Adaptive partitioning

Dennis Hofheinz*

January 22, 2017

Abstract

We present a new strategy for partitioning proofs, and use it to obtain new tightly secure encryption schemes. Specifically, we provide the following two conceptual contributions:

- A new strategy for tight security reductions that leads to compact public keys and ciphertexts.
- A relaxed definition of non-interactive proof systems for non-linear ("OR-type") languages. Our definition is strong enough to act as a central tool in our new strategy to obtain tight security, and is achievable both in pairing-friendly and DCR groups.

We apply these concepts in a generic construction of a tightly secure public-key encryption scheme. When instantiated in different concrete settings, we obtain the following:

- A public-key encryption scheme whose chosen-ciphertext security can be tightly reduced to the DLIN assumption in a pairing-friendly group. Ciphertexts, public keys, and system parameters contain 6, 24, and 2 group elements, respectively. This improves heavily upon a recent scheme of Gay et al. (Eurocrypt 2016) in terms of public key size, at the cost of using a symmetric pairing.
- The first public-key encryption scheme that is tightly chosen-ciphertext secure under the DCR assumption. While the scheme is not very practical (ciphertexts carry 28 group elements), it enjoys constant-size parameters, public keys, and ciphertexts.

*Karlsruhe Institute of Technology, Dennis.Hofheinz@kit.edu. Supported by DFG grants HO 4534/4-1 and HO 4534/2-2.

Contents

1 Introduction

Tight security. Ideally, the only way to attack a cryptographic scheme S should be to solve a well-investigated, presumably hard computational problem P (such as factoring large integers). In fact, most existing constructions of cryptographic schemes provide such security guarantees, by exhibiting a security reduction. A reduction shows that any attack that breaks the scheme with some probability ε_S implies a problem solver that succeeds with probability ε_P. Of course, we would like ε_P to be as large as possible, depending on ε_S.

Specifically, we could call the quotient $\ell := \varepsilon_S/\varepsilon_P$ the *security loss* of a reduction.[1] A small value of ℓ is desirable, since it indicates a tight coupling of the security of the scheme to the hardness of the computational problem. It is also desirable that ℓ does not depend, e.g., on the number of considered instances of the scheme. Namely, when ℓ is linear in the number of instances, the scheme's security guarantees might vanish quickly in large settings. This can be a problem when being forced to choose concrete key sizes for schemes in settings whose size is not even known at setup time.

Hence, let us call a security reduction *tight* if its security loss ℓ only depends on a global security parameter (but not, e.g., on the number of considered instances, or the number of usages). Most existing cryptographic reductions are not tight. Specifically, it appears to be a nontrivial problem to construct tightly secure public-key primitives, such as public-key encryption, or digital signature schemes. (A high-level explanation of the arising difficulties can be found in [17].)

Existing work on tight security. The importance of a tight security reduction was already pointed out in 2000 by Bellare, Boldyreva, and Micali [4]. However, the first chosen-ciphertext secure (CCA secure) public-key encryption (PKE) scheme with a tight security reduction from a standard assumption was only proposed in 2012, by Hofheinz and Jager [17]. Their scheme is rather inefficient, however, with several hundred group elements in the ciphertext. A number of more efficient schemes were then proposed in [2, 7, 5, 25, 20, 26, 3, 14, 16, 12]. In particular, Chen and Wee [7] introduced a very useful partitioning strategy to conduct tight security reductions. Their strategy leads to very compact ciphertexts (of as few as 3 group elements [12], plus the message size), but also to large public keys. We will describe their strategy in more detail later, when explaining our techniques. Conversely, Hofheinz [16] presented a different partitioning strategy that leads to compact public keys, but larger ciphertexts (of 60 group elements). We give an overview over existing tightly secure PKE schemes (and some state-of-the-art schemes that are not known to be tightly secure for reference) in Fig. 1.

Our contribution. In this work, we propose a new strategy to obtain tightly secure encryption schemes. This strategy leads to new tightly secure PKE schemes with simultaneously compact public keys and compact ciphertexts (cf. Fig. 1). In particular, our technique yields a practical pairing-based PKE scheme that compares well even with the recent tightly secure PKE scheme of Gay, Hofheinz, Kiltz, and Wee [12]. However, we should also note that our scheme relies on a symmetric pairing (unlike the scheme of [12], which can be instantiated even in DDH groups). Hence, the price we pay for a significantly smaller public key is that the scheme of [12] is clearly superior to ours in terms of computational efficiency. Besides, the use of a symmetric pairing might entail larger group sizes for comparable security.

Our technique also yields the first PKE scheme whose security can be tightly reduced to the Decisional Composite Residuosity (DCR [28]) assumption in groups of the form $\mathbb{Z}_{N^2}^*$ for RSA numbers $N = PQ$. To obtain the DCR instance of our scheme, we also introduce a new type of "OR-proofs" (i.e., a proof system to show disjunctions of simpler statements) in the DCR setting. We give more details on these proofs below.

[1] Technically, we also need to take into account the complexity of the attacks on S and P. However, for this exposition, let us simply assume that the complexity of these attacks is comparable.

Scheme	$\|pk\|$	$\|C\| - \|M\|$	sec. loss	assumption	pairing
CS98 [8]	3	3	$\mathbf{O}(q)$	DDH	no
KD04, HK07 [24, 18]	$k+1$	$k+1$	$\mathbf{O}(q)$	k-LIN ($k \geq 1$)	no
HJ12 [17]	$\mathbf{O}(1)$	$\mathbf{O}(\lambda)$	$\mathbf{O}(1)$	DLIN	yes
ADKNO13 [2]	$\mathbf{O}(1)$	$\mathbf{O}(\lambda)$	$\mathbf{O}(1)$	DLIN	yes
HKS15 [20]	$\mathbf{O}(\lambda)$	2	$\mathbf{O}(\lambda)$	subgroup	yes
LPJY15 [25, 26]	$\mathbf{O}(\lambda)$	47	$\mathbf{O}(\lambda)$	DLIN	yes
AHY15 [3]	$\mathbf{O}(\lambda)$	12	$\mathbf{O}(\lambda)$	DLIN	yes
GCDCT16 [14]	$\mathbf{O}(\lambda)$	$6k+4$	$\mathbf{O}(\lambda)$	k-LIN ($k \geq 1$)	yes
H16 [16]	2	60	$\mathbf{O}(\lambda)$	SXDH	yes
GHKW16 [12]	$2k\lambda$	$3k$	$\mathbf{O}(\lambda)$	k-LIN ($k \geq 1$)	no
This work	$2k(k+5)$	$k+4$	$\mathbf{O}(\lambda)$	k-LIN ($k \geq 2$)	yes
CS02 [9]	9	2	$\mathbf{O}(q)$	DCR	—
CS03 [6]	3	2	$\mathbf{O}(q)$	DCR	—
This work	20	28	$\mathbf{O}(\lambda)$	DCR	—

Figure 1: Comparison of CCA-secure public-key encryption schemes. λ is the security parameter, and q is the number of challenge ciphertexts. The sizes $|pk|$ and $|C| - |M|$ of public key (excluding public parameters) and ciphertext overhead are counted in group elements. For the ciphertext overhead $|C|-|M|$, we do not count smaller components (like MACs) inherited from the used symmetric encryption scheme.

We remark that our main scheme is completely generic, and can be instantiated both with prime-order groups, and in the DCR setting. Only some of our building blocks (such as the "OR-proofs" mentioned above) require setting-dependent instantiations, which we give both in a prime-order, and in the DCR setting.

Hence, we view our main contribution as conceptual. Indeed, in terms of computational efficiency, our encryption schemes do not outperform existing (non-tightly secure) schemes, even when taking into account our tight security reduction in the choice of key sizes. Still, we believe that specializations of our technique can lead to schemes whose efficiency is at least on par with that of existing non-tightly secure schemes.

1.1 Technical overview

Technical goal. To explain our approach, consider the following security game with an adversary $\mathcal{A}$. First, $\mathcal{A}$ obtains a public key, and then may ask for many encryptions of arbitrary messages. Depending on a single bit b chosen by the security game, $\mathcal{A}$ then either always gets an encryption of the desired message, or an encryption of a random message. Also, $\mathcal{A}$ has access to a decryption oracle, and is finally supposed to guess b (i.e., whether the encrypted ciphertexts contain the desired, or random messages). If no efficient $\mathcal{A}$ can predict b non-negligibly better than guessing, the used PKE scheme is considered CCA secure in the multi-challenge setting. Note that regular (i.e., single-challenge) CCA security implies CCA security in the multi-challenge setting using a hybrid argument (over the challenge encryptions $\mathcal{A}$ gets), but this hybrid argument incurs a large security loss. Hence, the difficulty in proving multi-challenge security is to randomize many challenge ciphertexts in as few steps as possible.

General paradigm. All of the mentioned works on tightly secure PKE follow a general paradigm. Namely, in these schemes, each ciphertext $C = (c, \pi)$ carries some kind of "consistency proof" π that the plaintext message encrypted in c is intact. What this concretely means varies in different schemes. For instance, in some works [17, 2, 25, 26, 16], π is explicit and proves knowledge of the plaintext *or* of a valid signature on c. In other works [7, 5, 20, 3, 14, 12], π is implicit, and proves knowledge of the plaintext *or* of a special authentication tag for that ciphertext. All of these works, however, use π to enable the security reduction to get leverage over the adversary $\mathcal{A}$, as follows. For instance, in the signature-based works

above, the security reduction will be able to produce proofs π for ciphertexts with unknown plaintexts (by proving knowledge of a signature), while an adversary can only construct proofs from which the plaintext can be extracted. This enables the security reduction to implement a decryption oracle, while being able to randomize plaintexts encrypted for $\mathcal{A}$.

Chen and Wee's approach. Chen and Wee [7] implement the above approach with an economic partitioning strategy (that in turn draws from an argument of Naor and Reingold [27]). Specifically, in their scheme, π implicitly proves knowledge of the plaintext *or* of a special tag T. Initially, T is constant, and committed to in the public key. In their security analysis, Chen and Wee introduce dependencies of T on the corresponding c. Specifically, in the i-th step of their analysis, they set $T = \mathbf{F}(\tau_{..i})$, where $\mathbf{F}$ is a random function, and $\tau_{..i}$ is the i-bit prefix of the hash τ of c. After a small number of such steps, T is a random value that is individual to each ciphertext. At this point, T is unpredictable for $\mathcal{A}$ on fresh ciphertexts, and hence $\mathcal{A}$'s decryption queries must prove knowledge of the respective plaintext. At the same time, the security game (which defines $\mathbf{F}$) can also prepare valid ciphertexts with unknown messages, and thus randomize all challenge ciphertexts at once.

We call the approach of Chen and Wee a partitioning strategy, since each hybrid step above proceeds as follows:

1. Partition the ciphertext space into two halves (in this case, according to the i-th bit of τ).
2. Change the definition of the "authentication tag" T for all ciphertexts from one half. (Keep the authentication tag for ciphertexts from the other half unchanged.)

In particular, the second step introduces an additional dependency of T on the bit τ_i. Most existing works use a partitioning strategy based on the individual bits of (the hash of) the ciphertext. An exception is the recent work [16], which implements a similar strategy based on an algebraic predicate of the ciphertext. This latter approach leads to shorter public keys, but requires relatively complex proofs π, and thus not only entails larger ciphertexts, but also requires a pairing.

Our approach. Here, we also follow the generic paradigm sketched above, but refine the partitioning strategy of Chen and Wee. Namely, instead of partitioning the ciphertext space statically (e.g., through the hash of c), we add a special (encrypted) bit to π that determines the half in which the corresponding ciphertext is supposed to be. In contrast to previous works, that bit is not always known, not even to the security reduction itself. This change has several consequences:

- The bit that determines the partitioning in each ciphertext is easily accessible with a suitable decryption key, and so leads to a simple consistency proof π (and thus small ciphertexts). (This is in contrast to the scheme from [16], which proves complex statements in π.)
- The partitioning bit can by changed dynamically in challenge ciphertexts in different steps of the proof. Hence, a single "bit slot" can be used to partition the ciphertext space in many different ways during the proof. Eventually, this leads to compact public keys, since only few statements (about this single bit slot) need to be proven. (This is in contrast to partitioning schemes in which one proof for each bit position is generated.)
- However, since also the adversary can dynamically determine the partitioning of his ciphertexts from decryption queries, the security analysis becomes more complicated. Specifically, the reduction must cope with a situation in which an adversary submits a ciphertext for which the partitioning bit is not known.

In particular the last consequence will require additional measures in our security analysis. Namely, we will in some cases need to accept several authentication tags T in $\mathcal{A}$'s decryption queries, simply because we do not know in which half of the partitioning the corresponding ciphertext is. In fact, we will not be able to force $\mathcal{A}$ to use "the right" authentication tag in his decryption queries. We will only be able to force $\mathcal{A}$ to use an authentication tag T from a previous challenge ciphertext (since all other tags are unpredictable to $\mathcal{A}$). Hence, in order to eventually exclude that $\mathcal{A}$ produces ciphertexts without a proof of knowledge of the corre-

sponding plaintext, we will need to work a bit more.

At this point, our main conceptual idea will be to introduce a dependency of T on a suitable value τ that is individual to each ciphertext. (While the construction in our scheme is slightly more complicated, one can think of τ as being simply the hash of the ciphertext.) Hence, in the first part of our analysis, we force $\mathcal{A}$ to reuse a tag T from a previous challenge ciphertext, while we tie this T to a ciphertext-unique value τ in the second part. When this is done, $\mathcal{A}$'s proofs π from decryption queries must prove knowledge of the encrypted plaintext message, *or* break the collision-resistance of the used hash function. Since the hash function will be assumed to be collision-resistant, $\mathcal{A}$ must prove knowledge of the respective plaintext in each decryption query. Hence, we can proceed with a proof of CCA security as in previous schemes.

Building blocks. To implement our strategy, we require a variety of building blocks. Specifically, like previous works, we require re-randomizable (chosen-plaintext-secure) encryption, and universal hash proof systems for linear languages. We also require tightly secure one-time signatures, for which we give the first construction in the DCR setting. However, apart from our new partitioning strategy, the main technical innovation from our work is the construction of a non-interactive proof system for disjunctions (of simpler statements) in the DCR setting.

Namely, our proof system allows to prove that, given two ciphertexts c_1, c_2, at least one of them decrypts to zero. (In fact, the syntactics are a little more complicated, and in particular, honest proofs can only be formulated when the *first* ciphertext decrypts to zero. However, proofs that *one* of the two ciphertexts decrypts to zero can always be simulated using a special trapdoor, and we have soundness even in the presence of such simulated proofs.)

Such a proof system for disjunctions already exists in pairing-friendly groups [1]. However, a construction without pairings is far from obvious. Intuitively, the reason is that the language of pairs (c_1, c_2) as above (with at least one c_i that encrypts zero) is not closed under addition (of the respective plaintexts). Hence, disjunctions as above do not correspond to linear languages, and most common constructions (e.g., for universal hash proof systems [9, 22]) do not apply. Our DCR-based construction thus is not linear, and relies on new techniques.

Concretely, our proof system can be viewed as a randomized variant of a universal hash proof system. Namely, depending on how many of the c_i do not encrypt zero, a valid proof reveals zero, one, or two linear equations about the secret verification key of our system. However, proofs in our system are randomized, and the revealed equations are also blinded with precisely one random value. Hence, up to one equation about the secret key is completely blinded. But as soon as both c_i encrypt nonzero values, a valid proof contains nontrivial information about the secret key. Thus, such proofs cannot be produced by an adversary who only sees proofs for valid statements (with at least one c_i that encrypts zero). Hence, soundness follows as with regular universal hash proof systems.

Acknowledgements. I would like to thank Antonio Faonio for pointing out a problem in the formulation of Definition 4.4, and Dingding Jia and Ryo Nishimaki for a careful proofreading. In particular, Dingding spotted a mistake in the description of honest key derivation. I am also indebted to Lin Lyu, who found a flaw in an earlier version of the DCR-based one-time signature scheme $\mathbf{OTS}_{\mathbf{DCR}}$, a gap in the proof of Lemma 6.3 (that in fact made Lemma 6.4 necessary), and many smaller mistakes in an earlier version in a very thorough proofreading. Finally, I would like to thank the reviewers for helpful comments concerning the presentation.

2 Preliminaries

Notation and conventions. For a group $\mathbb{G}$ of order $|\mathbb{G}|$, a group element $g \in \mathbb{G}$, and a vector $\mathbf{u} = (u_1, \ldots, u_n)^\top \in \mathbb{Z}_{|\mathbb{G}|}^n$, we write $g^{\mathbf{u}} := (g^{u_1}, \ldots, g^{u_n})^\top \in \mathbb{G}^n$. Similarly, we define $g^{\mathbf{M}} \in \mathbb{G}^{n\times m}$ for matrices $\mathbf{M} \in \mathbb{Z}_{|\mathbb{G}|}^{n\times m}$. For integers $x, N \in \mathbb{Z}$ with $N > 0$, we define $[x]_N := x \bmod N$, and $[x]^N$ to be the unique integer with $x = [x]_N + N \cdot [x]^N$. Furthermore, we define the "absolute

modular value" $|x|_N$ through

$$|x|_N := \begin{cases} [x]_N & \text{if } [x]_N < N/2 \\ [-x]_N & \text{if } [x]_N \geq N/2, \end{cases}$$

such that $0 \leq |x|_N \leq N/2$ in any case. Finally, we let $\left(\frac{x}{N}\right)$ denote the Jacobi symbol of x modulo N. For a bit $b \in \{0,1\}$, we denote with $\overline{b} = 1 - b$ the complement of b. For a bitstring $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, we denote with $x_{..i} = (x_1, \ldots, x_i)$ the i-bit prefix of x, and with $x_{i..} = (x_i, \ldots, x_n)$ the $(n-i+1)$-bit postfix of x. For random variables $X, Y \in \{0,1\}^*$, we let $\mathbf{SD}(X; Y)$ denote their statistical distance, and $H_\infty(X)$ the min-entropy of X.

Global public parameters. To simplify notation, we assume that all algorithms in this work (including adversaries) implicitly receive public parameters pp as input. In our case, these public parameters will contain the description of algebraic groups and related algorithms, and a collision-resistant and a universal hash function. We give more details on these parameters when we discuss the algebraic setting, collision-resistant hashing, and our key extractor (which uses the universal hash function).

Collision-resistant hashing. We require collision-resistant hashing, which we define now:

Definition 2.1 (Collision-resistant hashing)**.** *A hash function generator is a PPT algorithm* **CRHF** *that, on input 1^λ, outputs (the description of) an efficiently computable function $H : \{0,1\}^* \to \{0,1\}^{\ell_H}$. We say that* **CRHF** *outputs collision-resistant hash functions H (or, slightly abusing notation, that* **CRHF** *is collision-resistant), if*

$$\mathrm{Adv}^{\mathrm{crhf}}_{\mathbf{CRHF},\mathcal{A}}(\lambda) = \Pr\left[x \neq x' \wedge H(x) = H(x') \;\middle|\; H \leftarrow \mathbf{CRHF}(1^\lambda),\ (x, x') \leftarrow \mathcal{A}(1^\lambda, H)\right]$$

is negligible for every PPT adversary $\mathcal{A}$.

We assume that the public parameters pp contain a function H sampled with a hash function generator **CRHF**.

Universal hashing, and randomness extraction. We also assume a family $\mathbf{UHF} = \mathbf{UHF}_\lambda$ of universal hash functions $h : \{0,1\}^* \to \{0,1\}^\lambda$. Since universal hash functions are good randomness extractors, we in particular have that for any random variable X with min-entropy $H_\infty(X) \geq 3\lambda$,

$$\mathbf{SD}\big((h,\, h(X));\, (h, R)\big) \leq 1/2^\lambda,$$

where $h \in \mathbf{UHF}_\lambda$ and $R \in \{0,1\}^\lambda$ are uniformly chosen.

Key encapsulation mechanisms, and multi-challenge security. A key encapsulation mechanism (KEM) **KEM** consists of PPT algorithms $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$. Key generation $\mathbf{Gen}(1^\lambda)$ outputs a public key pk and a secret key sk. Encapsulation $\mathbf{Enc}(pk)$ takes a public key pk, and outputs a ciphertext c, and a session key K. Decapsulation $\mathbf{Dec}(sk, c)$ takes a secret key sk, and a ciphertext c, and outputs a session key K. For correctness, we require that for all (pk, sk) in the range of $\mathbf{Gen}(1^\lambda)$, and all (c, K) in the range of $\mathbf{Enc}(pk)$, we always have $\mathbf{Dec}(sk, c) = K$. Security is defined as follows:

Definition 2.2 (Multi-challenge ciphertext indistinguishability)**.** *Given a key encapsulation scheme* **KEM***, consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *$\mathcal{C}$ samples a keypair through $(pk, sk) \leftarrow \mathbf{Gen}(1^\lambda)$, and chooses a uniform bit $b \leftarrow \{0,1\}$.*
2. *$\mathcal{A}$ is invoked on input $(1^\lambda, pk)$, and with (many-time) access to the following oracles:*
   - *$\mathcal{O}_{\mathbf{enc}}()$ runs $(c, K) \leftarrow \mathbf{Enc}(pk)$, sets $K_0 = K$, samples a fresh $K_1 \leftarrow \{0,1\}^\lambda$, and returns (c, K_b).*
   - *$\mathcal{O}_{\mathbf{dec}}(c)$ returns $\perp$ if c is a previous output of $\mathcal{O}_{\mathbf{enc}}$. Otherwise, $\mathcal{O}_{\mathbf{dec}}$ returns $K \leftarrow \mathbf{Dec}(sk, c)$.*
3. *Finally, $\mathcal{A}$ outputs a bit b', and $\mathcal{C}$ outputs 1 iff $b = b'$.*

Let

$$\mathrm{Adv}^{\mathrm{mcca}}_{\mathbf{KEM},\mathcal{A}}(\lambda) \;=\; \Pr[\mathcal{C} \textit{ outputs } 1] - 1/2.$$

We say that **KEM** *has indistinguishable ciphertexts under chosen-ciphertext attacks in the multi-challenge setting (short: is IND-MCCA secure) iff* $\mathrm{Adv}^{\mathrm{mcca}}_{\mathbf{KEM},\mathcal{A}}(\lambda)$ *is negligible for all PPT* $\mathcal{A}$*.*

We note that secure KEM schemes imply secure PKE schemes [8], and that the corresponding security reduction is tight also in the multi-challenge setting. Hence, like [12], we will focus on obtaining an IND-MCCA secure KEM scheme in the following.

3 The generic algebraic setting

3.1 The generic setting

3.1.1 Groups and public parameters

In the following, let $\mathbb{G}$ be a group of order $|\mathbb{G}|$. We require that $|\mathbb{G}|$ is square-free, and only has prime factors larger than 2^λ. Furthermore, we assume two subgroups $\mathbb{G}_1, \mathbb{G}_2 \subseteq \mathbb{G}$ of order $|\mathbb{G}_1|$ and $|\mathbb{G}_2|$, respectively, and such that $\mathbb{G}_1 \cdot \mathbb{G}_2 = \{h_1 \cdot h_2 \mid h_1 \in \mathbb{G}_1, h_2 \in \mathbb{G}_2\} = \mathbb{G}$. Note that we neither require nor exclude that $|\mathbb{G}|$ (or $|\mathbb{G}_1|$ or $|\mathbb{G}_2|$) is prime, or that $\mathbb{G}_1 \cap \mathbb{G}_2$ is trivial.

We assume that the global public parameters pp include

- (descriptions of) $\mathbb{G}$, $\mathbb{G}_1$, and $\mathbb{G}_2$,
- fixed generators g of $\mathbb{G}$, g_1 of $\mathbb{G}_1$ and g_2 of $\mathbb{G}_2$,
- the group order $|\mathbb{G}_2|$ of $\mathbb{G}_2$,
- a positive integer $\ell_{\mathbf{B}}$, and a matrix $g_1^{\mathbf{B}}$, for $\mathbf{B} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}} \times \ell_{\mathbf{B}}}$.[2]

We stress that these parameters may depend on λ, and note that $|\mathbb{G}|$, $|\mathbb{G}_1|$, and $\mathbf{B}$ do not need to be public. However, we do require that there are efficient algorithms for the following tasks:

- performing the group operation in $\mathbb{G}$,
- sampling uniformly distributed $\mathbb{Z}_{|\mathbb{G}_1|}$-elements,
- recognizing $\mathbb{G}$ (i.e., deciding group membership in $\mathbb{G}$).

Since we assume $|\mathbb{G}_2|$ to be public, we also have algorithms for deciding membership in $\mathbb{G}_2$, and for uniformly sampling from $\mathbb{Z}_{|\mathbb{G}_2|}$ and $\mathbb{G}_2$, and thus also from $\mathbb{Z}_{|\mathbb{G}|}$ and $\mathbb{G}$.

3.1.2 Computational assumptions

In our generic setting, we will use an assumption that can be seen as a combination of the Extended Decisional Diffie-Hellman assumption from [15], and the Matrix Decisional Diffie-Hellman assumption from [10].

Definition 3.1 (Generalized DDH, combining [15, 10])**.** *We say that the Generalized Decisional Diffie-Hellman (GDDH) assumption holds in our setting if the following advantage is negligible for every PPT adversary* $\mathcal{A}$*, and for uniformly chosen* $\boldsymbol{\omega}, \mathbf{r} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}}}$*:*

$$\mathrm{Adv}^{\mathrm{gddh}}_{\mathbb{G},\mathcal{A}}(\lambda) \;=\; \frac{1}{2}\big(\Pr\left[\mathcal{A}(1^\lambda, g_1^{\boldsymbol{\omega}^\top \mathbf{B}}, g_1^{\mathbf{B}\mathbf{r}}, g_1^{\boldsymbol{\omega}^\top \mathbf{B}\mathbf{r}}) = 1\right] - \Pr\left[\mathcal{A}(1^\lambda, g_1^{\boldsymbol{\omega}^\top \mathbf{B}}, g_1^{\mathbf{B}\mathbf{r}}, g_1^{\boldsymbol{\omega}^\top \mathbf{B}\mathbf{r}} g_2) = 1\right]\big).$$

Besides GDDH, we will also assume that it is infeasible to find a nontrivial element $g_2^u \in \mathbb{G}_2$ that does not already generate $\mathbb{G}_2$:

Definition 3.2 ($\mathbb{G}_2$-factoring assumption)**.** *We say that the factoring* $\mathbb{G}_2$ *is hard in our setting if the following advantage is negligible for every PPT adversary* $\mathcal{A}$ *whose output* $(g_2^{u_1}, \ldots, g_2^{u_q}) \in \mathbb{G}_2^q$ *is always a vector of* $\mathbb{G}_2$*-elements:*

$$\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{A}}(\lambda) \;=\; \Pr\left[\exists i : \gcd(|\mathbb{G}_2|, u_i) \notin \{1, |\mathbb{G}_2|\} \;\middle|\; (g_2^{u_1}, \ldots, g_2^{u_q}) \leftarrow \mathcal{A}(1^\lambda)\right].$$

[2] How $\ell_{\mathbf{B}}$ and $\mathbf{B}$ are chosen depends in the concrete instance. In the prime-order setting, $\ell_{\mathbf{B}}$ and $\mathbf{B}$ determine what concrete computational problem is reduced to. Conversely, in the DCR setting, $\ell_{\mathbf{B}} = 1$, and $\mathbf{B} = 1$ is trivial.

3.1.3 Generalized ElGamal encryption

To simplify our notation, and to structure our presentation, we consider the following generalized variant of ElGamal:

Keypairs. Keypairs (epk, esk) are of the form $(epk, esk) = (g_1^{\omega^\top \mathbf{B}}, \omega)$ for $\omega \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$.

Encryption. To encrypt $u \in \mathbb{Z}_{|\mathbb{G}_2|}$ with random coins $\mathbf{r} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$, compute

$$\mathbf{E}_{epk}(u; \mathbf{r}) \;=\; \mathbf{c} \;=\; (\mathbf{c}_0, c_1) \;=\; (g_1^{\mathbf{B}\mathbf{r}}, g_1^{\omega^\top \mathbf{B}\mathbf{r}} g_2^u) \;\in \mathbb{G}^{\ell_\mathbf{B}} \times \mathbb{G}.$$

If we omit $\mathbf{r}$ and only write $\mathbf{E}_{epk}(u)$, then $\mathbf{r}$ is implicitly chosen uniformly from $\mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$.

Decryption. A ciphertext $\mathbf{c} = (\mathbf{c}_0, c_1) = (g^\gamma, g^\delta)$ is decrypted to

$$\mathbf{D}_{esk}(\mathbf{c}) \;=\; g^{\delta - \omega^\top \gamma} \;\in \mathbb{G}.$$

Note that we encrypt exponents, while decryption only retrieves the respective group element.

It will also be useful to generalize this encryption to vectors of plaintexts with reused random coins: for $\mathbf{pk} = (epk_1, \ldots, epk_n)$ and $\mathbf{sk} = (esk_1, \ldots, esk_n)$ with $(epk_i, esk_i) = (g_1^{\omega_i^\top \mathbf{B}}, \omega_i)$, and $\mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{Z}_{|\mathbb{G}_2|}^n$, let

$$\begin{aligned} \mathbf{E}_{\mathbf{pk}}(\mathbf{u}; \mathbf{r}) \;&=\; (\mathbf{c}_0, (c_1, \ldots, c_n)) \;=\; (g_1^{\mathbf{B}\mathbf{r}}, (g_1^{\omega_1^\top \mathbf{B}\mathbf{r}} g_2^{u_1}, \ldots, g_1^{\omega_n^\top \mathbf{B}\mathbf{r}} g_2^{u_n})) \;\in \mathbb{G}^{\ell_\mathbf{B}} \times \mathbb{G}^n \\ \mathbf{D}_{\mathbf{sk}}(\mathbf{c}) \;&=\; (g^{\delta_1 - \omega_1^\top \gamma}, \ldots, g^{\delta_n - \omega_n^\top \gamma}) \;\in \mathbb{G}^n \quad \text{for} \quad \mathbf{c} = (g^\gamma, (g^{\delta_1}, \ldots, g^{\delta_n})). \end{aligned}$$

When no confusion is possible, we may write $(\mathbf{c}_0, c_1, \ldots, c_n)$ instead of the more cumbersome $(\mathbf{c}_0, (c_1, \ldots, c_n))$. Sometimes, it will also be convenient to write $\mathbf{\Omega} = (\omega_1 \| \ldots \| \omega_n) \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B} \times n}$, such that $\mathbf{pk} = g_1^{\mathbf{\Omega}^\top \mathbf{B}}$ and

$$\begin{aligned} \mathbf{E}_{\mathbf{pk}}(\mathbf{u}; \mathbf{r}) \;&=\; (g_1^{\mathbf{B}\mathbf{r}}, g_1^{\mathbf{\Omega}^\top \mathbf{B}\mathbf{r}} g_2^{\mathbf{u}}) \\ \mathbf{D}_{\mathbf{sk}}(\mathbf{c}) \;&=\; g^{\gamma - \mathbf{\Omega}^\top \delta} \quad \text{for} \quad \mathbf{c} = (g^\gamma, g^\delta) \in \mathbb{G}^{\ell_\mathbf{B}} \times \mathbb{G}^n. \end{aligned}$$

While this variant of ElGamal encryption will mainly be a notational tool, it is also a very simple tightly (chosen-plaintext) secure encryption scheme:

Definition 3.3 (IND-MCCPA security game for $(\mathbf{E}, \mathbf{D})$)**.** *Consider the following game (which we call the IND-MCCPA security game, for "indistinguishability against multiple (partial) corruptions and chosen-plaintext attacks") between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *$\mathcal{A}(1^\lambda)$ picks $n \in \mathbb{N}$, and an index $i^* \in \{1, \ldots, n\}$.*
2. *$\mathcal{C}$ samples $b \in \{0, 1\}$, and $\omega_1, \ldots, \omega_n \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$, and sets $(epk_i, esk_i) = (g_1^{\omega_i^\top \mathbf{B}}, \omega_i)$, and $\mathbf{pk} = (epk_1, \ldots, epk_n)$ and $\mathbf{sk} = (esk_1, \ldots, esk_n)$.*
3. *Next, $\mathcal{A}$ is run on input $(epk_i)_{i=1}^{\ell_\mathbf{B}}$, $(esk_i)_{i \neq i^*}$, and with (many-time) access to the following oracle:*
   - *$\mathcal{O}_{\mathbf{enc}}(\mathbf{u}^{(0)}, \mathbf{u}^{(1)})$, for $\mathbf{u}^{(j)} = (u_1^{(j)}, \ldots, u_n^{(j)}) \in \mathbb{Z}_{|\mathbb{G}_2|}^n$ ($j \in \{0, 1\}$), first checks that $u_i^{(0)} = u_i^{(1)}$ for all $i \neq i^*$, and returns $\perp$ if not. Then, $\mathcal{O}_{\mathbf{enc}}$ computes and returns $\mathbf{c} = \mathbf{E}_{\mathbf{pk}}(\mathbf{u}^{(b)})$.*
4. *If $\mathcal{A}$ terminates with output b', then $\mathcal{C}$ outputs 1 iff $b = b'$.*

Let

$$\mathrm{Adv}_{\mathbb{G},\mathcal{A}}^{\mathrm{mccpa}}(\lambda) = \Pr[\mathcal{C} \text{ outputs } 1] - 1/2.$$

Lemma 3.4 (Tight security of $(\mathbf{E}, \mathbf{D})$)**.** *For every $\mathcal{A}$, there exists an adversary $\mathcal{B}$ (of essentially the same complexity as the IND-MCCPA game with $\mathcal{A}$) for which*

$$\mathrm{Adv}_{\mathbb{G},\mathcal{B}}^{\mathrm{gddh}}(\lambda) \;=\; \mathrm{Adv}_{\mathbb{G},\mathcal{A}}^{\mathrm{mccpa}}(\lambda). \tag{1}$$

Proof. $\mathcal{B}$ gets $epk^* = g_1^{\omega^{*\top}\mathbf{B}}$ and $\mathbf{c}^* = (\mathbf{c}_0^*, c_1^*) = (g_1^{\mathbf{B}\mathbf{r}^*}, g_1^{\omega^{*\top}\mathbf{B}\mathbf{r}^*} g_2^b)$ (for unknown $b \in \{0,1\}$) as input. Now $\mathcal{B}$ first runs $\mathcal{A}$ to obtain n and i^*. Then, $\mathcal{B}$ generates public and secret keys as follows:

- For $i \neq i^*$, $\mathcal{B}$ samples $\boldsymbol{\omega}_i \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}}}$, and sets $(epk_i, esk_i) = (g^{\boldsymbol{\omega}_i^\top \mathbf{B}}, \boldsymbol{\omega}_i)$.
- $\mathcal{B}$ sets $epk_{i^*} = g_1^{\omega^{*\top}\mathbf{B}}$, and thus implicitly defines $esk_{i^*} = \boldsymbol{\omega}_{i^*} = \boldsymbol{\omega}^*$.

Then, $\mathcal{B}$ runs $\mathcal{A}$, on input $\mathbf{pk} = (epk_i)_i$ and $(esk_i)_{i \neq i^*}$, and implements oracle $\mathcal{O}_{\mathbf{enc}}$ as follows:

- Upon an $\mathcal{O}_{\mathbf{enc}}(\mathbf{u}^{(0)}, \mathbf{u}^{(1)})$ query with $u_i^{(0)} = u_i^{(1)}$ for $i \neq i^*$, $\mathcal{B}$ first samples a fresh $\mathbf{r}' \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}}}$, implicitly defines $\mathbf{r} = (u_{i^*}^{(1)} - u_{i^*}^{(0)})\mathbf{r}^* + \mathbf{r}'$, and sets up

$$\begin{aligned}
\mathbf{c}_0 &= g_1^{(u_{i^*}^{(1)} - u_{i^*}^{(0)})\mathbf{B}\mathbf{r}^* + \mathbf{B}\mathbf{r}'} = g_1^{\mathbf{B}((u_{i^*}^{(1)} - u_{i^*}^{(0)})\mathbf{r}^* + \mathbf{r}')} = g_1^{\mathbf{B}\mathbf{r}} \\
c_i &= g_1^{(u_{i^*}^{(1)} - u_{i^*}^{(0)})\boldsymbol{\omega}_i^\top \mathbf{B}\mathbf{r}^*} g_1^{\boldsymbol{\omega}_i^\top \mathbf{B}\mathbf{r}'} g_2^{u_i^{(0)}} = g_1^{\boldsymbol{\omega}_i^\top \mathbf{B}\mathbf{r}} g_2^{u_i^{(0)}} \qquad \text{for } i \neq i^* \\
c_i &= g_1^{(u_{i^*}^{(1)} - u_{i^*}^{(0)})\boldsymbol{\omega}^{*\top}\mathbf{B}\mathbf{r}^* + \boldsymbol{\omega}^{*\top}\mathbf{B}\mathbf{r}'} g_2^{(u_{i^*}^{(1)} - u_{i^*}^{(0)})\cdot b + u_{i^*}^{(0)}} = g_1^{\boldsymbol{\omega}_{i^*}^\top \mathbf{B}\mathbf{r}} g_2^{u_{i^*}^{(b)}}
\end{aligned}$$

  For the resulting $\mathbf{c} = (\mathbf{c}_0, c_1, \ldots, c_n)$, we have that $\mathbf{c} = \mathbf{E}_{\mathbf{pk}}(\mathbf{u}^{(b)}; \mathbf{r})$ for (independently and uniformly distributed) random coins $\mathbf{r} = (u_{i^*}^{(1)} - u_{i^*}^{(0)})\mathbf{r}^* + \mathbf{r}'$. Hence, $\mathcal{O}_{\mathbf{enc}}$ returns $\mathbf{c}$.

Finally, $\mathcal{B}$ relays any guess b' from $\mathcal{A}$ as its own output.

Observe that $\mathcal{B}$ perfectly simulates the game from Lemma 3.4 (with the same challenge bit b). We obtain (1). □

3.2 The prime-order setting

The groups. We consider two concrete instantiations of our generic setting. The first is a prime-order setting, in which $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ has prime order $|\mathbb{G}| = |\mathbb{G}_1| = |\mathbb{G}_2|$. In these cases, we assume that $|\mathbb{G}| > 2^\lambda$ is public, and hence most syntactic requirements from Section 3.1.1 are trivially met. However, we will additionally need to assume that membership in $\mathbb{G}$ is efficiently decidable. We have numerous candidates for such groups (including, e.g., subgroups of $\mathbb{Z}_p^*$, or elliptic curves). In such groups, plausible assumptions include the Decisional Diffie-Hellman (DDH) assumption, the k-Linear (k-LIN) assumption [29, 18], or a whole class of assumptions called Matrix-DDH assumptions [10].

Hardness of the GDDH and factoring problems. All of the mentioned assumptions imply our GDDH assumption for suitable $\ell_{\mathbf{B}}$ and $\mathbf{B}$. For instance, GDDH with $\ell_{\mathbf{B}} = 1$ and uniform $\mathbf{B}$ is nothing but a reformulation of the DDH assumption. More generally, GDDH with uniform $\mathbf{B}$ is actually the so-called $\mathcal{U}_{\ell_{\mathbf{B}}}$-MDDH assumption. In particular, this means that the k-LIN assumption implies GDDH with $\ell_{\mathbf{B}} = k$ and uniform $\mathbf{B}$ (see [10]). Additionally, we note that the $\mathbb{G}_2$-factoring assumption we make is trivially satisfied in prime-order settings (since $\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{A}}(\lambda) = 0$ for all $\mathcal{A}$ if $|\mathbb{G}_2| = |\mathbb{G}|$ is prime).

Pairing-friendly groups. In Section 5.4.1, we also exhibit a building block in the prime-order setting that uses a symmetric pairing $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ (for a suitable target group $\mathbb{G}_T$). Also for such pairing-friendly groups, we have a variety of candidates in case $\ell_{\mathbf{B}} \geq 2$. (Unfortunately, for $\ell_{\mathbf{B}} = 1$, a symmetric pairing can be used to trivially break the GDDH assumption.)

3.3 The DCR setting

The public parameters. The second setting we consider is compatible with the Decisional Composite Residuosity (DCR) assumption [28]. In this case, the global public parameters include an integer $N = PQ$, for distinct safe primes P, Q (i.e., such that $P = 2P' + 1$ and

$Q = 2Q' + 1$ for prime $P', Q' > 2^\lambda$).[3] We also assume that P, Q, P', Q' are pairwise different, and that $\gcd(P + Q - 1, N) = 1$ (the latter of which ensures that N is invertible modulo $\varphi(N) = (P-1)(Q-1) = 4P'Q'$).

We implicitly set $\ell_{\mathbf{B}} = 1$, and the matrix $\mathbf{B} \in \mathbb{Z}_{|\mathbb{G}_1|\times|\mathbb{G}_1|}$ from Section 3.1.1 to be trivial (i.e., the identity matrix). Hence, neither $\ell_{\mathbf{B}}$ nor $g_1^{\mathbf{B}}$ will have to be included in the parameters. However, we also include a generator g_1 of $\mathbb{G}_1$ in the public parameters, chosen as described below.

The groups. We now define the groups $\mathbb{G}$, $\mathbb{G}_1$, and $\mathbb{G}_2$. Since $\mathbb{G}$ should only have large prime factors, we should avoid setting $\mathbb{G} = \mathbb{Z}^*_{N^2}$. Instead, we could set $\mathbb{G}_1$ and $\mathbb{G}_2$ to be the subgroups of order $\varphi(N)/4$ and N, respectively, and then $\mathbb{G} = \mathbb{G}_1 \cdot \mathbb{G}_2$. However, in this case, membership in $\mathbb{G}$ would not be efficiently decidable in an obvious way. So here, we define our groups in a slightly more complex way, following the approach of signed quadratic residues [13, 11, 19].

Equipped with the notation $|x|_N$ and $\left(\frac{x}{N}\right)$ from Section 2, we set

$$\begin{aligned}
\mathbb{G}_1 &= \left\{ |x^N|_{N^2} \;\middle|\; x \in \mathbb{Z}^*_{N^2}, \left(\frac{x^N}{N}\right) = 1 \right\} \subseteq \mathbb{Z}^*_{N^2} \\
\mathbb{G}_2 &= \left\{ |(1+N)^e|_{N^2} \;\middle|\; e \in \mathbb{Z}_N \right\} \subseteq \mathbb{Z}^*_{N^2} \\
\mathbb{G} &= \left\{ |y|_{N^2} \;\middle|\; y \in \mathbb{Z}^*_{N^2}, \left(\frac{y}{N}\right) = 1 \right\}.
\end{aligned}$$

These sets are groups, when equipped with the group operation $a \cdot b = |a \cdot b|_{N^2}$. Indeed, since $P, Q = 3 \bmod 4$, we have $\left(\frac{-1}{N}\right) = 1$, and thus $\left(\frac{|y_1 y_2|_{N^2}}{N}\right) = \left(\frac{y_1 y_2}{N}\right) = 1$ for $\left(\frac{y_1}{N}\right) = \left(\frac{y_2}{N}\right) = 1$. Hence, $\mathbb{G}_1$ and $\mathbb{G}$ are closed under group operation. It is then straightforward to check that $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}$ are groups.

A canonical generator g_2 of $\mathbb{G}_2$ is $|1+N|_{N^2}$, and a generator g_1 of $\mathbb{G}_1$ (to be included in the public parameters) can be randomly chosen as $|x^N|_{N^2}$ for a uniform $x \in \mathbb{Z}_{N^2}$.

Properties of the groups. We claim that $|\mathbb{G}_1| = \varphi(N)/4$. Indeed, we have that

$$\left|\left\{ |x^N|_N \;\middle|\; x \in \mathbb{Z}^*_{N^2} \right\}\right| = \left|\left\{ |x^N|_{N^2} \;\middle|\; x \in \mathbb{Z}^*_{N^2} \right\}\right| = \varphi(N)/2.$$

In other words, $|x^N|_N$ uniquely determines $|x^N|_{N^2}$. Furthermore, since N is invertible modulo $\varphi(N)$, the map $f : \mathbb{Z}^*_{N^2} \to \mathbb{Z}^*_N$ with $f(x) = x^N \bmod N$ is surjective. Hence, the set of all $|x^N|_N$ with $\left(\frac{x^N}{N}\right) = 1$ has cardinality $\varphi(N)/4$ (cf. [19, Lemma 1]). Using that $|x^N|_N$ fixes $|x^N|_{N^2}$, we obtain $|\mathbb{G}_1| = \varphi(N)/4$. Moreover, for $e \in \mathbb{Z}_N$, we can write $|(1+N)^e|_{N^2} = |1 + eN|_{N^2} = e/|e| + |e|_N N$, and thus $|\mathbb{G}_2| = N$. Finally, we have $\mathbb{G} = \mathbb{G}_1 \cdot \mathbb{G}_2$, since every $|y|_{N^2} \in \mathbb{G}$ can be written as $|y|_{N^2} = |x^N(1+N)^e|_{N^2}$ with $\left(\frac{x^N}{N}\right) = 1$. Hence, since $|\mathbb{G}_1| = \varphi(N)/4 = P'Q'$ and $|\mathbb{G}_2| = N = PQ$ are coprime, $|\mathbb{G}| = |\mathbb{G}_1| \cdot |\mathbb{G}_2| = N \cdot \varphi(N)/4$ is square-free.

We also note that the discrete logarithm problem is easy in $\mathbb{G}_2$. Indeed, for $g_2^u \in \mathbb{G}_2$, we have

$$g_2^u = |(1+N)^e|_{N^2} = |1 + eN|_{N^2} = \begin{cases} [e]_N N + 1 & \text{if } [e]_N < N/2 \\ [-e]_N N - 1 & \text{if } [e]_N > N/2. \end{cases}$$

A simple case distinction thus allows to compute $[e]_N$.

Membership testing and sampling exponents. It is left to note that membership in $\mathbb{G}$ can be efficiently decided (by checking that $y \in \mathbb{Z}_{N^2}$ is invertible, lies between $-N^2/2$ and $N^2/2$, and satisfies $\left(\frac{y}{N}\right) = 1$). However, since $|\mathbb{G}_1|$ will not be public, exponents $s \in \mathbb{Z}_{|\mathbb{G}_1|}$ can only be sampled approximatively, e.g., by uniformly sampling $s \in \mathbb{Z}_{\lfloor N/4 \rfloor}$. This incurs a statistical defect of $\mathbf{O}(1/2^\lambda)$ upon each such sampling. In the following, we will silently ignore these statistical defects (and assume that there is an algorithm that uniformly samples $s \in \mathbb{Z}_{\varphi(N)}$) in

[3] We note that our DCR-based OR-proofs from Section 5.4.2 require P, Q to be somewhat larger, although still compatible with practical parameter choices.

our generic constructions for simplicity and ease of presentation. However, we note that the concrete bound (22) also holds for such an approximative sampling in the DCR setting.

Hardness of the GDDH and factoring problems. We claim that in the setting described above, the Decisional Composite Residuosity (DCR) assumption [28] implies the GDDH assumption. This connection has already been established in [15, Theorem 2] for a slight variant of the groups $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$ above. (In their setting, $\mathbb{G}_1$ consists of elements $x^N \in \mathbb{Z}_{N^2}$ with $\left(\frac{x^N}{N}\right) = 1$, instead of elements $|x^N|_{N^2}$ with $\left(\frac{x^N}{N}\right) = 1$.) In fact, their proof applies also to our setting, and we obtain that the DCR assumption implies the GDDH assumption with $\ell = 1$ and trivial $\mathbf{B} = 1$ in $\mathbb{G}$ (as in Definition 3.1).

Furthermore, we note that the DCR assumption also implies the $\mathbb{G}_2$-factoring assumption (Definition 3.2). We sketch how any $\mathbb{G}_2$-factoring adversary $\mathcal{A}$ can be transformed into a DCR adversary $\mathcal{B}$. First, $\mathcal{B}$ runs $\mathcal{A}$, and obtains elements $g_2^{u_1}, \ldots, g_2^{u_q}$. Then, $\mathcal{B}$ uses that the discrete logarithm problem is easy in $\mathbb{G}_2$, and retrieves the corresponding $u_1, \ldots, u_q \in \mathbb{Z}_{|\mathbb{G}_2|}$. Now if $\gcd(|\mathbb{G}_2|, u_i) \notin \{1, |\mathbb{G}_2|\}$ for some u_i, then $\gcd(N, u_i) \in \{P, Q\}$ directly allows to factor N. Hence, if $\mathcal{A}$ succeeds, then $\mathcal{B}$ can factor N, and solve its own DCR challenge (e.g., by computing the order of its input).

4 Tightly secure building blocks

In this section, we describe two building blocks for our main KEM construction. The first, tightly secure one-time signature schemes, is fairly standard, but requires a new instantiation in the DCR setting to achieve tight security. The second is, key extractors, is new, but similar building blocks have been been used at least in the prime-order setting implicitly in previous works on tight security (e.g., [12]).

4.1 One-time signature schemes

Definition 4.1 (Signature scheme)**.** *A digital signature scheme* $\mathbf{OTS} = (\mathbf{SGen}, \mathbf{SSig}, \mathbf{SVer})$ *consists of the following PPT algorithms:*

- $\mathbf{SGen}(1^\lambda)$ *outputs a keypair* (ovk, osk)*. We call ovk and osk the verification, resp. signing key.*
- $\mathbf{SSig}(osk, M)$*, for a message* $M \in \{0,1\}^*$*, outputs a signature* σ*.*
- $\mathbf{SVer}(ovk, M, \sigma)$*, outputs either* 0 *or* 1*.*

We require correctness in the sense that for all (ovk, osk) *in the range of* $\mathbf{SGen}(1^\lambda)$*, all* $M \in \{0,1\}^*$*, and all* σ *in the range of* $\mathbf{SSig}(osk, M)$*, we always have* $\mathbf{SVer}(ovk, M, \sigma) = 1$*.*

We only require one-time security (and call a signature scheme secure in this sense also a one-time signature scheme):

Definition 4.2 (EUF-MOTCMA security)**.** *Let* **OTS** *be a digital signature scheme as in Definition 4.1, and consider the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*:*

1. *$\mathcal{C}$ runs $\mathcal{A}$ on input 1^λ, and with (many-time) oracle access to the following oracles:*
   - $\mathcal{O}_{\mathbf{gen}}()$ *samples a fresh keypair* $(ovk, osk) \leftarrow \mathbf{SGen}()$*, and returns ovk.*
   - $\mathcal{O}_{\mathbf{sig}}(ovk, M)$ *first checks if ovk has been generated by* $\mathcal{O}_{\mathbf{gen}}$*, and returns* $\perp$ *if not. Next,* $\mathcal{O}_{\mathbf{sig}}$ *checks if there has been a previous* $\mathcal{O}_{\mathbf{sig}}(ovk, \cdot)$ *query (i.e., an* $\mathcal{O}_{\mathbf{sig}}$ *query with the same ovk), and returns* $\perp$ *if so. Let osk be the corresponding secret key generated alongside ovk. (If ovk has been generated multiple times by* $\mathcal{O}_{\mathbf{gen}}$*, take the first such osk.)* $\mathcal{O}_{\mathbf{sig}}$ *returns* $\sigma \leftarrow \mathbf{SSig}(osk, M)$*.*
2. *If $\mathcal{A}$ returns* (ovk^*, M^*, σ^*)*, such that* $\mathbf{SVer}(ovk^*, M^*, \sigma^*) = 1$*, and* ovk^* *has been returned by* $\mathcal{O}_{\mathbf{gen}}$*, but* σ^* *has not been returned by* $\mathcal{O}_{\mathbf{sig}}(ovk^*, M^*)$*, then $\mathcal{C}$ returns* 1*. Otherwise, $\mathcal{C}$ returns* 0*.*

Let $\mathrm{Adv}^{\mathrm{ots}}_{\mathbf{OTS},\mathcal{A}}(\lambda)$ *be the probability that $\mathcal{C}$ finally outputs* 1 *in the above game. We say that* **OTS** *is strongly existentially unforgeable under many one-time chosen-message attacks (EUF-MOTCMA secure) iff for every PPT $\mathcal{A}$, the function* $\mathrm{Adv}^{\mathrm{ots}}_{\mathbf{OTS},\mathcal{A}}(\lambda)$ *is negligible.*

We remark, however, that our security notion is "strong", in the sense that a forger is already successful when he manages to generate a new signature for an already signed message.

4.1.1 A construction in the prime-order setting

In case $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ with $|\mathbb{G}|$ prime and public, [17] already give a simple construction of a digital signature scheme that achieves EUF-MOTCMA security under the discrete logarithm assumption. Most importantly for our case, their security reduction is tight (i.e., only loses a constant factor). We refer to their paper for details.

4.1.2 A construction in the DCR setting

In the DCR setting (as in Section 3.3), there exist simple and efficient EUF-MOTCMA secure signature schemes from the factoring [23] or RSA assumptions [21]. However, these schemes are not known to be tightly secure.

Hence, in this section, we construct a new digital signature scheme whose EUF-MOTCMA security can be tightly reduced to the GDDH assumption in the DCR setting.

Let $N = PQ$ and $\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2$ be defined as in Section 3.3. In particular, we have $|\mathbb{G}_1| = \varphi(N)/4$, and $|\mathbb{G}_2| = N$. We are going to assume that the global public parameters pp contain a hash function $H : \{0,1\}^* \to \{0,1\}^{\ell_H}$ from a collision-resistant hash function generator **CRHF**, and that $|\mathbb{G}_2| = N > 2^{\ell_H}$. Now our signature scheme $\mathbf{OTS_{DCR}}$ is defined by the following algorithms:

- **SGen**() uniformly samples $d_0, d_1 \in \mathbb{G}$, and outputs $ovk = (D_0, D_1) = (d_0^N, d_1^N) \in \mathbb{G}_1^2$ and $osk = (d_0, d_1)$.
- **SSig**(osk, M) uniformly samples $d_0', d_1' \in \mathbb{G}$, and computes

$$\begin{aligned} (D_0', D_1') &= (d_0'^N, d_1'^N) \in \mathbb{G}_1^2 \\ e' &= |d_1'^{\kappa'} d_0'|_N \in \{0, \ldots, \lfloor N/2 \rfloor\} \quad \text{for} \quad \kappa' = H(M) \\ e &= |d_1^{\kappa} d_0|_N \in \{0, \ldots, \lfloor N/2 \rfloor\} \quad \text{for} \quad \kappa = H(D_0', D_1'). \end{aligned} \tag{2}$$

  The signature is defined as $\sigma = (D_0', D_1', e', e) \in \mathbb{G}^2 \times \{0, \ldots, \lfloor N/2 \rfloor\}^2$.
- **SVer**(ovk, M, σ), for ovk as above, outputs 1 iff $\sigma = (D_0', D_1', e', e) \in \mathbb{G}^2 \times \{0, \ldots, \lfloor N/2 \rfloor\}^2$, and the following equations hold, *where* e, e' *are interpreted*[4] *as elements of* $\mathbb{G}$:

$$\begin{aligned} e'^N &= D_1'^{\kappa'} D_0' \quad \text{for} \quad \kappa' = H(M) \\ e^N &= D_1^{\kappa} D_0 \quad \text{for} \quad \kappa = H(D_0', D_1'). \end{aligned} \tag{3}$$

Correctness of this scheme follows from (2), (3), and Footnote 4. Essentially, this scheme is the combination of two instances of a simpler, non-adaptively secure one-time signature scheme. Thus, we could explain our scheme in a more modular way. However, since we only use that scheme as a building block, we prefer to give a brief exposition and analysis. Specifically:

Lemma 4.3 (EUF-MOTCMA security of $\mathbf{OTS_{DCR}}$)**.** *Under the GDDH assumption, and if* **CRHF** *is collision-resistant, the above signature scheme* $\mathbf{OTS_{DCR}}$ *is EUF-MOTCMA secure. Concretely, for every adversary* $\mathcal{A}$*, there exist adversaries* $\mathcal{B}^{\text{gddh}}$ *and* $\mathcal{B}^{\text{crhf}}$ *(of essentially the same complexity as the EUF-MOTCMA experiment with* $\mathbf{OTS_{DCR}}$ *and* $\mathcal{A}$*) such that*

$$\mathrm{Adv}^{\text{ots}}_{\mathbf{OTS_{DCR}},\mathcal{A}}(\lambda) \leq 2\mathrm{Adv}^{\text{gddh}}_{\mathbb{G},\mathcal{B}^{\text{gddh}}}(\lambda) + \mathrm{Adv}^{\text{crhf}}_{\mathbf{CRHF},\mathcal{B}^{\text{crhf}}}(\lambda). \tag{4}$$

[4] To interpret an element $|x|_N \in \{0, \ldots, \lfloor N/2 \rfloor\}$ as a $\mathbb{G}$-element, we use the embedding $\phi : \{0, \ldots, \lfloor N/2 \rfloor\} \to \mathbb{G}$ with $\phi(y) = y \bmod N^2$. By our definition of $\mathbb{G}$, this means that for every $x \in \mathbb{G}$, there is an $i \in \mathbb{Z}_N$ with $\phi(|x|_N) = x + i \cdot N \in \mathbb{G}$. In particular, note that $\phi(|x|_N)^N = x^N \in \mathbb{G}$ for every $x \in \mathbb{G}$. This last property will imply correctness.

Proof. In the following, write (ovk^*, M^*, σ^*) with $ovk^* = (D_0^*, D_1^*)$ and $\sigma^* = (D_0'^*, D_1'^*, e'^*, e^*)$ for $\mathcal{A}$'s forgery. Let $\mathbf{bad}_{\text{coll}}$ be the event that $\mathcal{A}$'s forgery induces a hash collision (in the sense that $H(M^*) = H(M)$ for some previously signed $M \neq M^*$, or that $H(D_0'^*, D_1'^*) = H(D_0', D_1')$ for some $(D_0', D_1') \neq (D_0'^*, D_1'^*)$ from a previous $\mathcal{O}_{\mathbf{sig}}$ query). A straightforward reduction shows

$$\Pr[\mathbf{bad}_{\text{coll}}] \leq \text{Adv}^{\text{crhf}}_{\mathbf{CRHF},\mathcal{B}^{\text{crhf}}}(\lambda) \tag{5}$$

for an adversary $\mathcal{B}^{\text{crhf}}$ on **CRHF** that simulates the EUF-MOTCMA game with $\mathbf{OTS}_{\mathbf{DCR}}$ and $\mathcal{A}$ and outputs any such collision upon $\mathbf{bad}_{\text{coll}}$.

Next, let $\mathbf{bad}_{\text{fresh}}$ denote the event that $\mathbf{bad}_{\text{coll}}$ does not occur, and $\mathcal{A}$ submits a valid forgery for some $(D_0'^*, D_1'^*)$ that has *not* been output upon a previous $\mathcal{O}_{\mathbf{sig}}(ovk, M)$ query with $ovk = ovk^*$. (In other words, $\mathbf{bad}_{\text{fresh}}$ occurs if $\mathcal{A}$ forges a signature with a fresh pair $(D_0'^*, D_1'^*)$ that has been not generated previously by the game itself.) To bound the probability for $\mathbf{bad}_{\text{fresh}}$, consider the following adversary $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ on the DCR assumption. $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ gets as input a value $Z = g_1^r g_2^b$, for random $r \in \mathbb{Z}_{|\mathbb{G}_1|}$ and $b \in \{0, 1\}$. $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ internally simulates the EUF-MOTCMA game with $\mathbf{OTS}_{\mathbf{DCR}}$ and $\mathcal{A}$, with the following exceptions:

- Upon an $\mathcal{O}_{\mathbf{gen}}$ query from $\mathcal{A}$, $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ *first* generates a fresh pair (d_0', d_1') (as necessary for signatures), sets $(D_0', D_1') = (d_0'^N, d_1'^N)$ and $\kappa = H(D_0', D_1')$, computes $(D_0, D_1) = (s_0^N / Z^\kappa, s_1^N Z)$ for uniformly chosen $s_0, s_1 \in \mathbb{G}$, and outputs $ovk = (D_0, D_1)$.
- Upon an $\mathcal{O}_{\mathbf{sig}}$ query from $\mathcal{A}$, $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ uses the previously generated corresponding values (d_0', d_1') to generate e' as in (2). The value e is generated as $e = |s_1^\kappa s_0|_N$. (Note that this implies $e^N = s_1^{N\kappa} s_0^N = D_1^\kappa D_0$, cf. Footnote 4.) The final signature is $\sigma = (D_0', D_1', e', e)$.

Finally, $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ outputs 1 iff $\mathbf{bad}_{\text{fresh}}$ occurs.

We turn to $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$'s analysis. First, if $b = 0$ (i.e., if $Z = g_1^r$), then $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ perfectly simulates the EUF-MOTCMA game with $\mathbf{OTS}_{\mathbf{DCR}}$ and $\mathcal{A}$. However, if $b = 1$, then $\mathbf{bad}_{\text{fresh}}$ cannot occur. Indeed, for contradiction, assume a valid forgery (ovk^*, M^*, σ^*) with $ovk^* = (D_0^*, D_1^*)$, and $\sigma^* = (D_0'^*, D_1'^*, e'^*, e^*)$. Now $\mathbf{bad}_{\text{fresh}}$ would imply that $(D_0'^*, D_1'^*) \neq (D_0', D_1')$ for the pair (D_0', D_1') previously generated upon $\mathcal{A}$'s $\mathcal{O}_{\mathbf{gen}}$ query. Since $\mathbf{bad}_{\text{fresh}}$ implies $\neg\mathbf{bad}_{\text{coll}}$, we may also assume that $\kappa^* = H(D_0'^*, D_1'^*) \neq H(D_0', D_1') = \kappa$. However, the key $ovk^* = (D_0^*, D_1^*)$ output by $\mathcal{O}_{\mathbf{gen}}$ satisfies $(D_1^*)^{\kappa^*} D_0^* = (s_1^\kappa s_0)^N Z^{\kappa^* - \kappa} = g_1^z g_2^{\kappa^* - \kappa}$ for some $z \in \mathbb{Z}_{|\mathbb{G}_1|}$. Hence, there is no value e with $e^N = (D_1^*)^{\kappa^*} D_0^*$ (since the order of e^N divides $\varphi(N)/4$, while the order of $(D_1^*)^{\kappa^*} D_0^* = g_1^z g_2^{\kappa^* - \kappa}$ is a nontrivial multiple of N). Thus, σ^* cannot be valid in the sense of (3).

In summary, we have

$$\Pr[\mathbf{bad}_{\text{fresh}}] \leq |\text{Adv}^{\text{gddh}}_{\mathbb{G},\mathcal{B}^{\text{gddh}}_{\text{fresh}}}(\lambda)|. \tag{6}$$

Similarly, let $\mathbf{bad}_{\text{reused}}$ denote the event that $\mathbf{bad}_{\text{coll}}$ does not occur, and $\mathcal{A}$ submits a valid forgery as above, but with a pair $(D_0'^*, D_1'^*)$ that *has* been output upon a previous $\mathcal{O}_{\mathbf{sig}}(ovk, M)$ query with $ovk = ovk^*$. (In other words, $\mathbf{bad}_{\text{reused}}$ occurs if $\mathcal{A}$ reuses a pair $(D_0'^*, D_1'^*)$.) To bound the probability for $\mathbf{bad}_{\text{reused}}$, consider the following adversary $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ on the DCR assumption. Like $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ above, $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ internally simulates the EUF-MOTCMA game with $\mathbf{OTS}_{\mathbf{DCR}}$ and $\mathcal{A}$, but embeds its own challenge $Z = g_1^r g_2^b$ into the (D_0', D_1') pairs. Specifically, $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ computes $(D_0', D_1') = (s_0^N / Z^{\kappa'}, s_1^N Z)$ for fresh random s_0, s_1 upon each $\mathcal{O}_{\mathbf{sig}}$ queries (where $\kappa' = H(M)$ as with **SSig**). Finally, $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ outputs 1 if $\mathbf{bad}_{\text{reused}}$ occurs.

As with $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$, it is easy to see that $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ perfectly simulates the EUF-MOTCMA game if $b = 0$. We also claim that $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ never outputs 1 if $b = 1$. For contradiction, assume a valid forgery as above, but with reused $(D_0'^*, D_1'^*) = (D_0', D_1')$ generated by $\mathcal{B}^{\text{gddh}}_{\text{reused}}$. We may assume that $M^* \neq M$ for the message M previously signed by $\mathcal{O}_{\mathbf{sig}}$ under ovk^*. Indeed, $M^* = M$ would imply $e'^N = (e'^*)^N$. But this means $e' = e'^*$ (since $e' \neq e'^*$ would imply that $e'/e'^* \in \mathbb{G}$

is a nonzero element whose order divides $\varphi(N)$), and similarly $e = e^*$. Hence, any successful forgery with $M^* = M$ would imply $\sigma^* = \sigma$, and thus be rejected by the EUF-MOTCMA game.

Since $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ implies $\neg\mathbf{bad}_{\text{coll}}$, we may even assume $\kappa^* \neq \kappa$. However, $(D_1'^*)^{\kappa^*} D_0'^* = (D_1')^{\kappa^*} D_0' = g_1^z g_2^{\kappa^* - \kappa}$ for a suitable $g_1^z \in \mathbb{Z}_{|\mathbb{G}_1|}$, and thus no e' with $e'^N = (D_1'^*)^{\kappa^*} D_0'^*$ exists. As a consequence, σ^* cannot be valid in the sense of (3), and we get

$$\Pr[\mathbf{bad}_{\text{reused}}] \leq |\text{Adv}^{\text{gddh}}_{\mathbb{G},\mathcal{B}^{\text{gddh}}_{\text{reused}}}(\lambda)|. \tag{7}$$

Now any successful forgery either implies $\mathbf{bad}_{\text{coll}}$, or $\mathbf{bad}_{\text{fresh}}$, or $\mathbf{bad}_{\text{reused}}$. Hence, combining (5), (6), and (7) yields (4) if we combine the two adversaries $\mathcal{B}^{\text{gddh}}_{\text{fresh}}$ and $\mathcal{B}^{\text{gddh}}_{\text{reused}}$ into one. □

4.2 Key extractors

Intuition. Intuitively, a key extractor derives a pseudorandom key K from a given encryption $\mathbf{c} = \mathbf{E}(0;\mathbf{r})$ of 0. This K can be derived either publicly, using a public extraction key *xpk* and the witness $\mathbf{r}$, or secretly, using a secret extraction key *xsk* and only the ciphertext $\mathbf{c}$. We desire security in the sense keys derived secretly (i.e., using *xsk*) from random ciphertexts $\mathbf{c} = \mathbf{E}(R;\mathbf{r})$ for random R cannot be distinguished from truly random bitstrings K. This should hold even for many such challenges, and in the face of oracle access to *xsk* on "consistent" ciphertexts $\mathbf{c} = \mathbf{E}(0;\mathbf{r})$.

In this sense, key extractors give a computational form of the soundness guarantee provided by universal hash proof systems. We also note that a similar tool has been implicitly used in [12] for a similar purpose in the prime-order setting. Hence, we abstract and generalize their construction in a straightforward way.

Definition. In the following, fix a function $\ell_{\text{ext}} = \ell_{\text{ext}}(\lambda)$. In the following definition, we will choose the value R encrypted in random ciphertexts uniformly from $\mathbb{Z}_{2^{\ell_{\text{ext}}}}$. Our subsequent generic construction of key extractors works for any $\ell_{\text{ext}} \geq 3\lambda$ (and $|\mathbb{G}_2| \geq 2^{3\lambda}$).

Definition 4.4 (Key extractor)**.** *A key extractor* $\mathbf{EXT} = (\mathbf{ExtGen}, \mathbf{Ext}_{\text{pub}}, \mathbf{Ext}_{\text{priv}})$ *for* $\mathbb{G}$ *consists of the following PPT algorithms*

- $\mathbf{ExtGen}(1^\lambda, \mathit{epk})$*, on input a public encryption key* $\mathit{epk} = g_1^{\omega^\top \mathbf{B}} \in \mathbb{G}_1^{\ell_{\mathbf{B}}}$ *for* $(\mathbf{E},\mathbf{D})$ *(as in Section 3.1.3), outputs a keypair* $(\mathit{xpk}, \mathit{xsk})$*. We call xpk the public and xsk the private extraction key.*
- $\mathbf{Ext}_{\text{pub}}(\mathit{xpk}, \mathbf{c}, \mathbf{r})$*, for* $\mathbf{c} = \mathbf{E}_{\mathit{epk}}(0;\mathbf{r})$*, outputs a key* $K \in \{0,1\}^\lambda$*.*
- $\mathbf{Ext}_{\text{priv}}(\mathit{xsk}, \mathbf{c})$ *also outputs a session key* $K \in \{0,1\}^\lambda$*.*

We require the following:

Correctness. *For all* $\mathit{epk} = g_1^{\omega^\top \mathbf{B}}$*, all keypairs* $(\mathit{xpk}, \mathit{xsk})$ *in the range of* $\mathbf{ExtGen}(1^\lambda, \mathit{epk})$*, all* $\mathbf{r} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}}}$*, and all* $\mathbf{c} = \mathbf{E}_{\mathit{epk}}(0;\mathbf{r})$*, we always have* $\mathbf{Ext}_{\text{pub}}(\mathit{xpk}, \mathbf{c}, \mathbf{r}) = \mathbf{Ext}_{\text{priv}}(\mathit{xsk}, \mathbf{c})$*.*

Indistinguishability. *Consider the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*:*

1. *$\mathcal{C}$ uniformly samples* $\boldsymbol{\omega} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}}}$ *and sets* $(\mathit{epk}, \mathit{esk}) = (g_1^{\omega^\top \mathbf{B}}, \boldsymbol{\omega})$*. Then, $\mathcal{C}$ generates an* $\mathbf{EXT}$ *keypair* $(\mathit{xpk}, \mathit{xsk}) \leftarrow \mathbf{ExtGen}(1^\lambda, \mathit{epk})$*, and finally samples* $b \in \{0,1\}$*.*
2. *$\mathcal{A}$ is run on input* $(1^\lambda, \mathit{epk}, \mathit{xpk})$*, with (many-time) access to oracles* $\mathcal{O}_{\mathbf{cha}}$ *and* $\mathcal{O}_{\mathbf{ext}}$ *that operate as follows:*
   - $\mathcal{O}_{\mathbf{cha}}()$ *uniformly chooses a fresh* $R \in \mathbb{Z}_{2^{\ell_{\text{ext}}}}$*, computes* $\mathbf{c} \leftarrow \mathbf{E}_{\mathit{epk}}(R)$ *and* $K_0 = \mathbf{Ext}_{\text{priv}}(\mathit{xsk}, \mathbf{c})$*, and uniformly chooses* $K_1 \in \{0,1\}^\lambda$*. Finally,* $\mathcal{O}_{\mathbf{cha}}$ *returns* $(\mathbf{c}, K_b)$*.*
   - $\mathcal{O}_{\mathbf{ext}}(\mathbf{c})$ *first checks if* $\mathbf{D}_{\mathit{esk}}(\mathbf{c}) = g_2^0$*. If not, then we say that $\mathcal{A}$ fails, and $\mathcal{C}$ terminates with output 0 immediately. Otherwise,* $\mathcal{O}_{\mathbf{ext}}$ *computes and returns* $K = \mathbf{Ext}_{\text{priv}}(\mathit{xsk}, \mathbf{c})$*.*
   - *Finally, $\mathcal{A}$ outputs a bit* b'*, and $\mathcal{C}$ outputs 1 iff* $b = b'$ *(and 0 otherwise).*

Let $\text{Adv}^{\text{ext}}_{\mathbf{EXT},\mathcal{A}}(\lambda) = \Pr[\mathcal{C} \text{ outputs } 1] - 1/2$*. We require that for all PPT* $\mathcal{A}$*,* $\text{Adv}^{\text{snd}}_{\mathbf{PS},\mathcal{A}}(\lambda) \leq \varepsilon$ *for a negligible function* $\varepsilon = \varepsilon(\lambda)$*.*

4.2.1 A generic construction

For our GDDH-based key extractor, we assume that a function h chosen from a family of universal hash functions $\mathbf{UHF}_\lambda$ is made public in the global public parameters pp. Then, our extractor $\mathbf{EXT}^{\text{gddh}} = (\mathbf{ExtGen}^{\text{gddh}}, \mathbf{Ext}^{\text{gddh}}_{\text{pub}}, \mathbf{Ext}^{\text{gddh}}_{\text{priv}})$ is defined as follows:

- $\mathbf{ExtGen}^{\text{gddh}}(1^\lambda, \mathit{epk})$, for $\mathit{epk} = g_1^{\omega^\top \mathbf{B}}$, uniformly samples $\mathbf{s} \in \mathbb{Z}_{|\mathbb{G}|}^{\ell_\mathbf{B}}$ and $t \in \mathbb{Z}_{|\mathbb{G}|}$, and computes $g_1^{\mathbf{w}^\top} := g_1^{\mathbf{s}^\top \mathbf{B} + t \cdot \omega^\top \mathbf{B}} \in \mathbb{G}_1^{\ell_\mathbf{B}}$. The output of $\mathbf{ExtGen}^{\text{gddh}}$ is $\mathit{xpk} = g_1^{\mathbf{w}^\top}$ and $\mathit{xsk} = (\mathbf{s}, t)$.
- $\mathbf{Ext}^{\text{gddh}}_{\text{pub}}(\mathit{xpk}, \mathbf{c}, \mathbf{r})$, for xpk as above and $\mathbf{c} = \mathbf{E}_{\mathit{epk}}(0; \mathbf{r})$, outputs $K = h(g_1^{\mathbf{w}^\top \cdot \mathbf{r}})$.
- $\mathbf{Ext}^{\text{gddh}}_{\text{priv}}(\mathit{xsk}, \mathbf{c})$, for $\mathbf{c} = (g^\gamma, g^\delta) \in \mathbb{G}^{\ell_\mathbf{B}} \times \mathbb{G}$, outputs $K = h(g^{\mathbf{s}^\top \gamma + t \cdot \delta})$.

Given $g_1^{\mathbf{w}^\top} = g_1^{\mathbf{s}^\top \mathbf{B} + t \cdot \omega^\top \mathbf{B}}$ and a ciphertext $\mathbf{c} = \mathbf{E}(0; \mathbf{r}) = (g^\gamma, g^\delta) = (g_1^{\mathbf{B}\mathbf{r}}, g_1^{\omega^\top \mathbf{B}\mathbf{r}})$, we have

$$g_1^{\mathbf{w}^\top \mathbf{r}} = g_1^{\mathbf{s}^\top \mathbf{B}\mathbf{r} + t \cdot \omega^\top \mathbf{B}\mathbf{r}} = g^{\mathbf{s}^\top \gamma + t \cdot \delta},$$

and correctness follows. Indistinguishability follows from the following lemma:

Lemma 4.5. *For $\ell_{\text{ext}} \geq 3\lambda$ and $|\mathbb{G}_2| \geq 2^{3\lambda}$, $\mathbf{EXT}^{\text{gddh}}$ above satisfies the indistinguishability property of Definition 4.4, assuming GDDH in $\mathbb{G}$. Specifically, for every adversary $\mathcal{A}$ that makes at most q oracle queries, there is an adversary $\mathcal{B}$ (with roughly the same complexity as the indistinguishability experiment with $\mathbf{EXT}^{\text{gddh}}$ and $\mathcal{A}$), such that*

$$\mathrm{Adv}^{\text{ext}}_{\mathbf{EXT}^{\text{gddh}},\mathcal{A}}(\lambda) \;\leq\; \mathrm{Adv}^{\text{gddh}}_{\mathbb{G},\mathcal{B}}(\lambda) + q/2^\lambda. \tag{8}$$

Proof. Fix an adversary $\mathcal{A}$ in the sense of Definition 4.4, and consider the following GDDH adversary $\mathcal{B}$. First, $\mathcal{B}$ gets as input $(g_1^{\omega^\top \mathbf{B}}, g_1^{\mathbf{B}\mathbf{r}}, g_1^{\omega^\top \mathbf{B}\mathbf{r}} g_2^\beta)$ for $\beta \in \{0,1\}$, and sets $\mathit{epk} = g_1^{\omega^\top \mathbf{B}}$. Then, $\mathcal{B}$ simulates the indistinguishability experiment from Definition 4.4 with $\mathbf{EXT}^{\text{gddh}}$ and $\mathcal{A}$, as follows.

$\mathcal{B}$ begins by choosing $\mathbf{s}' \in \mathbb{Z}_{|\mathbb{G}|}^{\ell_\mathbf{B}}$, $t' \in \mathbb{Z}_{|\mathbb{G}|}$, and a bit $b \in \{0,1\}$ uniformly, and then sets $g_1^{\mathbf{w}'^\top} = g_1^{\mathbf{s}'\mathbf{B} + t' \cdot \omega^\top \mathbf{B}}$, and $\mathit{xpk} = g_1^{\mathbf{w}'^\top}$. Then, $\mathcal{B}$ runs $\mathcal{A}$ on input $(1^\lambda, \mathit{epk}, \mathit{xpk})$, and implements $\mathcal{A}$'s oracles $\mathcal{O}_{\mathbf{cha}}$ and $\mathcal{O}_{\mathbf{ext}}$ as follows:

- $\mathcal{O}_{\mathbf{cha}}()$ first chooses uniformly $\mathbf{r}' \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$ and $R \in \mathbb{Z}_{2^{\ell_{\text{ext}}}}$, and sets

$$\mathbf{c} \;=\; (g^\gamma, g^\delta) \;=\; (g_1^{R \cdot \omega^\top \mathbf{B}\mathbf{r} + \omega^\top \mathbf{B}\mathbf{r}'}, g_1^{R \cdot \omega^\top \mathbf{B}\mathbf{r} + \omega^\top \mathbf{B}\mathbf{r}'} g_2^{R\beta}) \;=\; (g_1^{\omega^\top \mathbf{B}(R\mathbf{r} + \mathbf{r}')}, g_1^{\omega^\top \mathbf{B}(R\mathbf{r} + \mathbf{r}')} g_2^{R\beta}).$$

  Observe that this way, $\mathbf{c}$ is a fresh encryption of $R\beta$. Next, $\mathcal{O}_{\mathbf{cha}}$ sets $K_0 = h(g^{\mathbf{s}'^\top \gamma + t' \cdot \delta} g_2^R)$, samples $K_1 \in \{0,1\}^\lambda$, and returns $(\mathbf{c}, K_b)$ to $\mathcal{A}$.
- $\mathcal{O}_{\mathbf{ext}}(\mathbf{c})$, for $\mathbf{c} = (g^\gamma, g^\delta)$, computes and returns $K = (g^{\mathbf{s}'^\top \gamma + t' \cdot \delta})$ to $\mathcal{A}$.

Finally, $\mathcal{B}$ returns 1 iff $\mathcal{A}$ finally correctly predicts b.

Observe that $\mathcal{B}$'s internal simulation differs from the indistinguishability experiment with $\mathbf{EXT}^{\text{gddh}}$ and $\mathcal{A}$ in two aspects:

- Ciphertexts prepared by $\mathcal{B}$'s oracle $\mathcal{O}_{\mathbf{cha}}$ only encrypt R (for random $R \in \mathbb{Z}_{2^{\ell_{\text{ext}}}}$) if $\mathcal{B}$'s own challenge has $\beta = 1$. (Otherwise, $\mathcal{O}_{\mathbf{cha}}$ prepares ciphertexts that always encrypt 0.)
- If $b = 0$, "raw keys" $g^{\mathbf{s}'^\top \gamma + t' \cdot \delta} g_2^R$ prepared by $\mathcal{B}$'s $\mathcal{O}_{\mathbf{cha}}$ oracle have an extra g_2^R term.

In a nutshell, these changes have the following effect: if $\beta = 1$, then $\mathcal{B}$ perfectly simulates the indistinguishability experiment, *unless* $\mathcal{A}$ fails (in the sense of Definition 4.4). But if $\beta = 0$, then $\mathcal{B}$ outputs 1 with probability essentially $1/2$. After proving these claims formally, we will show that combining them yields (8).

We start by formalizing and proving the first claim, namely

$$\Pr[\mathcal{B} \text{ outputs } 1 \mid \beta = 1] \;\geq\; \mathrm{Adv}^{\text{ext}}_{\mathbf{EXT},\mathcal{A}}(\lambda) + 1/2. \tag{9}$$

(Note that the right-hand side of (9) is precisely the probability that $\mathcal{C}$ outputs 1 in the indistinguishability experiment of Definition 4.4.)

To show (9), let **fail** be the event that $\mathcal{A}$ fails, i.e., that $\mathcal{A}$ submits a ciphertext $\mathbf{c}$ to $\mathcal{O}_{\mathbf{ext}}$ with $\mathbf{D}_{esk}(\mathbf{c}) \neq g_2^0$. Furthermore, recall that $epk = g_1^{\omega^\top \mathbf{B}}$. Hence, if we set $(\mathbf{s}, t) = (\mathbf{s}' - \boldsymbol{\omega}, t' + 1)$, we get that $(\mathbf{s}, t)$ and $(\mathbf{s}', t')$ are two secret keys that act identically on ciphertexts of the form $\mathbf{c} = \mathbf{E}_{epk}(0)$:

$$\mathbf{D}_{(\mathbf{s},t)}(\mathbf{c}) = g_1^{(\mathbf{s}+t\cdot\boldsymbol{\omega})^\top \mathbf{Br}} = g_1^{(\mathbf{s}'+t\cdot\boldsymbol{\omega})^\top \mathbf{Br}} = \mathbf{D}_{(\mathbf{s}',t')}(\mathbf{c}) \qquad \text{for} \qquad \mathbf{c} = (g_1^{\mathbf{Br}}, g_1^{\omega^\top \mathbf{Br}}).$$

Further, for $\mathbf{c} = (g^\gamma, g^\delta) = (g_1^{\mathbf{Br}}, g_1^{\omega^\top \mathbf{Br}} g_2^R)$, we get $\mathbf{D}_{(\mathbf{s},t)}(\mathbf{c}) = \mathbf{D}_{(\mathbf{s}',t')}(\mathbf{c}) g_2^R$. In other words, if $\beta = 1$, and unless **fail** occurs, $\mathcal{B}$ perfectly emulates the indistinguishability game with secret key $(\mathbf{s}, t)$ (instead of $(\mathbf{s}', t')$). Since the challenger $\mathcal{C}$ outputs 0 upon **fail** in the indistinguishability game, we get (9).

Our second claim is

$$\Pr[\mathcal{B} \text{ outputs } 1 \mid \beta = 0] \;\leq\; 1/2 + q/2^\lambda. \tag{10}$$

Indeed, observe that when $\beta = 0$, $\mathcal{O}_{\mathbf{cha}}$ queries contain ciphertexts $\mathbf{c} = (g_1^{\mathbf{Br}}, g_1^{\omega^\top \mathbf{Br}})$ that do not depend on the respective (freshly and independently chosen) $R \in \mathbb{Z}_{2^{\ell_{\text{ext}}}}$. Thus, if $b = 0$, the "raw keys" $g_1^{\mathbf{s}'^\top \mathbf{Br} + t'\cdot\omega^\top \mathbf{Br}} g_2^R$ have at least 3λ bits of min-entropy (since $\ell_{\text{ext}} \geq 3\lambda$, and the order of g_2 is greater than $2^{3\lambda}$). Hence, the keys $K = K_0 = h(g_1^{\mathbf{s}'^\top \mathbf{Br} + t'\cdot\omega^\top \mathbf{Br}} g_2^R)$ are statistically $1/2^\lambda$-close to uniform. But also if $b = 1$, the corresponding keys $K = K_1$ are truly random by definition. Hence, $\mathcal{A}$ receives an (at least almost) independently random K in either case. Taking into account the statistical defect from h, we obtain (10).

Finally, combining (9) and (10), we directly get (8). □

Summing up, we obtain

Theorem 4.6. *Under the GDDH assumption, and for $\ell_{\text{ext}} \geq 3\lambda$ and $|\mathbb{G}_2| \geq 2^{3\lambda}$, $\mathbf{EXT}^{\text{gddh}}$ is a key extractor in the sense of Definition 4.4.*

5 Benign proof systems

Intuition. Benign proof systems are the central technical tool in our KEM construction. Intuitively, a benign proof system for some language $\mathcal{L}$ is a non-interactive designated-verifier zero-knowledge proof system with strong soundness guarantees. Concretely, the system guarantees soundness even if simulated proofs for potentially false statements $x \notin \mathcal{L}$ are known. However, we do not quite require "simulation-soundness", in the sense that this should hold for simulated proofs for arbitrary false statements. (We note that simulation-sound proof systems are extremely useful in the context of tight security proofs, but they are also very hard to construct.)

Instead, we only require that no adversary can forge proofs for statements $x \notin \mathcal{L}$ that are "more false" than any statement for which a simulated proof is known. A little more specifically, we require that even if simulated proofs for statements $x \in \mathcal{L}' \supseteq \mathcal{L}$ are known, an adversary cannot forge a proof for some $x \notin \mathcal{L}'$. The main benefit over existing soundness notions is that $\mathcal{L}'$ does not even have to be known during the construction of the scheme. (For instance, our first proof system provides a "graceful soundness degradation", in the sense that it is sound in this sense for arbitrary linear languages $\mathcal{L}' \supseteq \mathcal{L}$.)

Overview over our constructions. Apart from the abstraction, we also provide generic and setting-specific constructions of benign proof systems. Our generic constructions (for a linear, and a "dynamically parameterized" linear language) can be viewed as abstractions and generalizations of universal hash proof systems. For $\mathcal{L}' = \mathcal{L}$, soundness in the above sense follows immediately from the correctness property of hash proof systems. (Indeed, hash proofs

for valid instances $x \in \mathcal{L}$ are unique and completely determined by public information.) For $\mathcal{L}' \supsetneq \mathcal{L}$, we will use additional properties of specific (existing) hash proof systems. In fact, the mentioned "graceful degradation" guarantees have already been used implicitly in the work of [12].

However, we also consider a somewhat nonstandard (and in our application crucial) "OR-language". Here, we give a prime-order instantiation in pairing-friendly groups (which is directly implied by the universal hash proof systems for disjunctions from [1]), and a new instance in the DCR setting. This DCR instance will be the key to the DCR-based instantiation of our KEM.

5.1 Definition

Definition 5.1 (Proof system). *Let $\mathcal{L} = \{\mathcal{L}_{pars}\}$ be a family of languages[5] with $\mathcal{L}_{pars} \subseteq \mathcal{X}_{pars}$, and with efficiently computable witness relation $\mathcal{R}$. A non-interactive designated-verifier proof system (NIDVPS)* $\mathbf{PS} = (\mathbf{PGen}, \mathbf{PPrv}, \mathbf{PVer}, \mathbf{PSim})$ *for $\mathcal{L}$ consists of the following PPT algorithms:*

- $\mathbf{PGen}(1^\lambda, pars)$ *outputs a keypair (ppk, psk). We call ppk the public and psk the private key.*
- $\mathbf{PPrv}(ppk, x, w)$*, for $x \in \mathcal{L}$ and $\mathcal{R}(x, w) = 1$, outputs a proof π.*
- $\mathbf{PVer}(psk, x, \pi)$*, for $x \in \mathcal{X}$ and a proof π, outputs a verdict $b \in \{0, 1\}$.*
- $\mathbf{PSim}(psk, x)$*, for $x \in \mathcal{L}$, outputs a proof π.*

We require correctness in the following sense:

Completeness. *For all pars, all (ppk, psk) in the range of $\mathbf{PGen}(1^\lambda, pars)$, all $x \in \mathcal{L}$, and all w with $\mathcal{R}(pars, x, w) = 1$, we always have $\mathbf{PVer}(psk, x, \mathbf{PPrv}(ppk, x, w)) = 1$.*

All relevant security properties of a NIDVPS are condensed in the following definition.

Definition 5.2 (Benign proof system). *Let $\mathbf{PS}$ be an NIDVPS for $\mathcal{L}$ as in Definition 5.1, and let $\mathcal{L}^{\mathrm{sim}} = \{\mathcal{L}^{\mathrm{sim}}_{pars}\}$, $\mathcal{L}^{\mathrm{ver}} = \{\mathcal{L}^{\mathrm{ver}}_{pars}\}$, and $\mathcal{L}^{\mathrm{snd}} = \{\mathcal{L}^{\mathrm{snd}}_{pars}\}$ be families of languages. We say that $\mathbf{PS}$ is* $(\mathcal{L}^{\mathrm{sim}}, \mathcal{L}^{\mathrm{ver}}, \mathcal{L}^{\mathrm{snd}})$-benign *if the following properties hold:*

(Perfect) zero-knowledge. *For all pars, all (ppk, psk) in the range of $\mathbf{PGen}(1^\lambda, pars)$, and all $x \in \mathcal{L}$ and w with $\mathcal{R}(pars, x, w) = 1$, we have the following equivalence of distributions:*

$$\mathbf{PPrv}(ppk, x, w) \quad \equiv \quad \mathbf{PSim}(psk, x).$$

(Statistical) $(\mathcal{L}^{\mathrm{sim}}, \mathcal{L}^{\mathrm{ver}}, \mathcal{L}^{\mathrm{snd}})$**-soundness.** *Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

1. *$\mathcal{A}$ is run on input 1^λ, and chooses pars.*
2. *$\mathcal{C}$ generates $(ppk, psk) \leftarrow \mathbf{PGen}(1^\lambda, pars)$.*
3. *$\mathcal{A}$ is run again on input $(1^\lambda, ppk)$, and with (many-time) access to oracles $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ that operate as follows:*
   - *$\mathcal{O}_{\mathbf{sim}}(x)$ checks if $x \in \mathcal{L}^{\mathrm{sim}}_{pars}$, and if yes, returns $\mathbf{PSim}(psk, x)$. Otherwise, $\mathcal{O}_{\mathbf{sim}}$ returns $\perp$.*
   - *$\mathcal{O}_{\mathbf{ver}}(x, \pi)$ checks if $x \in \mathcal{L}^{\mathrm{ver}}_{pars}$, and, if so, returns $\mathbf{PVer}(psk, x, \pi)$. Otherwise, $\mathcal{O}_{\mathbf{ver}}$ returns $\perp$.*

Finally, $\mathcal{A}$ wins iff it has queried $\mathcal{O}_{\mathbf{ver}}$ with (x, π) such that $x \in \mathcal{X}_{pars} \setminus \mathcal{L}^{\mathrm{snd}}_{pars}$ and $\mathbf{PVer}(psk, x, \pi) = 1$. Let $\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS},\mathcal{A}}(\lambda)$ the probability that $\mathcal{A}$ wins. We require that for all (not necessarily computationally bounded) $\mathcal{A}$ that only make a polynomial number of oracle queries, $\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS},\mathcal{A}}(\lambda)$ is negligible.

Intuitively, the soundness condition of Definition 5.2 thus states that no proofs for $\mathcal{X} \setminus \mathcal{L}^{\mathrm{snd}}_{pars}$-statements can be forged, even when (simulated) proofs for $\mathcal{L}^{\mathrm{sim}}_{pars}$-statements are available, and proofs for $\mathcal{L}^{\mathrm{ver}}_{pars}$-statements can be verified.

[5] These languages may also implicitly depend on the global public parameters pp.

5.2 The generic linear language

We will be interested in proof systems for "linear languages", in the sense that instances are vectors of group elements, and the language is closed under vector addition (i.e., component-wise group operation).

In the following, let $D \in \mathbb{N}$ and $\mathbf{pk} = (epk_1, \ldots, epk_D) = (g_1^{\omega_1^\top \mathbf{B}}, \ldots, g_1^{\omega_D^\top \mathbf{B}}) \in (\mathbb{G}_1^{\ell_\mathbf{B}})^D$. For a concise notation, write $\mathbf{\Omega} = (\omega_1 \| \ldots \| \omega_D) \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B} \times D}$. Also, fix a $\mathbb{Z}_{|\mathbb{G}_2|}$-module

$$\mathfrak{U} \;=\; \{\mathbf{M}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}_{|\mathbb{G}_2|}^d\} \;\subseteq\; \mathbb{Z}_{|\mathbb{G}_2|}^D \tag{11}$$

defined by a matrix $\mathbf{M} \in \mathbb{Z}_{|\mathbb{G}_2|}^{D \times d}$. Our languages are parameterized over $pars_{\mathrm{lin}} = (\mathbf{pk}, \mathbf{M})$, although $\mathcal{L}_{\mathbf{pk}}^{\mathrm{lin}}$ only depends on $\mathbf{pk}$, and not on $\mathbf{M}$. Namely, consider

$$\begin{aligned} \mathcal{L}_{\mathbf{pk}}^{\mathrm{lin}} &= \left\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}) \;\middle|\; \mathbf{r} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}},\ \mathbf{u} = \mathbf{0} \in \mathbb{Z}_{|\mathbb{G}_2|}^D\right\} \\ \mathcal{L}_{\mathrm{sim},(\mathbf{pk},\mathbf{M})}^{\mathrm{lin}} &= \mathcal{L}_{\mathrm{ver},(\mathbf{pk},\mathbf{M})}^{\mathrm{lin}} \;=\; \mathcal{L}_{\mathrm{snd},(\mathbf{pk},\mathbf{M})}^{\mathrm{lin}} \\ &= \left\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}) \;\middle|\; \mathbf{r} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}},\ \mathbf{u} \in \mathfrak{U}\right\} \\ \mathcal{X}^{\mathrm{lin}} &= \mathbb{G}^{\ell_\mathbf{B}+D}, \end{aligned} \tag{12}$$

and set $\mathcal{L}^{\mathrm{lin}} = \{\mathcal{L}_{\mathbf{pk}}^{\mathrm{lin}}\}$ and $\mathcal{L}_{\mathrm{sim}}^{\mathrm{lin}} = \mathcal{L}_{\mathrm{ver}}^{\mathrm{lin}} = \mathcal{L}_{\mathrm{snd}}^{\mathrm{lin}} = \{\mathcal{L}_{\mathrm{sim},(\mathbf{pk},\mathbf{M})}^{\mathrm{lin}}\}$. A witness for $x \in \mathcal{L}_{\mathbf{pk}}^{\mathrm{lin}}$ is $\mathbf{r}$.

5.2.1 A generic construction

Our generic construction is a simple variant of a universal hash proof system [9]. Like $\mathcal{L}_{\mathbf{pk}}^{\mathrm{lin}}$, our construction itself only depends on $\mathbf{pk}$, and not on $\mathbf{M}$. Hence, parameter generation only takes $\mathbf{pk}$ as input. Specifically, our $(\mathcal{L}_{\mathrm{sim}}^{\mathrm{lin}}, \mathcal{L}_{\mathrm{ver}}^{\mathrm{lin}}, \mathcal{L}_{\mathrm{snd}}^{\mathrm{lin}})$-benign proof system $\mathbf{PS}^{\mathrm{lin}}$ for $\mathcal{L}^{\mathrm{lin}}$ is given by the following algorithms:

- $\mathbf{PGen}^{\mathrm{lin}}(1^\lambda, \mathbf{pk})$, for $\mathbf{pk} = (g_1^{\omega_1^\top \mathbf{B}}, \ldots, g_1^{\omega_D^\top \mathbf{B}}) \in \mathbb{G}_1^{D \times \ell_\mathbf{B}}$, uniformly chooses $\mathbf{s} \in \mathbb{Z}_{|\mathbb{G}|}^{\ell_\mathbf{B}}$ and $\mathbf{t} \in \mathbb{Z}_{|\mathbb{G}|}^D$, and then computes and outputs

$$\begin{aligned} ppk_{\mathrm{lin}} &= g_1^{\mathbf{w}^\top} = g_1^{\mathbf{s}^\top \mathbf{B} + \mathbf{t}^\top \mathbf{\Omega}^\top \mathbf{B}} \in \mathbb{G}_1^{\ell_\mathbf{B}} \\ psk_{\mathrm{lin}} &= (\mathbf{s}, \mathbf{t}). \end{aligned} \tag{13}$$

- $\mathbf{PPrv}^{\mathrm{lin}}(ppk_{\mathrm{lin}}, x, \mathbf{r})$ (with $ppk_{\mathrm{lin}} = g_1^{\mathbf{w}^\top} \in \mathbb{G}_1^{\ell_\mathbf{B}}$ and $x = \mathbf{E}_{\mathbf{pk}}(\mathbf{0};\mathbf{r})$) computes and outputs

$$\pi_{\mathrm{lin}} \;=\; g_1^{\mathbf{w}^\top \mathbf{r}} \;\in\; \mathbb{G}_1.$$

- $\mathbf{PVer}^{\mathrm{lin}}(psk_{\mathrm{lin}}, x, \pi_{\mathrm{lin}})$ (with $psk_{\mathrm{lin}} = (\mathbf{s}, \mathbf{t})$ as above, $x = (g^\gamma, g^\delta) \in \mathbb{G}^{\ell_\mathbf{B}} \times \mathbb{G}^D$, and $\pi_{\mathrm{lin}} \in \mathbb{G}$) outputs 1 iff

$$\pi_{\mathrm{lin}} \;=\; g^{\mathbf{s}^\top \gamma + \mathbf{t}^\top \delta}. \tag{14}$$

- $\mathbf{PSim}^{\mathrm{lin}}(psk_{\mathrm{lin}}, x)$ (for $psk_{\mathrm{lin}} = (\mathbf{s}, \mathbf{t})$ and $x = (g^\gamma, g^\delta)$ as above) computes π_{lin} as in (14).

Correctness and the zero-knowledge property of $\mathbf{PS}^{\mathrm{lin}}$ follow from

$$g^{\mathbf{s}^\top \gamma + \delta^\top \mathbf{t}} = g_1^{\mathbf{s}^\top \mathbf{B}\mathbf{r} + \mathbf{t}^\top \mathbf{\Omega}^\top \mathbf{B}\mathbf{r}} = g_1^{\mathbf{w}^\top \mathbf{r}}$$

for $g^\gamma = g_1^{\mathbf{B}\mathbf{r}}$ and $g^\delta = g_1^{\mathbf{\Omega}^\top \mathbf{B}\mathbf{r}}$. The soundness property is proved conceptually similarly to the smoothness of universal hash proof systems [9] (see also [22]):

Lemma 5.3. $\mathbf{PS}^{\mathrm{lin}}$ *is statistically* $(\mathcal{L}_{\mathrm{sim}}^{\mathrm{lin}}, \mathcal{L}_{\mathrm{ver}}^{\mathrm{lin}}, \mathcal{L}_{\mathrm{snd}}^{\mathrm{lin}})$*-sound in the sense of Definition 5.2. Concretely, for an adversary* $\mathcal{A}$ *in the soundness game from Definition 5.2 that makes at most* $q = q(\lambda)$ *oracle queries,*

$$\mathrm{Adv}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{A}}^{\mathrm{snd}}(\lambda) \;\leq\; q/2^\lambda. \tag{15}$$

Proof. Fix any $\mathbf{pk} = (g_1^{\omega_1^\top \mathbf{B}}, \ldots, g_1^{\omega_D^\top \mathbf{B}}) \in (\mathbb{G}_1^{\ell_\mathbf{B}})^D$ and $\mathbf{M} \in \mathbb{Z}_{|\mathbb{G}_2|}^{D\times d}$. Let us consider the information made available to $\mathcal{A}$ through ppk_{lin} and oracle queries:

- By (13), ppk_{lin} only reveals $\mathbf{s}^\top\mathbf{B} + \mathbf{t}^\top\boldsymbol{\Omega}^\top\mathbf{B} = (\mathbf{s}+\boldsymbol{\Omega}\mathbf{t})^\top\mathbf{B} \bmod |\mathbb{G}_1|$.
- Now consider an $\mathcal{O}_{\mathbf{sim}}$ query on some $x = (g^\gamma, g^\delta)$. We may assume that $x \in \mathcal{L}^{\text{lin}}_{\text{sim}}$ (since otherwise, $\mathcal{O}_{\mathbf{sim}}(x) = \perp$), which means that $x = \mathbf{E}_{\mathbf{pk}}(\mathbf{M}\mathbf{x};\mathbf{r})$ for some $\mathbf{r} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$ and $\mathbf{x} \in \mathbb{Z}_{|\mathbb{G}_2|}^{d}$. Hence, by (14), the corresponding proof π_{lin} computed by $\mathcal{O}_{\mathbf{sim}}$ only depends on ppk_{lin} and $\mathbf{t}^\top\mathbf{M} \bmod |\mathbb{G}_2|$.
- Since proofs π_{lin} are unique, the result of any $\mathcal{O}_{\mathbf{ver}}$ query can be computed from the result of the corresponding $\mathcal{O}_{\mathbf{sim}}$ query. Hence, $\mathcal{O}_{\mathbf{ver}}$ queries do not reveal more information than $\mathcal{O}_{\mathbf{sim}}$ queries.

Hence, $\mathcal{A}$'s view depends only on $(\mathbf{s}+\boldsymbol{\Omega}\mathbf{t})^\top\mathbf{B} \bmod |\mathbb{G}_1|$ and $\mathbf{t}^\top\mathbf{M} \bmod |\mathbb{G}_2|$. It remains to bound the probability that $\mathcal{A}$ manages to submit an $\mathcal{O}_{\mathbf{ver}}$ query (x, π_{lin}) with $x \in \mathcal{X}^{\text{lin}} \setminus \mathcal{L}^{\text{lin}}_{\text{snd},(\mathbf{pk},\mathbf{M})}$ and a valid proof π_{lin}. So consider any $\mathcal{O}_{\mathbf{ver}}$ query (x, π_{lin}) with $x = (g^\gamma, g^\delta) \in \mathcal{X}^{\text{lin}} \setminus \mathcal{L}^{\text{lin}}_{\text{snd},(\mathbf{pk},\mathbf{M})}$. This means that $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ cannot be represented as $\boldsymbol{\gamma} = \frac{|\mathbb{G}|}{|\mathbb{G}_1|}\cdot\mathbf{B}\mathbf{r} \bmod |\mathbb{G}|$ and $\boldsymbol{\delta} = \frac{|\mathbb{G}|}{|\mathbb{G}_1|}\cdot\boldsymbol{\Omega}^\top\mathbf{B}\mathbf{r} + \frac{|\mathbb{G}|}{|\mathbb{G}_2|}\cdot\mathbf{M}\mathbf{x} \bmod |\mathbb{G}|$ for suitable $\mathbf{r}$ and $\mathbf{x}$. Hence, since $|\mathbb{G}|$ was assumed to be square-free, there is a prime factor p of $|\mathbb{G}|$ such that $\boldsymbol{\gamma} \bmod p$ and $\boldsymbol{\delta} \bmod p$ cannot be represented as $\boldsymbol{\gamma} = \frac{|\mathbb{G}|}{|\mathbb{G}_1|}\cdot\mathbf{B}\mathbf{r} \bmod p$ and $\boldsymbol{\delta} = \frac{|\mathbb{G}|}{|\mathbb{G}_1|}\cdot\boldsymbol{\Omega}^\top\mathbf{B}\mathbf{r} + \frac{|\mathbb{G}|}{|\mathbb{G}_2|}\cdot\mathbf{M}\mathbf{x} \bmod p$.

Let $\mathfrak{V}_p \subseteq \mathbb{Z}_p^{\ell_\mathbf{B}+D}$ be the vector space spanned by the columns of $\frac{|\mathbb{G}|}{|\mathbb{G}_1|}\cdot\begin{pmatrix}\mathbf{B}\\ \boldsymbol{\Omega}^\top\mathbf{B}\end{pmatrix} \bmod p$, and by the columns of $\frac{|\mathbb{G}|}{|\mathbb{G}_2|}\cdot\begin{pmatrix}\mathbf{0}\\ \mathbf{M}\end{pmatrix} \bmod p$, where $\mathbf{0} \in \mathbb{Z}_p^{\ell_\mathbf{B}\times d}$ is the all-zero matrix. By the discussion above, the view of $\mathcal{A}$ only depends on $(\mathbf{s}^\top, \mathbf{t}^\top)\cdot\mathfrak{U}_p$ (i.e., on the value of the inner products $(\mathbf{s}^\top, \mathbf{t}^\top)\cdot\mathbf{v} \bmod p$ for suitable $\mathbf{v} \in \mathfrak{U}_p$). However, for any $\mathbf{v} \in \mathbb{Z}_p^{\ell_\mathbf{B}+D} \setminus \mathfrak{U}_p$, we have that $(\mathbf{s}^\top, \mathbf{t}^\top)\cdot\mathbf{v} \bmod p$ is independent of $\mathcal{A}$'s view. In particular, $\begin{pmatrix}\boldsymbol{\gamma}\\ \boldsymbol{\delta}\end{pmatrix} \bmod p \notin \mathfrak{U}_p$, and thus, $\mathbf{s}^\top\boldsymbol{\gamma} + \mathbf{t}^\top\boldsymbol{\delta} \bmod p$ looks random given $\mathcal{A}$'s view. Since $p > 2^\lambda$ by assumption about $|\mathbb{G}|$, this means that $\mathcal{A}$ would have to predict an independently random value $\mathbf{s}^\top\boldsymbol{\gamma} + \mathbf{t}^\top\boldsymbol{\delta} \bmod p$ in order to submit an $x \in \mathcal{X}^{\text{lin}} \setminus \mathcal{L}^{\text{lin}}_{\text{snd},(\mathbf{pk},\mathbf{M})}$ with a valid proof π_{lin}. Hence, the probability that any particular query (x, π_{lin}) with $x \in \mathcal{X}^{\text{lin}} \setminus \mathcal{L}^{\text{lin}}_{\text{snd},(\mathbf{pk},\mathbf{M})}$ has a valid proof π_{lin} is bounded by $1/2^\lambda$. Since $\mathcal{A}$ makes at most q such queries, we get (15). □

Summing up, we get

Theorem 5.4. $\mathbf{PS}^{\text{lin}}$ *is an* $(\mathcal{L}^{\text{lin}}_{\text{sim}}, \mathcal{L}^{\text{lin}}_{\text{ver}}, \mathcal{L}^{\text{lin}}_{\text{snd}})$*-benign NIDVPS for* $\mathcal{L}^{\text{lin}}$.

5.3 A dynamically parameterized linear language

In our scheme, we will also use a slight variant of the generic linear language above. Specifically, we will consider a simple "dynamically parameterized" linear language, where one parameter (i.e., coefficient) is determined by the language instance. For a formal description, let $pars_{\text{hash}} = \mathbf{pk} = (epk_1, epk_2) \in (\mathbb{G}_1^{\ell_\mathbf{B}})^2$, and

$$\begin{aligned}
\mathcal{L}^{\text{hash}}_{\mathbf{pk}} &= \left\{\left(\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}),\ \tau\right) \mid \mathbf{u} = \mathbf{0} \in \mathbb{Z}^2_{|\mathbb{G}_2|}\right\}\\
\mathcal{L}^{\text{hash}}_{\text{sim},\mathbf{pk}} &= \mathcal{L}^{\text{hash}}_{\text{ver},\mathbf{pk}} = \mathcal{L}^{\text{hash}}_{\text{snd},\mathbf{pk}}\\
&= \left\{\left(\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}),\ \tau\right) \mid \mathbf{u} = (u_1, u_2)^\top \in \mathbb{Z}^2_{|\mathbb{G}_2|},\ u_2 = \tau u_1\right\}\\
\mathcal{X}^{\text{hash}}_{\mathbf{pk}} &= \left\{\left(\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}),\ \tau\right) \mid \mathbf{u} \in \mathbb{Z}^2_{|\mathbb{G}_2|}\right\},
\end{aligned} \tag{16}$$

where $\mathbf{r}$ and τ always range over $\mathbb{Z}_{|\mathbb{G}_1|}^{\ell_\mathbf{B}}$ and $\mathbb{Z}_{|\mathbb{G}_2|}$, respectively. A witness for $x \in \mathcal{L}^{\text{hash}}$ is $\mathbf{r}$. The families $\mathcal{L}^{\text{hash}}$, $\mathcal{L}^{\text{hash}}_{\text{sim}}$, $\mathcal{L}^{\text{hash}}_{\text{ver}}$, and $\mathcal{X}^{\text{hash}}$ are defined in the obvious way.

5.3.1 A generic construction

We can give a generic construction of a benign proof system for $\mathcal{L}^{\text{hash}}$ essentially by combining two instances of our generic benign proof system $\mathbf{PS}^{\text{lin}}$ from Section 5.2.1. That is, our $(\mathcal{L}^{\text{hash}}_{\text{sim}}, \mathcal{L}^{\text{hash}}_{\text{ver}}, \mathcal{L}^{\text{hash}}_{\text{snd}})$-benign proof system $\mathbf{PS}^{\text{hash}}$ for $\mathcal{L}^{\text{hash}}$ is given by the following algorithms:

- $\mathbf{PGen}^{\text{hash}}(1^\lambda, \mathbf{pk})$ samples two public keys

$$(ppk_{\text{lin}}, psk_{\text{lin}}) \leftarrow \mathbf{PGen}^{\text{lin}}(1^\lambda, epk_1) \qquad (ppk'_{\text{lin}}, psk'_{\text{lin}}) \leftarrow \mathbf{PGen}^{\text{lin}}(1^\lambda, epk_2)$$

  and outputs $ppk_{\text{hash}} = (ppk_{\text{lin}}, ppk'_{\text{lin}})$ and $psk_{\text{hash}} = (psk_{\text{lin}}, psk'_{\text{lin}})$.
- $\mathbf{PPrv}^{\text{hash}}(ppk_{\text{hash}}, x, \mathbf{r})$ (for $ppk_{\text{lin}} = (ppk_{\text{lin}}, ppk'_{\text{lin}})$ and $x = (\mathbf{c}, \tau)$ for $\mathbf{c} = (g^\gamma, g^\delta) = \mathbf{E}(\mathbf{0}; \mathbf{r})$ and $\delta = (\delta_1, \delta_2)$) computes

$$\pi_{\text{lin}} = \mathbf{PPrv}^{\text{lin}}(ppk_{\text{lin}}, (g^\gamma, g^{\delta_1}), \mathbf{r}) \qquad \pi'_{\text{lin}} = \mathbf{PPrv}^{\text{lin}}(ppk'_{\text{lin}}, (g^\gamma, g^{\delta_2}), \mathbf{r})$$

  and outputs $\pi_{\text{hash}} = \pi_{\text{lin}}^\tau \cdot \pi'_{\text{lin}} \in \mathbb{G}$.
- $\mathbf{PVer}^{\text{hash}}(psk_{\text{hash}}, x, \pi_{\text{hash}})$ (with $psk_{\text{hash}} = (psk_{\text{lin}}, psk'_{\text{lin}})$ and $x = (\mathbf{c}, \tau)$ with $\mathbf{c} = (g^\gamma, g^\delta)$ for $\delta = (\delta_1, \delta_2)$), outputs 1 iff

$$\pi_{\text{hash}} = \big(\mathbf{PSim}^{\text{lin}}(psk_{\text{lin}}, (g^\gamma, g^{\delta_1}))\big)^\tau \cdot \mathbf{PSim}^{\text{lin}}(psk'_{\text{lin}}, (g^\gamma, g^{\delta_2})). \tag{17}$$

- $\mathbf{PSim}^{\text{hash}}(psk_{\text{hash}}, x)$ (for psk_{hash} and x as in $\mathbf{PVer}^{\text{hash}}$), outputs π_{hash} as defined in (17).

Our proof system is generic in the sense that it does not make any assumption about the underlying group structure. (Specifically, it works both in the prime-order, and in the DCR setting.) However, we stress that it builds on the specific (generic) proof system $\mathbf{PS}^{\text{lin}}$ from Section 5.2.1.

Completeness and the zero-knowledge property of $\mathbf{PS}^{\text{hash}}$ follow from those properties of $\mathbf{PS}^{\text{lin}}$. Soundness can be proved with the same ideas as for $\mathbf{PS}^{\text{lin}}$:

Lemma 5.5. $\mathbf{PS}^{\text{hash}}$ *is statistically* $(\mathcal{L}^{\text{hash}}_{\text{sim}}, \mathcal{L}^{\text{hash}}_{\text{ver}}, \mathcal{L}^{\text{hash}}_{\text{snd}})$*-sound in the sense of Definition 5.2. Concretely, for an adversary* $\mathcal{A}$ *in the soundness game from Definition 5.2 that makes at most* $q = q(\lambda)$ *oracle queries,*

$$\text{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{hash}},\mathcal{A}}(\lambda) \leq q/2^\lambda. \tag{18}$$

Proof. We proceed as in the proof of Lemma 5.3, and in particular use the notation from there. In that notation, ppk_{hash} only reveals $(\mathbf{s} + \boldsymbol{\omega}_1 t)^\top \mathbf{B} \bmod |\mathbb{G}_1|$ and $(\mathbf{s}' + \boldsymbol{\omega}_2 t')^\top \mathbf{B} \bmod |\mathbb{G}_1|$ about the secret key $psk_{\text{hash}} = (psk_{\text{lin}}, psk'_{\text{lin}}) = ((\mathbf{s}, t), (\mathbf{s}', t'))$.

Furthermore, consider an $\mathcal{O}_{\mathbf{sim}}$ query $x = (g^\gamma, g^\delta) \in \mathcal{L}^{\text{hash}}_{\text{sim}}$ with $\delta = (\delta_1, \delta_2)$. By definition of $\mathcal{L}^{\text{hash}}_{\text{sim}}$, we can write

$$\begin{pmatrix} \gamma \\ \delta_1 \\ \delta_2 \end{pmatrix} = \frac{|\mathbb{G}|}{|\mathbb{G}_1|} \cdot \begin{pmatrix} \mathbf{1} \\ \boldsymbol{\omega}_1^\top \\ \boldsymbol{\omega}_2^\top \end{pmatrix} \mathbf{B}\mathbf{r} + \frac{|\mathbb{G}|}{|\mathbb{G}_2|} \cdot \begin{pmatrix} 0 \\ u_1 \\ u_2 \end{pmatrix} \bmod |\mathbb{G}| \tag{19}$$

for the identity matrix $\mathbf{1} \in \mathbb{Z}_{|\mathbb{G}_1|}^{\ell_{\mathbf{B}} \times \ell_{\mathbf{B}}}$, $u_2 = \tau u_1$, and $u_1, u_2, \tau \in \mathbb{Z}_{|\mathbb{G}_2|}$. Hence, the query only reveals

$$\begin{aligned} \tau \cdot (\mathbf{s}^\top, t) \begin{pmatrix} \gamma \\ \delta_1 \end{pmatrix} + (\mathbf{s}'^\top, t') \begin{pmatrix} \gamma \\ \delta_2 \end{pmatrix} &\overset{(19)}{=} \frac{|\mathbb{G}|}{|\mathbb{G}_1|} \cdot \big(\tau(\mathbf{s} + \boldsymbol{\omega}_1 t) + \mathbf{s}' + \boldsymbol{\omega}_2 t'\big)^\top \mathbf{B}\mathbf{r} + \frac{|\mathbb{G}|}{|\mathbb{G}_2|}\big(\tau t u_1 + t' u_2\big) \\ &\overset{u_2 = \tau u_1}{=} \frac{|\mathbb{G}|}{|\mathbb{G}_1|} \cdot \big(\tau(\mathbf{s} + \boldsymbol{\omega}_1 t) + \mathbf{s}' + \boldsymbol{\omega}_2 t'\big)^\top \mathbf{B}\mathbf{r} + \frac{|\mathbb{G}|}{|\mathbb{G}_2|}(t + t')u_2 \bmod |\mathbb{G}|. \end{aligned}$$

Specifically, aside from information already revealed through ppk_{hash}, this query only depends on $t + t' \bmod |\mathbb{G}_2|$. Since an $\mathcal{O}_{\mathbf{ver}}$ query can be implemented with an $\mathcal{O}_{\mathbf{sim}}$ query in case of $\mathbf{PS}^{\text{hash}}$, the same holds for $\mathcal{O}_{\mathbf{ver}}$ queries. In particular, ppk and all oracle queries yield $2\ell_{\mathbf{B}} + 1$ linear equations over $\mathbb{Z}_{|\mathbb{G}|}$ for the $2\ell_{\mathbf{B}} + 2$ unknowns from $\mathbf{s}, \mathbf{s}', t, t'$. A similar argument as in the proof of Lemma 5.3 shows that any $\mathcal{O}_{\mathbf{ver}}$ query with $x \in \mathcal{X}^{\text{hash}} \setminus \mathcal{L}^{\text{hash}}_{\text{snd}}$ and a valid proof π_{hash} would have to correctly guess a previously unknown linear equation about $t, t' \bmod p$ for a suitable prime $p > 2^\lambda$. Since $\mathcal{A}$ makes at most q oracle queries, (18) follows as desired. □

Summing up, we obtain

Theorem 5.6. $\mathbf{PS}^{\mathsf{hash}}$ *is an* $(\mathcal{L}^{\mathsf{hash}}_{\mathsf{sim}}, \mathcal{L}^{\mathsf{hash}}_{\mathsf{ver}}, \mathcal{L}^{\mathsf{hash}}_{\mathsf{snd}})$*-benign NIDVPS for* $\mathcal{L}^{\mathsf{hash}}$.

5.4 The generic OR-language

We will also be interested in the following family $\mathcal{L}^{\vee}$, together with its "simulation", "verification" and "soundness" counterparts $\mathcal{L}^{\vee}_{\mathsf{sim}}$, $\mathcal{L}^{\vee}_{\mathsf{ver}}$ and $\mathcal{L}^{\vee}_{\mathsf{snd}}$. Here, the actual languages in $\mathcal{L}^{\vee}$ are linear like those in $\mathcal{L}^{\mathsf{lin}}$. However, soundness also holds when $\mathcal{L}^{\vee}_{\mathsf{sim}}$-instances are simulated, and those instances have an "OR flavor".

The language parameters are $pars_{\vee} = (\mathbf{pk}, \ell_{\vee})$ for $\mathbf{pk} = (epk_1, epk_2) \in (\mathbb{G}_1^{\ell_{\mathbf{B}}})^2$, and a function $\ell_{\vee} = \ell_{\vee}(\lambda)$. The families $\mathcal{L}^{\vee}$, $\mathcal{L}^{\vee}_{\mathsf{sim}}$, $\mathcal{L}^{\vee}_{\mathsf{ver}}$, $\mathcal{L}^{\vee}_{\mathsf{snd}}$, and $\mathcal{X}^{\vee}$ are comprised of the following languages, where we consider all $\mathbf{r} \in \mathbb{Z}^{\ell_{\mathbf{B}}}_{|\mathbb{G}_1|}$, and $\mathbf{u} = (u_1, u_2) \in (\mathbb{Z}^{*}_{|\mathbb{G}_2|} \cup \{0\})^2$:

$$\begin{aligned}
\mathcal{L}^{\vee}_{\mathbf{pk}} = \mathcal{L}^{\vee}_{\mathsf{ver},\mathbf{pk}} &= \left\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}) \mid u_1 = 0\right\} \\
\mathcal{L}^{\vee}_{\mathsf{sim},(\mathbf{pk},\ell_{\vee})} &= \left\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}) \mid u_1 = 0 \ \vee \ (|u_1| < 2^{\ell_{\vee}} \wedge u_2 = 0)\right\} \\
\mathcal{L}^{\vee}_{\mathsf{snd},\mathbf{pk}} &= \left\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}) \mid u_1 = 0 \vee u_2 = 0\right\} \\
\mathcal{X}^{\vee}_{\mathbf{pk}} &= \left\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r})\right\}.
\end{aligned}$$

Here, the value $|u_1|$ (in the definition of $\mathcal{L}^{\vee}_{\mathsf{sim},(\mathbf{pk},\ell_{\vee})}$ is to be understood simply as the absolute value for signed $\mathbb{Z}_{|\mathbb{G}_2|}$-values in the prime-order setting, and as $|u_1| = |u_1|_N$ in the DCR setting. Observe that $\mathcal{L}^{\vee}_{\mathbf{pk}} \subseteq \mathcal{L}^{\vee}_{\mathsf{sim},(\mathbf{pk},\ell_{\vee})} \subseteq \mathcal{L}^{\vee}_{\mathsf{snd},\mathbf{pk}} \subseteq \mathcal{X}^{\vee}_{\mathbf{pk}}$. A valid witness for $x \in \mathcal{L}^{\vee}$ is $\mathbf{r}$.

5.4.1 A construction in pairing-friendly groups

Now assume that $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ is a prime-order group equipped with a symmetric pairing. Then, a benign proof system for $\mathcal{L}^{\vee}$ can be constructed from the universal hash proof systems for disjunctions from [1]. Specifically, [1] construct universal hash proof systems for languages of the form $\mathcal{L} = \{(x_1, x_2) \mid x_1 \in \mathcal{L}_1 \vee x_2 \in \mathcal{L}_2\}$, where $\mathcal{L}_i \subseteq \mathbb{G}^{\ell}$ are linear languages (i.e., vector spaces over $\mathbb{Z}_{|\mathbb{G}|}$). In our case, given $\mathbf{pk} = (epk_1, epk_2)$, we can thus set

$$\begin{aligned}
\mathcal{L}_1 &= \left\{\mathbf{E}_{epk_1}(0;\mathbf{r})\right\} \\
\mathcal{L}_2 &= \left\{\mathbf{E}_{epk_2}(0;\mathbf{r})\right\} \\
\mathcal{L} &= \left\{x = (\mathbf{c}_0, c_1, c_2) \mid (\mathbf{c}_0, c_1) \in \mathcal{L}_1 \vee (\mathbf{c}_0, c_2) \in \mathcal{L}_2\right\}.
\end{aligned} \tag{20}$$

Invoking [1] with these languages yields a NIDVPS $\mathbf{PS}^{\vee}_{\mathbf{pair}}$ that achieves:

Theorem 5.7. $\mathbf{PS}^{\vee}_{\mathbf{pair}}$ *is an* $(\mathcal{L}^{\vee}_{\mathsf{sim}}, \mathcal{L}^{\vee}_{\mathsf{ver}}, \mathcal{L}^{\vee}_{\mathsf{snd}})$*-benign NIDVPS for* $\mathcal{L}^{\vee}$.

5.4.2 A construction in the DCR setting

In the following, we assume an $N = PQ$, and groups $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$ as in Section 3.3. In particular, we have $\ell_{\mathbf{B}} = 1$, and $\mathbf{B}$ is the trivial (identity) matrix. Furthermore, fix an $\ell_{\vee} = \ell_{\vee}(\lambda)$. We additionally assume that $P, Q > 2^{\ell_{\vee}+4\lambda}$. Recall that $g_1, epk_1, epk_2 \in \mathbb{G}_1$ are of order $|\mathbb{G}_1| = \varphi(N)/4$, and that $g_2 \in \mathbb{G}_2$ is of order $|\mathbb{G}_2| = N$.

Our $(\mathcal{L}^{\vee}_{\mathsf{sim}}, \mathcal{L}^{\vee}_{\mathsf{ver}}, \mathcal{L}^{\vee}_{\mathsf{snd}})$-benign proof system $\mathbf{PS}^{\vee}_{\mathbf{DCR}}$ for $\mathcal{L}^{\vee}$ is given by the following algorithms:

- $\mathbf{PGen}^{\vee}(1^{\lambda})$ uniformly picks $s_1, s_2 \in \mathbb{Z}_{\lfloor N^2/4 \rfloor}$ and outputs $ppk_{\vee} = (epk_1^{s_1}, epk_1^{s_2})$ and $psk_{\vee} = (s_1, s_2)$.

- $\mathbf{PPrv}^{\vee}(ppk_{\vee}, x, r)$ (with $ppk_{\vee} = (epk_1^{s_1}, epk_1^{s_2})$, and $x = (c_0, c_1, c_2) = (g_1^r, epk_1^r, epk_2^r g_2^{u_2})$) uniformly chooses $t_1, t_2 \in \mathbb{Z}_N$, and outputs

$$\pi_{\vee} \;=\; (\pi_0, \pi_1, \pi_2) \;=\; \big(c_2^{t_1 + N \cdot t_2},\; (epk_1^{s_1})^r \cdot g_2^{t_1},\; (epk_1^{s_2})^r \cdot g_2^{t_2}\big).$$

- $\mathbf{PVer}^{\vee}(psk_{\vee}, x, \pi_{\vee})$ (with $psk = (s_1, s_2)$, $x = (c_0, c_1, c_2)$, and $\pi_{\vee} = (\pi_0, \pi_1, \pi_2)$) first checks that $\pi_1 / c_1^{s_1} = g_2^{t_1}$ and $\pi_2 / c_1^{s_2} = g_2^{t_2}$ for some $t_1, t_2 \in \mathbb{Z}_N$ (and outputs 0 if not). **PVer** then computes[6] these t_1, t_2, and outputs 1 iff $\pi_0 = c_2^{t_1 + N \cdot t_2}$.
- $\mathbf{PSim}^{\vee}(psk_{\vee}, x)$ (for $psk = (s_1, s_2)$ and $x = (c_0, c_1, c_2)$) uniformly picks $t_1, t_2 \in \mathbb{Z}_{N^2}$ and outputs

$$\pi_{\vee} \;=\; (\pi_0, \pi_1, \pi_2) \;=\; \big(c_2^{t_1 + N \cdot t_2},\; c_1^{s_1} \cdot g_2^{t_1},\; c_1^{s_2} \cdot g_2^{t_2}\big).$$

The completeness and zero-knowledge properties of $\mathbf{PS}^{\vee}_{\mathbf{DCR}}$ follow directly from the fact that $c_1^{s_i} = (epk_1^r)^{s_i} = (epk_1^{s_i})^r$. To show the soundness of $\mathbf{PS}^{\vee}_{\mathbf{DCR}}$, we prove a helpful technical lemma:

Lemma 5.8. *Let s_1, s_2, t_1, t_2 be distributed as in $\mathbf{PS}^{\vee}_{\mathbf{DCR}}$, and fix any $u \in \mathbb{Z}$ with $|u| < 2^{\ell_{\vee}}$. Let*[7]

$$aux \;:=\; ([s_1]_{\varphi(N)/4},\; [s_2]_{\varphi(N)/4},\; [t_1 + N \cdot t_2]_{\varphi(N)/4},\; [u s_1 + t_1]_N,\; [u s_2 + t_2]_N),$$

and write[8] $w_1 := [s_1/\alpha]_N$ *(with the division performed in $\mathbb{Z}_N$) for $\alpha := [N]_{\varphi(N)/4}$. Then, for an independently random $R \in \mathbb{Z}_{2^\lambda}$, we have*

$$\varepsilon \;:=\; \mathbf{SD}\big(([w_1]_{2^\lambda}, aux)\,;\,(R, aux)\big) \;\le\; 3/2^\lambda.$$

In other words, w_1 (and thus s_1) is unpredictable, even given *aux*.

Proof. Without loss of generality, assume $u \ge 0$. (For $u < 0$, we can invoke the lemma with $-u$, $-t_1$, and $-t_2$ in place of u, t_1 and t_2.) We proceed in steps, in each step modifying *aux*, and bounding the impact on ε. Specifically, in the following, we will define a number of random variables aux_i, and abbreviate $\varepsilon_i := \mathbf{SD}\big(([w_1]_{2^\lambda}, aux_i)\,;\,(R, aux_i)\big)$. As a starting point, consider

$$aux_1 \;:=\; ([t_1 + N \cdot t_2]_{\varphi(N)/4},\; [u s_1 + t_1]_N,\; [u s_2 + t_2]_N).$$

Now note that $w_1 = [s_1/\alpha]_N$ and the $[u s_i + t_i]_N$ (for $i \in \{1, 2\}$) only depend on $[s_i]_N$. However, our uniform choice of $s_i \in \mathbb{Z}_{\lfloor N^2/4 \rfloor}$ is statistically $2/2^{\ell_{\vee}+4\lambda}$-close to a uniform choice of $s_i \in \mathbb{Z}_{N \cdot \varphi(N)/4}$ (in which case $[s_i]_N$ and $[s_i]_{\varphi(N)/4}$ are independently and uniformly random). Hence, the $[s_i]_{\varphi(N)/4}$ are essentially independent of w_1 and aux_1, and we obtain $\varepsilon \le \varepsilon_1 + 4/2^{\ell_{\vee}+4\lambda}$. Next, consider

$$aux_2 \;:=\; ([t_1]_\alpha,\; [t_2]^\alpha,\; [t_1]^\alpha + [t_2]_\alpha,\; [u s_1 + t_1]_N,\; [u s_2 + t_2]_N).$$

Since $t_1 + \alpha \cdot t_2 = [t_1]_\alpha + \alpha \cdot ([t_1]^\alpha + [t_2]_\alpha) + \alpha^2 \cdot [t_2]^\alpha$, we have that aux_1 is a function of aux_2, and so $\varepsilon_1 \le \varepsilon_2$. Similarly, we can refine the last two components of aux_2 to obtain

$$aux_3 \;:=\; ([t_1]_\alpha,\; [t_2]^\alpha,\; [t_1]^\alpha + [t_2]_\alpha,\; [u s_1 + \alpha \cdot [t_1]^\alpha]_N,\; [u s_2 + [t_2]_\alpha]_N).$$

Again, $\varepsilon_2 \le \varepsilon_3$ since aux_3 fully defines aux_2. Similar to our first step, now $[t_1]_\alpha$ and $[t_2]^\alpha$ are essentially independent of the remaining parts of *aux* (up to a statistical defect of at most $2/2^{\ell_{\vee}+4\lambda}$ for each). Hence, for

$$aux_4 \;:=\; ([t_1]^\alpha + [t_2]_\alpha,\; [u s_1 + \alpha \cdot [t_1]^\alpha]_N,\; [u s_2 + [t_2]_\alpha]_N),$$

[6] Here, we implicitly use that computing discrete logarithms in $\mathbb{G}_2$ is easy, see Section 3.3.

[7] In this lemma and its proof, we heavily rely on the notation of $[s]_N$ and $[s]^N$ from Section 2.

[8] Here, we use our assumption that $[N]_{\varphi(N)/4} = P + Q - 1$ and N are coprime.

we get that $\varepsilon_3 \leq \varepsilon_4 + 4/2^{\ell_\vee+4\lambda}$. Now let $w_2 := [s_2]_N$, and consider

$$aux_5 \;:=\; ([t_1]^\alpha + [t_2]_\alpha,\; uw_1 + [t_1]^\alpha,\; uw_2 + [t_2]_\alpha).$$

Since aux_4 can be computed from aux_5, we have $\varepsilon_4 \leq \varepsilon_5$. Next, we release $w_1 + w_2$ (over $\mathbb{Z}$):

$$aux_6 \;:=\; ([t_1]^\alpha + [t_2]_\alpha,\; w_1 + w_2,\; uw_1 + [t_1]^\alpha).$$

Again, aux_5 can be computed from aux_6, and hence $\varepsilon_5 \leq \varepsilon_6$. Since we consider the statistical distance between $[w_1]_{2^\lambda}$ and R, we can release (and then drop) $[w_1]^{2^\lambda}$. Concretely, consider

$$\begin{aligned}
aux_7 &:= ([t_1]^\alpha + [t_2]_\alpha,\; [w_1]_{2^\lambda} + w_2,\; u \cdot [w_1]_{2^\lambda} + [t_1]^\alpha,\; [w_1]^{2^\lambda}),\\
aux_8 &:= ([t_1]^\alpha + [t_2]_\alpha,\; [w_1]_{2^\lambda} + w_2,\; u \cdot [w_1]_{2^\lambda} + [t_1]^\alpha)\\
aux_9 &:= ([t_1]^\alpha + [t_2]_\alpha,\; u \cdot [w_1]_{2^\lambda} + [t_1]^\alpha).
\end{aligned}$$

Here, $\varepsilon_6 \leq \varepsilon_7$ since aux_6 can be computed from aux_7. Moreover, recall that $N > 2^{2\ell_\vee+8\lambda}$ by our choice of $P, Q > 2^{\ell_\vee+4\lambda}$. Hence, $\varepsilon_7 \leq \varepsilon_8 + 1/2^{2\ell_\vee+7\lambda}$, since $[w_1]_{2^\lambda}$ and $[w_1]^{2^\lambda}$ are independent up to a statistical defect of at most $1/2^{2\ell_\vee+7\lambda}$. Finally, $\varepsilon_8 \leq \varepsilon_9 + 1/2^{2\ell_\vee+7\lambda}$, since w_2 is uniformly and independently random chosen from $\mathbb{Z}_N$.

Similarly, we can show that $[t_2]_\alpha$ blinds $[[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}}$:

$$\begin{aligned}
aux_{10} &:= ([[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}} + [t_2]_\alpha,\; u \cdot [w_1]_{2^\lambda} + [[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}},\; [[t_1]^\alpha]^{2^{\ell_\vee+2\lambda}}),\\
aux_{11} &:= ([[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}} + [t_2]_\alpha,\; u \cdot [w_1]_{2^\lambda} + [[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}},\\
aux_{12} &:= (u \cdot [w_1]_{2^\lambda} + [[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}}).
\end{aligned}$$

With the same reasoning as in aux_7-aux_9 (and using that $\alpha, N/\alpha > 2^{\ell_\vee+4\lambda}/2$ by $P, Q > 2^{\ell_\vee+4\lambda}$), we get $\varepsilon_9 \leq \varepsilon_{10}$, as well as $\varepsilon_{10} \leq \varepsilon_{11} + 1/2^{2\lambda}$, and $\varepsilon_{11} \leq \varepsilon_{12} + 1/2^{2\lambda}$. Finally, if we set $aux_{13} := ()$ to be the empty sequence, we get $\varepsilon_{12} \leq \varepsilon_{13} + 1/2^\lambda + 2/2^{\ell_\vee+4\lambda}$, since $[t_1]^\alpha$ is $2/2^{\ell_\vee+4\lambda}$-close to uniform over $\mathbb{Z}_{\lceil N/\alpha \rceil}$ (which implies that $[[t_1]^\alpha]_{2^{\ell_\vee+2\lambda}}$ blinds $u \cdot [w_1]_{2^\lambda}$). It is left to observe that $\varepsilon_{13} = \mathbf{SD}\big([w_1]_{2^\lambda}\,;\, R\big) \leq 1/2^{2\ell_\vee+7\lambda}$, since $w_1 \in \mathbb{Z}_N$ is uniformly random. Summing up, we get $\varepsilon \leq 1/2^\lambda + 2/2^{2\lambda} + 10/2^{\ell_\vee+4\lambda} + 3/2^{2\ell_\vee+7\lambda} \leq 3/2^\lambda$, as desired. □

We can now proceed to show the soundness of $\mathbf{PS}^\vee_{\mathbf{DCR}}$:

Lemma 5.9. $\mathbf{PS}^\vee_{\mathbf{DCR}}$ *is statistically* $(\mathcal{L}^\vee_{\text{sim}}, \mathcal{L}^\vee_{\text{ver}}, \mathcal{L}^\vee_{\text{snd}})$*-sound in the sense of Definition 5.2. Concretely, for an adversary* $\mathcal{A}$ *that makes at most* $q = q(\lambda)$ *oracle queries in the soundness game from Definition 5.2,*

$$\mathrm{Adv}^{\text{snd}}_{\mathbf{PS}^\vee_{\mathbf{DCR}},\mathcal{A}}(\lambda) \;\leq\; 4q/2^\lambda. \tag{21}$$

Proof. Fix $\ell_\vee$ and $\mathbf{pk}$, and let $\mathbf{view}_\mathcal{A}$ be $\mathcal{A}$'s view in a run of the computational soundness game from Definition 5.2. Specifically, $\mathbf{view}_\mathcal{A}$ consists of $\mathcal{A}$'s input $ppk_\vee = (epk_1^{s_1}, epk_1^{s_2})$, as well as all oracle queries (and the corresponding answers). We first consider to what extent $\mathbf{view}_\mathcal{A}$ determines the secret key $psk_\vee = (s_1, s_2)$.

- $\mathcal{A}$'s input $ppk_\vee = (epk_1^{s_1}, epk_1^{s_2})$ only depends on $[s_1]_{\varphi(N)/4}$ and $[s_2]_{\varphi(N)/4}$ (since epk_1 has order $\varphi(N)/4$).
- Each $\mathcal{O}_{\mathbf{sim}}$ oracle query of $\mathcal{A}$ reveals a value $\pi_\vee = (\pi_0, \pi_1, \pi_2) = (c_2^{t_1+N\cdot t_2}, c_1^{s_1} \cdot g_2^{t_1}, c_1^{s_2} \cdot g_2^{t_2})$ for $\mathcal{A}$-supplied c_1, c_2 and fresh t_1, t_2. We may assume that $c_1 = epk_1^r \cdot g_2^{u_1}$ and $c_2 = epk_2^r \cdot g_2^{u_2}$ with $u_1 = 0$ or $|u_1|_N < 2^{\ell_\vee} \wedge u_2 = 0$ (since otherwise, $\mathcal{O}_{\mathbf{sim}}$ rejects the query). Hence, such a query reveals

  $$(\pi_0, \pi_1, \pi_2) \;=\; (epk_2^{r(t_1+N\cdot t_2)} g_2^{u_2 t_1},\; epk_1^{rs_1} \cdot g_2^{u_1 s_1 + t_1},\; epk_1^{rs_2} \cdot g_2^{u_1 s_2 + t_2}),$$

  which only depends on $[s_1]_{\varphi(N)/4}$, $[s_2]_{\varphi(N)/4}$, $[t_1 + N\cdot t_2]_{\varphi(N)/4}$, $[u_2 t_1]_N$, as well as $[u_1 s_1 + t_1]_N$ and $[u_1 s_2 + t_2]_N$. Thus, if $u_1 = 0$, the query reveals only $[s_1]_{\varphi(N)/4}$ and $[s_2]_{\varphi(N)/4}$ about

(s_1, s_2). But if $u_1 \neq 0$ (and thus $u_2 = 0$), we can apply Lemma 5.8 with $u := u_1$, where we represent $u_1 \in \mathbb{Z}_N$ as an integer between $-N/2$ and $N/2$. This yields that the query leaves $[w_1]_{2^\lambda}$ undetermined, up to a small statistical defect. A hybrid argument over all of $\mathcal{A}$'s $\mathcal{O}_{\mathbf{sim}}$ queries shows that the overall statistical defect is bounded by $3q/2^\lambda$.

- An $\mathcal{O}_{\mathbf{ver}}$ query on input $(x, \pi_\vee)$ yields $\perp$ unless $x \in \mathcal{L}^\vee_{\mathrm{ver},(\mathbf{pk},\ell_\vee)} = \mathcal{L}^\vee_{(\mathbf{pk},\ell_\vee)}$. But for $x = (c_0, c_1, c_2) = (g_1^r, epk_1^r, epk_2^r g_2^{u_2}) \in \mathcal{L}^\vee_{(\mathbf{pk},\ell_\vee)}$, we get that $\mathcal{O}_{\mathbf{ver}}$'s output only depends on $c_1^{s_i} = epk_1^{rs_i}$, and hence only on $[s_i]_{\varphi(N)/4}$ (for $i = 1, 2$).

To summarize, $\mathbf{view}_\mathcal{A}$ is essentially independent of $[w_1]_{2^\lambda}$, up to a statistical defect of $3q/2^\lambda$.

It remains to prove that any $\mathcal{O}_{\mathbf{ver}}$ query on some $(x, \pi_\vee)$ with $x \in \mathcal{X}^\vee \setminus \mathcal{L}^\vee_{\mathrm{snd},\mathbf{pk}}$ (i.e., an x with $x = (c_0, c_1, c_2) = (g_1^r, epk_1^r \cdot g_2^{u_1}, epk_2^r \cdot g_2^{u_2})$ for $u_1, u_2 \in \mathbb{Z}_N^*$) is invalid in the sense that $\mathbf{PVer}(psk_\vee, x, \pi_\vee) = 0$ with high probability. To this end, write

$$\pi_\vee = (\pi_0, \pi_1, \pi_2) = (epk_2^{\rho_0} \cdot g_2^{\alpha_0},\ epk_1^{\rho_1} \cdot g_2^{\alpha_1},\ epk_1^{\rho_2} \cdot g_2^{\alpha_2})$$

for suitable $\rho_0, \rho_1, \rho_2, \alpha_0, \alpha_1, \alpha_2$. Recall that $(x, \pi_\vee)$ is valid only if for $i = 1, 2$, we have $\pi_i / c_1^{s_i} = g_2^{t_i}$ for some $t_i \in \mathbb{Z}_N$, and if $\pi_0 = c_2^{t_1 + N \cdot t_2}$ for those t_i. Hence, if $(x, \pi_\vee)$ is valid, then the following holds for some t_1, t_2:

$$\begin{aligned} \rho_0 &= [r(t_1 + N \cdot t_2)]_{\varphi(N)/4} & \alpha_0 &= [u_2 t_1]_N \\ \rho_1 &= [r s_1]_{\varphi(N)/4} & \alpha_1 &= [u_1 s_1 + t_1]_N \\ \rho_2 &= [r s_2]_{\varphi(N)/4} & \alpha_2 &= [u_1 s_2 + t_2]_N. \end{aligned}$$

By assumption, $u_2 \in \mathbb{Z}_N^*$, and thus α_0 determines t_1. Using also $u_1 \in \mathbb{Z}_N^*$, hence α_0 and α_1 determine $[s_1]_N$, and thus also $w_1 = [s_1/\alpha]_N$. However, as we have argued above, $\mathbf{view}_\mathcal{A}$ is essentially independent of $[w_1]_{2^\lambda}$. The probability that $\mathcal{A}$ correctly guesses an independently and uniformly random $[w_1]_{2^\lambda}$ with a single query is exactly $1/2^\lambda$. Since $\mathcal{A}$ makes at most q guesses, the probability for a correct guess is bounded by $q/2^\lambda$. Taking into account the mentioned statistical defect in $\mathbf{view}_\mathcal{A}$, we obtain (21). □

Taking things together, we obtain

Theorem 5.10. $\mathbf{PS}^\vee_{\mathbf{DCR}}$ *is an* $(\mathcal{L}^\vee_{\mathrm{sim}}, \mathcal{L}^\vee_{\mathrm{ver}}, \mathcal{L}^\vee_{\mathrm{snd}})$*-benign NIDVPS for* $\mathcal{L}^\vee$.

6 The key encapsulation scheme

In the following, we present our main construction of an IND-MCCA secure key encapsulation (KEM) scheme. (This directly implies a PKE scheme with the same security properties [8].)

6.1 The construction

Ingredients and public parameters. In our construction, we use the following ingredients:

- groups $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$ with $|\mathbb{G}_2| > 2^{3\lambda}$ (see Section 3.1.1 for a description of the generic setting),
- the generalized ElGamal scheme $(\mathbf{E}, \mathbf{D})$ implicitly defined through $\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$ (Section 3.1.3),
- an EUF-MOTCMA secure one-time signature scheme $\mathbf{OTS} = (\mathbf{SGen}, \mathbf{SSig}, \mathbf{SVer})$ (Section 4.1),
- a key extractor $\mathbf{EXT} = (\mathbf{ExtGen}, \mathbf{Ext}_{\mathrm{pub}}, \mathbf{Ext}_{\mathrm{priv}})$ for $\mathbb{G}$ (see Section 4.2) with $\ell_{\mathrm{ext}} = 3\lambda$,
- an $(\mathcal{L}^{\mathrm{lin}}_{\mathrm{sim}}, \mathcal{L}^{\mathrm{lin}}_{\mathrm{ver}}, \mathcal{L}^{\mathrm{lin}}_{\mathrm{snd}})$-benign proof system $\mathbf{PS}^{\mathrm{lin}} = (\mathbf{PGen}^{\mathrm{lin}}, \mathbf{PPrv}^{\mathrm{lin}}, \mathbf{PVer}^{\mathrm{lin}}, \mathbf{PSim}^{\mathrm{lin}})$ for $\mathcal{L}^{\mathrm{lin}}$ (Section 5.2),
- a $(\mathcal{L}^{\mathrm{hash}}_{\mathrm{sim}}, \mathcal{L}^{\mathrm{hash}}_{\mathrm{ver}}, \mathcal{L}^{\mathrm{hash}}_{\mathrm{snd}})$-benign proof system $\mathbf{PS}^{\mathrm{hash}} = (\mathbf{PGen}^{\mathrm{hash}}, \mathbf{PPrv}^{\mathrm{hash}}, \mathbf{PVer}^{\mathrm{hash}}, \mathbf{PSim}^{\mathrm{hash}})$ for $\mathcal{L}^{\mathrm{hash}}$ (Section 5.3),

- an $(\mathcal{L}^{\vee}_{\text{sim}}, \mathcal{L}^{\vee}_{\text{ver}}, \mathcal{L}^{\vee}_{\text{snd}})$-benign proof system $\mathbf{PS}^{\vee} = (\mathbf{PGen}^{\vee}, \mathbf{PPrv}^{\vee}, \mathbf{PVer}^{\vee}, \mathbf{PSim}^{\vee})$ for $\mathcal{L}^{\vee}$ (Section 5.4) with $\ell_{\vee} = 3\lambda$, and
- a collision-resistant hash function generator **CRHF** (Section 2) with $\ell_{\mathsf{H}} = 2\lambda$.[9]

We can use the presented generic constructions for **EXT**, $\mathbf{PS}^{\text{lin}}$, and $\mathbf{PS}^{\text{hash}}$, and, in the prime-order and DCR settings, the presented concrete constructions for **OTS** and $\mathbf{PS}^{\vee}$. (We note, however, that the DCR-based proof system $\mathbf{PS}^{\vee}_{\mathbf{DCR}}$ additionally requires that $|\mathbb{G}|$ has no prime factors smaller than $2^{7\lambda}$.) Specifically, we obtain instantiations both in the prime-order (with symmetric pairing) and DCR settings.

We also assume public parameters pp that contain whatever public parameters our building blocks require. Specifically, pp defines groups $\mathbb{G}$, $\mathbb{G}_1$, and $\mathbb{G}_2$ (as described in Section 3.1.1), and contains a hash function H output by **CRHF**.

The algorithms. Now our KEM **KEM** is defined through the following algorithms:

- $\mathbf{Gen}(1^{\lambda})$ first uniformly picks $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_4 \in \mathbb{Z}^{\ell_{\mathbf{B}}}_{|\mathbb{G}_1|}$, and sets $(\mathbf{pk}, \mathbf{sk}) = (epk_{\mathsf{i}}, esk_{\mathsf{i}})^4_{\mathsf{i}=1} = (g_1^{\boldsymbol{\omega}_{\mathsf{i}}^{\top}\mathbf{B}}, \boldsymbol{\omega}_{\mathsf{i}})^4_{\mathsf{i}=1}$. Next, **Gen** samples

$$\begin{aligned}
(ppk_{\text{lin}}, psk_{\text{lin}}) &\leftarrow \mathbf{PGen}^{\text{lin}}(1^{\lambda}, \mathbf{pk}) \\
(ppk_{\text{hash}}, psk_{\text{hash}}) &\leftarrow \mathbf{PGen}^{\text{hash}}(1^{\lambda}, (epk_1, epk_2/epk_3)) \\
(ppk_{\vee,1}, psk_{\vee,1}) &\leftarrow \mathbf{PGen}^{\vee}(1^{\lambda}, (epk_1, epk_1)) \\
(ppk_{\vee,2}, psk_{\vee,2}) &\leftarrow \mathbf{PGen}^{\vee}(1^{\lambda}, (epk_4, epk_4)) \\
(ppk_{\vee,3}, psk_{\vee,3}) &\leftarrow \mathbf{PGen}^{\vee}(1^{\lambda}, (epk_4, epk_1)) \\
(ppk_{\vee,4}, psk_{\vee,4}) &\leftarrow \mathbf{PGen}^{\vee}(1^{\lambda}, (epk_2/epk_3, epk_4)) \\
(ppk_{\vee,5}, psk_{\vee,5}) &\leftarrow \mathbf{PGen}^{\vee}(1^{\lambda}, (epk_2/epk_3, epk_4)) \\
(ppk_{\vee,6}, psk_{\vee,6}) &\leftarrow \mathbf{PGen}^{\vee}(1^{\lambda}, (epk_2/epk_3, epk_1)) \\
(xpk, xsk) &\leftarrow \mathbf{ExtGen}(1^{\lambda}, epk_2),
\end{aligned}$$

  sets $\mathbf{ppk} = (ppk_{\text{lin}}, ppk_{\text{hash}}, ppk_{\vee,1}, \ldots, ppk_{\vee,6})$ and $\mathbf{psk} = (psk_{\text{lin}}, psk_{\text{hash}}, psk_{\vee,1}, \ldots, psk_{\vee,6})$, and finally outputs

$$pk = (\mathbf{pk}, \mathbf{ppk}, xpk) \qquad\qquad sk = (\mathbf{sk}, \mathbf{psk}, xsk).$$

- $\mathbf{Enc}(pk)$ (for pk as above) selects a random $\mathbf{r}$, and computes

$$\begin{aligned}
\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_4) &= \mathbf{E}(\mathbf{pk}, \mathbf{0}; \mathbf{r}) \\
(ovk, osk) &\leftarrow \mathbf{SGen}() \\
\tau &= \mathsf{H}(ovk) \\
\pi_{\text{lin}} &\leftarrow \mathbf{PPrv}^{\text{lin}}(ppk_{\text{lin}}, \mathbf{c}, \mathbf{r}) \\
\pi_{\text{hash}} &\leftarrow \mathbf{PPrv}^{\text{hash}}(ppk_{\text{hash}}, ((\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2/\mathbf{c}_3), \tau), \mathbf{r}) \\
\pi_{\vee,1} &\leftarrow \mathbf{PPrv}^{\vee}(ppk_{\vee,1}, (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_1/\mathbf{g}_2), \mathbf{r}) \\
\pi_{\vee,2} &\leftarrow \mathbf{PPrv}^{\vee}(ppk_{\vee,2}, (\mathbf{c}_0, \mathbf{c}_4, \mathbf{c}_4/\mathbf{g}_2), \mathbf{r}) \\
\pi_{\vee,3} &\leftarrow \mathbf{PPrv}^{\vee}(ppk_{\vee,3}, (\mathbf{c}_0, \mathbf{c}_4, \mathbf{c}_1/\mathbf{g}_2), \mathbf{r}) \\
\pi_{\vee,4} &\leftarrow \mathbf{PPrv}^{\vee}(ppk_{\vee,4}, (\mathbf{c}_0, \mathbf{c}_2/\mathbf{c}_3, \mathbf{c}_4), \mathbf{r}) \\
\pi_{\vee,5} &\leftarrow \mathbf{PPrv}^{\vee}(ppk_{\vee,5}, (\mathbf{c}_0, \mathbf{c}_2/\mathbf{c}_3, \mathbf{c}_4/\mathbf{g}_2), \mathbf{r}) \\
\pi_{\vee,6} &\leftarrow \mathbf{PPrv}^{\vee}(ppk_{\vee,6}, (\mathbf{c}_0, \mathbf{c}_2/\mathbf{c}_3, \mathbf{c}_1/\mathbf{g}_2), \mathbf{r})
\end{aligned}$$

[9] Since we assume collision-resistance (and not only target collision-resistance), we will have to take into account, e.g., birthday attacks on the hash function. This unfortunately entails $\ell_{\mathsf{H}} \geq 2\lambda$.

$$\begin{aligned}\boldsymbol{\pi} &= (\pi_{\text{lin}}, \pi_{\text{hash}}, \pi_{\vee,1}, \ldots, \pi_{\vee,6})\\ \sigma &\leftarrow \mathbf{SSig}(osk, (\mathbf{c}, \boldsymbol{\pi}))\\ K &= \mathbf{Ext}_{\text{pub}}(xpk, (\mathbf{c}_0, \mathbf{c}_2), \mathbf{r}).\end{aligned}$$

Here, we interpret $\tau = (\tau_1, \ldots, \tau_{2\lambda}) \in \{0,1\}^{2\lambda}$ as an integer $\tau = \sum_{i=1}^{2\lambda} 2^{i-1}\tau_i \in \{0, \ldots, 2^{2\lambda}-1\}$, with τ_1 being interpreted as the least significant bit.
The final output of **Enc** is $C = (\mathbf{c}, \boldsymbol{\pi}, ovk, \sigma)$ and K.

- **Dec**(sk, C) (for sk and C as above), first verifies σ and all proofs in $\boldsymbol{\pi}$ using ovk and $\mathbf{sk}$, and, if all are valid, returns
$$K = \mathbf{Ext}_{\text{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2)).$$

Explanation. The proofs in $\boldsymbol{\pi}$ require some explanation. They prove various (seemingly highly redundant) properties of the vector $\mathbf{u} = (u_i)_{i=1}^4 \in \mathbb{Z}_{|\mathbb{G}_2|}^4$ encrypted in $\mathbf{c}$. Some of these properties will be violated in different steps of our security analysis already by the security game, and we will then rely on the remaining properties. For instance, π_{lin} always guarantees that the vectors $\mathbf{u}$ encrypted in decryption queries lie in the subspace spanned by the vectors $\mathbf{u}$ encrypted in challenge ciphertexts. (That subspace is initially trivial, since honest encryptions contain $\mathbf{u} = \mathbf{0}$, but will be larger in later parts of the analysis.) π_{hash} guarantees that $\tau u_1 = u_2 - u_3$ in $\mathcal{A}$'s decryption queries (unless generated challenge ciphertexts already violate that relation).

The $\mathbf{PS}^\vee$-proofs $\pi_{\vee,i}$ are a bit more delicate. First, $\pi_{\vee,1}$ and $\pi_{\vee,2}$ guarantee that $u_1, u_4 \in \{0,1\}$. The condition $u_1 \in \{0,1\}$ only simplifies the analysis, but $u_4 \in \{0,1\}$ is instrumental to enforce our partitioning strategy. In particular, u_4 will be the bit that determines the partitioning of ciphertexts in our partitioning argument. Depending on the value of u_4, $\pi_{\vee,4}$ and $\pi_{\vee,5}$ give further guarantees: $\pi_{\vee,4+b}$ guarantees $u_2 = u_3 \vee u_4 = b$. At each point in our analysis, at least one of these conditions (for one value of b) is never violated. Hence, $u_2 = u_3$ is guaranteed in decryption queries whenever $u_4 \neq b$. Finally, the proofs $\pi_{\vee,3}$ and $\pi_{\vee,6}$ ensure technical conditions ($u_4 = 0 \vee u_1 = 1$ and $u_2 = u_3 \vee u_1 = 1$) that will help to deal with the somewhat limited soundness guarantees of $\mathbf{PS}^\vee$. (In particular, these proofs help to cope with the fact that the soundness game of $\mathbf{PS}^\vee$only allows a limited type of verification queries.)

Correctness. The correctness of **KEM** follows directly from the correctness of the underlying primitives.

6.2 Security analysis

Theorem 6.1 (Security of **KEM**). *If the ingredients from Section 6.1 are secure, then* **KEM** *is IND-MCCA secure. Specifically, for every IND-MCCA adversary $\mathcal{A}$ that makes at most q oracle queries, there are adversaries $\mathcal{B}^{\text{crhf}}$, $\mathcal{B}^{\text{ots}}$, $\mathcal{B}^{\text{fact}}$, $\mathcal{B}^{\text{mccpa}}$, $\mathcal{B}^{\text{lin}}$, $\mathcal{B}^{\text{hash}}$, and $\mathcal{B}^\vee$ with*

$$\begin{aligned}|\text{Adv}^{\text{mcca}}_{\mathbf{KEM},\mathcal{A}}(\lambda)| \leq\ & \text{Adv}^{\text{crhf}}_{\mathbf{CRHF},\mathcal{B}^{\text{crhf}}}(\lambda) + \text{Adv}^{\text{ots}}_{\mathbf{OTS},\mathcal{B}^{\text{ots}}}(\lambda)\\ &+ \mathbf{O}(\lambda)\text{Adv}^{\text{fact}}_{\mathbb{G}_2,\mathcal{B}^{\text{fact}}}(\lambda) + \mathbf{O}(\lambda)\text{Adv}^{\text{mccpa}}_{\mathbb{G},\mathcal{B}^{\text{mccpa}}}(\lambda) + \mathbf{O}(\lambda)\text{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{lin}},\mathcal{B}^{\text{lin}}}(\lambda)\\ &+ \mathbf{O}(\lambda)\text{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{hash}},\mathcal{B}^{\text{hash}}}(\lambda) + \mathbf{O}(\lambda)\text{Adv}^{\text{snd}}_{\mathbf{PS}^\vee,\mathcal{B}^\vee}(\lambda) + \mathbf{O}(\lambda q)/2^\lambda. \quad (22)\end{aligned}$$

6.2.1 The main proof

Outline. The goal of our proof will be to randomize all keys handed out by $\mathcal{O}_{\mathbf{enc}}$ along with challenge ciphertexts. In order to do so, we rely on the indistinguishability of the key extractor **EXT**. However, to apply **EXT**'s indistinguishability (Definition 4.4), we first need to establish a certain kind of "unfairness". Specifically, we will randomize the u_2 component of all challenge ciphertexts, while rejecting all decryption queries with $u_2 \neq 0$. (Note that this in particular means that the experiment does not need to be able to decrypt challenge ciphertexts.)

Establishing this unfairness thus is the key to proving chosen-ciphertext security. But it will also form the main difficulty of the proof, and we will outsource this process into several helper lemmas (Lemmas 6.2 to 6.4).

Proof of Theorem 6.1. Our proof is game-based. Let ε_i be the probability that $\mathcal{A}$ wins in Game i. We give an overview of the games in Fig. 2 (on p. 31).

Game M0 is the original IND-MCCA game with **KEM** and $\mathcal{A}$. Of course,

$$\varepsilon_{M0} \;=\; \mathrm{Adv}^{\mathrm{mcca}}_{\mathbf{KEM},\mathcal{A}}(\lambda). \tag{23}$$

Game M1 rejects every $\mathcal{O}_{\mathbf{dec}}$ query from $\mathcal{A}$ which reuses a value τ from a previous $\mathcal{O}_{\mathbf{enc}}$ ciphertext. Note that any such query that would not already have been rejected in Game M0 implies either a H-collision (in case the *ovk* values are different), or an **OTS** forgery (in case also *ovk* is reused). Hence, reductions to the collision-resistance of **CRHF**, and the unforgeability of **OTS** yield

$$|\varepsilon_{M0} - \varepsilon_{M1}| \;\le\; \mathrm{Adv}^{\mathrm{crhf}}_{\mathbf{CRHF},\mathcal{B}^{\mathrm{crhf}}_{M1}}(\lambda) + \mathrm{Adv}^{\mathrm{ots}}_{\mathbf{OTS},\mathcal{B}^{\mathrm{ots}}_{M1}}(\lambda) \tag{24}$$

for the following adversaries $\mathcal{B}^{\mathrm{crhf}}_{M1}$ and $\mathcal{B}^{\mathrm{ots}}_{M1}$. First, $\mathcal{B}^{\mathrm{crhf}}_{M1}(1^\lambda, H)$ simulates Game M1, embedding H into *pars*, and outputs any H-collision $H(ovk) = \tau = H(ovk')$ for ovk, ovk' from $\mathcal{O}_{\mathbf{enc}}$, resp. $\mathcal{O}_{\mathbf{dec}}$ queries. Second, $\mathcal{B}^{\mathrm{ots}}_{M1}(1^\lambda)$ also simulates Game M1, and uses its own $\mathcal{O}_{\mathbf{gen}}$ and $\mathcal{O}_{\mathbf{sig}}$ oracles to create **OTS** public keys and signatures for $\mathcal{O}_{\mathbf{enc}}$, and outputs any **OTS**-forgery from an $\mathcal{O}_{\mathbf{dec}}$ query.

In **Game** M2, we prepare challenge ciphertexts $C = (\mathbf{c}, \boldsymbol{\pi}, ovk, \sigma)$ and the corresponding keys K slightly differently. Namely, instead of using a witness $\mathbf{r}$ to prepare $\boldsymbol{\pi}$ and K, we now use the secret keys **psk** and *xsk*, as follows:

$$\begin{aligned}
\pi_{\mathrm{lin}} &\leftarrow \mathbf{PSim}^{\mathrm{lin}}(psk_{\mathrm{lin}}, \mathbf{c}) \\
\pi_{\mathrm{hash}} &\leftarrow \mathbf{PSim}^{\mathrm{hash}}(psk_{\mathrm{hash}}, ((\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2/\mathbf{c}_3), \tau)) \\
\pi_{\vee,1} &= \mathbf{PSim}^{\vee}(psk_{\vee,1}, (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_1/\mathbf{g}_2)) \\
\pi_{\vee,2} &= \mathbf{PSim}^{\vee}(psk_{\vee,2}, (\mathbf{c}_0, \mathbf{c}_4, \mathbf{c}_4/\mathbf{g}_2)) \\
\pi_{\vee,3} &= \mathbf{PSim}^{\vee}(psk_{\vee,3}, (\mathbf{c}_0, \mathbf{c}_4, \mathbf{c}_1/\mathbf{g}_2)) \\
\pi_{\vee,4} &= \mathbf{PSim}^{\vee}(psk_{\vee,4}, (\mathbf{c}_0, \mathbf{c}_2/\mathbf{c}_3, \mathbf{c}_4)) \\
\pi_{\vee,5} &= \mathbf{PSim}^{\vee}(psk_{\vee,5}, (\mathbf{c}_0, \mathbf{c}_2/\mathbf{c}_3, \mathbf{c}_4/\mathbf{g}_2)) \\
\pi_{\vee,6} &= \mathbf{PSim}^{\vee}(psk_{\vee,6}, (\mathbf{c}_0, \mathbf{c}_2/\mathbf{c}_3, \mathbf{c}_1/\mathbf{g}_2)) \\
K &\leftarrow \mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2)).
\end{aligned}$$

Intuitively, this change enables the game to "forget" the witness $\mathbf{r}$, and change $\mathbf{c}$ subsequently. By the perfect zero-knowledge property of $\mathbf{PS}^{\mathrm{lin}}$ and $\mathbf{PS}^{\vee}$, and the perfect correctness of **EXT**, we have

$$\varepsilon_{M1} \;=\; \varepsilon_{M2}. \tag{25}$$

In **Game** M3, we let $\mathcal{O}_{\mathbf{enc}}$ encrypt the vector $\mathbf{u} = (1, \mathbf{G}(\tau), 0, 0)^\top$ in each challenge ciphertext (using $\mathbf{c} = \mathbf{E}(\mathbf{pk}, \mathbf{u}; \mathbf{r})$), where $\mathbf{G} : \{0,1\}^* \to \mathbb{Z}_{2^{3\lambda}}$ is a truly random function. Since at this point, we neither use $\mathbf{r}$ nor esk_1 or esk_2, we can invoke the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$ (twice) to obtain

$$\varepsilon_{M2} - \varepsilon_{M3} \;=\; 2\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}_{M3}}(\lambda) \tag{26}$$

for an adversary $\mathcal{B}^{\mathrm{mccpa}}_{M3}$ that simulates Game M2, resp. Game M3, depending on the challenge bit b of the IND-MCCPA game.

Game M4 rejects all $\mathcal{O}_{\mathbf{dec}}$ queries with $u_1 \neq 0$. (In general, when we speak of values u_i from an $\mathcal{O}_{\mathbf{dec}}$ query we mean the values u_i defined through $g_2^{u_i} = \mathbf{D}_{esk_i}(\mathbf{c}_0, \mathbf{c}_i)$, where we set $u_i = \perp$

if $\mathbf{D}_{esk_i}(\mathbf{c}_0, \mathbf{c}_i) \notin \mathbb{G}_2$.) In other words, while Game M4 keeps preparing challenge ciphertexts that are inconsistent (in the sense that they have $u_1 \neq 0$), now all inconsistent decryption queries from $\mathcal{A}$ are rejected. The justification for this game change is somewhat complex, and outsourced into Lemma 6.2 (see below). We obtain:

$$\begin{aligned}|\varepsilon_{M3} - \varepsilon_{M4}| \;\leq\; & (12\lambda+1)\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{M4}}(\lambda) + (24\lambda+3)\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}_{M4}}(\lambda) + (8\lambda+3)\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{M4}}(\lambda) \\ & + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{hash}},\mathcal{B}^{\mathrm{hash}}_{M4}}(\lambda) + (32\lambda+1)\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}_{M4}}(\lambda) + \mathbf{O}(q/2^{\lambda}) \quad (27)\end{aligned}$$

for suitable $\mathcal{B}^{\mathrm{fact}}_{M4}$, $\mathcal{B}^{\mathrm{mccpa}}_{M4}$, $\mathcal{B}^{\mathrm{lin}}_{M4}$, $\mathcal{B}^{\mathrm{hash}}_{M4}$, and $\mathcal{B}^{\vee}_{M4}$.

Game M5 finally randomizes the keys K output by $\mathcal{O}_{\mathbf{enc}}$. Specifically, Game M5 chooses $K \in \{0,1\}^{\lambda}$ uniformly and independently for each challenge ciphertext. Essentially, we are going to justify this change with the indistinguishability of **EXT**. However, since the **EXT** indistinguishability experiment provides only a comparatively weak extraction oracle, we will need to make a few preparations first. Specifically, we will argue that a number of annoying "bad events" can only happen with negligible probability in Game M4 and Game M5.

Namely, let $\mathbf{bad}_{\mathbf{c}}$ denote the event that $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with valid $\boldsymbol{\pi}$, but in which $\mathbf{c}$ is not even in the range of $\mathbf{E}(\mathbf{pk}, \mathbf{u})$ for *any* $\mathbf{u}$. Also, let $\mathbf{bad}_{u_3}$ denote the event that $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with valid $\boldsymbol{\pi}$, but with $u_1 = 0$ and $u_3 \neq 0$. We can use the soundness of $\mathbf{PS}^{\mathrm{lin}}$ to exclude $\mathbf{bad}_{\mathbf{c}}$ and $\mathbf{bad}_{u_3}$ both in Game M4 and Game M5. Specifically, an adversary $\mathcal{B}^{\mathrm{lin}}_{M5}$ on $\mathbf{PS}^{\mathrm{lin}}$'s soundness simulates Game M5 until $\mathbf{pk}$ is generated, then submits its own language parameter $(\mathbf{pk}, \mathbf{M})$ for

$$\mathbf{M} \;=\; \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix},$$

and then embeds the received public key ppk_{lin} into its own simulation. To generate and verify $\mathbf{PS}^{\mathrm{lin}}$-proofs, $\mathcal{B}^{\mathrm{lin}}_{M5}$ uses its own $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ oracles as follows. First, to generate $\mathbf{PS}^{\mathrm{lin}}$ proofs for encryptions generated by $\mathcal{B}^{\mathrm{lin}}_{M5}$'s internally simulated $\mathcal{O}_{\mathbf{enc}}$ oracle, $\mathcal{B}^{\mathrm{lin}}_{M5}$ submits the corresponding ciphertext $\mathbf{c}$ to its $\mathcal{O}_{\mathbf{sim}}$ oracle. By definition of Game M5, we have $\mathbf{c} \in \mathcal{L}^{\mathrm{lin}}_{\mathrm{sim},(\mathbf{pk},\mathbf{M})}$, and thus $\mathcal{O}_{\mathbf{sim}}$ returns a simulated proof π_{lin} that $\mathcal{B}^{\mathrm{lin}}_{M5}$ can then embed into its own simulation. Furthermore, to verify proofs π_{lin} during an $\mathcal{O}_{\mathbf{dec}}$ query from $\mathcal{A}$, $\mathcal{B}^{\mathrm{lin}}_{M5}$ submits π_{lin} with the corresponding $\mathbf{c}$ to its $\mathcal{O}_{\mathbf{ver}}$ oracle. If $\mathcal{O}_{\mathbf{ver}}$ returns $\perp$ (which means that $\mathbf{c} \notin \mathcal{L}^{\mathrm{lin}}_{\mathrm{ver},(\mathbf{pk},\mathbf{M})}$, i.e., that $\mathbf{c}$ is not in the range of $\mathbf{E}$, or that $u_3 \neq 0$ or $u_4 \neq 0$), then $\mathcal{B}^{\mathrm{lin}}_{M5}$ counts the proof as invalid, and thus rejects $\mathcal{A}$'s $\mathcal{O}_{\mathbf{dec}}$ query.

Observe that this $\mathcal{B}^{\mathrm{lin}}_{M5}$ perfectly simulates Game M5 until $\mathcal{A}$ submits an $\mathcal{O}_{\mathbf{dec}}$ query with $\mathbf{c} \notin \mathcal{L}^{\mathrm{lin}}_{\mathrm{ver},(\mathbf{pk},\mathbf{M})}$ but valid $\boldsymbol{\pi}$. (In this case, $\mathcal{B}^{\mathrm{lin}}_{M5}$ rejects that query, while Game M5 would accept it.) However, in that case, $\mathcal{B}^{\mathrm{lin}}_{M5}$ would have won its own soundness game, since it has submitted a $\mathbf{c} \notin \mathcal{L}^{\mathrm{lin}}_{\mathrm{snd},(\mathbf{pk},\mathbf{M})}$ with a valid proof π_{lin} to its own $\mathcal{O}_{\mathbf{ver}}$ oracle. Since $\mathbf{bad}_{\mathbf{c}} \vee \mathbf{bad}_{u_3}$ implies $\mathbf{c} \notin \mathcal{L}^{\mathrm{lin}}_{\mathrm{snd},(\mathbf{pk},\mathbf{M})}$, we can deduce that

$$\Pr\left[\mathbf{bad}_{\mathbf{c}} \vee \mathbf{bad}_{u_3} \text{ in Game M5}\right] \;\leq\; \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{M5}}. \quad (28)$$

Analogously, we get

$$\Pr\left[\mathbf{bad}_{\mathbf{c}} \vee \mathbf{bad}_{u_3} \text{ in Game M4}\right] \;\leq\; \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{M4}} \quad (29)$$

for an adversary $\mathcal{B}^{\mathrm{lin}}_{M4}$ on $\mathbf{PS}^{\mathrm{lin}}$ that simulates Game M4 instead of Game M5.

Similarly, let $\mathbf{bad}_{\mathrm{fact}}$ denote the event that $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with $u_1 \in \mathbb{Z}_{|\mathbb{G}_2|} \setminus (\mathbb{Z}^*_{|\mathbb{G}_2|} \cup \{0\})$ or $u_2 - u_3 \in \mathbb{Z}_{|\mathbb{G}_2|} \setminus (\mathbb{Z}^*_{|\mathbb{G}_2|} \cup \{0\})$. In other words, $\mathbf{bad}_{\mathrm{fact}}$ occurs when $\mathcal{A}$ produces an encryption

of a "funny message" that would allow to factor $\mathbb{G}_2$ in the sense of Definition 3.2. Hence, a straightforward reduction to the $\mathbb{G}_2$-factoring assumption yields

$$\begin{aligned} \Pr[\mathbf{bad}_{\mathrm{fact}} \text{ in Game M4}] &\leq \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{\mathrm{M4}}} \\ \Pr[\mathbf{bad}_{\mathrm{fact}} \text{ in Game M5}] &\leq \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{\mathrm{M5}}} \end{aligned} \tag{30}$$

for adversaries $\mathcal{B}^{\mathrm{fact}}_{\mathrm{M4}}$ and $\mathcal{B}^{\mathrm{fact}}_{\mathrm{M5}}$ that simulate Game M4, resp. Game M5, and use $\mathbf{sk}$ to retrieve and output $\mathbf{u}$ upon every $\mathcal{O}_{\mathbf{dec}}$ query from $\mathcal{A}$.

Finally, let $\mathbf{bad}_{u_2}$ denote the event that $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with $u_1 = 0$ and $u_2 \neq u_3$. We will rely on the soundness of the 6-th instance of $\mathbf{PS}^{\vee}$ to exclude $\mathbf{bad}_{u_2}$, and a corresponding adversary $\mathcal{B}^{\vee,6}_{\mathrm{M5}}$ can proceed similarly to $\mathcal{B}^{\mathrm{lin}}_{\mathrm{M5}}$ above. However, it is important to note that $\mathcal{B}^{\vee,6}_{\mathrm{M5}}$ feeds its $\mathcal{O}_{\mathbf{sim}}$ oracle only instances $(\mathbf{c}_0, c_2/c_3, c_1/g_2) \in \mathcal{L}^{\vee}_{\mathrm{sim},((epk_2/epk_3,epk_1),\ell_{\vee})}$ for $\ell_{\vee} = 3\lambda$, since $|u_2 - u_3|_N < 2^{3\lambda}$ and $u_1 = 1$ in ciphertexts prepared by $\mathcal{O}_{\mathbf{enc}}$. (Hence, $\mathcal{B}^{\vee,6}_{\mathrm{M5}}$ can obtain from $\mathcal{O}_{\mathbf{sim}}$ all proofs $\pi_{\vee,6}$ as necessary to implement $\mathcal{O}_{\mathbf{enc}}$.) The analysis of $\mathcal{O}_{\mathbf{ver}}$ queries requires a little more care, since the "verification" languages in $\mathcal{L}^{\vee}_{\mathrm{ver}}$ are rather restricted. In particular, $\mathcal{O}_{\mathbf{ver}}$ returns $\perp$ (in which case $\mathcal{B}^{\vee,6}_{\mathrm{M5}}$ rejects the query) as soon as $\mathbf{bad}_{\mathbf{c}}$ or $\mathbf{bad}_{\mathrm{fact}}$ occur, or $u_2 \neq u_3$ holds (independently of u_1). However, if $u_1 \neq 0$, the $\mathcal{O}_{\mathbf{dec}}$ query would have been rejected by the rules of Game M5 anyway, and if $u_1 = 0$, then a valid proof $\pi_{\vee,6}$ would constitute a $\mathbf{PS}^{\vee}$-forgery (since *both* branches of the "OR" statement are violated). Hence, $\mathcal{B}^{\vee,6}_{\mathrm{M5}}$ only deviates from Game M5 if it can produce a valid forgery, or if $\mathbf{bad}_{\mathbf{c}} \vee \mathbf{bad}_{\mathrm{fact}}$ occurs. Since a valid forgery implies $u_2 \neq u_3$ and thus $\mathbf{bad}_{u_2}$, we obtain

$$\begin{aligned} \Pr[\mathbf{bad}_{u_2} \mid \neg(\mathbf{bad}_{\mathbf{c}} \vee \mathbf{bad}_{\mathrm{fact}}) \text{ in Game M4}] &\leq \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee,6}_{\mathrm{M4}}} \\ \Pr[\mathbf{bad}_{u_2} \mid \neg(\mathbf{bad}_{\mathbf{c}} \vee \mathbf{bad}_{\mathrm{fact}}) \text{ in Game M5}] &\leq \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee,6}_{\mathrm{M5}}} \end{aligned} \tag{31}$$

where $\mathcal{B}^{\vee,6}_{\mathrm{M4}}$ is defined like $\mathcal{B}^{\vee,6}_{\mathrm{M5}}$, except that it simulates Game M4 (and not Game M5).

We are now ready for our final reduction to the indistinguishability of $\mathbf{EXT}$. Concretely, consider an adversary $\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}$ that initially obtains a public key epk from its own indistinguishability game, along with access to challenge and extraction oracles $\mathcal{O}_{\mathbf{cha}}$ and $\mathcal{O}_{\mathbf{ext}}$. Internally, $\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}$ simulates Game M5, and embeds epk into that simulation as epk_2. Upon an $\mathcal{O}_{\mathbf{enc}}$ query from $\mathcal{A}$, $\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}$ queries $\mathcal{O}_{\mathbf{cha}}$, and embeds the obtained ciphertext as $\mathbf{c}_0$ and c_2 into its own simulation. (The corresponding c_i for $i \in \{1,3,4\}$ can be generated from $\mathbf{c}_0$ and esk_i as in the proof of Lemma 3.4.) The obtained key K is used in place of a real key K_0 in Game M5. The keys K for decryption queries from $\mathcal{A}$ are obtained via $\mathcal{O}_{\mathbf{ext}}(\mathbf{c}_0, c_2)$. Finally, $\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}$ outputs whatever the game outputs. Note that $\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}$ fails in the indistinguishability game only if $\mathcal{A}$ manages to submit a decryption query not in the range of $\mathbf{E}$, or one that satisfies $u_2 \neq 0$. However, this event would imply at least one of $\mathbf{bad}_{\mathbf{c}}$, $\mathbf{bad}_{u_2}$, or $\mathbf{bad}_{u_3}$. Hence, using (28), (29), and (31), and combining adversaries, we get

$$\varepsilon_{\mathrm{M4}} - \varepsilon_{\mathrm{M5}} = 3\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{\mathrm{M4M5}}}(\lambda) + 2\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{\mathrm{M4M5}}}(\lambda) + 2\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee,6}_{\mathrm{M4M5}}}(\lambda) + \mathrm{Adv}^{\mathrm{ext}}_{\mathbf{EXT},\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}}(\lambda) \tag{32}$$

for suitable $\mathcal{B}^{\mathrm{fact}}_{\mathrm{M4M5}}$, $\mathcal{B}^{\mathrm{lin}}_{\mathrm{M4M5}}$, $\mathcal{B}^{\vee,6}_{\mathrm{M4M5}}$, and $\mathcal{B}^{\mathrm{ext}}_{\mathrm{M5}}$. At this point, note that $\varepsilon_{\mathrm{M5}} = 0$, since $\mathcal{A}$ gets no information about the challenge bit b anymore. Combining this observation with (23)-(32) yields (22). □

Of course, the main work still lies ahead of us, since we need to prove (27). We do so now.

6.2.2 Enforcing consistent decryption queries

Outline. The next lemma establishes the "unfairness" mentioned in the outline of the main proof. In particular, it will force the adversary to use $u_1 = 0$ in decryption queries, while chal-

#	u	K computed as	game knows	$\mathcal{O}_{\mathbf{dec}}$ checks	remark
M0	$\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$	in $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{Ext}_{\mathrm{pub}}(xpk, (\mathbf{c}_0, \mathbf{c}_2), \mathbf{r})$ in $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$	for $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{ppk}, xpk, \mathbf{r}$ for $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{psk}, xsk$	—	IND-MCCA game
M1	$\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$	in $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{Ext}_{\mathrm{pub}}(xpk, (\mathbf{c}_0, \mathbf{c}_2), \mathbf{r})$ in $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$	for $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{ppk}, xpk, \mathbf{r}$ for $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{psk}, xsk$	τ fresh	**CRHF** collision-res., **OTS**-unforgeability
M2	$\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$	in $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$ in $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$	for $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{psk}, xsk$ for $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{psk}, xsk$	τ fresh	ZK of $\mathbf{PS}^{\mathrm{lin}}$, $\mathbf{PS}^{\vee}$, **EXT**-correctness
M3	$\begin{pmatrix}1\\\mathbf{G}(\tau)\\0\\0\end{pmatrix}$	in $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$ in $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$	for $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{psk}, xsk$ for $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{psk}, xsk$	τ fresh	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
M4	$\begin{pmatrix}1\\\mathbf{G}(\tau)\\0\\0\end{pmatrix}$	in $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$ in $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, (\mathbf{c}_0, \mathbf{c}_2))$	for $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{psk}, xsk$ for $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{psk}, xsk, esk_1$	τ fresh $u_1 = 0$	Lemma 6.2
M5	$\begin{pmatrix}1\\\mathbf{G}(\tau)\\0\\0\end{pmatrix}$	in $\mathcal{O}_{\mathbf{enc}}$: random in $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{Ext}_{\mathrm{priv}}(xsk, \mathbf{c})$	for $\mathcal{O}_{\mathbf{enc}}$: $\mathbf{psk}, xsk$ for $\mathcal{O}_{\mathbf{dec}}$: $\mathbf{psk}, xsk, esk_1$	τ fresh $u_1 = 0$	$\mathbb{G}_2$-factoring, $\mathbf{PS}^{\mathrm{lin}}$-soundness, $\mathbf{PS}^{\vee}$-soundness, **EXT**-indist.

Figure 2: Games in the main proof of Theorem 6.1. Here, "**u**" denotes the vector encrypted upon $\mathcal{O}_{\mathbf{enc}}$ queries, "K computed as" denotes how the session key K is computed during $\mathcal{O}_{\mathbf{enc}}$ and $\mathcal{O}_{\mathbf{dec}}$ queries from C, "game knows" denotes which information is necessary to perform the game, and "$\mathcal{O}_{\mathbf{dec}}$ checks" indicates additional consistency checks that $\mathcal{O}_{\mathbf{dec}}$ performs on its inputs. (If one of those checks fails, $\mathcal{O}_{\mathbf{dec}}$ outputs $\perp$.)

lenge ciphertexts contain $u_1 = 1$. (This unfairness can be used in the main proof to randomize u_2 in challenge ciphertexts, while enforcing $u_2 = 0$ in decryption queries.)

We can view this unfairness as a special kind of simulation-soundness. Namely, the proofs $\boldsymbol{\pi}$ (together with the encrypted u_2, u_3, u_4) from ciphertexts act as consistency proofs that show (among other things) that $u_1 = 0$. With this view, we now show that we can simulate valid-looking consistency proofs $\boldsymbol{\pi}$ for inconsistent ciphertexts, while preserving soundness for adversarially generated proofs.

To establish unfairness, we revisit the idea of "authentication tags" from the introduction. Namely, we first establish that all ciphertexts with $u_1 \neq 0$ must carry a valid authentication tag in u_2. Initially, that authentication tag will be a random (but constant) value X. Next, through a series of hybrid games (outsourced into Lemmas 6.3 and 6.4), we change the definition of valid authentication tags. In the j-th challenge ciphertext, we will have $u_2 = X + \tau^{(j)}$, where $\tau^{(j)}$ is the corresponding hash value τ from that ciphertext. Additionally, we will force the adversary to reuse one such value $u_2 = X + \tau^{(j)}$ (for a $\tau^{(j)}$ from a previous challenge ciphertext) in his own decryption queries with $u_1 \neq 0$. (This somewhat unusual "reusal rule" arises naturally out of our randomization strategy in Lemma 6.3) constitutes a departure from previous strategies to obtain chosen-ciphertext security or simulation-soundness.)

Finally, we employ the dynamically parameterized hash proof system $\mathbf{PS}^{\mathrm{hash}}$ to tie the value of the authentication tag u_2 to $X + \tau$ for the hash value τ of the *current* ciphertext. In particular, since all challenge ciphertexts (with $u_2 = X + \tau^{(j)}$) now satisfy the relation enforced by $\mathbf{PS}^{\mathrm{hash}}$, the soundness of $\mathbf{PS}^{\mathrm{hash}}$ guarantees that any decryption query must do so as well. In particular, $u_1 \neq 0$ must now satisfy contradictory requirements: $\mathbf{PS}^{\mathrm{hash}}$ enforces $u_2 = X + \tau$ for the hash value τ of that ciphertext, while the reusal rule established before en-

forces $u_2 = X + \tau^{(j)}$ for a hash value $\tau^{(j)}$ from a previous challenge ciphertext. Taken together (and using the uniqueness of hash values), this implies that all decryption queries with $u_1 \neq 0$ are rejected, as desired.

Lemma 6.2. *In the proof of Theorem 6.1, (27) holds, i.e., we have*

$$|\varepsilon_{M3} - \varepsilon_{M4}| \;\leq\; (12\lambda+1)\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}}(\lambda) + (24\lambda+3)\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}}(\lambda) + (8\lambda+3)\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{hash}},\mathcal{B}^{\mathrm{hash}}}(\lambda) + (32\lambda+1)\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}}(\lambda) + \mathbf{O}(q/2^{\lambda})$$

for adversaries $\mathcal{B}^{\mathrm{fact}}$, $\mathcal{B}^{\mathrm{mccpa}}$, $\mathcal{B}^{\mathrm{lin}}$, $\mathcal{B}^{\mathrm{hash}}$, and $\mathcal{B}^{\vee}$ of essentially the same complexity as Game M4*.*

Proof. Observe first that the only difference between Games M3 and M4 is that Game M4 explicitly checks and rejects $\mathcal{O}_{\mathbf{dec}}$ queries with $u_1 \neq 0$. Denote with $\mathbf{bad}_{u_1}$ the event that $\mathcal{A}$ submits an $\mathcal{O}_{\mathbf{dec}}$ query with $u_1 \neq 0$ that would not be rejected according to the rules of Game M3. In the following, we will bound the probability that $\mathbf{bad}_{u_1}$ occurs in Game M3. This bound will automatically yield a bound for the difference $|\varepsilon_{M3} - \varepsilon_{M4}|$.

It will be useful to call an $\mathcal{O}_{\mathbf{dec}}$ query of $\mathcal{A}$ a *critical query* if it has $u_1 \neq 0$, but would be accepted by $\mathcal{O}_{\mathbf{dec}}$ in Game M3 (i.e., carries a fresh τ, and a valid $\boldsymbol{\pi}$). Note that $\mathbf{bad}_{u_1}$ occurs iff $\mathcal{A}$ places a critical query at some point. We proceed in a sequence of games, and denote with ε_i the probability that Game i eventually outputs 1. Fig. 3 (on p. 35) provides an overview over the games in this proof.

Game C0 is defined like Game M3, except that each $\mathcal{O}_{\mathbf{dec}}$ query is checked to be critical. Furthermore, Game C0 outputs 1 iff $\mathbf{bad}_{u_1}$ occurs, i.e., if $\mathcal{A}$ places a critical query. Without loss of generality, $\mathcal{A}$ immediately terminates (with output 1) after a critical query. (This way, there is at most one critical query.) By the above discussion,

$$|\varepsilon_{M3} - \varepsilon_{M4}| \;\leq\; \varepsilon_{C0}. \tag{33}$$

Game C1 still immediately terminates after a critical query, but then only outputs 1 if the query satisfies $u_1 = 1$ (i.e., if c_1 decrypts to g_2 under esk_1). Hence, the difference between Game C0 and Game C1 can be bounded by the probability that $\mathcal{A}$ manages to submit a critical query with $u_1 \neq 1$. We claim that suitable adversaries $\mathcal{B}^{\mathrm{lin}}_{C1}$, $\mathcal{B}^{\vee,1}_{C1}$, and $\mathcal{B}^{\mathrm{fact}}_{C1}$ achieve

$$\varepsilon_{C0} \;\leq\; \varepsilon_{C1} + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{C1}} + \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{C1}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee,1}_{C1}}(\lambda). \tag{34}$$

Similar to Game M5, let $\mathbf{bad}_{\mathbf{c}}$ denote the event that $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with $\mathbf{c}$ not in the range of $\mathbf{E}(\mathbf{pk}, \mathbf{u})$ for any $\mathbf{u}$. Also, let $\mathbf{bad}_{\mathrm{fact}}$ denote the event that $\mathcal{A}$ submits an $\mathcal{O}_{\mathbf{dec}}$ query that satisfies $u_1 \in \mathbb{Z}_{|\mathbb{G}_2|} \setminus (\mathbb{Z}^*_{|\mathbb{G}_2|} \cup \{0\})$ or $u_1 - 1 \in \mathbb{Z}_{|\mathbb{G}_2|} \setminus (\mathbb{Z}^*_{|\mathbb{G}_2|} \cup \{0\})$. As in Game M5, we can construct adversaries $\mathcal{B}^{\mathrm{lin}}_{C1}$ and $\mathcal{B}^{\mathrm{fact}}_{C1}$ with $\Pr[\mathbf{bad}_{\mathbf{c}}] \leq \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{C1}}(\lambda)$ and $\Pr[\mathbf{bad}_{\mathrm{fact}}] \leq \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{C1}}(\lambda)$.

Moreover, we describe an adversary $\mathcal{B}^{\vee,1}_{C1}$ that attacks $\mathbf{PS}^{\vee}$. Specifically, $\mathcal{B}^{\vee,1}_{C1}$ internally simulates Game C1, and embeds its own public key $ppk_{\vee}$ as $ppk_{\vee,1}$. To generate and verify $\mathbf{PS}^{\vee}$-proofs, $\mathcal{B}^{\vee,1}_{C1}$ uses its own oracles as follows. Upon an $\mathcal{O}_{\mathbf{enc}}$ query from $\mathcal{A}$, $\mathcal{B}^{\vee,1}_{C1}$ uses its own $\mathcal{O}_{\mathbf{sim}}$-oracle to generate a proof $\pi_{\vee,1}$. (This is possible, since the corresponding statement satisfies $u_1 = 1$.) Upon an $\mathcal{O}_{\mathbf{dec}}$ query from $\mathcal{A}$, $\mathcal{B}^{\vee,1}_{C1}$ feeds the corresponding proof $\pi_{\vee,1}$ into its $\mathcal{O}_{\mathbf{ver}}$ oracle. If $\mathcal{O}_{\mathbf{ver}}$ replies with $\perp$, then $\mathcal{B}^{\vee,1}_{C1}$ interprets this as a 0 (i.e., as a failed verification), and continues its simulation of Game C1. $\mathcal{B}^{\vee,1}_{C1}$ finally outputs whatever Game C1 outputs.

Let us denote with $\mathbf{bad}_{C1}$ the event that $\mathcal{A}$ submits a critical query (i.e., an $\mathcal{O}_{\mathbf{dec}}$ query with fresh τ, valid $\boldsymbol{\pi}$, but $u_1 \neq 0$). Observe that $\mathcal{B}^{\vee,1}_{C1}$ perfectly simulates Game C1 (and in fact also Game C0), unless $\mathbf{bad}_{C1}$ occurs. Namely, upon $\mathbf{bad}_{C1}$, $\mathcal{B}^{\vee,1}_{C1}$ would reject that query, while Game C1 and Game C0 would recognize the query as critical and terminate. More precisely, Game C0 would always output 1 upon $\mathbf{bad}_{C1}$, while Game C1 would only output 1 if $u_1 = 1$. Hence, Game C0 and Game C1 only differ if $\mathbf{bad}_{C1}$ occurs with $u_1 \neq 1$. But if $\mathbf{bad}_{C1}$ occurs

with $u_1 \neq 1$, then either $\mathbf{bad_c} \vee \mathbf{bad}_{\text{fact}}$ occurs (if $u_1 \notin \mathbb{Z}^*_{|\mathbb{G}_2|}$ or $u_1 - 1 \notin \mathbb{Z}^*_{|\mathbb{G}_2|}$), or $\mathcal{B}^{\vee}_{C1}{}^1$ wins its own soundness game. Using our bounds on $\Pr[\mathbf{bad_c}]$ and $\Pr[\mathbf{bad}_{\text{fact}}]$, we get (34).

Game C2 initially chooses a uniformly random value $X \in \mathbb{Z}_{2^{3\lambda}}$ and encrypts the vector $\mathbf{u} = (1, X, X, 0)^\top$ in each challenge ciphertext. We can invoke the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$ (i.e., Lemma 3.4) twice to obtain

$$\varepsilon_{C1} \;\leq\; \varepsilon_{C2} + 2\mathrm{Adv}^{\text{mccpa}}_{\mathbb{G},\mathcal{B}^{\text{mccpa}}_{C2}}(\lambda). \tag{35}$$

for an adversary $\mathcal{B}^{\text{mccpa}}_{C2}$.

Game C3 further refines the additional check enforced on critical queries. Concretely, Game C3 checks not only $u_1 = 1$, but also $u_2 = X$. We stress again that we do not change the definition of a critical query, only the requirements for the game to output 1 upon a critical query. (In any case, the game terminates upon a critical query.) Hence, the difference between Game C2 and Game C3 can be bounded by the probability that $\mathcal{A}$ manages to submit a critical query with $u_2 \neq X$. Now if we let $\mathbf{M} = (1, X, X, 0)^\top \in \mathbb{Z}^{4\times 1}_{|\mathbb{G}_2|}$, we can invoke the soundness of $\mathbf{PS}^{\text{lin}}$ for $\mathbf{pk}$ and $\mathbf{M}$ (see (11) and (12)). This gives

$$\varepsilon_{C2} \;\leq\; \varepsilon_{C3} + \mathrm{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{lin}},\mathcal{B}^{\text{lin}}_{C3}}(\lambda) \tag{36}$$

for an adversary $\mathcal{B}^{\text{lin}}_{C3}$ that internally simulates Game C3, and uses its own oracles $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ as follows (and much like $\mathcal{B}^{\text{lin}}_{C1}$ above). $\mathcal{O}_{\mathbf{sim}}$ is used to generate $\mathbf{PS}^{\text{lin}}$-proofs for challenge ciphertexts, and $\mathcal{O}_{\mathbf{ver}}$ is used to determine the validity of $\mathbf{PS}^{\text{lin}}$-proofs π_{lin} in $\mathcal{O}_{\mathbf{dec}}$ queries from $\mathcal{A}$. In case $\mathcal{O}_{\mathbf{ver}}$ answers $\perp$ (which means that the corresponding $\mathbf{u}$ is not in the span of $\mathbf{M}$), $\mathcal{B}^{\text{lin}}_{C3}$ rejects that $\mathcal{O}_{\mathbf{dec}}$ query and proceeds with the game as if π_{lin} had been invalid. Observe that $\mathcal{B}^{\text{lin}}_{C3}$'s internal simulation perfectly simulates Game C3 *and* Game C2, *unless* $\mathcal{A}$ manages to construct a ciphertext with $\mathbf{u} \notin \mathrm{span}(\mathbf{M})$ and valid π_{lin}. In that case, however, $\mathcal{B}^{\text{lin}}_{C3}$ wins its own soundness game, and follows.

In **Game** C4, we let $\mathcal{O}_{\mathbf{enc}}$ encrypt the vector $\mathbf{u} = (1, \mathbf{F}(\tau), \mathbf{F}(\tau), 0)^\top$ in a challenge ciphertext with hash value τ. Here, $\mathbf{F} : \{0,1\}^* \to \mathbb{Z}_{2^{3\lambda}}$ is a truly random function chosen by the game. Additionally, we change the additional winning condition introduced in Game C1 and refined in Game C3. To describe our change, let $\tau^{(j)}$ denote the value τ from the j-th $\mathcal{O}_{\mathbf{enc}}$ query of $\mathcal{A}$. Now we let Game C4 finally only output 1 if $u_1 = 1$ and if there is an index j of an $\mathcal{O}_{\mathbf{enc}}$ query such that the critical query satisfies $u_2 = \mathbf{F}(\tau^{(j)})$. (In other words, $\mathcal{A}$ must reuse a value $\mathbf{F}(\tau^{(j)})$ previously obtained through $\mathcal{O}_{\mathbf{enc}}$ in order to win.)

The justification for this step requires a somewhat complex hybrid argument, which we outsource into Lemma 6.3. This lemma yields

$$\begin{aligned}\varepsilon_{C3} \;\leq\; \varepsilon_{C4} + 2\lambda \cdot \big(&3\mathrm{Adv}^{\text{fact}}_{\mathbb{G}_2,\mathcal{B}^{\text{fact}}_{C4}}(\lambda) + 6\mathrm{Adv}^{\text{mccpa}}_{\mathbb{G},\mathcal{B}^{\text{mccpa}}_{C4}}(\lambda) \\ &+ 8\mathrm{Adv}^{\text{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}_{C4}}(\lambda) + 2\mathrm{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{lin}},\mathcal{B}^{\text{lin}}_{C4}}(\lambda) + q/2^{3\lambda}\big)\end{aligned} \tag{37}$$

for suitable $\mathcal{B}^{\text{fact}}_{C4}$, $\mathcal{B}^{\text{mccpa}}_{C4}$, $\mathcal{B}^{\vee}_{C4}$, and $\mathcal{B}^{\text{lin}}_{C4}$.

Game C5 again alters the vectors $\mathbf{u}$ encrypted by $\mathcal{O}_{\mathbf{enc}}$, in the following way. Namely, instead of $\mathbf{u} = (1, \mathbf{F}(\tau), \mathbf{F}(\tau), 0)^\top$, we let $\mathcal{O}_{\mathbf{enc}}$ now encrypt the vector $\mathbf{u} = (1, X + \tau, X + \tau, 0)^\top$ for a single, initially randomly chosen $X \in \mathbb{Z}_{2^{3\lambda}}$. Besides, a final critical query is checked for $u_1 = 1$ and $u_2 = X + \tau^{(j)}$. Also here, an argument that we outsource into Lemma 6.4 shows

$$\begin{aligned}\varepsilon_{C4} \;\leq\; \varepsilon_{C5} + 2\lambda \cdot \big(&3\mathrm{Adv}^{\text{fact}}_{\mathbb{G}_2,\mathcal{B}^{\text{fact}}_{C5}}(\lambda) + 6\mathrm{Adv}^{\text{mccpa}}_{\mathbb{G},\mathcal{B}^{\text{mccpa}}_{C5}}(\lambda) \\ &+ 8\mathrm{Adv}^{\text{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}_{C5}}(\lambda) + 2\mathrm{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{lin}},\mathcal{B}^{\text{lin}}_{C5}}(\lambda) + \mathbf{O}(q/2^{\lambda})\big)\end{aligned} \tag{38}$$

for suitable $\mathcal{B}^{\text{fact}}_{C5}$, $\mathcal{B}^{\text{mccpa}}_{C5}$, $\mathcal{B}^{\vee}_{C5}$, and $\mathcal{B}^{\text{lin}}_{C5}$.

In **Game** C6, we change the third vector component u_3 in challenges prepared by $\mathcal{O}_{\mathbf{enc}}$. Specifically, instead of $\mathbf{u} = (1, X + \tau, X + \tau, 0)^\top$, we now let $\mathcal{O}_{\mathbf{enc}}$ encrypt $\mathbf{u} = (1, X + \tau, X, 0)^\top$. As in Game C2, we can invoke Lemma 3.4 to obtain

$$\varepsilon_{C5} \;\leq\; \varepsilon_{C6} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}_{C6}}(\lambda) \tag{39}$$

for a suitable adversary $\mathcal{B}^{\mathrm{mccpa}}_{C6}$.

Game C7 finally extends the final check on a critical query. Namely, while Game C6 only checked for $u_1 = 1$ and $u_2 = X + \tau^{(j)}$, Game C7 now checks the following equalities (and only outputs 1 if all of them hold):

$$u_1 \;=\; 1 \tag{40}$$

$$u_2 \;=\; X + \tau^{(j)} \qquad \text{for some } \tau^{(j)} \text{ from a previous } \mathcal{O}_{\mathbf{enc}} \text{ query} \tag{41}$$

$$u_2 \;=\; u_3 + \tau \tag{42}$$

$$u_3 \;=\; X. \tag{43}$$

(40) and (41) are already enforced in Game C6, but (42) and (43) require some justification.

First, all ciphertexts prepared by $\mathcal{O}_{\mathbf{enc}}$ satisfy $u_3 = Xu_1$. Thus, an adversary $\mathcal{B}^{\mathrm{lin}}_{C7}$ on $\mathbf{PS}^{\mathrm{lin}}$ can generate proofs π_{lin} through its $\mathcal{O}_{\mathbf{sim}}$ oracle, and use its $\mathcal{O}_{\mathbf{ver}}$ oracle to process $\mathcal{O}_{\mathbf{dec}}$ queries from $\mathcal{A}$. An analysis as in Game C3 shows that the introduction of (43) causes a statistical defect of at most $\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{C7}}(\lambda)$.

Next, (42) is guaranteed by the soundness of $\mathbf{PS}^{\mathrm{hash}}$. Namely, all ciphertexts prepared by $\mathcal{O}_{\mathbf{enc}}$ satisfy $u_2 = u_3 + \tau u_1$. Hence, an adversary $\mathcal{B}^{\mathrm{hash}}_{C7}$ on $\mathbf{PS}^{\mathrm{hash}}$ can generate proofs π_{hash} through its own $\mathcal{O}_{\mathbf{sim}}$ oracle, and use its $\mathcal{O}_{\mathbf{ver}}$ oracle to check the proofs π_{lin} in $\mathcal{A}$'s $\mathcal{O}_{\mathbf{dec}}$ queries. (If $\mathcal{O}_{\mathbf{ver}}$ answers $\perp$, $\mathcal{B}^{\mathrm{hash}}_{C7}$ rejects that decryption query and proceeds with the game.) An analysis as in Game C3 shows that the introduction of (42) causes a statistical defect of at most $\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{hash}},\mathcal{B}^{\mathrm{hash}}_{C7}}(\lambda)$. We note, however, that here, we also rely on π_{lin} to exclude the event that $\mathcal{A}$ submits an inconsistent ciphertext with $u_i = \perp$.

Taken together, we obtain

$$\varepsilon_{C6} \;\leq\; \varepsilon_{C7} + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{hash}},\mathcal{B}^{\mathrm{hash}}_{C7}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}_{C7}}(\lambda). \tag{44}$$

It is left to observe that the combined equations (40)-(43) imply that there is an index j with $\tau = \tau^{(j)}$. However, any such $\mathcal{O}_{\mathbf{dec}}$ query is already rejected by our change from Game M1, and thus the query cannot be critical. Thus, $\mathcal{A}$ cannot win in Game C7 anymore, and we get

$$\varepsilon_{C7} \;=\; 0. \tag{45}$$

Summing up (33)-(45) yields (27), as desired. □

6.2.3 Randomizing challenge ciphertexts

Outline. The following lemma is one of two core ingredients to establish the "reusal rule" mentioned in the outline of Lemma 6.2 (see Section 6.2.2). In a nutshell, this lemma randomizes the authentication tags in the u_2 component of challenge ciphertexts from $u_2 = X$ to $u_2 = \mathbf{F}(\tau^{(j)})$ (for a random function $\mathbf{F}$, and the hash value $\tau^{(j)}$ from that challenge ciphertext). Additionally, the adversary is forced to reuse one of those $\mathbf{F}(\tau^{(j)})$ values (for a previous $\tau^{(j)}$ from a challenge ciphertext) as u_2 in his decryption queries with $u_1 \neq 0$.

A little more concretely, the proof proceeds in a series of hybrids. The i-th hybrid enforces the rules above for $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$, where $\tau^{(j)}_{..i}$ is the i-bit prefix of $\tau^{(j)}$. To get from the i-th to the $(i+1)$-st hybrid, we again proceed in several steps:

#	u	M	game knows	winningcondition	remark
C0	$\begin{pmatrix}1\\ \mathbf{G}(\tau)\\ 0\\ 0\end{pmatrix}$	$\begin{pmatrix}1&0\\ 0&1\\ 0&0\\ 0&0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1	$u_1 \neq 0$	like M3, but outputs 1 iff $\mathcal{A}$ submits $\mathcal{O}_{\mathbf{dec}}$ query with $u_1 \neq 0$ that is not rejected
C1	$\begin{pmatrix}1\\ \mathbf{G}(\tau)\\ 0\\ 0\end{pmatrix}$	$\begin{pmatrix}1&0\\ 0&1\\ 0&0\\ 0&0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1	$u_1 = 1$	$\mathbb{G}_2$-factoring, $\mathbf{PS}^{\vee}$-soundness
C2	$\begin{pmatrix}1\\ X\\ X\\ 0\end{pmatrix}$	$\begin{pmatrix}1\\ X\\ X\\ 0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1	$u_1 = 1$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
C3	$\begin{pmatrix}1\\ X\\ X\\ 0\end{pmatrix}$	$\begin{pmatrix}1\\ X\\ X\\ 0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1, esk_2	$u_1 = 1$ $u_2 = X$	$\mathbf{PS}^{\text{lin}}$-soundness
C4	$\begin{pmatrix}1\\ \mathbf{F}(\tau)\\ \mathbf{F}(\tau)\\ 0\end{pmatrix}$	$\begin{pmatrix}1&0\\ 0&1\\ 0&1\\ 0&0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)})$ for some $\tau^{(j)}$ from a prev. $\mathcal{O}_{\mathbf{enc}}$ query	Lemma 6.3
C5	$\begin{pmatrix}1\\ X+\tau\\ X+\tau\\ 0\end{pmatrix}$	$\begin{pmatrix}1&0\\ 0&1\\ 0&1\\ 0&0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1, esk_2	$u_1 = 1$ $u_2 = X + \tau^{(j)}$	Lemma 6.4
C6	$\begin{pmatrix}1\\ X+\tau\\ X\\ 0\end{pmatrix}$	$\begin{pmatrix}1&0\\ 0&1\\ X&0\\ 0&0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1, esk_2	$u_1 = 1$ $u_2 = X + \tau^{(j)}$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
C7	$\begin{pmatrix}1\\ X+\tau\\ X\\ 0\end{pmatrix}$	$\begin{pmatrix}1&0\\ 0&1\\ X&0\\ 0&0\end{pmatrix}$	$\mathbf{psk}, xsk,$ esk_1, esk_2, esk_3	$u_1 = 1$ $u_2 = X + \tau^{(j)}$ $u_2 = u_3 + \tau$ $u_3 = X$	$\mathbf{PS}^{\text{hash}}$-soundness, $\mathbf{PS}^{\text{lin}}$-soundness

Figure 3: Games in the proof of Lemma 6.2. The "$\mathbf{u}$" column denotes the vector encrypted upon $\mathcal{O}_{\mathbf{enc}}$ queries, the columns of the matrix "$\mathbf{M}$" span the vector space generated by all "$\mathbf{u}$" values used in $\mathcal{O}_{\mathbf{enc}}$ queries, "game knows" denotes which information is necessary to perform the game, and "winning condition" denotes the conditions on a critical query under which the game finally outputs 1. (Any critical query that does not satisfy these conditions causes the game to output 0 immediately.)

Partitioning. First, we partition the set of challenge queries into two parts, according to the $(i+1)$-st bit $\tau^{(j)}_{i+1}$ of the hash value $\tau^{(j)}$ of that ciphertext. This partitioning is encoded into u_4 in the sense that $u_4 = \tau^{(j)}_{i+1}$ in all challenge queries. However, we stress that there is nothing that would force the adversary to use $u_4 = \tau_{i+1}$ for the corresponding hash value τ in his own decryption queries.

Decoupling the components of the partitioning. Next, we independently randomize the u_2 value in all challenge ciphertexts, in the sense that we set $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{..i})$ for two independent random functions $\mathbf{F}_0, \mathbf{F}_1$. To justify this step, we use a double encryption technique (that replicates u_2 in u_3) to modify encrypted values in the presence of a decryption oracle. This double encryption technique also entails that decryption needs to accept authentication tags $\mathbf{F}_b(\tau^{(j)}_{..i})$ for both $b = 0$ and $b = 1$.

Un-partitioning. Finally, if we set $\mathbf{F}(\tau^{(j)}_{..i+1}) := \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{..i})$, we obtain the $(i+1)$-st hybrid.

Lemma 6.3. *In the proof of Lemma 6.2, (37) holds. Concretely, there are $\mathcal{B}^{\text{fact}}$, $\mathcal{B}^{\text{mccpa}}$, $\mathcal{B}^{\text{lin}}$, and $\mathcal{B}^{\vee}$*

(with roughly the same complexity as Game C3*) such that*

$$\begin{aligned}\varepsilon_{C3} \;\leq\; \varepsilon_{C4} + 2\lambda \cdot \big(&3\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}}(\lambda) + 6\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}}(\lambda) \\ &+ 8\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}}(\lambda) + 2\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}}(\lambda) + \mathbf{O}(q/2^{\lambda})\big).\end{aligned} \tag{46}$$

Proof. We show (46) with a series of hybrids. To this end, recall our notation $\tau_{..i}$ for the i-bit prefix $(\tau_1, \ldots, \tau_i)$ of $\tau = (\tau_1, \ldots, \tau_{2\lambda})$. Now consider the following hybrid **Game** $\mathrm{Hyb}^{6.3}_i$, which is defined like Game C3, with the following exceptions:

- The vector $\mathbf{u}$ encrypted by $\mathcal{O}_{\mathbf{enc}}$ is $(1, \mathbf{F}(\tau_{..i}), \mathbf{F}(\tau_{..i}), 0)^\top$ (and not $(1, X, X, 0)^\top$).
- Analogously, a critical query is checked for $u_1 = 1$ and $u_2 = \mathbf{F}(\tau_{..i})$ (instead of $u_2 = X$).

By definition, these hybrids interpolate between Games C3 and C4, in the sense that

$$\varepsilon_{\mathrm{Hyb}^{6.3}_0} \;=\; \varepsilon_{C3} \qquad \text{and} \qquad \varepsilon_{\mathrm{Hyb}^{6.3}_{2\lambda}} \;=\; \varepsilon_{C4}. \tag{47}$$

Hence, to show the lemma, it suffices to show that for any $i \in \{0, \ldots, 2\lambda - 1\}$, the outputs of Games $\mathrm{Hyb}^{6.3}_i$ and $\mathrm{Hyb}^{6.3}_{i+1}$ are close. So fix an i, and consider the following sequence of games that interpolate between Game $\mathrm{Hyb}^{6.3}_i$ and Game $\mathrm{Hyb}^{6.3}_{i+1}$. An overview over the games is given in Fig. 4 (on p. 40).

Game R0 is defined exactly like Game $\mathrm{Hyb}^{6.3}_i$. Hence, by definition,

$$\varepsilon_{R0} \;=\; \varepsilon_{\mathrm{Hyb}^{6.3}_i}. \tag{48}$$

In **Game** R1, we change the vector $\mathbf{u}$ encrypted in ciphertexts generated by $\mathcal{O}_{\mathbf{enc}}$ from $(1, \mathbf{F}(\tau_{..i}), \mathbf{F}(\tau_{..i}), 0)^\top$ to $(1, \mathbf{F}(\tau_{..i}), \mathbf{F}(\tau_{..i}), \tau_{i+1})^\top$. (That is, we encrypt τ's $(i+1)$-th bit τ_{i+1} in u_4 now.) Since the secret key esk_4 is not used at this point, a straightforward reduction to the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$ yields

$$\varepsilon_{R0} \;\leq\; \varepsilon_{R1} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}_{R1}}(\lambda) \tag{49}$$

for a suitable adversary $\mathcal{B}^{\mathrm{mccpa}}_{R1}$.

In **Game** R2, we abort with output 0 if a critical query does not satisfy $u_4 \in \{0, 1\}$. We claim that

$$\varepsilon_{R1} \;\leq\; \varepsilon_{R2} + \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}_{R2}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee,2}_{R2}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee,3}_{R2}}(\lambda) \tag{50}$$

for adversaries $\mathcal{B}^{\mathrm{fact}}_{R2}$, $\mathcal{B}^{\vee,2}_{R2}$, and $\mathcal{B}^{\vee,3}_{R2}$ to be described. First, with an adversary $\mathcal{B}^{\mathrm{fact}}_{R2}$, we can bound the probability of any $\mathcal{O}_{\mathbf{dec}}$ query with u_4, $u_4 - 1$, or u_1 that lie in $\mathbb{Z}_{|\mathbb{G}_2|} \setminus (\mathbb{Z}^*_{|\mathbb{G}_2|} \cup \{0\})$. (This enables further reductions to the soundness of $\pi_{\vee,2}$ and $\pi_{\vee,3}$, and can be formalized as in Game M5.)

Next, $\mathcal{B}^{\vee,2}_{R2}$ simulates Game R2, and uses its own $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ oracles to generate and check $\mathbf{PS}^{\vee}$-proofs $\pi_{\vee,2}$. (Recall that $\pi_{\vee,2}$ intuitively enforces $u_4 \in \{0, 1\}$.) As in previous reductions, $\mathcal{B}^{\vee,2}_{R2}$ considers a proof $\pi_{\vee,2}$ invalid if $\mathcal{O}_{\mathbf{ver}}$ returns $\perp$. This happens if $u_4 \neq 0$ in the corresponding ciphertext, or if the ciphertext is no possible output of $\mathbf{E}$. The analysis of $\mathcal{B}^{\vee,2}_{R2}$ is similar to the one of $\mathcal{B}^{\mathrm{lin}}_{C3}$ from Game C3, but with an important difference.

Namely, $\mathcal{B}^{\vee,2}_{R2}$ may falsely reject $\mathcal{O}_{\mathbf{dec}}$ queries (with $u_1 = 0$ and $u_4 = 1$) that neither constitute $\mathbf{PS}^{\vee}$-forgeries nor would lead to the immediate termination of Game R1 or Game R2. Such queries are problematic, as they lead to a divergence of $\mathcal{B}^{\vee,2}_{R2}$'s simulation from Game R1 and Game R2, without obtaining a forged proof from that divergence. Let $\mathbf{bad}_2$ be the event that $\mathcal{A}$ places such a query (i.e., one with $u_1 = 0$, $u_4 = 1$, and valid proofs).

We additionally use the soundness of $\pi_{\vee,3}$ (which intuitively ensures $u_4 = 0 \vee u_1 = 1$) to exclude that $\mathbf{bad}_2$ happens *before* $\mathcal{B}^{\vee,2}_{R2}$ submits a valid forgery to its $\mathcal{O}_{\mathbf{ver}}$ oracle. Consider an adversary $\mathcal{B}^{\vee,3}_{R2}$ on $\mathbf{PS}^{\vee}$ that also simulates Game R2, and uses its own $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ oracles to generate and check $\mathbf{PS}^{\vee}$-proofs $\pi_{\vee,3}$. Again, if $\mathcal{O}_{\mathbf{ver}}$ returns $\perp$ (which is the case when $u_4 \neq 0$),

then $\mathcal{B}_{R2}^{\vee,3}$ considers the proof invalid. As above, $\mathcal{B}_{R2}^{\vee,3}$ perfectly simulates Game R2 until $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with valid $\boldsymbol{\pi}$ and $u_4 \neq 0$. Let us call that event $\mathbf{bad}_3$.

Observe that $\mathcal{B}_{R2}^{\vee,3}$ and $\mathcal{B}_{R2}^{\vee,2}$ perfectly simulate Game R2 and Game R1, as long as neither $\mathbf{bad}_2$, nor $\mathbf{bad}_3$, nor a critical query (i.e., one with $u_1 \neq 0$ and valid $\boldsymbol{\pi}$) occurs. However, if $\mathbf{bad}_2$ occurs first, then $\mathcal{B}_{R2}^{\vee,3}$ can extract a $\mathbf{PS}^{\vee}$-forgery from that query. Similarly, if $\mathbf{bad}_3$ occurs first, then $\mathcal{B}_{R2}^{\vee,2}$ can extract a $\mathbf{PS}^{\vee}$-forgery. Finally, if a critical query occurs, then $\mathcal{B}_{R2}^{\vee,2}$ can extract a $\mathbf{PS}^{\vee}$-forgery if that query distinguishes Game R1 and Game R2 (i.e., if $u_1 = 1$ and $u_4 \neq \{0,1\}$). Taken together, we get (50).

In **Game** R3, we initially independently and uniformly guess a bit $\beta \in \{0,1\}$, and later on abort (with output 0) if a critical query does not satisfy $u_4 = \beta$. Since we already enforced $u_4 \in \{0,1\}$ in Game R2, this change exactly halves the probability to output 1:

$$\varepsilon_{R2} \;=\; 2\varepsilon_{R3}. \tag{51}$$

Game R4 changes the vector $\mathbf{u}$ encrypted in challenge ciphertexts. Specifically, on top of the already used random function $\mathbf{F} : \{0,1\}^* \to \mathbb{Z}_{2^{3\lambda}}$, Game R4 defines another, independently and uniformly random function $\widetilde{\mathbf{F}} : \{0,1\}^* \to \mathbb{Z}_{2^{3\lambda}}$. For an easier exposition, we will write $\mathbf{F}_\beta := \mathbf{F}$ and $\mathbf{F}_{\overline{\beta}} := \widetilde{\mathbf{F}}$. (Recall our notation $\overline{\beta} = 1 - \beta$.) Now Game R4 encrypts $\mathbf{u} = (1, \mathbf{F}(\tau_{..i}), \mathbf{F}_{\tau_{i+1}}(\tau_{..i}), \tau_{i+1})^\top$ in challenge ciphertexts. Hence, if $\tau_{i+1} = \beta$, then Game R4 encrypts $(1, \mathbf{F}(\tau_{..i}), \mathbf{F}(\tau_{..i}), \tau_{i+1})^\top$ as before, and if $\tau_{i+1} = \overline{\beta}$, then $(1, \mathbf{F}(\tau_{..i}), \widetilde{\mathbf{F}}(\tau_{..i}), \tau_{i+1})^\top$. (This means that compared to Game R3, we only change encryptions when $\tau_{i+1} = \overline{\beta}$.)

Since Game R4 only modifies u_3 (and does not use the corresponding secret key esk_3), our change can be justified with the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$. Specifically, we have

$$\varepsilon_{R3} \;\leq\; \varepsilon_{R4} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}_{R4}^{\mathrm{mccpa}}}(\lambda) \tag{52}$$

for a suitable adversary $\mathcal{B}_{R4}^{\mathrm{mccpa}}$.

Game R5 now uses esk_3 (instead of esk_2) in checking critical queries. Specifically, recall that Game R4 checks a critical query for three conditions:
(a) $u_1 = 1$,
(b) $u_2 = \mathbf{F}(\tau_{..i}^{(j)})$ for some $\tau^{(j)}$ from a previous $\mathcal{O}_{\mathbf{enc}}$ query, and
(c) $u_4 = \beta$.
Here, we only change (b). Namely, instead of (b), we let Game R5 check for
(b′) $u_3 = \mathbf{F}(\tau_{..i}^{(j)})$ for some $\tau^{(j)}$ from a previous $\mathcal{O}_{\mathbf{enc}}$ query.
Hence, Game R5 only differs from Game R4 if $\mathcal{A}$ manages to submit a critical query with $u_2 \neq u_3$. However, note that we cannot rely on the soundness of $\mathbf{PS}^{\mathrm{lin}}$ at this point, since Game R5 also generates challenge ciphertexts with $u_2 \neq u_3$.

Instead, observe that all challenge ciphertexts satisfy $u_2 = u_3 \vee u_4 = \overline{\beta}$ and $|u_2 - u_3|_N < 2^{2\lambda}$. Hence, no challenge violates the soundness of the $(4+\overline{\beta})$-th $\mathbf{PS}^{\vee}$instance, and we would like to conclude that also the critical query must satisfy $u_2 = u_3 \vee u_4 = \overline{\beta}$. (Since $u_4 = \beta$ is enforced at this point on critical queries, this means that $u_2 = u_3$.) Due to the same complications as in Game R2, however, we will also require the soundness of $\pi_{\vee,6}$, as well as the $\mathbb{G}_2$-factoring assumption.

First, we invoke the $\mathbb{G}_2$-factoring assumption to ensure that none of u_4, $u_4 - 1$, $u_2 - u_3$, and $u_1 - 1$ lie in $\mathbb{Z}_{|\mathbb{G}_2|} \setminus (\mathbb{Z}^*_{|\mathbb{G}_2|} \cup \{0\})$. A detailed reduction (including an adversary $\mathcal{B}_{R5}^{\mathrm{fact}}$ that simulates Game R5) works exactly as in Game M5.

Next, consider an adversary $\mathcal{B}_{R5}^{\vee,4+\overline{\beta}}$ that simulates Game R5, and uses its own $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ oracles to generate and check proofs $\pi_{\vee,4+\overline{\beta}}$. As usual, if $\mathcal{O}_{\mathbf{sim}}$ outputs $\perp$, the proof is considered invalid. Let $\mathbf{bad}_{4+\overline{\beta}}$ be the event that $\mathcal{A}$ places an $\mathcal{O}_{\mathbf{dec}}$ query with $u_1 = 0$, $u_2 \neq u_3$, $u_4 = \overline{\beta}$, and valid $\boldsymbol{\pi}$. Observe that the simulation in $\mathcal{B}_{R5}^{\vee,4+\overline{\beta}}$ perfectly simulates Game R5 until

it either extracts a forged $\mathbf{PS}^{\vee}$-proof, or $\mathbf{bad}_{4+\overline{\beta}}$ occurs. (In the latter case, $\mathcal{B}_{\mathsf{R5}}^{\vee,4+\overline{\beta}}$ falsely rejects the query without extracting a forged $\mathbf{PS}^{\vee}$-proof.)

To exclude $\mathbf{bad}_{4+\overline{\beta}}$, consider an adversary $\mathcal{B}_{\mathsf{R5}}^{\vee,6}$ on $\mathbf{PS}^{\vee}$that also simulates Game R5, and uses its own $\mathcal{O}_{\mathbf{sim}}$ and $\mathcal{O}_{\mathbf{ver}}$ oracles to generate and check proofs $\pi_{\vee,6}$. Let us denote with $\mathbf{bad}_6$ the event that $\mathcal{A}$ submits an $\mathcal{O}_{\mathbf{dec}}$ query with $u_1 \neq 0$ and $u_2 = u_3$. Similar to the above, $\mathcal{B}_{\mathsf{R5}}^{\vee,6}$ perfectly simulates Game R5 until it extracts a forged $\mathbf{PS}^{\vee}$-proof, or $\mathbf{bad}_6$ occurs.

Like in Game R2, the combination of these reductions excludes $\mathbf{bad}_{4+\overline{\beta}}$ and $\mathbf{bad}_6$. Specifically, $\mathcal{B}_{\mathsf{R5}}^{\vee,4+\overline{\beta}}$ excludes $\mathbf{bad}_6$ (unless $\mathbf{bad}_{4+\overline{\beta}}$ occurs first), and $\mathcal{B}_{\mathsf{R5}}^{\vee,6}$ excludes $\mathbf{bad}_{4+\overline{\beta}}$ (unless $\mathbf{bad}_6$ occurs first). Furthermore, unless $\mathbf{bad}_{4+\overline{\beta}}$ occurs, $\mathcal{B}_{\mathsf{R5}}^{\vee,4+\overline{\beta}}$ shows that $u_2 = u_3$ in a critical query (which is the condition enforced in Game R5). Hence, we obtain

$$\varepsilon_{\mathsf{R4}} \leq \varepsilon_{\mathsf{R5}} + \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}_{\mathsf{R5}}^{\mathrm{fact}}} + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}_{\mathsf{R5}}^{\vee,4}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}_{\mathsf{R5}}^{\vee,5}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}_{\mathsf{R5}}^{\vee,6}}(\lambda). \tag{53}$$

Game R6 now exploits the fact that esk_2 is no longer used. Specifically, instead of encrypting $\mathbf{u} = (1, \mathbf{F}(\tau_{..i}), \mathbf{F}_{\tau_{i+1}}(\tau_{..i}), \tau_{i+1})^{\top}$, Game R6 encrypts $\mathbf{u} = (1, \mathbf{F}_{\tau_{i+1}}(\tau_{..i}), \mathbf{F}_{\tau_{i+1}}(\tau_{..i}), \tau_{i+1})^{\top}$ in challenge ciphertexts. Compared to Game R5, this means only u_2 changes, and thus we can once again rely on the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$ to justify our change. We obtain

$$\varepsilon_{\mathsf{R5}} \leq \varepsilon_{\mathsf{R6}} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}_{\mathsf{R6}}^{\mathrm{mccpa}}}(\lambda) \tag{54}$$

for an adversary $\mathcal{B}_{\mathsf{R6}}^{\mathrm{mccpa}}$ that simulates Game R6 or Game R5, depending on its challenge.

In **Game** R7, we reverse our change from Game R5. That is, we again use esk_2 (and not esk_3) in checks on critical queries. We can justify this change in two alternative ways: first, an argument completely analogous to that of Game R5 can justify this change with the soundness of $\mathbf{PS}^{\vee}$. Alternatively, observe that all challenge ciphertexts encrypted in Game R7 already satisfy $u_2 = u_3$. Hence, the soundness of $\mathbf{PS}^{\mathrm{lin}}$ also guarantees that $u_2 = u_3$ in critical queries.

Indeed, for our analysis, we justify our change with the soundness of $\mathbf{PS}^{\mathrm{lin}}$. Concretely, an argument as in Game C3 yields

$$\varepsilon_{\mathsf{R6}} \leq \varepsilon_{\mathsf{R7}} + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}_{\mathsf{R7}}^{\mathrm{lin}}}(\lambda) \tag{55}$$

for a suitable adversary $\mathcal{B}_{\mathsf{R7}}^{\mathrm{lin}}$ on $\mathbf{PS}^{\mathrm{lin}}$.

In **Game** R8, we reverse the change from Game R3. Concretely, recall that Game R7 checks a critical query for three conditions (where we use that $\mathbf{F}_{\beta} = \mathbf{F}$ by definition):
(a) $u_1 = 1$,
(b) $u_2 = \mathbf{F}_{\beta}(\tau^{(j)}_{..i})$ for some $\tau^{(j)}$ from a previous $\mathcal{O}_{\mathbf{enc}}$ query, and
(c) $u_4 = \beta$.
Observe that $\mathcal{A}$'s view in Game R7 does not depend on β. Hence, we can perform Game R7 with two independent random functions $\mathbf{F}_0, \mathbf{F}_1$, choosing β only when a critical query is submitted. Since this situation is completely symmetric with respect to β, we at least double the probability of a 1-output if we only check
(b′) $u_2 = \mathbf{F}_0(\tau^{(j)}_{..i})$ *or* $u_2 = \mathbf{F}_1(\tau^{(j)}_{..i})$ for some $\tau^{(j)}$ from a previous $\mathcal{O}_{\mathbf{enc}}$ query.
Game R8 proceeds exactly like this: it initially starts with $\mathbf{F}_0, \mathbf{F}_1$, and finally only checks (a) and (b′) (but not (c)) upon a critical query. We obtain:

$$2\varepsilon_{\mathsf{R7}} \leq \varepsilon_{\mathsf{R8}}. \tag{56}$$

(Note that even $2\varepsilon_{\mathsf{R7}} < \varepsilon_{\mathsf{R8}}$ can occur if $\mathcal{A}$ sometimes submits critical queries with $u_4 \notin \{0, 1\}$.)

Game R9 again changes the critical query condition. Specifically, Game R9 checks critical queries for $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{..i})$ for some $\tau^{(j)}$ (instead of condition (b′) from Game R8). Note that Game R9 hence implements a more stringent condition than Game R8, since Game R9 only

considers all $\mathbf{F}_{\tau_{i+1}^{(j)}}(\tau_{..i}^{(j)})$, where Game R8 considered all $\mathbf{F}_{\tau_{i+1}^{(j)}}(\tau_{..i}^{(j)})$ *and* all $\mathbf{F}_{\overline{\tau_{i+1}^{(j)}}}(\tau_{..i}^{(j)})$. However, to provoke a different behavior of these games, $\mathcal{A}$ would have to submit a critical query with

$$\begin{aligned} u_2 &\neq \mathbf{F}_{\tau_{i+1}^{(j)}}(\tau_{..i}^{(j)}) && \text{for all } \tau^{(j)} \text{ from previous } \mathcal{O}_{\mathbf{enc}} \text{ queries, but} \\ u_2 &= \mathbf{F}_{\overline{\tau_{i+1}^{*}}}(\tau_{..i}^{*}) && \text{for a } \tau^{*} \text{ from a previous } \mathcal{O}_{\mathbf{enc}} \text{ query.} \end{aligned} \tag{57}$$

Such a critical query would lead to a 1-output in Game R8, but to a 0-output in Game R9. Of course, a critical query that achieves (57) can only occur if no previous $\tau^{(j)}$ satisfies $\tau_{..i}^{(j)} = \tau_{..i}^{*}$ and $\tau_{i+1}^{(j)} = \overline{\tau_{i+1}^{*}}$. In other words, $\mathbf{F}_{\overline{\tau_{i+1}^{*}}}$ has not been queried previously on $\tau_{..i}^{*}$, and $\mathcal{A}$'s view is thus independent of $\mathbf{F}_{\overline{\tau_{i+1}^{*}}}(\tau_{..i}^{*})$. Hence, $\mathcal{A}$ would have to correctly guess a uniformly random value $\mathbf{F}_{\overline{\tau_{i+1}^{*}}}(\tau_{..i}^{*}) \in \mathbb{Z}_{2^{3\lambda}}$ in order to provoke a difference between Game R8 and Game R9. Since there are at most q previous $\tau^{(j)}$ (and thus potential matches for $\mathcal{A}$'s guess), we obtain

$$\varepsilon_{R8} \;\leq\; \varepsilon_{R9} + q/2^{3\lambda}. \tag{58}$$

Game R10 always encrypts $u_4 = 0$ (instead of $u_4 = \tau_{i+1}$) in challenge ciphertexts. Since esk_4 is no longer used in Game R10, a straightforward reduction to the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$ yields

$$\varepsilon_{R9} \;\leq\; \varepsilon_{R10} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}_{R10}}(\lambda) \tag{59}$$

for a suitable IND-MCCPA adversary $\mathcal{B}^{\mathrm{mccpa}}_{R10}$.

Furthermore, if we simply set $\mathbf{F}(\tau_{..i+1}) := \mathbf{F}_{\tau_{i+1}}(\tau_{..i})$, we see that Game R10 is simply a reformulation of Game $\mathrm{Hyb}^{6.3}_{i+1}$. Hence,

$$\varepsilon_{R10} \;=\; \varepsilon_{\mathrm{Hyb}^{6.3}_{i+1}}. \tag{60}$$

Putting (48)-(60) together, and combining adversaries on the same scheme into one, we obtain

$$\begin{aligned} \varepsilon_{\mathrm{Hyb}^{6.3}_{i}} \;\leq\; \varepsilon_{\mathrm{Hyb}^{6.3}_{i+1}} &+ 3\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}}(\lambda) + 6\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}}(\lambda) \\ &+ 8\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}}(\lambda) + 2\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}}(\lambda) + q/2^{3\lambda}. \end{aligned}$$

If we let these adversaries guess $i \in \{0, \ldots, 2\lambda - 1\}$ uniformly, we obtain (46). □

6.2.4 Derandomizing challenge ciphertexts and adding dependencies

Outline. In the next lemma, we will turn the random value $\mathbf{F}(\tau^{(j)})$ encrypted in the u_2 component of challenge ciphertexts into a "less random" value $X + \tau^{(j)}$, where X is a random (but initially chosen and constant) value. Analogously, the decryption rule will require that $u_2 = X + \tau^{(j)}$ (for a hash value $\tau^{(j)}$ from a previous challenge ciphertext) in all decryption queries with $u_1 \neq 0$. We note that establishing this decryption rule has been the goal from the start. However, the intermediate randomization of u_2 seems to be necessary to achieve this goal, since it enables the bitwise introduction of $\tau^{(j)}$ "behind a shielding random value $\mathbf{F}(\tau^{(j)})$".

The concrete steps of modifying u_2 are very similar to those from the proof of Lemma 6.3. A little more specifically, we will use a hybrid argument, where the i-th hybrid encrypts (and checks) $u_2 = \mathbf{F}(\tau_{i+1..}^{(j)}) + \tau_{..i}^{(j)}$, where $\tau_{..i}^{(j)}$ and $\tau_{i+1..}^{(j)}$ are the i-bit prefix, resp. the $(2\lambda - i)$-bit postfix of $\tau^{(j)}$. In other words, each hybrid step trades one bit of $\mathbf{F}$-input for one bit outside of the $\mathbf{F}$-evaluation. As in the proof of Lemma 6.3, each step will use a double encryption technique (that involves u_2 and u_3).

#	u	M	game knows	winning condition	remark
R0	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{..i}) \\ \mathbf{F}(\tau_{..i}) \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$ for some $\tau^{(j)}$ from a prev. $\mathcal{O}_{\mathbf{enc}}$ query	same as $\mathrm{Hyb}^{6.3}_{i}$
R1	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{..i}) \\ \mathbf{F}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
R2	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{..i}) \\ \mathbf{F}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$ $u_4 \in \{0,1\}$	$\mathbb{G}_2$-factoring, $\mathbf{PS}^{\vee}$-soundness
R3	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{..i}) \\ \mathbf{F}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$ $u_4 = \beta$	halves ε ($\beta \in \{0,1\}$ random)
R4	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$ $u_4 = \beta$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA $(\mathbf{F}_\beta(\tau_{..i}) = \mathbf{F}(\tau_{..i}),$ $\mathbf{F}_{\bar{\beta}}(\tau_{..i}) = \tilde{\mathbf{F}}(\tau_{..i}))$
R5	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_3, esk_4	$u_1 = 1$ $u_3 = \mathbf{F}(\tau^{(j)}_{..i})$ $u_4 = \beta$	$\mathbb{G}_2$-factoring, $\mathbf{PS}^{\vee}$-soundness
R6	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_3, esk_4	$u_1 = 1$ $u_3 = \mathbf{F}(\tau^{(j)}_{..i})$ $u_4 = \beta$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
R7	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{..i})$ $u_4 = \beta$	$\mathbf{PS}^{\mathrm{lin}}$-soundness
R8	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}_b(\tau^{(j)}_{..i})$ for $b = 0$ or $b = 1$	at least doubles ε
R9	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \tau_{i+1} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{..i})$	equivalent up to guessing unknown $\mathbf{F}_0, \mathbf{F}_1$-images
R10	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ \mathbf{F}_{\tau_{i+1}}(\tau_{..i}) \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{..i})$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA, same as $\mathrm{Hyb}^{6.3}_{i+1}$

Figure 4: Games in the proof of Lemma 6.3. The columns carry the same meaning as in Fig. 3: "**u**" denotes the vector encrypted upon $\mathcal{O}_{\mathbf{enc}}$ queries, the columns of "**M**" span the vector space generated by all "**u**" values, "game knows" denotes which information is necessary to perform the game, and "winning condition" denotes the conditions on a critical query under which the game finally outputs 1.

Lemma 6.4. *In the proof of Lemma 6.2, (38) holds. Concretely, there are* $\mathcal{B}^{\mathrm{fact}}$, $\mathcal{B}^{\mathrm{mccpa}}$, $\mathcal{B}^{\mathrm{lin}}$, *and* $\mathcal{B}^{\vee}$ *(with roughly the same complexity as Game* C4*) such that*

$$\begin{aligned} \varepsilon_{\mathrm{C4}} \leq \varepsilon_{\mathrm{C5}} + 2\lambda \cdot \big(&3\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}}(\lambda) + 6\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}}(\lambda) \\ &+ 8\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\vee},\mathcal{B}^{\vee}}(\lambda) + 2\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}}(\lambda) + \mathbf{O}(q/2^{\lambda})\big). \end{aligned} \tag{61}$$

Proof. We show (61) with a series of hybrids that is very similar to that from the proof of

Lemma 6.3. Again, recall our notation $\tau_{..i}$ (resp. $\tau_{i..}$) for the i-bit prefix $(\tau_1, \ldots, \tau_i)$ (resp. for the $(2\lambda - i + 1)$-bit postfix $(\tau_i, \ldots, \tau_{2\lambda})$) of $\tau = (\tau_1, \ldots, \tau_{2\lambda})$. Furthermore, recall that we interpret τ also as an integer $\tau = \sum_{i=1}^{2\lambda} 2^{i-1}\tau_i$ when convenient (i.e., in addition of exponents).

Now consider the following hybrid **Game** $\mathrm{Hyb}_i^{6.4}$, which is defined like Game C4, with the following exceptions:

- $\mathcal{O}_{\mathbf{enc}}$ encrypts a vector $\mathbf{u} = (1, \mathbf{F}(\tau_{i+1..}) + \tau_{..i}, \mathbf{F}(\tau_{i+1..}) + \tau_{..i}, 0)^\top$ (and not $(1, \mathbf{F}(\tau), \mathbf{F}(\tau), 0)^\top$).
- A critical query is checked for $u_1 = 1$ and $u_2 = \mathbf{F}(\tau_{i+1..}) + \tau_{..i}$ (instead of $u_2 = \mathbf{F}(\tau)$).

By definition, these hybrids interpolate between Games C4 and C5, in the sense that

$$\varepsilon_{\mathrm{Hyb}_0^{6.4}} = \varepsilon_{\mathrm{C4}} \qquad \text{and} \qquad \varepsilon_{\mathrm{Hyb}_{2\lambda}^{6.4}} = \varepsilon_{\mathrm{C5}}. \tag{62}$$

Hence, it suffices to show that for any $i \in \{0, \ldots, 2\lambda - 1\}$, the outputs of Games $\mathrm{Hyb}_i^{6.4}$ and $\mathrm{Hyb}_{i+1}^{6.4}$ are close. So fix an i, and consider the following sequence of games that interpolate between Games $\mathrm{Hyb}_i^{6.4}$ and $\mathrm{Hyb}_{i+1}^{6.4}$. The games are summarized in Fig. 5 (on p. 44).

Game H0 is defined exactly like Game $\mathrm{Hyb}_i^{6.4}$. Hence, by definition,

$$\varepsilon_{\mathrm{H0}} = \varepsilon_{\mathrm{Hyb}_i^{6.4}}. \tag{63}$$

Our first modifications are completely analogous to those from the proof of Lemma 6.3. Specifically, **Game** H1 changes the encrypted u_4 from 0 to τ_{i+1}. This change can be justified with the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$, and we get

$$\varepsilon_{\mathrm{H0}} \leq \varepsilon_{\mathrm{H1}} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G}, \mathcal{B}_{\mathrm{H1}}^{\mathrm{mccpa}}}(\lambda) \tag{64}$$

for a straightforward adversary $\mathcal{B}_{\mathrm{H1}}^{\mathrm{mccpa}}$. Next, **Game** H2 aborts with output 0 if a critical query does not satisfy $u_4 \in \{0, 1\}$. Just like in the proof of Lemma 6.3, we obtain

$$\varepsilon_{\mathrm{H1}} \leq \varepsilon_{\mathrm{H2}} + \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2, \mathcal{B}_{\mathrm{H2}}^{\mathrm{fact}}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^\vee, \mathcal{B}_{\mathrm{H2}}^{\vee,2}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^\vee, \mathcal{B}_{\mathrm{H2}}^{\vee,3}}(\lambda) \tag{65}$$

for suitable $\mathcal{B}_{\mathrm{H2}}^{\mathrm{fact}}$, $\mathcal{B}_{\mathrm{H2}}^{\vee,2}$, and $\mathcal{B}_{\mathrm{H2}}^{\vee,3}$. **Game** H3 initially chooses $\beta \in \{0, 1\}$, and aborts (with output 0) if a critical query does not satisfy $u_4 = \beta$. This change exactly halves the probability to output 1:

$$\varepsilon_{\mathrm{H2}} = 2\varepsilon_{\mathrm{H3}}. \tag{66}$$

In **Game** H4, we write the evaluation $\mathbf{F}(\tau_{i+1..})$ used in u_2, u_3 of encrypted values, and in the final check on critical queries differently. Namely, instead of $\mathbf{F}(\tau_{i+1..})$, we use $\mathbf{F}_{\tau_{i+1}}(\tau_{i+2..})$ for two independently chosen random functions $\mathbf{F}_0, \mathbf{F}_1$. This change is purely conceptual, and we obtain

$$\varepsilon_{\mathrm{H3}} = \varepsilon_{\mathrm{H4}}. \tag{67}$$

In the next four games, we modify the values encrypted in u_2 and u_3, with exactly the same reasoning as in Games R4 – R7. Concretely, **Game** H5 encrypts $u_3 = \mathbf{F}_\beta(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1}$ (instead of encrypting $u_3 = \mathbf{F}_{\tau_{i+1}}(\tau_{i+2..}) + \tau_{..i}$). Since Game H5 does not use esk_3, a straightforward reduction to the security of $(\mathbf{E}, \mathbf{D})$ yields

$$\varepsilon_{\mathrm{H4}} \leq \varepsilon_{\mathrm{H5}} + \mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G}, \mathcal{B}_{\mathrm{H5}}^{\mathrm{mccpa}}}(\lambda) \tag{68}$$

for a suitable adversary $\mathcal{B}_{\mathrm{H5}}^{\mathrm{mccpa}}$. Next, **Game** H6 uses esk_3 (instead of esk_2) in checking critical queries. An argument completely analogous to that of Game R5 yields

$$\varepsilon_{\mathrm{H5}} \leq \varepsilon_{\mathrm{H6}} + \mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2, \mathcal{B}_{\mathrm{H6}}^{\mathrm{fact}}} + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^\vee, \mathcal{B}_{\mathrm{H6}}^{\vee,4}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^\vee, \mathcal{B}_{\mathrm{H6}}^{\vee,5}}(\lambda) + \mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^\vee, \mathcal{B}_{\mathrm{H6}}^{\vee,6}}(\lambda) \tag{69}$$

for suitable adversaries $\mathcal{B}^{\text{fact}}_{\text{H6}}$, $\mathcal{B}^{\vee,4}_{\text{H6}}$, $\mathcal{B}^{\vee,5}_{\text{H6}}$, and $\mathcal{B}^{\vee,6}_{\text{H6}}$. **Game** H7 encrypts $u_2 = \mathbf{F}_\beta(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1}$ (not $u_2 = \mathbf{F}_{\tau_{i+1}}(\tau_{i+2..}) + \tau_{..i}$). Since Game H7 does not use esk_2, we get

$$\varepsilon_{\text{H6}} \;\leq\; \varepsilon_{\text{H7}} + \text{Adv}^{\text{mccpa}}_{\mathbb{G},\mathcal{B}^{\text{mccpa}}_{\text{H7}}}(\lambda) \tag{70}$$

for a suitable adversary $\mathcal{B}^{\text{mccpa}}_{\text{H7}}$. Finally, **Game** H8 again uses esk_2 (and not esk_3) to check critical queries. Again, an argument as in Game R7 shows of an adversary $\mathcal{B}^{\text{lin}}_{\text{H8}}$ with

$$\varepsilon_{\text{H7}} \;\leq\; \varepsilon_{\text{H8}} + \text{Adv}^{\text{snd}}_{\mathbf{PS}^{\text{lin}},\mathcal{B}^{\text{lin}}_{\text{H8}}}(\lambda). \tag{71}$$

In **Game** H9, we slightly change the check on critical queries. Namely, recall that Game H8 outputs 1 upon a critical query if $u_1 = 1$, $u_4 = \beta$, and $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ for some $\tau^{(j)}$ from a previous $\mathcal{O}_{\textbf{enc}}$ query. Instead, Game H9 outputs 1 if $u_1 = 1$, $u_4 = \beta$, and $u_2 = \mathbf{F}_\beta(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ from a previous $\mathcal{O}_{\textbf{enc}}$ query with $\tau^{(j)}_{i+1} = \beta$. (Thus, compared to Game H8, Game H9 does not consider $\tau^{(j)}$ with $\tau^{(j)}_{i+1} = \overline{\beta}$.) To justify this change, observe that Game H9 only uses $\mathbf{F}_\beta$ (but never $\mathbf{F}_{\overline{\beta}}$) in its own encryptions. Hence, $\mathcal{A}$'s view is independent of $\mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..})$ for $\tau^{(j)}_{i+1} \neq \beta$. Thus, any critical query that would be accepted by Game H8 but rejected by Game H9 would imply that $\mathcal{A}$ has guessed one out of at most q independently random and hidden $\mathbf{F}_{\overline{\beta}}$-images. Hence,

$$\varepsilon_{\text{H8}} \;\leq\; \varepsilon_{\text{H9}} + q/2^{3\lambda}. \tag{72}$$

Next, in **Game** H10, we use a freshly chosen random function $\mathbf{F} : \{0,1\}^* \to \mathbb{Z}_{2^{3\lambda}}$ in place of $\mathbf{F}_\beta(\tau^{(j)}_{i+2..}) - \beta 2^i$. Specifically, Game H10 encrypts $u_2 = u_3 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i+1}$, and finally checks a critical query for $u_2 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \beta 2^i + \tau^{(j)}_{..i}$ (for some $\tau^{(j)}$ with $\tau^{(j)}_{i+1} = \beta$). Note that this check on critical queries then becomes equivalent to checking $u_2 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i+1}$, since $\tau^{(j)}_{i+1} = \beta$ implies that

$$\beta 2^i + \tau^{(j)}_{..i} \;=\; \tau^{(j)}_{i+1} 2^i + \sum_{\ell=1}^{i} 2^{\ell-1}\tau^{(j)}_\ell \;=\; \sum_{\ell=1}^{i+1} 2^{\ell-1}\tau^{(j)}_\ell \;=\; \tau^{(j)}_{..i+1}.$$

Furthermore, we claim that the respective output values of Game H9 and Game H10 are statistically close. Indeed, for every input x, we have $\mathbf{SD}\big(\mathbf{F}_\beta(x) - \beta 2^i\,;\, \mathbf{F}(x)\big) \leq 1/2^\lambda$. Hence, summing up this statistical distance over all possible $q+1$ evaluations of $\mathbf{F}$, we obtain

$$\varepsilon_{\text{H9}} \;\leq\; \varepsilon_{\text{H10}} + (q+1)/2^\lambda. \tag{73}$$

In **Game** H11, we do not initially guess β, and instead check a possible critical query for $u_2 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i+1}$ for any previous $\tau^{(j)}$. Hence, Game H11 considers all previous $\mathcal{O}_{\textbf{enc}}$ queries $\tau^{(j)}$ (and not only those with $\tau^{(j)}_{i+1} = \beta$) in checking critical queries. Since $\mathcal{A}$'s view is completely independent of β already in Game H10, by symmetry we have

$$2\varepsilon_{\text{H10}} \;\leq\; \varepsilon_{\text{H11}}. \tag{74}$$

Finally, **Game** H12 always encrypts $u_4 = 0$ (instead of $u_4 = \tau_{i+1}$) in challenge ciphertexts. Since esk_4 is no longer used in Game H12, a straightforward reduction to the IND-MCCPA security of $(\mathbf{E}, \mathbf{D})$ yields

$$\varepsilon_{\text{H11}} \;\leq\; \varepsilon_{\text{H12}} + \text{Adv}^{\text{mccpa}}_{\mathbb{G},\mathcal{B}^{\text{mccpa}}_{\text{H12}}}(\lambda) \tag{75}$$

for a suitable IND-MCCPA adversary $\mathcal{B}^{\text{mccpa}}_{\text{H12}}$. Furthermore, Game H12 is simply a reformulation of Game $\text{Hyb}^{6.4}_{i+1}$. Hence,

$$\varepsilon_{\text{H12}} \;=\; \varepsilon_{\text{Hyb}^{6.4}_{i+1}}. \tag{76}$$

Putting (63)-(76) together, and combining adversaries on the same scheme into one, we obtain

$$\varepsilon_{\mathrm{Hyb}_i^{6.4}} \leq \varepsilon_{\mathrm{Hyb}_{i+1}^{6.4}} + 3\mathrm{Adv}^{\mathrm{fact}}_{\mathbb{G}_2,\mathcal{B}^{\mathrm{fact}}}(\lambda) + 6\mathrm{Adv}^{\mathrm{mccpa}}_{\mathbb{G},\mathcal{B}^{\mathrm{mccpa}}}(\lambda) + 8\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^\vee,\mathcal{B}^\vee}(\lambda) + 2\mathrm{Adv}^{\mathrm{snd}}_{\mathbf{PS}^{\mathrm{lin}},\mathcal{B}^{\mathrm{lin}}}(\lambda) + \mathbf{O}(q/2^\lambda).$$

If we let these adversaries guess $i \in \{0,\dots,2\lambda-1\}$ uniformly, we obtain (61). □

6.3 Security in the multi-user setting

Like in similar works (e.g., [25, 26, 12]), our main security analysis (Theorem 6.1) considers the multi-challenge, but single-user setting. This choice was made in order to simplify the presentation. However, the analysis also applies almost verbatim in the multi-user, multi-challenge setting. We sketch the necessary modifications to the analysis in the following. (The scheme **KEM** itself does not require any modifications.)

- The IND-MCCA security definition (Definition 2.2) needs to be adapted to many keypairs (pk_j, sk_j). The adversary $\mathcal{A}$ initially obtains all pk_j, and can select an index j that selects the scheme instance upon each $\mathcal{O}_{\mathbf{enc}}$ and $\mathcal{O}_{\mathbf{dec}}$ query. However, we stress that the challenge bit b chosen by the experiment applies to all instances. (That is, depending on b, either all keys K returned by $\mathcal{O}_{\mathbf{enc}}$ are real for all instances j, or all are random.)
- Similarly, the IND-MCCPA definition (Definition 3.3) needs to be adapted to many vectors $\mathbf{pk}_j$ and $\mathbf{sk}_j$. However, each of these $\mathbf{pk}_i$ contains the same number n of component public keys *epk*, and the "challenge index" i^* that $\mathcal{A}$ initially selects applies to all instances.
- As a result, the generalized ElGamal encryption from Section 3.1.3 must be proved tightly secure in a setting with multiple public vectors $\mathbf{pk}_j$ of public keys. This requires a slight change to the randomization in Lemma 3.4. Namely, instead of setting $epk_{i^*} = g_1^{\omega^{*\top}\mathbf{B}}$, we set $epk_{j,i^*} = g_1^{(\omega^*+\omega'_{j,i^*})^\top\mathbf{B}}$ for the i^*-th component public key of the j-th scheme instance, where $\omega'_{j,i^*} \in \mathbb{Z}^{\ell_{\mathbf{B}}}_{|\mathbb{G}_1|}$ is a freshly chosen blinding value. This allows to embed a single GDDH challenge into all i^*-th component public keys, and to generalize the argument from Lemma 3.4 to many scheme instances. (The corresponding reduction to GDDH is still completely tight.)
- The key extractor indistinguishability definition (Definition 4.4), and the corresponding analysis of **EXT** (Lemma 4.5) need to be adapted to support many public key instances epk_i. This can be done as with the IND-MCCPA security of $(\mathbf{E},\mathbf{D})$, by setting up $esk_j = g_1^{\omega+\omega'_j}$, where ω is from the GDDH challenge, and ω'_j is an instance-dependent randomization value. $\mathcal{O}_{\mathbf{cha}}$ and $\mathcal{O}_{\mathbf{ext}}$ queries are then answered using the corresponding blinded key $\omega + \omega'_j$.
- The one-time signature construction and its analysis support many instances in the first place, and do not need to be adapted. Similarly, the security of all considered proof systems is statistical, and hence their analysis also does not need to be adapted.
- In the security analysis of our KEM (Theorem 6.1), all changes in hybrid games are applied to all scheme instances j simultaneously. Hence, the structure of the proofs does not change, only hybrid modifications are now justified with the appropriate *multi-instance* security of the underlying building blocks.

6.4 Performance and optimizations

The efficiency characteristics of our scheme of course highly depend on the used building blocks, and thus also on the considered setting. In the following, we hence distinguish between the prime-order and the DCR setting. In both cases, however, we rely on the generic

#	u	game knows	winning condition	remark
H0	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ 0 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+1..}) + \tau^{(j)}_{..i}$ for some $\tau^{(j)}$ from a prev. $\mathcal{O}_{\mathbf{enc}}$ query	same as $\mathrm{Hyb}^{6.4}_{i}$
H1	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+1..}) + \tau^{(j)}_{..i}$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
H2	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+1..}) + \tau^{(j)}_{..i}$ $u_4 \in \{0,1\}$	$\mathbb{G}_2$-factoring, $\mathbf{PS}^{\vee}$-soundness
H3	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \mathbf{F}(\tau_{i+1..}) + \tau_{..i} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+1..}) + \tau^{(j)}_{..i}$ $u_4 = \beta$	halves ε ($\beta \in \{0,1\}$ random)
H4	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{i+2..}) + \tau_{..i} \\ \mathbf{F}_{\tau_{i+1}}(\tau_{i+2..}) + \tau_{..i} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ $u_4 = \beta$	conceptual change ($\mathbf{F}_0, \mathbf{F}_1$ independent)
H5	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{i+2..}) + \tau_{..i} \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ $u_4 = \beta$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
H6	$\begin{pmatrix} 1 \\ \mathbf{F}_{\tau_{i+1}}(\tau_{i+2..}) + \tau_{..i} \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_3, esk_4	$u_1 = 1$ $u_3 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ $u_4 = \beta$	$\mathbb{G}_2$-factoring, $\mathbf{PS}^{\vee}$-soundness
H7	$\begin{pmatrix} 1 \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_3, esk_4	$u_1 = 1$ $u_3 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ $u_4 = \beta$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA
H8	$\begin{pmatrix} 1 \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}_{\tau^{(j)}_{i+1}}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ $u_4 = \beta$	$\mathbf{PS}^{\mathrm{lin}}$-soundness
H9	$\begin{pmatrix} 1 \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \mathbf{F}_{\beta}(\tau_{i+2..}) - \beta 2^i + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}_{\beta}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i}$ for some $\tau^{(j)}$ with $\tau^{(j)}_{i+1} = \beta$ $u_4 = \beta$	$\mathbf{F}_{\overline{\beta}}$ hidden
H10	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+2..}) + \tau_{..i+1} \\ \mathbf{F}(\tau_{i+2..}) + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i+1}$ for some $\tau^{(j)}$ with $\tau^{(j)}_{i+1} = \beta$ $u_4 = \beta$	$\mathbf{F}_{\beta}(\tau^{(j)}_{i+2..})$ and $\mathbf{F}_{\beta}(\tau^{(j)}_{i+2..}) + \beta 2^i$ statistically close
H11	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+2..}) + \tau_{..i+1} \\ \mathbf{F}(\tau_{i+2..}) + \tau_{..i+1} \\ \tau_{i+1} \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2, esk_4	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i+1}$ for some $\tau^{(j)}$	at least doubles ε
H12	$\begin{pmatrix} 1 \\ \mathbf{F}(\tau_{i+2..}) + \tau_{..i+1} \\ \mathbf{F}(\tau_{i+2..}) + \tau_{..i+1} \\ 0 \end{pmatrix}$	**psk**, *xsk*, esk_1, esk_2	$u_1 = 1$ $u_2 = \mathbf{F}(\tau^{(j)}_{i+2..}) + \tau^{(j)}_{..i+1}$	$(\mathbf{E}, \mathbf{D})$ IND-MCCPA, same as $\mathrm{Hyb}^{6.4}_{i+1}$

Figure 5: Games in the proof of Lemma 6.4. The columns have the same meaning as in Fig. 3.

benign proof systems from Section 5.2.1 (for the generic linear language), Section 5.3.1 (for the parameterized linear language), and on the generic key extractor from Section 4.2.1.

6.4.1 In the prime-order setting

Without any optimizations. If $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ for prime $|\mathbb{G}|$, we can use the pairing-based benign proof system $\mathbf{PS}^{\vee}_{\mathbf{pair}}$ from Section 5.4.1, and the one-time signature scheme from Section 4.1.1. This leads to public keys of $12\ell_{\mathbf{B}}^2 + 20\ell_{\mathbf{B}}$ group elements, and ciphertexts of $\ell_{\mathbf{B}} + 14$ group elements and 2 exponents. Hence, if we desire a DLIN-based (i.e., 2-LIN-based) scheme, we obtain public keys of 88 group elements, and ciphertexts of 16 group elements and 2 exponents. Moreover, the corresponding matrix $g_1^{\mathbf{B}} \in \mathbb{G}^{2\times 2}$ in the public parameters takes up 2 group elements (since in case of DLIN, only the diagonal of $\mathbf{B}$ is nontrivial [10]). However, by optimizing our scheme (and using certain non-generic, setting-specific properties), we can do much better. We give details on these optimizations in the following.

Dropping the proofs from the ciphertext. We first observe that in the prime-order setting, all involved proof systems are actually hash proof systems (in the sense of [9]). Hence, proofs π can be computed from the respective statements x deterministically, using either a witness and the public key *ppk*, or the secret key *psk*. Furthermore, proofs for false statements look uniformly random (and are not just unpredictable, as in Definition 5.2). Hence, we can use a standard trick (from [8, 9, 24]), drop $\boldsymbol{\pi}$ from the ciphertext, and instead define the key K as

$$K \;=\; \mathbf{Ext}_{\mathrm{pub}}(\mathit{xpk}, (\mathbf{c}_0, c_1), \mathbf{r}) \;\oplus\; h(\boldsymbol{\pi}),$$

where h is the universal hash function from **EXT**. The resulting KEM will only be secure under *constrained* chosen-ciphertext attacks [18], a notion which however is sufficient for (tightly secure) efficient hybrid encryption.

Dropping unnecessary proofs. Dropping proofs from the ciphertext compresses ciphertexts, but leaves the public key unchanged. However, we can also compress the public key by dropping certain proof systems altogether. Concretely, the pairing-based universal hash proof system $\mathbf{PS}^{\vee}_{\mathbf{pair}}$ for disjunctions from [1] (cf. Section 5.4.1) actually proves more than membership in $\mathcal{L}^{\vee}$. In the notation from Section 5.4, this proof system is $(\mathcal{L}^{\vee}_{\mathrm{sim},\mathbf{pk}}, \mathcal{L}^{\vee}_{\mathrm{ver},\mathbf{pk}}, \mathcal{L}^{\vee}_{\mathrm{snd},\mathbf{pk}})$-benign for

$$\mathcal{L}^{\vee}_{\mathrm{sim},\mathbf{pk}} \;=\; \mathcal{L}^{\vee}_{\mathrm{ver},\mathbf{pk}} \;=\; \mathcal{L}^{\vee}_{\mathrm{snd},\mathbf{pk}} \;=\; \bigl\{\mathbf{E}_{\mathbf{pk}}(\mathbf{u};\mathbf{r}) \;\mid\; u_1 = 0 \vee u_2 = 0\bigr\}.$$

Hence, in this case, the "helper proofs" $\pi_{\vee,3}$ and $\pi_{\vee,6}$ (that help manage the weak original $\mathcal{L}^{\vee}_{\mathrm{ver}}$ definition from Section 5.4) are not necessary, and can be dropped from the scheme. For similar reasons, the proof $\pi_{\vee,1}$ can be omitted. As a result, the scheme really only requires 3 instances of $\mathbf{PS}^{\vee}_{\mathbf{pair}}$ (for $\pi_{\vee,2}$, $\pi_{\vee,4}$, and $\pi_{\vee,5}$).

Dropping the one-time signature scheme. Moreover, we can generate the tag τ used in **KEM** not through a one-time signature scheme, but simply as $\tau = H(\mathbf{c}_0)$ for the collision-resistant hash function H. This change is possible since, by now, the ciphertext only contains $\mathbf{c}$, and in regular ciphertexts (that encrypt $\mathbf{0}$ as in the scheme), $\mathbf{c}_0$ completely determines that $\mathbf{c}$. Hence, in a security analysis, we can initially establish a special decryption rule that rejects all decryption queries with a reused $\mathbf{c}_0$ from a previous encryption query. (Initially, $\mathbf{PS}^{\mathrm{lin}}$ ensures that such ciphertexts must be completely identical to that previous encryption query, and thus can be rejected by the rules of the IND-MCCA security game.) A similar trick has been used in [12] for a similar purpose.

Optimized efficiency. If we apply all of the above optimizations, we obtain a practical scheme. Concretely, public keys consist of the following:

- a vector of 4 generalized ElGamal public keys ($\ell_{\mathbf{B}}$ group elements each)
- a public key $\mathit{ppk}_{\mathrm{lin}}$ for $\mathbf{PS}^{\mathrm{lin}}$ ($\ell_{\mathbf{B}}$ group elements),

- a public key ppk_{hash} for $\mathbf{PS}^{\text{hash}}$ ($2\ell_{\mathbf{B}}$ group elements),
- public keys $ppk_{\vee,2}$, $ppk_{\vee,4}$, and $ppk_{\vee,5}$ for $\mathbf{PS}^{\vee}$($2\ell_{\mathbf{B}}(\ell_{\mathbf{B}}+1)$ group elements each), and
- an extractor public key xpk ($\ell_{\mathbf{B}}$ group elements).

This amounts to $2\ell_{\mathbf{B}}^2 + 10\ell_{\mathbf{B}}$ group elements. Ciphertexts consist of $\ell_{\mathbf{B}} + 4$ group elements. The public parameters consist of $g_1^{\mathbf{B}}$, plus of course descriptions of H, h, and the groups $\mathbb{G}$, $\mathbb{G}_1$, and $\mathbb{G}_2$. (Depending on the considered assumption, $g_1^{\mathbf{B}}$ can be represented in a compact form [10].)

For instance, a DLIN-based scheme has ciphertexts, public keys, and public parameters of 6, 24, and 2 group elements. Thus, compared with the recent pairing-free tightly secure KEM of [12], our scheme has a significantly smaller public key, and comparable ciphertexts (of 6 group elements) in the DLIN setting. However, when instantiated as described in the prime-order setting, our scheme relies on symmetric pairings, and thus is computationally less efficient, and cannot be instantiated under the DDH assumption.

6.4.2 In the DCR setting

Efficiency calculation. In the DCR setting from Section 3.3, we can use the OR-proofs from Section 5.4.2, and the one-time signature scheme from Section 4.1.2. Hence, public keys contain the following:

- a vector of 4 generalized ElGamal public keys (1 group element each)
- a public key ppk_{lin} for $\mathbf{PS}^{\text{lin}}$ (1 group element),
- a public key ppk_{hash} for $\mathbf{PS}^{\text{hash}}$ (2 group elements),
- 6 public keys for $\mathbf{PS}^{\vee}$(2 group elements each), and
- an extractor public key xpk (1 group element).

This amounts to 20 group elements. Ciphertexts contain the following:

- a vector $\mathbf{c}$ of 4 ElGamal encryptions with reused randomness (5 group elements in total),
- proofs π_{lin} and π_{hash} (1 group element each),
- 6 $\mathbf{PS}_{\mathbf{DCR}}^{\vee}$ proofs (3 group elements each), and
- a verification key, and a signature of $\mathbf{OTS}_{\mathbf{DCR}}$ (2, resp. 3 group elements[10]).

This amounts to 30 group elements.

Hence, our DCR-based scheme is not very practical. However, it is the first DCR-based tightly CCA-secure KEM, and can potentially be made much more efficient by using more compact, or more expressive OR-proofs.

Optimizations. We remark that the proof systems $\mathbf{PS}^{\text{lin}}$ and $\mathbf{PS}^{\text{hash}}$ are in fact hash proof systems. Hence, with the same reasoning as in the prime-order setting, we can integrate π_{lin} and π_{hash} into the session key K, at the price of obtaining only constrained chosen-ciphertext security. This saves 2 group elements in the ciphertext (which however still contains 28 group elements in that case).

References

[1] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. "Disjunctions for Hash Proof Systems: New Constructions and Applications". In: *Proc. EUROCRYPT (2) 2015*. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 69–100.

[2] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. "Tagged One-Time Signatures: Tight Security and Optimal Tag Size". In: *Proc. Public Key Cryptography 2013*. Vol. 7778. Lecture Notes in Computer Science. Springer, 2013, pp. 312–331.

[3] Nuttapong Attrapadung, Goichiro Hanaoka, and Shota Yamada. "A Framework for Identity-Based Encryption with Almost Tight Security". In: *Proc. ASIACRYPT (1) 2015*. Vol. 9452. Lecture Notes in Computer Science. Springer, 2015, pp. 521–549.

[10] Here, we count the two elements $e, e' \in \{0, \ldots, \lfloor N/2 \rfloor\}$ together as one $\mathbb{G}$-element.

[4] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. "Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements". In: *Proc. EUROCRYPT 2000*. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 259–274.

[5] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. "(Hierarchical) Identity-Based Encryption from Affine Message Authentication". In: *Proc. CRYPTO (1) 2014*. Vol. 8616. Lecture Notes in Computer Science. Springer, 2014, pp. 408–425.

[6] Jan Camenisch and Victor Shoup. "Practical Verifiable Encryption and Decryption of Discrete Logarithms". In: *Proc. CRYPTO 2003*. Vol. 2729. Lecture Notes in Computer Science. Springer, 2003, pp. 126–144.

[7] Jie Chen and Hoeteck Wee. "Fully, (Almost) Tightly Secure IBE and Dual System Groups". In: *Proc. CRYPTO (2) 2013*. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 435–460.

[8] Ronald Cramer and Victor Shoup. "Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack". In: *SIAM J. Comput.* 33.1 (2003), pp. 167–226.

[9] Ronald Cramer and Victor Shoup. "Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption". In: *Proc. EUROCRYPT 2002*. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 45–64.

[10] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. "An Algebraic Framework for Diffie-Hellman Assumptions". In: *Proc. CRYPTO (2) 2013*. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 129–147.

[11] Roger Fischlin and Claus-Peter Schnorr. "Stronger Security Proofs for RSA and Rabin Bits". In: *Proc. EUROCRYPT 1997*. Vol. 1233. Lecture Notes in Computer Science. Springer, 1997, pp. 267–279.

[12] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. "Tightly CCA-Secure Encryption Without Pairings". In: *Proc. EUROCRYPT (1) 2016*. Vol. 9665. Lecture Notes in Computer Science. Springer, 2016, pp. 1–27.

[13] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks". In: *SIAM J. Comput.* 17.2 (1988), pp. 281–308.

[14] Junqing Gong, Jie Chen, Xiaolei Dong, Zhenfu Cao, and Shaohua Tang. "Extended Nested Dual System Groups, Revisited". In: *Proc. Public Key Cryptography (1) 2016*. Vol. 9614. Lecture Notes in Computer Science. Springer, 2016, pp. 133–163.

[15] Brett Hemenway and Rafail Ostrovsky. "Extended-DDH and Lossy Trapdoor Functions". In: *Proc. Public Key Cryptography 2012*. Vol. 7293. Lecture Notes in Computer Science. Springer, 2012, pp. 627–643.

[16] Dennis Hofheinz. "Algebraic Partitioning: Fully Compact and (almost) Tightly Secure Cryptography". In: *Proc. TCC (A1) 2016*. Vol. 9562. Lecture Notes in Computer Science. Springer, 2016, pp. 251–281.

[17] Dennis Hofheinz and Tibor Jager. "Tightly Secure Signatures and Public-Key Encryption". In: *Proc. CRYPTO 2012*. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 590–607.

[18] Dennis Hofheinz and Eike Kiltz. "Secure Hybrid Encryption from Weakened Key Encapsulation". In: *Proc. CRYPTO 2007*. Vol. 4622. Lecture Notes in Computer Science. Springer, 2007, pp. 553–571.

[19] Dennis Hofheinz and Eike Kiltz. "The Group of Signed Quadratic Residues and Applications". In: *Proc. CRYPTO 2009*. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 637–653.

[20] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. "Identity-Based Encryption with (Almost) Tight Security in the Multi-instance, Multi-ciphertext Setting". In: *Proc. Public Key Cryptography 2015*. Vol. 9020. Lecture Notes in Computer Science. Springer, 2015, pp. 799–822.

[21] Susan Hohenberger and Brent Waters. "Realizing Hash-and-Sign Signatures under Standard Assumptions". In: *Proc. EUROCRYPT 2009*. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 333–350.

[22] Eike Kiltz and Hoeteck Wee. "Quasi-Adaptive NIZK for Linear Subspaces Revisited". In: *Proc. EUROCRYPT (2) 2015*. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 101–128.

[23] Hugo Krawczyk and Tal Rabin. "Chameleon Signatures". In: *Proc. NDSS 2000*. The Internet Society, 2000.

[24] Kaoru Kurosawa and Yvo Desmedt. "A New Paradigm of Hybrid Encryption Scheme". In: *Proc. CRYPTO 2004*. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 426–442.

[25] Benoît Libert, Marc Joye, Moti Yung, and Thomas Peters. "Concise Multi-challenge CCA-Secure Encryption and Signatures with Almost Tight Security". In: *Proc. ASIACRYPT (2) 2014*. Vol. 8874. Lecture Notes in Computer Science. Springer, 2014, pp. 1–21.

[26] Benoît Libert, Thomas Peters, Marc Joye, and Moti Yung. "Compactly Hiding Linear Spans - Tightly Secure Constant-Size Simulation-Sound QA-NIZK Proofs and Applications". In: *Proc. ASIACRYPT (1) 2015*. Vol. 9452. Lecture Notes in Computer Science. Springer, 2015, pp. 681–707.

[27] Moni Naor and Omer Reingold. "Number-theoretic constructions of efficient pseudo-random functions". In: *J. ACM* 51.2 (2004), pp. 231–262.

[28] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Proc. EUROCRYPT 1999*. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238.

[29] Hovav Shacham. *A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants*. IACR ePrint Archive, report 2007/74. `http://eprint.iacr.org/2007/74`. 2007.