substack{\ell\in[1,n_1],\ell\neq i\\ j\in[1,n_2]}}a_{\ell,j}k_{3,i,\ell,j}\right.\right.$$
$$\left.\left. + \quad k_{4,i,\pi(i)} \quad + \quad \sum_{\substack{\ell\in[1,n_1],\ell\neq i\\ t\in[0,T]}}\pi(\ell)^t k_{5,i,\ell,t}\right)\right]$$

$$= \sum_{i\in\Upsilon}\varepsilon_i\left[c_2k_{1,i} - c_1\left(\sum_{j\in[1,n_2]}a_{i,j}k_{2,i,j} \quad + \quad \sum_{\substack{\ell\in[1,n_1],\ell\neq i\\ j\in[1,n_2]}}a_{\ell,j}k_{3,i,\ell,j}\right.\right.$$
$$\left.\left. + \quad k_{4,i,\pi(i)} \quad + \quad \sum_{\substack{\ell\in[1,n_1],\ell\neq i\\ t\in[0,T]}}\pi(\ell)^t k_{5,i,\ell,t}\right)\right]$$

$$= \sum_{i\in\Upsilon}\varepsilon_i\left(sr_i\sum_{\substack{i\in[1,n_1]\\ j\in[1,n_2]}}a_{i,j}b_{i,j} \quad + \quad sr_i\sum_{\substack{i\in[1,n_1]\\ t\in[0,T]}}\pi(i)^t b'_{i,t}\right.$$
$$- \quad sr_i\sum_{j\in[1,n_2]}a_{i,j}b_{i,j} \quad + \quad s\sum_{j\in[1,n_2]}a_{i,j}v_j \quad - \quad sr_i\sum_{\substack{\ell\in[1,n_1],\ell\neq i\\ j\in[1,n_2]}}a_{\ell,j}b_{\ell,j}$$
$$\left. - \quad sr_i\sum_{t\in[0,T]}\pi(i)^t b'_{i,t} \quad - \quad sr_i\sum_{\substack{\ell\in[1,n_1],\ell\neq i\\ t\in[0,T]}}\pi(\ell)^t b'_{\ell,t}\right)$$

$$= s\sum_{i\in\Upsilon}\varepsilon_i\sum_{j\in[1,n_2]}a_{i,j}v_j \quad = \quad \alpha s.$$